

Génération Physiquement Guidée de Terrains Numériques

Aryamaan Jain

Centre Inria d'Université Côte d'Azur – Équipe GraphDeco

Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur

Dirigée par : Guillaume Cordonnier

Soutenue le : 18 Septembre 2026

Devant le jury composé de :

Eric Guérin
Professeur, INSA Lyon

Jean-Michel Dischler
Professeur, Université de Strasbourg

Oliver Deussen
Professor, University of Konstanz

Wojciech Palubicki
Assistant Professor, Adam Mickiewicz University

Patrick Pérez
Research Scientist, Kyutai

Guillaume Cordonnier
Chargé de Recherche, Centre Inria d'Université Côte d'Azur

Inria

Génération Physiquement Guidée de Terrains Numériques

Physically Guided Generation of Digital Terrains

Jury:

Président du jury / President of the jury

Eric Guérin, Professeur, INSA Lyon

Rapporteurs / Reviewers

Eric Guérin, Professeur, INSA Lyon

Jean-Michel Dischler, Professeur, Université de Strasbourg

Examinateurs / Examiners

Oliver Deussen, Professor, University of Konstanz

Wojciech Palubicki, Assistant Professor, Adam Mickiewicz University

Patrick Pérez, Research Scientist, Kyutai

Directeur de thèse / Thesis supervisor

Guillaume Cordonnier, Chargé de Recherche, Centre Inria d'Université Côte d'Azur

Acknowledgments

I would like to begin by expressing my heartfelt gratitude to my supervisor, Guillaume Cordonnier. Over the past three years, your support, guidance, patience, and encouragement have meant a great deal to me. You helped me grow as a researcher and gave me the confidence to pursue my ideas. I am sincerely grateful for everything I have learned from you, and for the invaluable role you have played in the successful completion of this PhD.

My sincere thanks also go to Bedrich Benes. Your advice and expertise helped shape my work in meaningful ways. Thank you as well for inviting me to visit your group at Purdue for two months. It was a memorable experience, and I believe that visit, together with our many discussions, opened new perspectives for me.

I would also like to thank my collaborators: James Gain, Bernhard Kerbl, Marie-Paule Cani, Zhanyu Yang, Zhaopeng Wang, and Brandon Finley. I have enjoyed working with you, and I hope we will have many more opportunities to collaborate in the future.

I am thankful to my teammates for listening to me talk through my scientific challenges, for the discussions we had, and for the good moments we shared. Thank you to Panagiotis, Petros, Ishaan, Berend, Yohan, Henro, Alexandre, Nicolas V., Nicolas R., Andreas, Alban, Melike, Marzia, Simon, Linus, Jeffrey, Anran, Jackson, Gilda, Vishal, Loic, and Boris.

I would like to thank our team leader, George Drettakis. Your steady support helped make the group a place where ideas could develop freely, and your guidance at important moments helped me think with confidence about what comes next. I also thank Adrien Bousseau for his contributions to the group's scientific atmosphere, and Sophie Honnorat and Nathalie Nordmann for their outstanding administrative support.

I am especially grateful to everyone who supported and guided me during these three years and before: my friends, my former teachers, my family, and all those who have had a lasting impact on my life.

Finally, I want to thank my father, Anil, and my mother, Shilpa. You gave me a wonderful environment to grow up in, and helped me develop the curiosity and passion for the work I do today.

Résumé

Les terrains numériques sont au cœur de la création de vastes environnements naturels pour le cinéma, les jeux vidéo, la réalité virtuelle et la visualisation géographique. Leur réalisme, toutefois, ne dépend pas seulement de leur apparence géométrique : des paysages convaincants suggèrent également les processus naturels qui les ont formés. L'eau organise les réseaux de vallées, les sédiments sont érodés, transportés et déposés, les pentes instables produisent des coulées de débris, le vent abrase la roche exposée, et les propriétés des matériaux déterminent quelles structures subsistent. Reproduire intégralement cette histoire géologique est irréalisable en informatique graphique. Cette thèse défend plutôt une génération de terrains *guidée par la physique*, dans laquelle des modèles physiques simplifiés, des algorithmes numériques efficaces et des a priori appris sont combinés afin de créer des paysages plausibles, contrôlables et cohérents avec les processus qu'ils évoquent.

La thèse développe cet argument à travers quatre contributions complémentaires. La première introduit un modèle d'érosion par coulées de débris pour les terrains montagneux à l'échelle de la vallée. Ce modèle complète l'érosion fluviale sur les pentes abruptes à faible drainage, en produisant des cicatrices d'érosion caractéristiques et des cônes de dépôt au sein d'une simulation GPU unifiée. La deuxième contribution est *FastFlow*, un ensemble d'algorithmes GPU pour le routage des écoulements et le routage des dépressions. Ces algorithmes accélèrent le calcul des réseaux d'écoulement hydrologiques sur de grands champs de hauteur et fournissent des primitives computationnelles pour l'érosion, la formation des rivières, les lacs, le transport sédimentaire et la simulation d'écosystèmes. La troisième contribution étend la modélisation de terrains fondée sur la physique au-delà des champs de hauteur grâce à *Arenite*, un simulateur volumétrique de grès qui combine résistance dépendante des contraintes, abrasion éolienne, érosion fluviale et dépôt fondé sur des particules afin de générer des arches, falaises, grottes, piliers, cheminées de fée et buttes. La quatrième contribution, *Pixels2Peaks*, aborde le problème inverse de la reconstruction de terrain à partir d'une image unique. *Pixels2Peaks* combine estimation géométrique, diffusion consciente du terrain, ainsi que des contraintes géomorphologiques, hydrologiques et de cohérence de vue pour synthétiser des cartes de

hauteur complètes à partir d'observations visuelles partielles.

Puisque la modélisation des trajectoires individuelles des fluides et des particules à l'échelle de chaînes de montagnes entières est computationnellement insoluble, nous avons besoin d'un substitut physiquement valide pour capturer leurs effets macroscopiques. Ce substitut, le *réseau d'écoulement*, encode les motifs de transport essentiels au réalisme géologique, régissant non seulement l'eau, mais aussi le mouvement des sédiments, des roches et des débris. Nos travaux démontrent la polyvalence de cette abstraction : le modèle de coulées de débris repose sur des réseaux d'écoulement approximatifs ; *FastFlow* formalise et accélère le routage exact des écoulements sur de grands champs de hauteur ; *Arenite* généralise ces principes de routage de surface en 2D à des domaines volumétriques en 3D ; et *Pixels2Peaks* intègre les réseaux d'écoulement comme contraintes structurelles au sein de modèles de diffusion.

Ensemble, ces contributions montrent que la génération de terrains peut être informée par des principes physiques sans sacrifier l'efficacité ni le contrôle attendus dans les flux de travail de l'informatique graphique. Lorsque la structure physique est choisie à l'échelle appropriée et formulée efficacement, elle organise l'aléatoire, réduit l'ambiguïté et confère aux terrains générés des formes qui demeurent cohérentes lors de l'inspection, de l'édition et des simulations en aval.

Mots-clés: Génération de terrains, érosion, dépôt, coulée de débris, routage des écoulements, routage des dépressions, simulation fondée sur la physique, phénomènes naturels, modélisation volumétrique, apprentissage automatique, modélisation à partir d'images, modèles de diffusion

Abstract

Digital terrains are central to the creation of large natural environments for film, games, virtual reality, and geographic visualization. Their realism, however, is not only a matter of geometric appearance: convincing landscapes also suggest the natural processes that formed them. Water organizes valley networks, sediment is eroded, transported, and deposited, unstable slopes produce debris flows, wind abrades exposed rock, and material properties determine which structures remain. Reproducing this geological history in full is impractical for computer graphics. This thesis instead argues for *physically guided* terrain generation, where simplified physical models, efficient numerical algorithms, and learned priors are combined to create landscapes that are plausible, controllable, and consistent with the processes they evoke.

The thesis develops this argument through four complementary contributions. The first introduces a debris-flow erosion model for valley-scale mountainous terrain. The model complements fluvial erosion on steep, low-drainage slopes, producing characteristic erosive scars and depositional cones within a unified GPU simulation. The second contribution is *FastFlow*, a set of GPU algorithms for flow routing and depression routing. These algorithms accelerate the computation of hydrological flow networks on large heightfields and provide computational primitives for erosion, river formation, lakes, sediment transport, and ecosystem simulation. The third contribution extends physically-based terrain modeling beyond heightfields through *Arenite*, a volumetric sandstone simulator that combines stress-dependent resistance, wind abrasion, fluvial erosion, and particle-based deposition to generate arches, cliffs, caves, pillars, hoodoos, and buttes. The fourth contribution, *Pixels2Peaks*, addresses the inverse problem of reconstructing terrain from a single image. *Pixels2Peaks* combines geometric estimation, terrain-aware diffusion, and geomorphological, hydrological, and view-consistency constraints to synthesize complete heightmaps from partial visual observations.

Because modeling individual fluid and particulate paths at the scale of entire mountain ranges is computationally intractable, we require a physically valid proxy to capture their macroscopic effects. This proxy, the *flow network*, encodes the transport patterns essential to geological realism, governing not just water, but the movement of sediments, rocks, and

debris. Our work demonstrates the versatility of this abstraction: the debris-flow model relies on approximate flow networks; *FastFlow* formalizes and accelerates exact flow routing on large heightfields; *Arenite* generalizes these 2D surface routing principles into 3D volumetric domains; and *Pixels2Peaks* embeds flow networks as structural constraints within diffusion models.

Together, these contributions show that terrain generation can be informed by physical principles without sacrificing the efficiency and control expected in computer graphics workflows. When the physical structure is chosen at the appropriate scale and formulated efficiently, it organizes randomness, reduces ambiguity, and gives generated terrains forms that remain coherent under inspection, editing, and downstream simulation.

Keywords: Terrain Generation, Erosion, Deposition, Debris Flow, Flow Routing, Depression Routing, Physically Based Simulation, Natural Phenomena, Volumetric Modeling, Machine Learning, Image-Based Modeling, Diffusion Models

Contents

Contents **vii**

1 Introduction **1**
- 1.1 Motivation: Digital Terrain as Process . . . 1
- 1.2 Goal: Physically Guided Terrain Generation . . . 2
- 1.3 Previous Work in Digital Terrain Generation . . . 3
- 1.4 Challenge: Geological Scale & Artistic Control . . . 4
- 1.5 Thesis Statement . . . 4
- 1.6 The Unifying Abstraction: Flow Networks . . . 5
- 1.7 Contributions by Chapter . . . 6
- 1.8 Publications . . . 7

2 Background & Related Work **9**
- 2.1 Digital Terrain Representations . . . 9
- 2.2 Modeling Paradigms . . . 11
- 2.3 Erosion and Deposition Across Scales . . . 13
- 2.4 Flow Networks and Depression Routing . . . 17
- 2.5 Example-based and Learned Synthesis . . . 19
- 2.6 Terrain Authoring and Single-Image Reconstruction . . . 20

3 Efficient Debris-flow Simulation for Steep Terrain Erosion **23**
- 3.1 Introduction . . . 23
- 3.2 Overview . . . 25
- 3.3 A New Debris-Flow Erosion Model . . . 27
- 3.4 Erosion and Deposition Algorithm . . . 30
- 3.5 Depression Routing . . . 33
- 3.6 Results and Validation . . . 37
- 3.7 Conclusion . . . 47

4 FastFlow: GPU Acceleration of Flow and Depression Routing for Landscape Simulation **49**
- 4.1 Introduction . . . 49
- 4.2 Overview . . . 52
- 4.3 Flow Routing . . . 54
- 4.4 Depression Routing . . . 57

4.5 Application to Landscape Simulation . . . 63
4.6 Results . . . 67
4.7 Conclusion . . . 76

5 Arenite: A Physics-based Sandstone Simulator 77
5.1 Introduction . . . 77
5.2 Overview . . . 79
5.3 Fluvial Erosion on Sandstones . . . 84
5.4 Deposition . . . 87
5.5 Implementation, Results and Validations . . . 88
5.6 Conclusion and Future Work . . . 95

6 Pixels2Peaks: Converting Terrain Images to Heightmaps 97
6.1 Introduction . . . 97
6.2 Overview . . . 100
6.3 Initial Terrain Estimation from a Single Image . . . 102
6.4 Guided Diffusion for Terrain Completion . . . 106
6.5 Terrain Dataset Curation . . . 114
6.6 Results and Evaluation . . . 116
6.7 Conclusion . . . 130

7 Conclusion 133
7.1 Physically Guided Terrain Generation . . . 133
7.2 Main Contributions . . . 133
7.3 Limitations and Future Work . . . 135
7.4 Closing Perspective . . . 137

Chapter 1

Introduction

1.1 Motivation: Digital Terrain as Process

Landscapes are among the largest and most visible assets in digital media. Open-world games and feature films routinely depict tens or hundreds of square kilometers of terrain that a player can traverse freely or a camera can approach from any angle, and the same demand now extends to virtual reality, geographic visualization, training simulators, and digital twins used for planning and hazard assessment. In each of these settings the terrain is the substrate on which everything else is built. It fixes the placement of rivers, paths, and settlements, establishes the scale and geological character of a scene, and is reused by downstream simulations of water, vegetation, erosion, and sediment transport.

In production, artists rarely rely on a single source. Scanned elevation data is realistic but limited to places that already exist, while procedural tools synthesize relief from noise, which provides roughness but not the organization of valleys and ridges. The erosion simulations embedded in these tools recover that organization, but they are slow to iterate, sensitive to their parameters, and limited to the few processes they model. Manual sculpting offers full control but does not scale to kilometers of detail. They alternate between these, iterating until a terrain looks right, without any guarantee that it will behave right when water is simulated over it, when a camera moves closer, or when a region is edited.

These uses make terrain realism more demanding than simple visual plausibility. A mountain range can have convincing silhouettes and still feel artificial if its valleys do not organize into coherent drainage basins. A cliff can look impressive and still fail under close inspection if erosion scars and deposits are unrelated to the slope and water flow that should have formed them [Rajasekaran et al., 2022; Scott and Dodgson, 2022]. A natural landscape is a record of processes acting across time: water accumulates and incises valleys, sediment erodes and settles, unstable slopes collapse into debris flows, wind abrades exposed rock, and material strength determines which structures remain.

A convincing digital terrain, therefore, encodes its history.

While natural sciences rely on computationally intensive simulations of geological evolution, reproducing that history in full is beyond the needs and constraints of computer graphics, which require rapid iteration and artistic control. The goal is not to simulate every rainfall event or every fracture over geological time. A more practical question is instead: which physical structures must be preserved so that a generated terrain remains plausible when viewed, edited, or built upon? This thesis contributes to this direction by exploring one possible approach: *physically guided* terrain generation, where selected physical abstractions are used to constrain terrain synthesis without requiring exhaustive geological simulation.

1.2 Goal: Physically Guided Terrain Generation

The goal of this thesis is to generate digital terrains that are plausible, controllable and efficient. These requirements are tightly coupled. Visual plausibility requires physical consistency with coherent valleys, slopes, deposits and ridges [Rajasekaran et al., 2022; Scott and Dodgson, 2022]. Control is necessary because computer graphics workflows are creative workflows: an artist must be able to specify intent and iterate. Efficiency is essential because terrain generation is rarely a one-shot offline task. Parameters are changed, views are revised, scenes are composed, and intermediate results must be evaluated repeatedly.

This thesis does not seek geological accuracy. Instead, it focuses on dominant macroscopic signatures of terrain-forming processes. For example, water can be represented by discharge rather than by resolving short-term fluid dynamics; debris flows can be modeled through their long-term erosive and depositional effects rather than as individual hazardous events; sandstone can be represented through stress-dependent resistance and erosion mechanisms rather than through a complete history of cementation and fracture; and image-based terrain reconstruction can be guided by hydrological and geomorphological consistency without claiming to recover the true hidden landscape.

This point of view treats physics as a source of structure. Physical guidance may appear as a simplified simulation model or a constraint in a generative model. In each case, the aim is to choose a level of abstraction that captures the visible consequence of a process while remaining usable in a graphics workflow.

1.3 Previous Work in Digital Terrain Generation

Digital terrain modeling in computer graphics spans procedural methods, data-driven synthesis, and physically-based simulation [Galin et al., 2019]. Procedural methods were among the earliest successful approaches, exploiting the visual similarity between stochastic roughness and natural landforms [Fournier et al., 1982; Perlin, 1985; Musgrave et al., 1989]. They remain attractive because they are fast, compact, and controllable. However, local fractal rules and noise functions do not by themselves guarantee global geological coherence. A procedural mountain may have convincing roughness while lacking drainage organization.

Physical simulations address this limitation by modeling the processes that shape terrain. Some methods simulate water motion and sediment exchange directly through hydraulic erosion [Benes, 2007; Krištof et al., 2009]; however, because they resolve short-term water dynamics, their spatial and temporal range is limited. The work in this thesis builds primarily on fluvial erosion models, which abstract long-term river incision through geomorphological laws based on discharge and slope [Cordonnier et al., 2016]. This abstraction shows that large-scale terrain structure can be captured by routing flow and accumulating its effects, rather than by resolving every short-term water movement. Despite this advantage, [Cordonnier et al., 2016] and its subsequent extensions often suffer from computational bottlenecks, approximations, and an insufficient range of geomorphological processes to produce realistic landscapes. The challenge is therefore to extend and accelerate these routing-based abstractions, while enabling intuitive user control.

Data-driven methods provide a different trade-off. By learning from real terrain data, they can synthesize visually plausible landscapes and support interactive conditioning through sketches, masks, classes, or learned latent controls [Guérin et al., 2017; Lochner et al., 2023; Hu et al., 2024]. Yet visual plausibility does not guarantee structural correctness. Learned terrains may contain interrupted river networks, unrealistic closed basins, inconsistent slope statistics, or plausible-looking details that fail under editing or downstream simulation.

Accordingly, this thesis treats physical guidance not as a separate category of terrain generation, but as a design principle. For each problem, it identifies the physical structure that governs the landform at the target scale and uses flow networks to represent transport in a computationally tractable form. In forward generation, these abstractions drive

efficient simulations of erosion and deposition; in inverse generation, they constrain the solutions generated by a learned prior. The common role of physics is therefore to constrain terrain generation without requiring an explicit simulation of every individual geological event.

1.4 Challenge: Geological Scale & Artistic Control

The central difficulty is scale. Natural landforms are produced by interactions that are local in mechanism but global in consequence. A drop of rain or a falling rock follows a local path, but the accumulated effect of countless such events produces mountain ranges, drainage basins, talus fans, canyons, arches, and valleys. Simulating these event-scale interactions directly across kilometers of terrain and thousands or millions of years is computationally intractable for most graphics applications.

This creates a micro-to-macro disconnect. The features that matter visually are often macroscopic: the organization of valleys, the transition from debris-flow-dominated slopes to fluvial channels, the connectivity of rivers through depressions, or the completion of hidden terrain behind a photographed ridge. These structures emerge from many small-scale events, but reproducing them does not require explicitly simulating every event. A practical terrain model should therefore identify the macroscopic state variable that summarizes the process.

The second difficulty within computer graphics lies in exposing meaningful controls without breaking the physical structure a model is meant to preserve. This tension runs through the thesis. The methods developed here seek fast iteration, GPU execution, user-defined initial conditions, editable masks and parameters, and guidance terms that can steer generation while still preserving the process signatures that make the output credible.

1.5 Thesis Statement

This thesis argues that physically guided terrain generation, built on efficient process-level abstractions, can produce landscapes that are plausible, controllable, and physically coherent without requiring exhaustive geological simulation. The key is to select physical structure at the right scale: detailed enough to constrain the generated terrain, but abstract enough to remain efficient and steerable in computer graphics workflows.



Figure 1.1: *Flow networks abstract the dominant transport structure of a landscape. On this mountainous terrain above Nice, France, the network forms an invisible skeleton that follows river valleys and erosion scars near ridgelines. The prominent features reveal the long-term imprint of gravity-driven flow across the landscape. Terrain data from Google Earth.*

1.6 The Unifying Abstraction: Flow Networks

A recurring abstraction in this thesis is the *flow network*. A flow network represents how material moves through a landscape at a macroscopic scale. In a heightfield, it can be seen as a directed graph that connects each terrain grid cell to its downstream recipient and accumulates upstream contributions into discharge, sediment flux, or related quantities (see Figure 1.1). When depressions are present, the network must also describe how flow fills and escapes local minima. More generally, a flow network encodes where water, sediment, rocks, and debris are likely to concentrate and move.

This abstraction is useful because it replaces countless microscopic trajectories with an organized transport structure. It does not simulate every particle. Instead, it captures the pathways that dominate visible terrain formation. River valleys, erosion channels, debris cones, lake outlets, vegetation moisture patterns, and hydrological losses in learned terrain generation, all depend on this structure. Flow networks therefore provide a physically meaningful bridge between local process and large-scale shape.

Across the thesis, flow networks serve as a family of process-level abstractions: approximate or exact, surface-based or volumetric, and embedded either in physical simulation or as constraints on learned generation. Their recurring role is to preserve coherent transport while allowing the representation and computational method to change with the model.

1.7 Contributions by Chapter

This thesis is organized around four contribution chapters, each addressing a different part of physically guided terrain generation. Across these chapters, flow networks develop from approximate (Chapter 3) to exact for GPU flow and depression routing (Chapter 4), before being generalized from surfaces to volumetric transport (Chapter 5) and used as a hydrological constraint on learned terrain generation (Chapter 6).

Chapter 3: Debris-flow erosion. The first contribution chapter addresses a process missing from fluvial terrain models: debris-flow erosion on steep, low-discharge mountain slopes. It introduces a debris-flow erosion and deposition model that captures long-term erosive scars and depositional cones, and couples this process with fluvial erosion in a unified GPU simulation. Because deposition can create new local minima that interrupt downstream transport, the chapter also introduces a GPU routing method for maintaining approximate flow networks through these evolving depressions. This work shows how introducing a process at the appropriate scale can recover missing valley-scale structure, while the routing difficulties created by deposition motivate the exact flow network computation developed in the next chapter.

Chapter 4: GPU flow and depression routing. Motivated by this routing problem, the second contribution chapter reformulates flow accumulation over stream trees and depression routing over terrain basins as parallel GPU algorithms, replacing the approximate treatment required in Chapter 3 with exact routing. This makes it practical to recompute hydrological structure during interactive or repeated simulations, and supports applications such as river generation, lake formation, fluvial erosion, sediment deposition, and ecosystem simulation. Having established an efficient and exact foundation for surface flow, the thesis next asks how process-based transport can be extended beyond heightfields to volumetric terrain.

Chapter 5: Volumetric sandstone simulation. Building on this foundation, the third contribution chapter extends flow networks from heightfields to simulate volumetric sandstone landforms. It combines stress-dependent resistance, material layering, wind erosion, fluvial erosion, and particle-based deposition to model the processes responsible for characteristic sandstone landforms. *Arenite* therefore broadens physically guided terrain generation from surfaces to erodible volumes.

Chapter 6: Image-based terrain reconstruction. Where the preceding chapters generate terrain forward from modeled processes, the final contribution chapter considers the inverse problem of reconstructing a complete terrain heightmap from a single landscape image. Since the visible image provides only partial view-dependent evidence, *Pixels2Peaks* combines camera and scale estimation with a diffusion model trained on elevation data. The generation is guided by geomorphological, hydrological, and view-consistency constraints, so that inferred terrain remains compatible with the visible photograph while also forming coherent drainage structure guided by flow networks in occluded regions. This final contribution completes the thesis's progression by demonstrating how physical processes and flow networks can constrain learned terrain inference when observations are incomplete or ambiguous.

1.8 Publications

The work presented in Chapter 3 has been published:

> Aryamaan Jain, Bedrich Benes, Guillaume Cordonnier. *Efficient Debris-flow Simulation for Steep Terrain Erosion.* In ACM Transactions on Graphics (Proc. SIGGRAPH), Vol. 43, No. 4. 2024.

The work presented in Chapter 4 has been published and awarded Best Paper:

> Aryamaan Jain, Bernhard Kerbl, James Gain, Brandon Finley, Guillaume Cordonnier. *FastFlow: GPU Acceleration of Flow and Depression Routing for Landscape Simulation.* In Computer Graphics Forum (Proc. Pacific Graphics), Vol. 43, No. 7. 2024.

The work presented in Chapter 5 has been published. My specific contribution was independently developing the 3D flow and depression routing, along with the deposition model:

> Zhanyu Yang, Aryamaan Jain, Guillaume Cordonnier, Marie-Paule Cani, Zhaopeng Wang, Bedrich Benes. *Arenite: A Physics-based Sandstone Simulator.* In ACM Transactions on Graphics (Proc. SIGGRAPH), Vol. 44, No. 4. 2025.

The work presented in Chapter 6 has been published:

> Aryamaan Jain, James Gain, Guillaume Cordonnier. *Pixels2Peaks: Converting Terrain Images to Heightmaps.* In ACM Transactions on Graphics (Proc. SIGGRAPH), Vol. 45, No. 4. 2026.

Chapter 2

Background & Related Work

Creating realistic digital terrain is fundamentally a trade-off between physical accuracy and computational cost. This chapter surveys key strands of work across computer graphics, machine learning and some elements from geomorphology that have navigated this balance, establishing the context for the methods proposed in this thesis. The review begins with the core geometric representations that dictate how terrain is stored (Section 2.1) and the high-level modeling paradigms used to generate it (Section 2.2). From there, the focus shifts to the specific geomorphic mechanics of erosion and deposition (Section 2.3). Because simulating these processes at scale requires efficient data structures, Section 2.4 covers the underlying flow-routing algorithms. The chapter concludes by examining how data-driven methods are changing the field, specifically through learned synthesis (Section 2.5) and inverse single-image reconstruction (Section 2.6).

2.1 Digital Terrain Representations

Digital terrain modeling begins with the choice of a geometric representation. This choice determines which landforms can be expressed and which generation or simulation methods remain computationally practical.

Elevation models. A terrain is commonly represented as an elevation function that assigns a height value to every point on a two-dimensional plane (see Figure 2.1). In practice, this function is usually sampled on a regular grid to form a *heightfield*. When the samples are measurements of a real landscape, the result is generally called a digital elevation model (DEM) [Galin et al., 2019]. Heightfield grids are straightforward to render, and particularly well suited to GPU computation and learning-based methods. Triangulated irregular networks (TINs) provide an alternative sampling in which resolution can be adapted to terrain complexity. Both representations describe terrain as a 2D surface embedded in 3D, with each horizontal position corresponding to exactly one

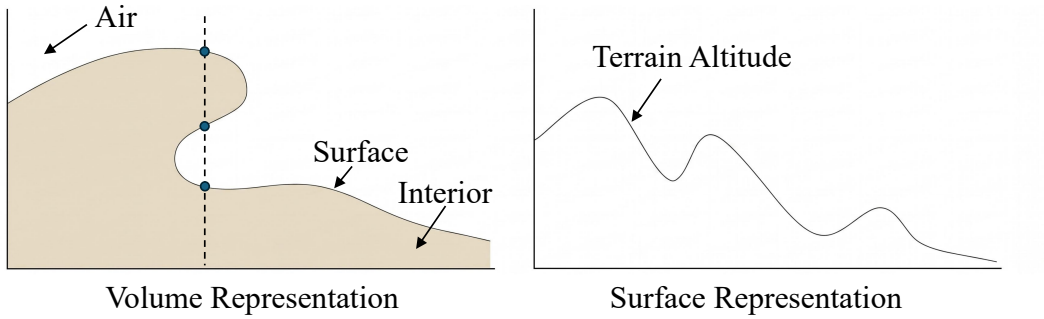


Figure 2.1: *Side-view comparison of volumetric and heightfield terrain representations. A volumetric representation can encode features such as overhangs or cavities, where a horizontal position may correspond to multiple elevations (left); a heightfield assigns exactly one elevation to each horizontal position (right). Adapted from [Cordonnier, 2018].*

elevation. This assumption is well matched to large-scale landscapes, typically spanning hundreds of meters to kilometers, where the dominant structures are ridges, valleys, slopes, channels, drainage basins, and sediment deposits. At this scale, local cavities or overhangs are usually less important than the organization of the visible surface and the movement of water and material across it. This is why heightfields are used in Chapter 3 for valley-scale debris-flow erosion, in Chapter 4 for flow and depression routing over large landscapes, and in Chapter 6 for single-image reconstruction of hills and mountains. The limitation is that heightfields cannot represent smaller-scale landforms containing folds, overhangs, caves, or other features for which a horizontal position corresponds to multiple elevations.

Volumetric models. Layered stacks provide an intermediate representation by extending heightfields with material thicknesses and subsurface structure, and can insert air intervals to encode overhangs and arches while retaining a column-wise organization (see Figure 2.2). Fully three-dimensional representations, including implicit fields, meshes, and voxels, give greater representational flexibility by assigning material throughout a volume. This flexibility becomes necessary at the scale of individual geological features and rock masses, often from meters to hundreds of meters, where the shape depends on cavities or overhangs rather than only on the top surface. Sandstone landforms are a typical example: arches, alcoves, caves, pillars, hoodoos, and buttes cannot be adequately described as single-valued heightfields. This motivates Chapter 5 to use a volumetric representation. The cost of this expressiveness is substantially higher computation and

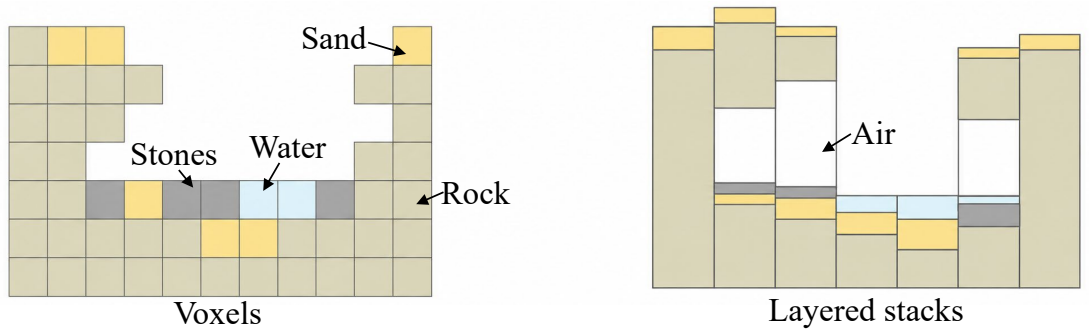


Figure 2.2: *Layered and voxel terrain representations. In a layered stack (right), each column stores a sequence of material layers, including air intervals that can encode overhangs while preserving a heightfield-like organization. In a voxel grid (left), material labels are assigned throughout the volume, allowing more general topology at higher computational and memory cost. Adapted from [Galin et al., 2019].*

memory use [Galin et al., 2019; Smelik et al., 2014]. Prior work balances that cost against control in several ways: 3D geomorphological objects can be interactively carved in a layered representation [Peytavie et al., 2009]; construction trees and open shape grammars arrange implicit primitives to generate diverse volumetric landforms [Paris et al., 2019a]; and curve-based primitives guide diffusion processes to model volumetric terrain features [Becher et al., 2019]. Such methods can directly author complex topology, whereas volumetric simulation lets that topology emerge from physical processes, but is generally more expensive and harder to direct.

These representations provide the geometry for terrain modeling, but do not determine how the geometry is produced. We next review the principal paradigms used to generate and modify those terrains.

2.2 Modeling Paradigms

Representation and generation methods are closely connected, but each representation can support several modeling strategies. Terrain-modeling techniques are conventionally divided into procedural, simulation-based, and example-based approaches, as illustrated in Figure 2.3. This section only sketches these paradigms and their trade-offs so as to position the thesis, and we refer the reader to the survey of Galin et al. [2019] for a comprehensive review. The strands most relevant to this work are examined in depth in Sections 2.3 to 2.6, and each chapter revisits the related work specific to its contribution.



Figure 2.3: *Three terrain-modeling paradigms. Procedural methods synthesize terrain from rules such as combined noise functions (left) [Ebert et al., 2002]; simulation-based methods generate landforms by modeling processes such as fluvial erosion (center) [Cordonnier et al., 2016]; and example-based or learned methods infer terrain structure from data, here through a generative adversarial network conditioned on a user sketch (right) [Guérin et al., 2017].*

Procedural methods. Procedural methods synthesize terrain from compact rules without simulating physical processes or drawing on real data. Early work exploited the visual similarity between fractal functions and mountainous relief, through midpoint-displacement subdivision [Fournier et al., 1982], faulting [Mandelbrot, 1983], and sums of gradient noise at increasing frequencies [Perlin, 1985; Ebert et al., 2002]. Noise-based generation remains the workhorse of interactive tools because each elevation can be evaluated independently, and hence in parallel, but for the same reason it cannot capture material transport, so fractal terrains mainly reproduce statistically self-similar roughness rather than the process-driven organization of eroded landscapes [Musgrave et al., 1989]. Perceptual studies confirm that realistic terrain also depends on characteristic ridges, valleys, and drainage structure [Rajasekaran et al., 2022; Scott and Dodgson, 2022]. A second family of procedural methods trades this global, dimensionless synthesis for explicit spatial control over specific landforms: coupling subdivision with the generation of river networks [Kelley et al., 1988; Prusinkiewicz and Hammel, 1993], diffusing elevation and slope constraints from feature curves [Hnaidi et al., 2010], combining landform primitives in construction trees [Génevaux et al., 2015], or growing a river network first and deriving the relief from it [Génevaux et al., 2013]. Constructive approaches to volumetric landforms and sketch-based authoring are discussed in Sections 2.1 and 2.6. These methods give artists direct control but leave the plausibility of the result to the designer of the rules.

Simulation-based methods. Simulation-based methods model the processes that form a landscape [Musgrave et al., 1989; Cordonnier et al., 2016]. Although their physical detail varies, most organize terrain formation around the detachment, transport, and deposition of material by water, gravity, ice, or wind, resolved at scales ranging from seconds of water flow over a few meters to millions of years of uplift and incision over mountain ranges. The appeal lies in the way plausible terrain arises from simulating recognizable geomorphic processes: drainage networks, talus slopes, and valley profiles emerge rather than being drawn. Depending on the processes included, a simulation either refines an existing procedural or sketched surface or, with tectonic uplift, grows an entire landscape as the balance between uplift and erosion. The drawback is that these models can be computationally demanding and hard to control, since their parameters are physical quantities such as erodibility or precipitation that do not map cleanly onto an artist's intended shape. Section 2.3 reviews these processes in detail, while Section 2.4 examines the flow-routing structures on which many of them depend.

Example-based and learned methods. Example-based and learned methods make the complementary trade-off. Rather than encoding the rules that produce terrain, they derive a prior from existing elevation data, whether by copying and blending patches of real terrain [Zhou et al., 2007] or by training a generative model on it [Guérin et al., 2017; Lochner et al., 2023; Hu et al., 2024]. Realism is thus inherited from the data rather than earned through simulation, results are produced quickly, and controls such as sketches, masks, or class labels can be exposed through conditioning. The limitations mirror these strengths. The output is bounded by the coverage and resolution of the exemplars, the statistics it reproduces are correlational rather than causal, and nothing in the training objective enforces process-level constraints, so plausible-looking terrains can fail when water is routed over them or when they are edited. Section 2.5 reviews how these methods synthesize terrains and enable terrain control.

2.3 Erosion and Deposition Across Scales

We now review terrain erosion and deposition models, focusing on the geomorphic processes they represent. Water is one of the principal agents in this transformation, and terrain models represent its action at different temporal and spatial scales. Hydraulic erosion methods resolve comparatively short-term water motion and its exchange of

sediment with the ground, while fluvial methods abstract that motion into long-term discharge and incision. Other erosion models address non-fluvial transport processes, such as thermal relaxation and debris flows on steep slopes. These approaches are complementary because no single process dominates every part of a landscape.

Hydraulic erosion. The first physically-based terrain-generation methods used hydraulic erosion to add water-carved structure to procedural relief [Musgrave et al., 1989]. Operating primarily at localized spatial scales (ranging from submeter resolutions to a few square kilometers) and short temporal scales (simulating fractions of a second up to several years), they approximate a shallow layer of water flowing over the terrain, using grid-based dynamics or particle formulations [Benes, 2007; Krištof et al., 2009] (Figure 2.4), and model mechanical interaction between the water and underlying material [Wojtan et al., 2007]. Later work refined the coupling to fluid motion and velocity fields [Neidhold et al., 2005; Anh et al., 2007; Chiba et al., 1998], reached interactive rates [Šťava et al., 2008], and used GPU tiling to process larger terrains [Vanek et al., 2011]. Erosion and deposition process simulations modeled the exchange of material during terrain evolution [Roudier et al., 1993], while layered terrain representations introduced strata with different resistance [Benes and Forsbach, 2001]. These models can produce compelling river-bed detail, but resolving water dynamics constrains their timestep and makes geological time spans expensive to approximate.

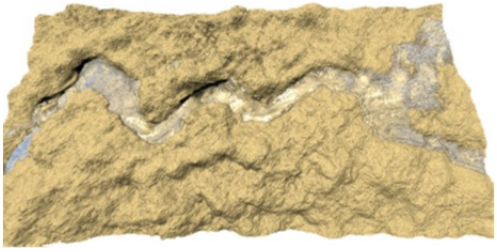


Figure 2.4: *Hydraulic erosion by a river [Krištof et al., 2009].*



Figure 2.5: *Large scale terrain generation [Cordonnier et al., 2016].*

Fluvial erosion. Geomorphological models avoid repeated short-term fluid simulation by representing water through discharge, which is the volume of water passing through a cross-section per unit time and is computed from accumulated upstream precipitation over the terrain. Stream-power laws relate erosion to discharge and local slope [Whipple and Tucker, 1999; Braun and Willett, 2013], enabling the formation of mountain ranges over much

larger regional spatial extents (tens to thousands of square kilometers) and geological time scales (thousands to millions of years) [Cordonnier et al., 2016] (Figure 2.5). Because uplift and erosion act together at this macroscopic scale, uplift fields have also been exposed as an authoring mechanism [Cordonnier et al., 2017a], and approximate GPU schemes propagate discharge locally to improve performance [Schott et al., 2023]. River networks are thus both a visible terrain feature [Génevaux et al., 2013] and the computational structure through which long-term erosion is evaluated. Related work has extended the process vocabulary to inverse "unerosion" [Yang et al., 2024], glacial landscapes [Cordonnier et al., 2023], and interactions with flora and fauna [Ecormier-Nocca et al., 2021; Cordonnier et al., 2017b].

The abstraction that makes fluvial erosion scalable also limits what it captures. Stream-power models are designed for large drainage areas and do not, merely by increasing resolution, recover the characteristic incision of steep hillslopes. This limitation is especially visible on cliffs exposed by glacial retreat, for which an empirical debris-flow approximation was introduced alongside glacial erosion [Cordonnier et al., 2023].

Thermal erosion and debris flows. Thermal erosion was introduced to relax slopes that exceed a material's angle of repose [Musgrave et al., 1989]. It serves as an inexpensive proxy for weathering and gravity-driven mass movement in computer graphics, but repeated local relaxation tends to produce smooth, uniform talus slopes. Geomorphological studies describe steep-slope patterns that this graphics abstraction does not reproduce. Incision can occur in low-drainage hillslopes through episodic debris flows, and the dominant erosion process can shift from debris-flow incision to fluvial incision as drainage area increases [Stock and Dietrich, 2006]. During intense rainfall, loose rock and sediment mix with water and move downslope as a dense mass [Takahashi, 1981]. Geomorphological research has consequently modeled the initiation, propagation, and deposition of individual events [Takahashi, 1978; Hutter et al., 1994], with an emphasis on predictive physical accuracy [Iverson, 1997; Xia et al., 2023]. Such event-scale simulations are too costly to integrate over the thousands of events implicated in long-term landscape evolution.

Reduced geomorphological laws therefore model the aggregate effect of repeated debris flows rather than each event [McGuire et al., 2022]. Chapter 3 follows this process-level abstraction by generalizing the effect represented by thermal relaxation and coupling erosion with deposition. This fills a process gap between hillslope relaxation and fluvial

erosion while retaining a time scale suitable for terrain generation.

Volumetric erosion and sandstone formation. Heightfield processes capture mountain scale topography but cannot produce the concavities and disconnected material characteristic of sandstone arches, alcoves, and caves. Computer graphics models of volumetric erosion address this representation problem by removing and transporting material in 3D domains. Such erosion has been formulated with Eulerian grids [Benes et al., 2006], while particle formulations provide a Lagrangian alternative to grid-based hydraulic erosion [Krištof et al., 2009]. Related approximations include directable weathering of concave rock [Jones et al., 2010], aeolian erosion models [Krs et al., 2020; Paris et al., 2019b], and particle-based weathering and transport across multiple terrain representations [Hartley et al., 2024].

These models show how erosion can act within volumes, but they do not by themselves explain why sandstone landscapes localize erosion into openings, pillars, arcades, and alcoves rather than eroding as homogeneous rock. Geomorphological studies point to the missing controls through the interaction of lithology, weathering, stress, and transport [Turkington and Paradise, 2005; Adamovic et al., 2006; Cílek et al., 2007]. Experiments and numerical models show that compressive stress can strengthen sandstone against erosion [Bruthans et al., 2014; Ostanin et al., 2017] and contribute to arcade formation [Safonov et al., 2020]. Wind-driven abrasion and deflation are likewise important in exposed, sparsely vegetated environments [Greeley and Iversen, 1985; Loope and Mason, 2009]. A broader geomorphological review is provided by Migoń [2021]. These studies identify mechanisms behind the patterns that generic volumetric erosion leaves unexplained, but they generally remain observations, experiments, or process-specific models rather than a controllable terrain-generation workflow. Chapter 5 combines these observations in a controllable volumetric simulator based on stress-dependent resistance, wind abrasion, fluvial erosion, deposition, and particle transport.

The process contributions above differ in representation and scale, but depend on coherent transport through terrain that changes during simulation. In particular, deposition continually alters drainage paths and creates local minima. The next section therefore moves from the processes that shape terrain to the flow and depression routing that keep those processes connected.

2.4 Flow Networks and Depression Routing

Flow routing assigns downstream connections to terrain samples, while flow accumulation propagates local inputs, such as rainfall, along those connections to compute discharge. This operation is fundamental in hydrology and geomorphology, but it also underpins terrain erosion, river generation, sediment transport, and ecological simulation in computer graphics.

Flow routing as a computational primitive. Large-scale erosion models accumulate precipitation from ridges toward outlets instead of explicitly resolving short-term water motion [Cordonnier et al., 2016; Braun and Willett, 2013]. The resulting dependency paths extend across the terrain and are less locally parallel than shallow-water updates [Vanek et al., 2011]. Approximate GPU erosion can propagate discharge only a few cells per timestep [Schott et al., 2023], but this requires many explicit iterations and assumes a sufficiently stable network, which deposition or time-dependent tectonics can invalidate. Efficient routing is also useful beyond geomorphology. Ecosystem models use precipitation-derived flow maps to estimate water availability, particularly in riparian areas [Gain et al., 2017]. The Topographic Wetness Index combines local slope with catchment area obtained by flow routing [Riihimäki et al., 2021], so repeated flow computation can become a substantial cost in ecosystem simulations evaluated at daily or weekly intervals.

Flow-direction models. Routing methods operate on regular grids [Garbrecht and Martz, 1997; O'Callaghan and Mark, 1984] or TINs [Banninger, 2007]. A grid typically connects each cell to its four orthogonal or eight surrounding neighbors. Single-flow-direction (SFD) methods select one lower neighbor as the receiver of each cell, producing a forest of drainage trees. The selection rule for the lower neighbor changes the dendritic patterns observed. For example, stochastic routing can avoid the overly regular patterns produced by always choosing the steepest descent [Tzathas et al., 2024]. Multiple-flow-direction (MFD) methods instead divide flow among several lower neighbors according to elevation differences [Quinn et al., 1991; Coatléven, 2020]. A common formulation, due to Holmgren [Holmgren, 1994], modulates the slope s_{ij} from cell i to each lower neighbor j by an exponent $p \in [1, \infty)$, sending the fraction $w_{ij} = s_{ij}^p / \sum_{k \in \mathcal{B}_i} s_{ik}^p$ of the flow to j, where $\mathcal{B}_i$ is the set of lower neighbors of i. Setting $p = 1$ distributes flow proportionally

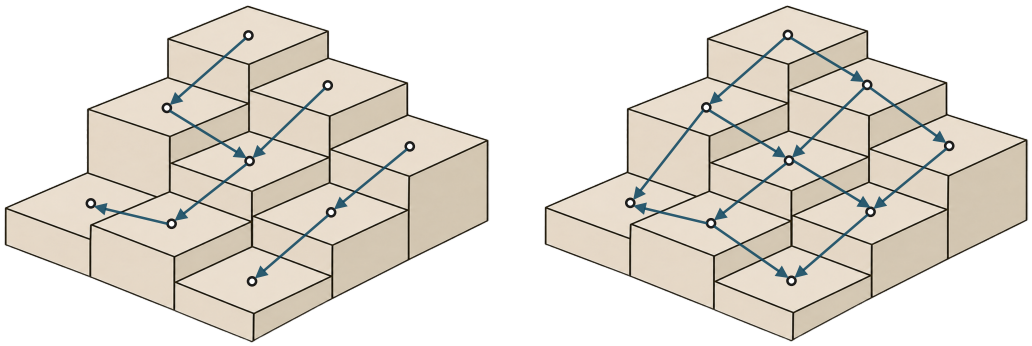


Figure 2.6: *Comparison of flow routing on the same 4-connected elevation grid. Single-flow-direction routing (left) sends all discharge from each cell to one downslope receiver, producing a tree-like drainage network. Multiple-flow-direction routing (right) splits discharge among all lower neighbors, representing divergent flow but yielding a directed acyclic graph rather than a tree.*

to slope, $p \to \infty$ recovers the steepest-descent receiver of SFD, and intermediate values interpolate smoothly between the two, with $4 \leq p \leq 6$ reported as most accurate. MFD methods represent divergent flow more naturally, but replace the tree structure of SFD with a directed acyclic graph and therefore require different parallel algorithms. Figure 2.6 illustrates this structural distinction on a 3×3 grid.

Previous acceleration strategies exploit parallelism across CPU cores [Barnes, 2017] or GPU threads [Barnes, 2019; Schott et al., 2023]. They nevertheless propagate information sequentially along each path, taking $\mathcal{O}(l)$ rounds for a longest channel of length l, which is commonly of order $\sqrt{n}$ for n terrain cells. Chapter 4 exploits the SFD tree structure to compress those paths in parallel and reduce routing to $\mathcal{O}(\log n)$ rounds.

Depressions. Ordinary downhill routing terminates at local minima. Some depressions correspond to lakes or genuine endorheic basins, whereas many are artifacts of measurement or synthesis. Hydrological terrain processing may fill them, breach their rims, or route their overflow [Wang et al., 2019]. For the simulations considered here, depression routing constructs a connected path from each interior minimum to a domain outlet, allowing a depression to fill and spill without interrupting the global flow network.

Priority-Flood algorithms process cells from the boundary inward using an elevation-ordered queue and obtain $\mathcal{O}(n \log n)$ complexity [Barnes et al., 2014]. Subsequent methods reduced their reliance on priority queues [Zhou et al., 2016; Wei et al., 2018] and

introduced CPU parallelism [Do et al., 2011; Barnes, 2016; Zhou et al., 2017; Barnes et al., 2020]. A different formulation observes that depression routing can be expressed through a minimum spanning tree and solved in linear time on a planar graph [Cordonnier et al., 2019]. Chapter 4 adapts this formulation to the GPU and combines it with logarithmic-time flow accumulation.

Chapter 5 extends these surface abstractions to volumes, where flow routing transports water and sediment over exposed 3D sandstone surfaces, while depression routing connects local minima to outlets.

2.5 Example-based and Learned Synthesis

So far, this chapter has focused on terrain generation through explicit models of natural processes. Example-based and learned methods take the complementary view that plausible terrain structure can instead be inferred from real-world elevation data. Such a data-driven prior is particularly useful when the desired landform is easier to demonstrate than to describe through simulation parameters. It also makes authoring controls more directly expressive of the intended result, from broad landscape categories [Zhao et al., 2019] and spatially varying styles [Perche et al., 2023] to specialized landform families such as river deltas and coastal terrain [Valencia-Rosado et al., 2020] and top-down sketches of ridges and drainage networks [Guérin et al., 2017; Lochner et al., 2023; Hu et al., 2024].

This line of work began with methods that adapted texture synthesis to transfer and recombine structure from elevation exemplars [Zhou et al., 2007; Gain et al., 2015; Scott and Dodgson, 2021]. Neural approaches subsequently shifted the emphasis from assembling observed patterns to learning distributions from collections of terrains. Regression models learned to recover fine detail from low-resolution elevation data [Argudo et al., 2018; Kubade et al., 2020], while generative adversarial networks (GANs) enabled conditional synthesis, editing, terrain amplification, and category-guided generation [Guérin et al., 2017; Zhao et al., 2019; Jain et al., 2024; Naik et al., 2022; Perche et al., 2023]. More recently, diffusion models have provided a flexible generative prior for terrain at multiple scales, including synthesis constrained by user sketches [Jain et al., 2022; Lochner et al., 2023; Hu et al., 2024].

Despite this progression, a learned terrain prior does not by itself guarantee hydrological consistency. Diffusion models can represent a broad and diverse range of terrain

while avoiding the training instability and repetitive artifacts often associated with conditional GANs [Guérin et al., 2017]. Their training objective does not, however, require water to reach an outlet, so generated terrains may contain large endorheic basins or small artificial pits. Prior example-based work addressed this problem by incorporating depression breaching into multi-resolution texture synthesis [Scott and Dodgson, 2021], but texture synthesis is less able than diffusion models to generate plausible terrain structure [Lochner et al., 2023]. Chapter 6 employs a diffusion prior for terrain synthesis while using flow-derived objectives to guide generation toward hydrologically consistent terrain.

2.6 Terrain Authoring and Single-Image Reconstruction

The preceding section considered learned terrain priors driven primarily by top-down constraints. Chapter 6 instead starts from a perspective photograph and asks how its partial, view-dependent evidence can constrain a heightfield. This inverse problem connects two bodies of work: first-person terrain-authoring interfaces, which translate image-space cues into geometric edits, and general single-image reconstruction methods, which infer 3D structure from visual evidence.

2.6.1 First-Person Terrain Authoring

First-person terrain authoring provides the closest interface-level precedent for controlling a landscape from a single depicted view. Terrain-authoring systems commonly adopt painting or sketching metaphors in either a top-down or first-person view [Galin et al., 2019]; in the latter case, strokes constrain the terrain in the same perspective as a photograph. Early systems projected the endpoints of silhouette strokes onto a ground plane or an existing heightfield and deformed the surface to interpolate them [Cohen et al., 2000]. Later methods localized wavelet-noise warping around silhouettes [Gain et al., 2009] (Figure 2.7), minimized diffusion-based deformation of an existing terrain [Tasse et al., 2014a,b], or cut and blended exemplar patches [dos Passos and Igarashi, 2013].

By construction, these interfaces reproduce the terrain in the drawn view, but a plausible silhouette does not determine a plausible landscape beyond it. The ground-plane projection of a stroke may be assumed linear [Cohen et al., 2000], edited separately in a top-down view [Gain et al., 2009], or inherited from deformed and blended source features [dos Passos and Igarashi, 2013; Tasse et al., 2014a]. These strategies either require

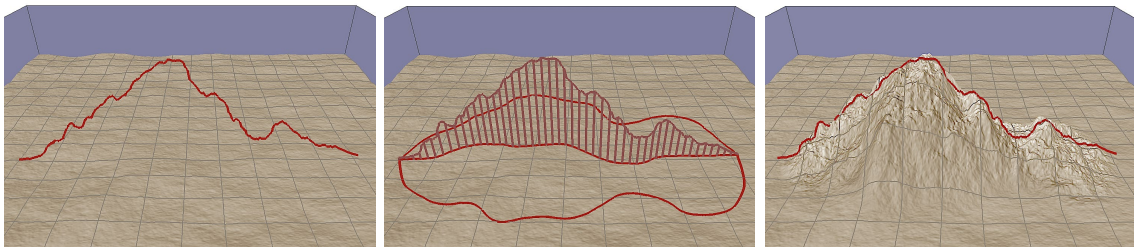


Figure 2.7: *Terrain Sketching workflow from [Gain et al., 2009]: left, the user sketches a silhouette curve; center, the shadow and boundary curves can be adjusted; right, a matching landscape is generated.*

additional author input or risk distorting drainage and ridge organization. A photograph replaces sparse strokes with richer depth cues, but converting those cues into terrain constraints requires moving from interactive authoring to inverse reconstruction.

2.6.2 Single-Image 3D Reconstruction

Where first-person authoring begins with sparse, user-drawn constraints, single-image reconstruction attempts to infer geometry directly from dense visual evidence. The problem nevertheless remains ill posed because many 3D scenes can produce the same projection. Classical methods relied on geometric cues and shape from shading [Horn, 1986; Zhang et al., 1999]. Deep learning models subsequently learned explicit voxel grids [Choy et al., 2016], point clouds [Fan et al., 2017], and meshes [Wang et al., 2018], or represented surfaces implicitly [Mescheder et al., 2019]. Neural radiance fields, initially designed for multiple view reconstruction [Mildenhall et al., 2021], have also been adapted to single-image novel-view synthesis [Yu et al., 2021].

Pretrained 2D diffusion models introduced an alternative source of 3D priors. DreamFusion uses Score Distillation Sampling (SDS) to optimize a neural radiance field without 3D training data [Poole et al., 2022]. Subsequent methods condition reconstruction on a single input image [Liu et al., 2023a; Qian et al., 2023]. Large feed-forward models can now generate detailed general-purpose assets from a single image [Hong et al., 2023; Zhang et al., 2024; Li et al., 2025; Tencent, 2025]. Although effective for common objects, these methods are not designed for large-scale natural landscapes or hydrological validity.

Terrain-specific single-image reconstruction remains comparatively scarce. The closest pipeline projects estimated depth and shadow-free color to a top-down heightfield, then completes missing elevation and appearance with a conditional GAN [Takahashi et al.,

2022]. Chapter 6 extends this approach with camera estimation, variable-resolution terrain diffusion, and geomorphological, hydrological, and view-consistency guidance.

Together, these works expose a recurring challenge: terrain models must balance geomorphological plausibility, computational efficiency, and controllability across representations and scales. The next chapter addresses this challenge at the valley scale by introducing an efficient debris-flow erosion and deposition model for steep terrain, bridging the gap between thermal relaxation and large-scale fluvial erosion.

Chapter 3

Efficient Debris-flow Simulation for Steep Terrain Erosion

3.1 Introduction

Natural landscapes are fascinating because of their monumental presence and striking diversity. Our deepening understanding of *orogeny* – the geological origins of mountainous landscapes – allows us now to simulate the emergence of consistent mountain ranges.

One such example is fluvial erosion, used to generate massive ranges at scales up to thousands of kilometers with simple and efficient simulation models [Cordonnier et al., 2016; Schott et al., 2023]. However, the drawback of using a simple model becomes apparent when fluvial erosion is used at intermediate scales (*e.g.*, a few valleys of up

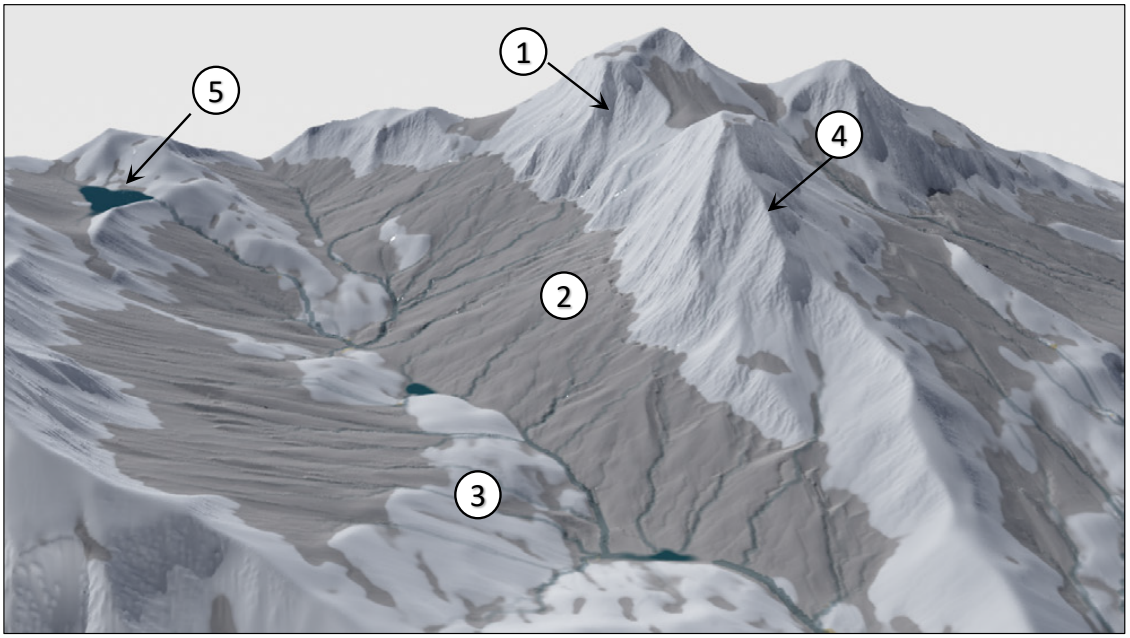


Figure 3.1: *A landscape carved by simulated erosion and deposition. (1) Deep scars of debris flow erosion, (2) deposit fan, (3) low-slope fluvially shaped hill, (4) sharp ridges carved by debris flow erosion, (5) lake found by our depression routing algorithm.*

to 10 km), where the generated terrain is smooth and uniform [Perron et al., 2009], and does not exhibit high-frequency landforms. This limitation was particularly visible in a recent work on glacial erosion [Cordonnier et al., 2023], where fluvial processes failed to convincingly erode steep cliffs, prompting a first empirical approximation for *debris flow* [Iverson, 1997] – a mixture of mud and rocks that flows down slopes with an extreme erosive capacity.

We observe typical erosion patterns in high valleys and ridges that emerge from this process. In these *low-discharge* regions where the action of water alone is not sufficient to carve significant landforms [Stock and Dietrich, 2006], debris flows are occasionally triggered by exceptional climatic events such as downpours. While simpler models for steep landscape evolution exist in geomorphology, they are often adapted to 1D convex valleys [McGuire et al., 2022]. Our goal is to extend these laws to the broader diversity of terrains encountered in computer graphics.

Improving upon the work on *fluvial erosion*, validated at what we call the *mountain range scale* (with a cell size Δx above $50 - 100\,m$), we now focus on a *valley scale* (Δx around $5 - 30\,m$), where we observe different erosion patterns depending on the distance to the ridges [Neely and DiBiase, 2023]. Debris flow leaves sharp erosive scars on steep mountain slopes while, farther, fluvial erosion flattens the main valleys. Fluvial erosion competes with debris flow deposition at the bottom of the steepest slopes, yielding a smooth transition between the two erosion regimes. To capture these intricate details (Figure 3.1), we propose a new unified erosion and deposition simulation algorithm that integrates fluvial processes and debris flow. We explore the interaction between these two processes and provide additional control parameters such as vegetation cover.

We observe that deposition leaves several *depressions* – or local minima in the topography – that severely reduce the ability of our algorithm to transport water and sediments. While several precise yet slow solutions to this problem exist on the CPU, we propose the first approximate algorithm on the GPU to compute a path out of the depressions. Eventually, we present how we derive our model for debris flow from different geological observations and show in particular that debris flow erosion generalizes the slope limiting effect of *thermal erosion* (a term mainly found in graphics, defined by Musgrave et al. [1989] as "a catch-all term for any process that knocks material loose, which material then falls down to pile up at the bottom of an incline"). Crucially, while thermal erosion tends to leave uniform slopes, our debris flow model enhances these areas with intricate,

physically consistent erosion scars.

In summary, the main contributions of this chapter are 1) A novel model for debris flow erosion and deposition derived from geomorphological observations; 2) A unified GPU erosion and deposition simulation that encompasses both fluvial and debris flow processes; and 3) A GPU algorithm to approximate flow paths in topographies with depressions.

3.2 Overview

Our focus is on debris flow, which cannot be modeled independently from other processes responsible for mountain formation. Therefore, we briefly review the geological origins and equations of all processes we consider.

Mountain formation from geomorphology Mountains emerge from the mutual influence of various geological processes (Figure 3.2). The collision of tectonic plates induces the compression of the Earth's crust accommodated by multi-scale folding and faulting [Willett et al., 1993]. The ensuing vertical growth rate is called *uplift* and denoted $\mathcal{U}$. It has been used in computer graphics to control the overall shape of the mountain [Cordonnier et al., 2017a; Schott et al., 2023].

Counteracting the effects of uplift, rainwater flows over the terrain, accumulates, erodes it, and forms valleys. This effect is modeled with *fluvial erosion* $\mathcal{F}$ that explains at large scale the distribution of mountains and valleys [Whipple and Tucker, 1999]. We extend the computer graphics implementations of the stream power fluvial erosion [Cor-

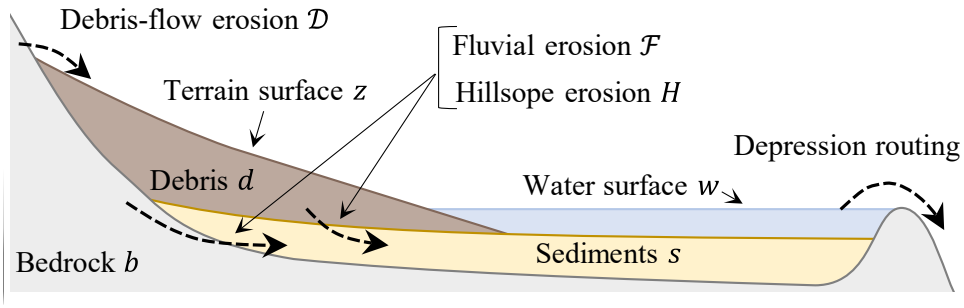


Figure 3.2: *We model interleaved erosion processes, which affect different terrain layers: debris flow erosion turns bedrock b into debris d, while hillslope and fluvial erosion turn bedrock and debris into sediments s. Depression routing allows us to compute the path of water and transported material out of local minima and deduce the water surface w.*

donnier et al., 2016; Schott et al., 2023] with sediment deposition, borrowing a law from geomorphology [Yuan et al., 2019]:

$$\mathcal{F} = -k_f \, Q_w^m \|\nabla z\|^n + k_d \, \frac{Q_s}{Q_w}, \tag{3.1}$$

where k_f and k_d are erosion and deposition constants, $m = 0.4$ and $n = 1$ are power law exponents, ∇z is the terrain surface gradient (with respect to horizontal coordinates, which means that $\|\nabla z\|$ is the terrain slope), Q_w (units $\mathrm{m}^3\mathrm{y}^{-1}$) is the water volumetric flow rate or discharge, and Q_s is the sediment volumetric flow rate. In particular, Q_w and Q_s are computed from the upstream domain A, which is defined, at a point $\mathbf{x}$, as the set of points $\mathbf{y}$ *upstream* from $\mathbf{x}$, *i.e.*, there exists a path of strictly increasing elevations between $\mathbf{x}$ and $\mathbf{y}$. Note that a rigorous definition of A should be relative to a channel width instead of a single point, but this single point approximation is valid under the assumption that the channel width is lower than the cell size [Yuan et al., 2019]. Then, the discharge is the upstream sum of precipitation p:

$$Q_w = \iint_A p \, \mathrm{d}x^2. \tag{3.2}$$

Assuming $p = 1$, Eq. 3.2 gives the drainage area, which, with the leftmost part of Eq. 3.1, yields the common expression for the stream power erosion. The sediment volumetric flow rate is given by the total upstream balance of fluvial erosion:

$$Q_s = -\iint_A \mathcal{F} \, \mathrm{d}x^2. \tag{3.3}$$

The upstream balance of fluvial erosion is always negative (the deposition amount is limited by upstream erosion). Therefore, Q_s is always positive.

Fluvial erosion cannot explain the erosion of low-drainage regions close to the ridges [Lague and Davy, 2003]. In these regions, material creep flow tends to smoothen the ridges, a phenomenon modeled with *hillslope erosion* $\mathcal{H}$ [Braun and Sambridge, 1997]:

$$\mathcal{H} = k_h \, \Delta z, \tag{3.4}$$

where k_h is a hillslope erosion coefficient, and Δz is the Laplacian of the terrain surface, which we discretize explicitly in time with a standard 4-neighbor Laplacian kernel.

In steeper regions, slopes are subject to debris flow, which yields a specific and intense erosion [Stock and Dietrich, 2006]. We will propose a new model for debris flow erosion

and deposition $\mathcal{D}$ in Section 3.3, and take advantage of the similarities between this model and fluvial erosion (Eq. 3.1) to model them through a unified algorithm in Section 3.4. In particular, our algorithm computes the propagation of water, sediment, and debris along flow paths, often interrupted by *depressions* or local minima in the topography [Scott and Dodgson, 2021]. This interruption is not realistic, as, in nature, the water accumulates, forms a pond or a lake, and eventually outflows. Therefore, in Section 3.5, we propose an approximate GPU algorithm to compute the path of water and sediments out of depressions.

Multi-layered deposition We handle sediment and debris deposition thanks to a layered terrain structure [Benes and Forsbach, 2001]. Our heightmap embeds the bedrock altitude b. The sediment and debris thicknesses s and d, and their aggregation gives the surface elevation $z = b + s + d$. Erosion acts directly on the terrain surface (with the notation $\square^{-/+} = \min / \max(0, \square)$):

$$\frac{\partial z}{\partial t} = \mathcal{F}^- + \mathcal{D}^- + \mathcal{H}^-, \tag{3.5}$$

and then any decrease in elevation δz is propagated, first to d and s in proportion (*e.g.*, $\delta d = d/(d+s)\delta z$), then to the bedrock if both d and s vanish. Since sediment and debris deposition interleave in time, we treat the two as a single well-mixed cover rather than tracking their stratigraphy; eroding a slab of this mixture removes each constituent according to its volume fraction, which conserves mass and leaves the composition of the cover unchanged. In contrast, deposition and uplift directly act on the corresponding layers:

$$\frac{\partial b}{\partial t} = \mathcal{U} \; ; \; \frac{\partial s}{\partial t} = \mathcal{F}^+ + \mathcal{H}^+ \text{ and } \frac{\partial d}{\partial t} = \mathcal{D}^+. \tag{3.6}$$

Depression routing also allows us to compute a water surface altitude w used to render and compute deposition inside lakes.

3.3 A New Debris-Flow Erosion Model

We propose a new expression for the erosion of steep slopes by debris flow. In nature, debris flow erosion results in constant slope profiles in low-drainage regions [Stock and Dietrich, 2006], similar to thermal erosion [Musgrave et al., 1989], but it also leaves small

erosion channels indicating a relationship between erosion and the local accumulation of flowing debris (Figure 3.3, left). Downstream, debris flows lose energy and stabilize, leaving typical cones of deposited sediments (Figure 3.1).



Figure 3.3: *Left: landscape carved by debris flow in the Pyrenees (capture from Google Earth of the Pyrenees. Data: Airbus.) Right: close-up view of a debris-flow channel in the Himalayas (©Dan Hobley, Creative Commons).*

We propose a novel expression of debris flow erosion that accounts for these three different effects:

$$\mathcal{D} = -E_{th} - E_{df} + D_{df}, \tag{3.7}$$

where E_{th} corresponds to the amount of loosening rocks that effectively trigger a debris flow, E_{df} the terrain erosion from the abrasion by debris flows, and D_{df} the downstream deposition.

Debris flow starts from a slope failure, described by a Coulomb criterion [Iverson, 1997]. Therefore, we model the triggering of debris flow with an equation similar to thermal erosion:

$$E_{th} = k_{th}(\|\nabla z\| - \tan\theta_c)^+, \tag{3.8}$$

where k_{th} is an erosion coefficient and $\theta_c \approx 30°$ is a critical *talus angle*.

Unsealed rocks and soaked sediments merge to form a fluid mixture that flows downslope with a strong abrasion capacity (Figure 3.3, right). Following previous work in geomorphology [McGuire et al., 2022], we express this abrasion as a power law:

$$E_{df} = k'_{df} t_{df} \|\nabla z\|^{\alpha'} h^{\beta'} \Theta, \tag{3.9}$$

where k'_{df}, α' and β' are erosion constants, t_{df}, is the time during which a single debris flow event actively flows at a given point, $\|\nabla z\|$ is the terrain surface gradient (the slope),

and h is the debris flow thickness. Friction at the bed interface slows the debris flows until they stabilize in gentle slopes. We model this effect and the accompanying reduction in erosive power with the threshold factor:

$$\Theta = \left(1 - \frac{\tau_y}{\tau}\right)^+, \tag{3.10}$$

where τ_y is the (constant) yield shear stress and the shear stress at the base of the flow is given by $\tau = \rho g\, h \|\nabla z\|$ with ρ and g the debris flow density and the gravitational acceleration.

An accurate computation of the debris flow thickness and duration would require a simulation of the flow dynamics and, therefore, strongly reduce the time domain. We found that the proposition of McGuire et al. [2022] to derive debris flow properties from the drainage area A leads to strong discontinuities in erosion, for instance, when a smooth ridge overlays a steep slope. Instead, we integrate the upstream mobilized material in a debris flow volumetric flow rate Q_{df}:

$$Q_{df} = -\iint_A \mathcal{D}\mathrm{d}x^2, \tag{3.11}$$

where A is the *upstream domain*, and express the debris flow thickness h and duration t_{df} as a function of Q_{df} and the surface slope $\|\nabla z\|$. As McGuire et al. [2022], we express the debris flow duration as $t_{df} = M_{df}/Q_{df}$, where the debris flow volume M_{df} is given by a power law of Q_{df}, yielding a power-law relationship between t_{df} and Q_{df}. We also follow McGuire et al. [2022] and assume a power law between Q_{df} and the debris-flow width w_{df}, and introduce a debris flow velocity v_{df} proportional to $h^2\,\|\nabla z\|$ [Rickenmann, 1999]. Noting that Q_{df} is also the flux of debris flowing through the channel cross section $Q_{df} = h\, w_{df} v_{df}$, we deduce a power law between h and the product of Q_{df} and $\|\nabla z\|$. Substituting these power laws in Eqs. 3.9 and 3.11, we obtain:

$$E_{df} = k_{df} \|\nabla z\|^{\alpha} Q_{df}^{\beta} \Theta, \tag{3.12}$$

and

$$\Theta = \left(1 - k_\tau \|\nabla z\|^{-\alpha_\tau} Q_{df}^{-\beta_\tau}\right)^+. \tag{3.13}$$

Here, k_{df}, k_τ, α, β, α_τ and β_τ are user-controllable erosion coefficients. Note that we obtained these expressions by assuming a power law relationship between the flow width

and the volumetric flow rate. This choice implies that the modeled flow is narrower than the cell size. A finer spatial and temporal discretization (*e.g.*, to accurately simulate individual events) would require a different formulation.

Eventually, the debris flow settles downslope when the threshold factor Θ (Eq. 3.13) approaches zero, quickly depositing the available rocks and sediments on the terrain:

$$D_{df} = \begin{cases} k_{dep,df}\, Q_{df}, & \text{if } \Theta = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

The condition of $\Theta = 0$ prevents our model from adding excessive piles of debris; a large deposition increases the slope, eventually increasing Θ. In practice, we do not set a value for $k_{dep,df}$: at the considered time scale, all available debris is deposited.

Note that our new formulation for debris flow erosion is structurally similar to the Stream Power Law with deposition (Eq. 3.1). In the next section, we propose a unified algorithm to solve both equations.

3.4 Erosion and Deposition Algorithm

We propose a new algorithm for solving erosion/deposition equations in the form:

$$\frac{\partial z}{\partial t} = -E(\|\nabla z\|, Q_w, Q_m) + D(Q_w, Q_m), \tag{3.15}$$

where Q_w is the upstream water discharge (Eq. 3.2), and Q_m the volumetric flow rate of upstream eroded material:

$$Q_m = -\iint_A \frac{\partial z}{\partial t} \mathrm{d}x^2, \tag{3.16}$$

a formulation that we find for both the fluvial sediment and the debris flow volumetric flow rates (Eqs. 3.3 and 3.11). Therefore, with the proper choice of functions E and D, Eq. 3.15 models both fluvial erosion (Eq. 3.1) and debris flow erosion (Eq. 3.7).

We discretize Eq. 3.15 in time with an explicit forward Euler scheme, which requires first computing the volumetric flow rates Q_w and Q_m and then evaluating erosion and deposition. We note that Eq. 3.16 requires the values of $\partial z/\partial t$ *upstream*. Therefore, there is no bidirectional dependency between distant cells of the terrain, and it is safe to use $\partial z/\partial t$ from the previous timestep. This choice simplifies the solution of the integral equation induced by the coupling of Eqs. 3.15 and 3.16.

Propagation of volumetric flow rate Let us take the example of the volumetric flow rate of water (water discharge Q_w); the same algorithm applies for Q_m, only changing the precipitation p by the rate of erosion $-\partial z/\partial t$ at the previous timestep. We denote by $Q(i)$ the discrete counterpart of a variable Q, evaluated at the location of a cell i. The discharge is defined as an upstream integral (Eq. 3.2), which is commonly expressed recursively as a downstream accumulation [Braun and Willett, 2013]. For a cell i, we define $R(i)$ as the set of *receiver* cells r strictly lower than i among the four neighbors of i, and conversely $G(i)$ as the set of *giver* cells g such that $i \in R(g)$. The discharge Q is then computed recursively:

$$Q_w(i) = p\,\Delta x^2 + \sum_{g \in G(i)} w^{g \to i} Q_w(g), \tag{3.17}$$

where $w^{g \to i}$ is the proportion of discharge moving from cell g to cell i, and Δx the cell size.

We reviewed different propositions from previous work for the computation of w. We observed that *multi-direction flow* [Holmgren, 1994], where w is chosen proportionally to the elevation offsets, tends to blur the erosion, while the *single-direction* choice [Cordonnier et al., 2016], which sets w to 0 for all receivers except for the one along the steepest slope, leaves strongly axis-aligned erosion patterns. Raising the offsets to an exponent [Holmgren, 1994; Schott et al., 2023] (Section 2.4) interpolates between the two and limits the blur, but only by biasing the flow toward the steepest neighbor, trading blur for directional artifacts as the exponent increases. We propose an alternative *random single-direction* strategy, where we randomly choose one receiver r per cell i and assign it a weight of one against zero for the other receivers. The probability that r is selected is proportional to the elevation difference $z(i) - z(r)$, which ensures that a trajectory following successive receivers will, on average, follow the steepest slope without axis bias, while each individual path stays sharp. An exponent could equally be applied to this probability, but since a single path does not blur, it would only bias the average direction.

We follow Schott et al. [2023] and iteratively update $Q_w(i)$ in parallel on the GPU. While this would, in theory, require n iterations to fully accumulate the discharge where n is the maximum river length, we observe that using a single iteration per timestep does not significantly change the resulting eroded terrain.

Erosion and deposition Equipped with the different transport flow rates, we now focus on slope computation, which is the last missing component to compute erosion and deposition. Eq. 3.15 is hyperbolic and therefore requires an upwind spatial discretization scheme. Previous discretization of the stream power law [Braun and Willett, 2013; Cordonnier et al., 2016; Schott et al., 2023] proposes a computation of the slope based on the elevation difference between a cell and its lowest neighbor, which tends to produce strongly axis-aligned slopes. Instead, we suggest a rotation-invariant discretization:

$$\nabla z = \frac{1}{\Delta x} \begin{pmatrix} (z(x,y) - z(x+\Delta x, y), z(x,y) - z(x-\Delta x, y))^+ \\ (z(x,y) - z(x, y+\Delta x), z(x,y) - z(x, y-\Delta x))^+ \end{pmatrix} \tag{3.18}$$

where Δx is the cell size, and the notation $(a,b)^+ = \max(0,a,b)$ selects the largest nonnegative elevation drop along each axis. Taking the largest drop along each axis and combining them through the norm also captures the slope of a diagonal descent, which the lowest neighbor alone underestimates.

When the deposition adds a significant amount of debris at a timestep (especially debris deposition), the excess sediments can locally interrupt the flow path. This situation is a numerical artifact and could not appear in nature: if the flow path is interrupted, no sediment can be deposited downward. To prevent this issue, we limit the total elevation of a cell after deposition to the average elevation of the upstream cells weighted proportionally to their sediment volume flow rate.

We modeled debris flow deposition with a large coefficient (Eq. 3.14). An interpretation of this formulation is that, during a timestep Δt, the maximum height of deposited material d_{max} is limited by the available debris volume and by the constraint that the threshold factor Θ should remain null. These two constraints give:

$$d_{max} = \min\left(\frac{\Delta t}{\Delta x^2} Q_{df}, d_\Theta\right), \tag{3.19}$$

where d_Θ is obtained implicitly by solving $\Theta = 0$:

$$\|\nabla(z + d_\Theta)\|^2 = \left(k_\tau Q_{df}^{-\beta_\tau}\right)^{-2\alpha_\tau}. \tag{3.20}$$

We use the discretization of Eq. 3.18 to compute the surface gradient with the conservative approximation that d_Θ is null downstream, which yields a second-order equation for d_Θ (if the equation does not have any solution, we discretize the slope as toward the lowest neighbor only).

3.5 Depression Routing

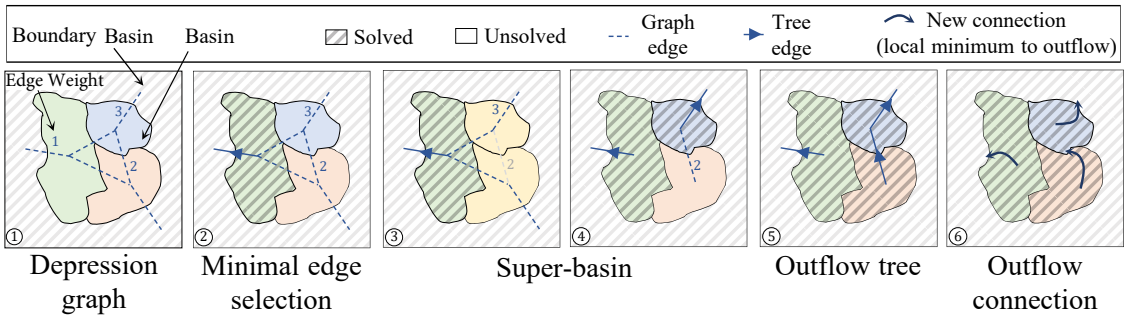


Figure 3.4: *Illustration of our depression routing algorithm. We start with a depression graph (1) that connects basins with edges weighted by the associated pass altitude (here* 1*,* 2*, and* 3 *meters, the other edges assumed to be higher). We initialized the boundary basins as solved and the internal basins as unsolved. Then, (2), we select the minimal edges (in parallel), remove them from the graph, and add them to the outflow tree. In (3), the lowest edge connects two unsolved basins. In this case, we merge both as a super-basin, which temporarily disables edge* 2*, and choose edge* 3 *as the next branch of the outflow tree (4). We mark the associated basin as solved, split the super-basin, and iterate. The final structure is a tree (5) (considering all boundary basins to be a single one), which we follow to transfer the discharge from the local minima to the selected outflows (6).*

Our algorithm for flow routing presented in Section 3.4 assumes that a path always exists from any cell to the bound of the domain. However, this assumption does not hold in the presence of *depressions* in the terrain, coming either from a noisy initial terrain, an irregular uplift, or converging deposition streams. These depressions, or local minima in the topography, are a common problem, for example, for the computation of discharge in a noisy terrain in hydrology [Wang et al., 2019]. *Depression routing* (Figure 3.4), therefore, seeks to add a connection between each local minimum and another cell, such as there exists a (cycle-free) path of minimum elevation between any position on the terrain and a bound. While several solutions exist on the CPU [Cordonnier et al., 2019; Scott and Dodgson, 2021], to the best of our knowledge, the available parallel solutions [Barnes, 2016] only focus on CPU parallel or distributed computing and are not suitable for the GPU.

Similar to the algorithm for the discharge (Section 3.4), we propose an iterative approximate GPU algorithm that *improves* the depression routing at each iteration, which therefore gives us the option for either a slow algorithm, where we iterate until conver-

gence at each timestep of the simulation, but ensures an accurate flow routing, or a fast approximate algorithm, where we update the depression routing by one iteration for each simulation timestep. The second option might yield inaccurate results, especially regarding sediment and discharge loss after sudden changes in the topography, such as the emergence of local minima, but progressively improves, provided that the terrain does not change significantly between iterations. Furthermore, this strategy does not slow the simulation as each iteration is fully parallelized.

Depression routing is challenging because when depressions connect, it is impossible to know *a priori* in which order one flows into another. Cordonnier et al. [2019] observed that the terrain could be segmented into independent *basins*, each one corresponding to the upstream area of a local minimum, and design a *basin graph* by connecting each pair of basins with an edge weighted by the altitude of corresponding *pass* (lowest cell neighboring both basins). They compute a path out of the depressions as a minimum spanning tree embedded on this graph and rooted at the exterior bounds of the terrain. We use a similar structure on our GPU algorithm, which consists of two steps: 1) the identification of the basins and 2) the estimation of the per-depression outflow.

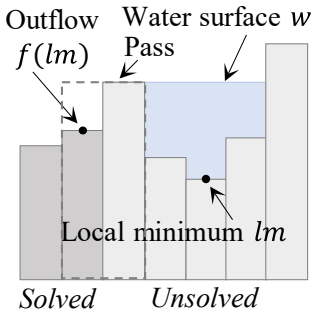
3.5.1 Basin Identification

We segment the terrain as a set of basins by storing the position of the corresponding local minimum in each cell, which is the one that would be found by flowing along the steepest slope. We use an iterative algorithm [Kotyra, 2023] to propagate the cell position from bottom to top. For a cell i, we store the *receiver* $r(i)$ that was randomly selected among the lower neighbors in Section 3.4 to distribute the discharge, and as $lm(i)$ the local minimum corresponding to i. We compute lm recursively, starting with $lm_0(i) = 0$ if i is a bound of the terrain and $lm_0(i) = i$ otherwise, and iteratively refining it with $lm_{k+1}(i) = lm_k(r(i))$. This algorithm converges in n iterations when n is the maximum distance between a cell and the corresponding local minimum. In practice, we do not require the algorithm to converge as we only need to compute lm up to potential outflows of the depressions. In practice, we do a single iteration of refinement for lm per simulation timestep because the gradual variation of the terrain does not cause a significant change in the basin definition over time.

This assumption that the terrain slowly changes can be challenged in specific conditions, for instance, in the presence of a noisy uplift, which tends to generate many small

depressions. We significantly reduce their impact by running the basin identification on a Gaussian-smoothed version of the terrain, computed at full resolution with a 9×9 kernel and $\sigma = 8$ cells. Even after smoothing, small depressions occasionally appear from fluctuations in the simulation. They have little impact on erosion but induce short-lived basins that are hard to maintain through a single-step basin identification. If a new local minimum appears at cell i, we heuristically determine if it corresponds to a new depression if the distance between i and $lm(i)$ exceeds a user-given threshold. In which case, we set $lm(i) = i$; otherwise, we do not change $lm(i)$. This heuristic, together with the smoothing, does not have a visible impact on erosion but significantly stabilizes basin identification.

3.5.2 Computation of Depression Outflow



When basins are identified, our goal is now to find an *outflow* for each local minimum l, as the lowest cell $f(l)$ adjacent to the basin of l (or, equivalently, $lm(f(l)) \neq l$). However, this condition is insufficient, as the path out of several adjacent depressions could form a cycle.

Following Cordonnier et al. [2019], we formalize the depression routing problem with a depression graph, where each basin is a vertex, and each pair of adjacent basins form an edge (Figure 3.4, (1)). We neglect the river depth (assuming that the river surface is the same as the terrain surface), implying that each depression has a single outflow. Therefore, the set of paths out of depressions forms an algorithmic tree that covers the depression graph. We call this tree the *outflow tree* (Figure 3.4, (5)). In particular, the outflow tree minimizes the potential energy needed for the water to overflow from each basin. If we weigh each edge by such energy, the outflow tree is then the minimum spanning tree of the depression graph.

For any pair of basins, the potential energy needed for the water to overflow from one basin into the other is given by the lowest energy path that connects the local minimum of both basins, where the energy is proportional to the maximum altitude along the path. In practice, we do not need the multiplicative factor and directly assign the altitude as the path energy and, therefore, as the edge weight. This path necessarily traverses a pair of terrain cells such that each of the two cells belongs to a different basin. We call this pair

of cells a *pass*. The pass between basins $B1$ and $B2$ is easily computed in parallel. The first cell of the pass is the lowest cell of $B1$ with a lower neighbor in $B2$, and the second cell is its direct neighbor in $B2$. Note that changing the order of $B1$ and $B2$ might give a different pass, in which case we choose the lowest of both. The elevation of the highest cell of the pass gives the corresponding edge weight.

From Boruvka's algorithm [Vineet et al., 2009], we know that any edge with minimum weight out of a vertex is a candidate for the minimum spanning tree, which motivates our iterative algorithm. We initialize all basins as *unsolved* except basins connected to the bound. Indeed, any local minima on the terrain boundary is a valid global outflow. Note that this choice can be altered by the user as long as they define at least one global outflow.

Then, at each iteration, in parallel, we compute the altitude of the pass between each pair of (solved-unsolved) basins. If for an unsolved basin U with local minimum l, the lowest pass is adjacent to a solved one S, we set as $f(l)$ the cell of the pass that is in S and mark U as *solved*. Then, we iterate until all lakes are solved (Figure 3.4, (2)). Note that a special case occurs if no minimum pass point to solved depressions *e.g.*, if two neighboring unsolved depressions share the same pass). In this case, we merge all connected basins as a *super-basin* (Figure 3.4, (3)) and find the outflow among the passes out of the super-basin (Figure 3.4, (4)). In the worst case, the complexity of this algorithm is proportional to the number of depressions. In contrast, in the best situation, all depressions can be routed in a single parallel pass. In practice, similarly to the identification of basins, we run one iteration at each simulation step. When the MST is fully computed, we keep the outflows for the discharge update (Figure 3.4, (6), Section 3.5.3) but reset all depressions to *unsolved* so that the connections are progressively updated in the subsequent iterations. We validate in Section 3.6.4 that after an initial number of iterations, this strategy does not result in a significant loss of discharge trapped in local minima.

Several parts require specific attention because of our iterative process. First, we allowed in Section 3.5.1 local minima l to point to another one $lm(l) \neq l$. In that case, we still need to define a path out of l and, therefore, set the outflow of l as $f(l) = f(lm(l))$. Furthermore, it is possible that, during the simulation, slopes change such that an outflow becomes part of the corresponding basin, eventually trapping the discharge and sediment in an infinite feedback loop. In that case, we update the outflow by parsing the cells upstream of $f(l)$ until we find one in the neighboring basins. We observed the new

outflow is always in the direct neighborhood of $f(l)$.

3.5.3 Discharge and Lakes Update

We update the discharge computation to take advantage of the depression routing by transferring all the discharge present in local minima to the corresponding outflow. This also defines a *water surface* w given by the altitude of the outflow:

$$w(i) = \max\left(z(i), z(f(lm(i)))\right). \tag{3.21}$$

We use the water surface for rendering but also to deposit sediment and debris below water level, proportionally to the amount of available sediment:

$$s(i) = s(i) + (w(i) - z(i)) \min(1, S/W), \tag{3.22}$$

where W is the total water volume above $lm(i)$ and $S = \Delta t\, Q_s$ the total sediment volume. We handle debris similarly and transfer any excess sediment to the outflow. Note that all these operations are easily performed in parallel.

3.6 Results and Validation

We implemented our method with PyTorch and PyTorch Scatter to optimize reduction operations in the depression routing algorithm. All results were generated on a desktop computer equipped with an Intel Xeon Gold CPU clocked at 2.10GHz, 128GB RAM, and an Nvidia RTX A6000 with 48GB of memory. We produced the 1024×1024 terrain for Figure 3.1, which required 1600 iterations in 32s (an average of 0.02s per iteration). We use Terragen (Planetside Software [2024]) for rendering, with a uniform color for each bedrock (light gray), sediment (yellow), debris (brownish dark gray), and water (dark blue) layer and without additional noise on the elevation or textures. While this requires ad hoc choices of realism of the depicted natural scenes, it allows us to control the output of our system precisely.

Here, we show some landscapes produced by our approach, validating that we observe the desired landforms. Then, we advocate for debris flow erosion by comparing it with thermal and fluvial erosion, and we validate our flow-routing approach. Unless stated otherwise, our results are terrains at 1024×1024 with $\Delta x = 8$ meters resolution eroded with parameters shown in Table 3.1. Our simulation is stable and without noticeable artifact with values of Δt ranging from 10 to 300 years. The visual variations in the

results come from different initial topography and simulation lengths. The simulation code is available at: `https://gitlab.inria.fr/landscapes/debrisflow`.

Table 3.1: *Parameters*

Name	Value	Unit
Uplift rate: $\mathcal{U}$	0.0012	$\mathrm{m y}^{-1}$
Fluvial erosion: k_f	0.0007	$\mathrm{m}^{1-2m}\mathrm{y}^{-1}$
Fluvial deposition: k_d	0.01	
Fluvial exp.: m, n	0.4, 1	
Precipitation: p	1	$\mathrm{m y}^{-1}$
Hillslope: k_h	0.04	$\mathrm{m}^2\mathrm{y}^{-1}$
Debris-flow erosion: k_{df}	0.001	$\mathrm{m}^{1-\alpha-2\beta}\mathrm{y}^{\alpha-1}$
Debris-flow exp.: α, β	0.08, 1	
Debris-flow threshold: k_τ	0.3	$\mathrm{m}^{3\alpha_\tau}\mathrm{y}^{-\alpha}$
Debris-flow threshold exp.: α_τ, β_τ	0.25,1	
Thermal erosion: k_{th}	0.005	$\mathrm{m y}^{-1}$
Thermal critical angle: θ_c	37	degree

3.6.1 Debris-Flow Eroded Landscapes

We first present some eroded landscapes with our method, highlighting the main emerging landforms.

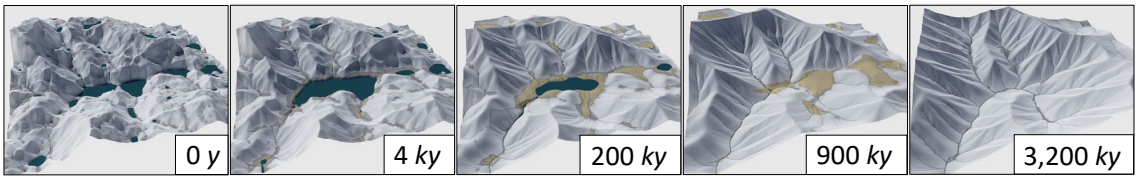
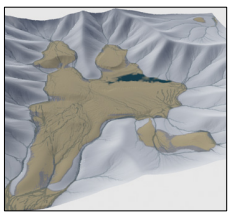


Figure 3.5: *Time lapse of simulation, from an initial noise-based terrain to a debris-flow eroded landscape. Note how the lakes fill, sediment, and how the sediments (in yellow) are progressively washed away when the landscape approaches steady-state.*

Large-scale landscape Figure 3.5 shows the progressive erosion of a terrain, which starts from a purely procedural noise-based shape and is progressively uplifted and eroded. We observe that fluvial erosion is mostly active at the bottom of valleys during the first few thousand years, while debris flow erosion actively flattens upstream



cliffs during the simulation period. Note how lakes progressively sediment and how lakes and sediments progressively disappear in uplifted regions, which is consistent with geological observations [Yuan et al., 2019]. Indeed, when the topography reaches a steady state, in which elevations no longer change because erosion balances uplift everywhere, the net effect of sedimentation and erosion must be negative to compensate for the uplift. The inset figure shows that if the uplift is disabled afterward, the valley refills with sediments.

Debris flow erosion patterns Figure 3.1 shows a close-up view of a valley carved by debris flow, with specific debris flow erosion scars (1) and talus deposit (2) next to a smoother fluvial-eroded slope (3). The steep slopes eroded by debris flow tend to form sharp ridges (4), while the remaining lakes (5) are found by our depression routing algorithm, and they are then progressively filled with sediments. Figure 3.6 demonstrates an erosion of a landscape with gentler slopes to highlight the difference between the debris-flow eroded region on the right and the overall fluvial landscape on the left. We observe that sediments deposit in lower, gentler regions while debris stabilizes at the foot of the steepest slopes.

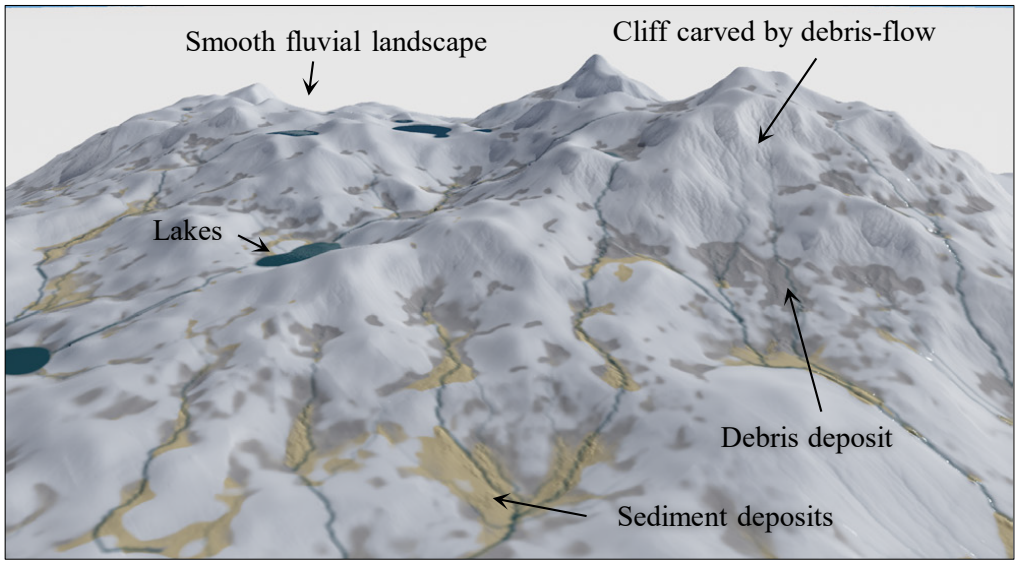


Figure 3.6: *An 8 × 8 km terrain eroded with our method to showcase the difference between a steeper debris flow eroded area on the right and a smooth region on the left eroded mostly by water.*

Effect of debris-flow on slopes Debris-flow affects steep slopes, which we demonstrate in Figure 3.7. We model terrain with a progressively increasing slope from gentle (left) to steep (right) and show the resulting erosion patterns. We notice, in particular, that the steepest slopes tend to exhibit more diverse erosion directions.

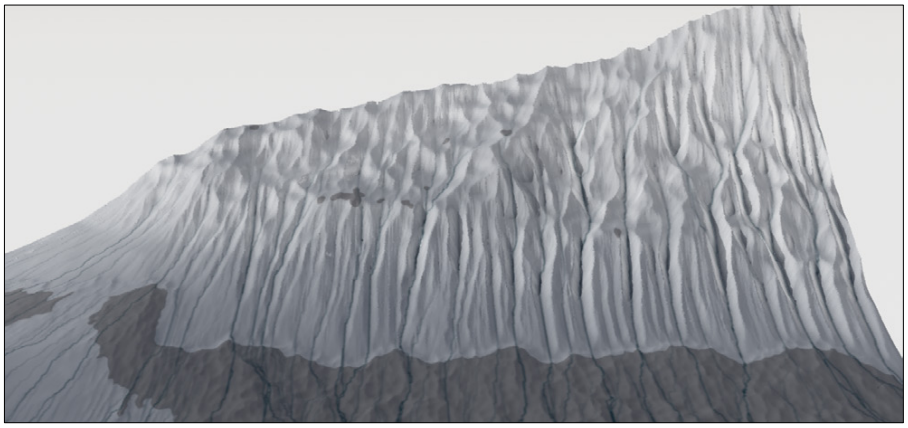


Figure 3.7: *The impact of debris erosion shown on different slopes by progressively varying the slope of terrain from left (gentle) to right (steep).*

3.6.2 Ablation and Parameters

Hillslope erosion First, we observe the importance of hillslope by changing the coefficient k_h. Figure 3.8 shows that hillslope controls the roughness of the erosion scars and the valley spacing.

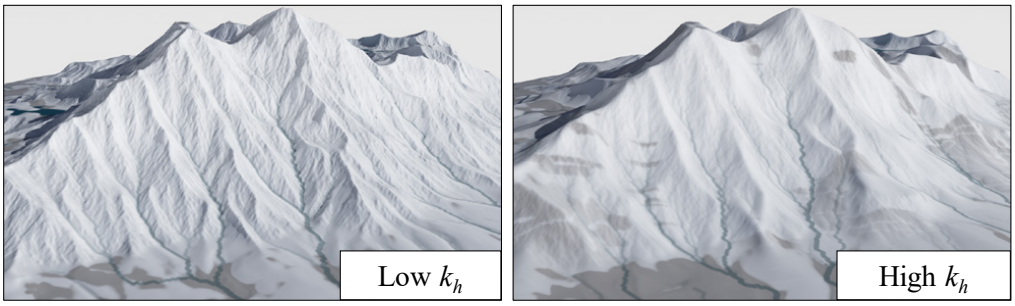


Figure 3.8: *We change the value of the coefficient k_h to highlight the effect of hillslope erosion. Higher values smooth the erosion patterns and increase valley spacing.*

Fluvial deposition Deposition plays an important role in the appearance of the terrain. We vary the fluvial deposition parameter k_d in Figure 3.9 to show the impact of fluvial

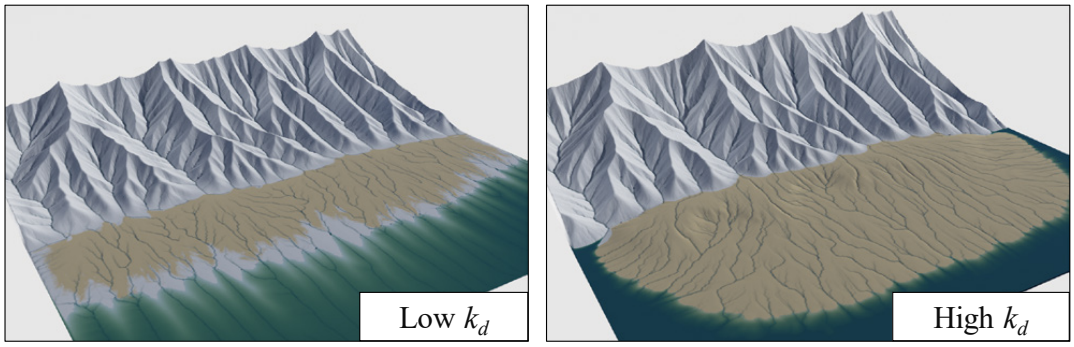


Figure 3.9: *Effect of fluvial deposition. Large values (right) yield steeper slopes in the uplifting region and fan-shaped deposits at the foot.*

deposition on a large-scale terrain ($\Delta x = 16$ meters) where only a further half of the landscape is subject to uplift. We increase the sea level at rendering to highlight the deposition patterns. A low deposition constant has only a marginal effect. A large value, however, leads to the accumulation of significant deposits, shaping deposition fans at the outlet of the main mountain rivers and deploying deltas into the sea. Deposition also impacts the upper part of the mountain, where it opposes erosion and leads to steeper slopes.

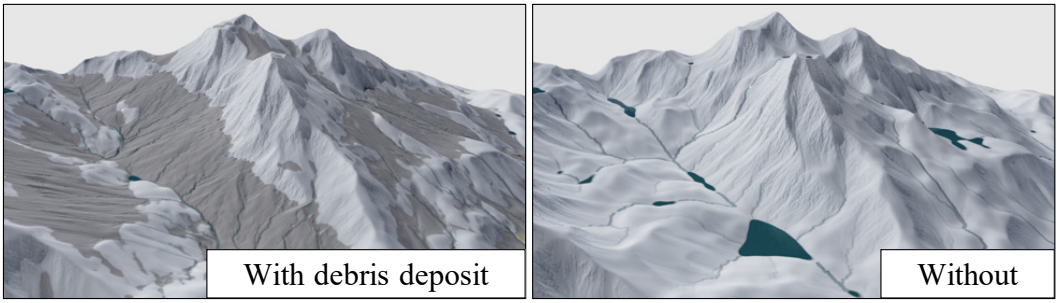


Figure 3.10: *We highlight the importance of debris deposition by comparing a replication of Figure 3.1, with (left) and without (right) debris deposition.*

Debris deposition While fluvial erosion yields deposits with gentle slopes, debris flow only occurs in steeper regions. Therefore, debris forms deposition patterns with strong angles and dramatically impacts the landscape. We can observe this behavior in Figure 3.10, where we compared a replication of Figure 3.1 with the same setup without debris deposition. The higher regions are similar, but the bottom of the valley fills with

debris, resulting in V-shaped patterns of constant slopes. We further show in Figure 3.11 how the threshold parameter k_τ controls the slope of the deposits, a low value giving gentle slopes, while a high value tends to produce steeper deposits.

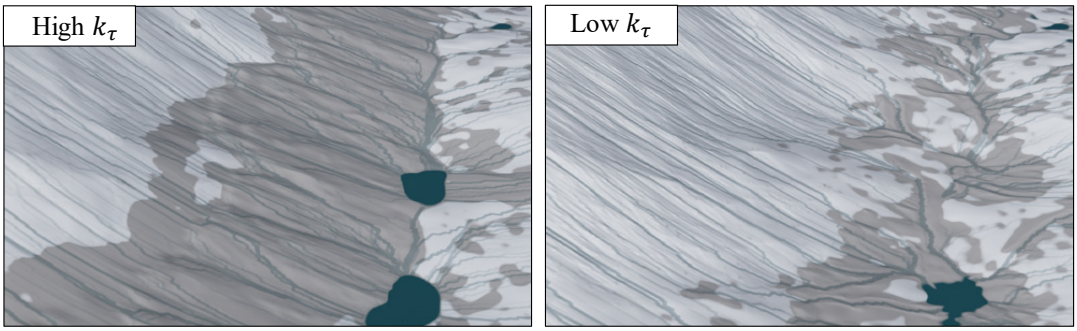


Figure 3.11: *The parameter k_τ controls the slope of the debris deposit, from steep with high k_τ to gentle with low k_τ.*

Vegetation We suggest using "vegetation" as a proxy control parameter, which mimics the smoothing effect of vegetation on the terrain. The vegetation parameter causes the displacement of some rocks and sediments and, in effect, increases hillslope erosion. However, it also shields the terrain from other erosion processes. We demonstrate this effect in Figure 3.12, where we introduce an altitude-controlled vegetation layer and show notably different erosion patterns below and above the vegetation limit.

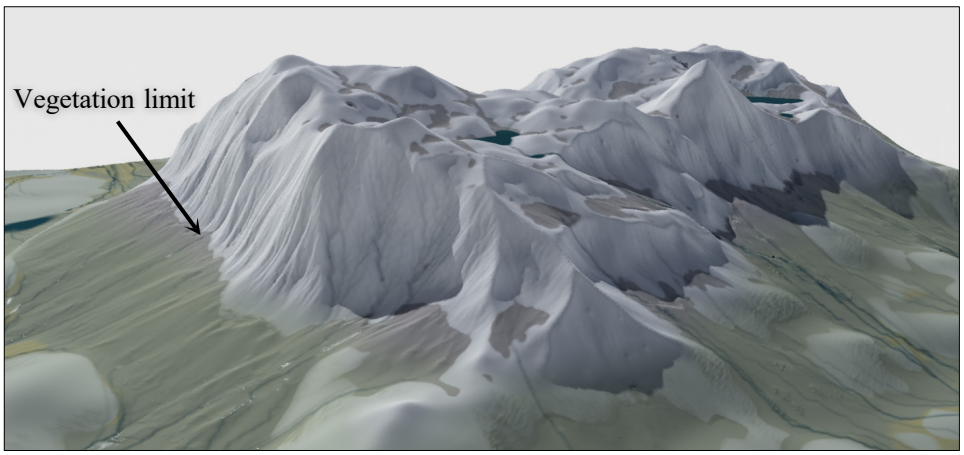


Figure 3.12: *We use an altitude-dependent vegetation layer (green overlay) to show how to control erosion roughness.*

3.6.3 Comparisons

Comparison with real landscapes. We qualitatively compare our results with a real landscape in the French Alps captured on Google Earth (Figure 3.13, left). Our simulation (Figure 3.13, right) is able to capture similar landforms: deep erosion scars left by debris flow, debris deposits, and fluvial erosion patterns below. The real slopes exhibit finer and denser scars than our simulation, a consequence of our 8 m resolution, of the smoothing induced by hillslope erosion (Figure 3.8), and of uniform rock erodibility; finer grids or spatially varying erosion coefficients are natural extensions to capture this detail.

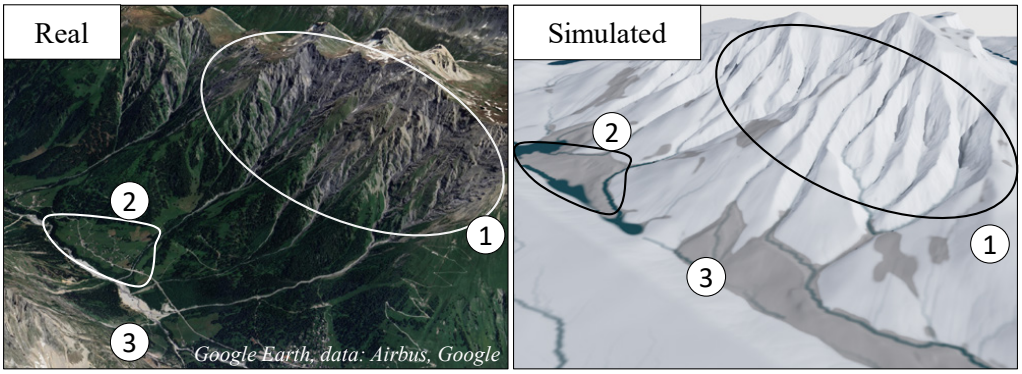


Figure 3.13: *Comparison between a real landscape in the French Alps (left, capture from Google Earth, data from Google and Airbus) and our method (right). We observe similar (1) deep scars left by debris flow, (2) debris deposits, and (3) fluvial erosion patterns.*

In Figure 3.14, we compare the slope-drainage graph of a terrain generated with our model (orange dots) with one from real data from the San Gabriel Mountains, California, USA [McGuire et al., 2022] (blue line). Each orange dot corresponds to the mean slope of a bin of terrain cells with similar drainage (computed to total 100 logarithmically spaced bins). The blue line fits real data [McGuire et al., 2022], showing a close match between our slope profiles and real data and the slope-reducing impact of debris flow at low drainage. To highlight this point, we add a curve of the theoretical profile (green) of a valley eroded purely with fluvial erosion, which shows increasingly steeper slopes when the drainage area reduces. To compute the fluvial profile, we observe that changing the equation of the blue curve from $S = k_1/(1 + k_2 A)$ (with S the slope) to $S = k_1/(k_2 A)$ yields a steady-state solution of fluvial erosion.

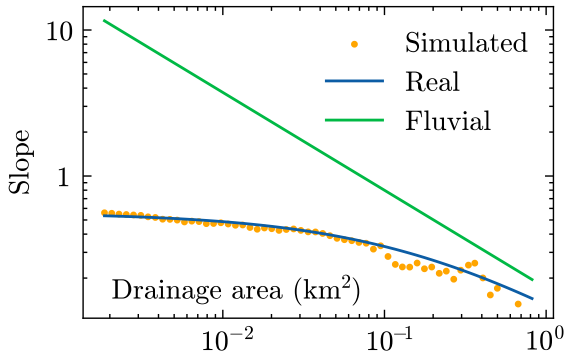


Figure 3.14: *Slopes calculated by our simulation (orange dots) tend to constant values with lower discharge areas, a feature attributed to debris flow. We compare simulated channel profiles with real terrain profiles from San Gabriel Mountain, California, USA [McGuire et al., 2022] (blue) and a theoretical fluvial profile (green).*

Comparison with thermal erosion We validate the results of our method by comparing them with thermal erosion. Figure 3.15 shows side by side comparison of a uniform slope with a small amount of added Perlin noise eroded with thermal (left) and debris flow erosion (right). Our method produces deep erosion scars, while thermal erosion keeps the slope uniform.

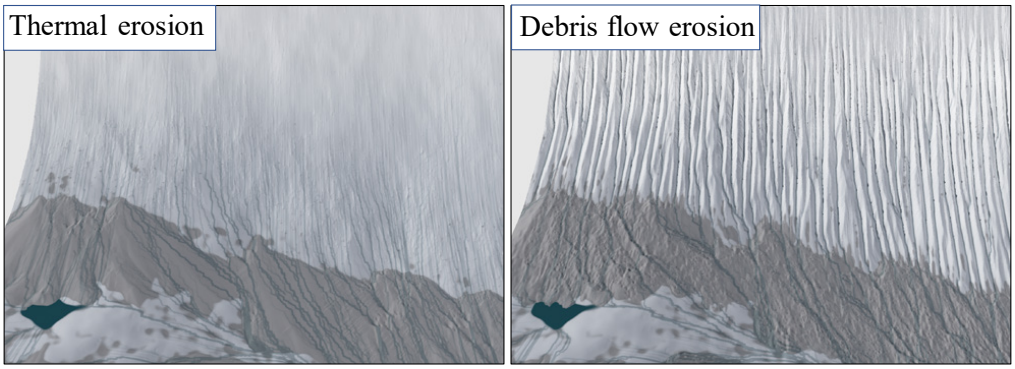


Figure 3.15: *Comparison of thermal erosion (left) with our method (right).*

Comparison with fluvial erosion Figure 3.16 demonstrates the necessity of debris flow erosion. While fluvial erosion alone is mostly smoothed by hillslope processes except for localized streams, debris flow erodes the steep slopes and provides more detailed structures.

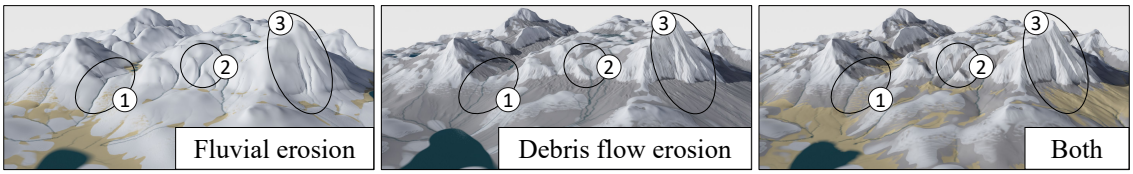


Figure 3.16: *Comparison of fluvial erosion only (left), debris flow erosion only (center), and combining both (right). Fluvial erosion only carves high drainage areas (1), while debris flow affects the steep slopes (3). The combination of both carves distinct features in high-drainage, high-slope areas (2).*

3.6.4 Validation of Depression Routing

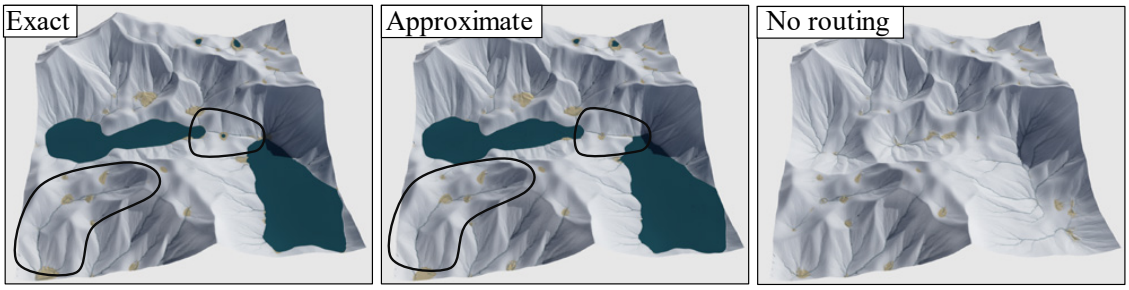


Figure 3.17: *We compare our approximate algorithm for depression routing with the exact version and no depression routing. We observe that connections established by our depression routing algorithm cause a high discharge on highlighted points and allow these points to be carved with erosion*

We first show in Figure 3.17 that depression routing is critical to correctly erode a terrain with depressions. Without depression routing, the discharge and sediments are trapped in local minima. In contrast, our algorithm, which approximates the exact solution, provides a new path for water and strongly erodes the outflow of the lakes. Figure 3.18 shows the impact of the number of iterations on the accuracy of our method by plotting the proportion of input discharge that exits the terrain. Our algorithm quickly converges to 100%, while no depression routing would trap 80% of the discharge. Finally, we compared our timings with an exact CPU implementation [Cordonnier et al., 2019]. On a 1024×1024 terrain, the CPU algorithm takes 500 ms, while a single step of our GPU implementation requires 1.8 ms. The $280\times$ speedup of our algorithm is significant for the 1-step approximate version, which, interleaved with the simulation, progressively improves its accuracy. However, if an exact solution is needed, our algorithm must iterate

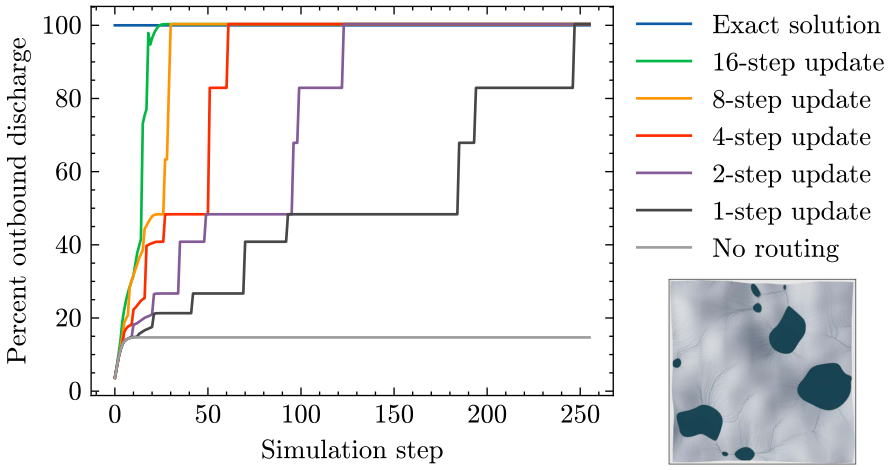


Figure 3.18: *We validate our depression routing algorithm by evaluating the amount of discharge that leaves the bound of a 1024× 1024 terrain (bottom right), and that is therefore not trapped in a local minimum. We observe our approximate algorithm converges to the exact solution and routes* 100% *of discharge while this number drops to* 20% *if we do not handle depressions.*

until convergence since basin labels advance by one cell per iteration. In practice this takes 200 to 250 iterations on our 1024×1024 terrains, which offsets the per-iteration speedup and results in a performance on par with the CPU version.

3.6.5 Limitations

Although the depression routing algorithm is fast, it comes with limitations of accuracy. Sudden changes in the terrain cause the lakes to be invalidated, requiring more iterations to converge. This behavior also occurs with the discharge propagation, preventing extreme parameters or high timesteps from being used. These limitations motivate the more robust flow and depression routing methods introduced in Chapter 4.

Furthermore, we introduce a random single flow receiver to propagate discharge and sediment flux in Section 3.4 and find it superior. A systematic comparison with steepest and multiple-receiver strategies remains to be done, for instance by measuring the grid alignment of channels and the sharpness of erosion scars on eroded terrains. Finally, while physically-based approaches ensure a global consistency that is hard for an artist to enforce by hand, more research is required to find intuitive handles for direct and expressive user control, such as painted maps of erodibility, or sketches of ridges and scars from which these parameters are inferred.

3.7 Conclusion

In this chapter, we introduced a new method for simulating erosion by debris flows. Our method is compatible with large-scale erosion models, such as fluvial erosion, and yields higher-resolution landscapes at the scale of the valley. We derived a new formulation for debris flow erosion from geomorphological observations, developed a unified algorithm to model erosion and deposition in both debris flow and fluvial processes, and proposed an iterative refining algorithm for depression routing. Our results adequately capture erosion scars left in cliffs by debris flow and foothill debris cones competing with fluvial erosion. We show that, compared to fluvial erosion, debris flow erosion enhances high-frequency patterns in eroded terrains. Finally, our implementation is fully parallelized on the GPU, which allows for fast terrain generation and erosion.

However, this efficiency comes at the cost of a loss in accuracy, with approximations that progressively correct over time, but prevent the use of large timesteps or overly strong parameters. Chapter 4 addresses these limitations through exact GPU flow and depression routing methods, and by coupling them with more stable implicit time-stepping schemes, providing a more robust basis for extending terrain simulation to larger scales.

Chapter 4

FastFlow: GPU Acceleration of Flow and Depression Routing for Landscape Simulation

4.1 Introduction

The previous chapter showed that routing flow through evolving terrain is one of the central computational bottlenecks of process-based terrain simulation. Debris-flow deposition, fluvial incision, lake formation, sediment transport, and vegetation modeling all depend on knowing where material moves and accumulates. In practice, this means repeatedly computing a flow network over a terrain that may change after every simulation step.

This computation is difficult because flow is not purely local. The discharge at a cell depends on all upstream cells, while local minima interrupt downhill paths and require

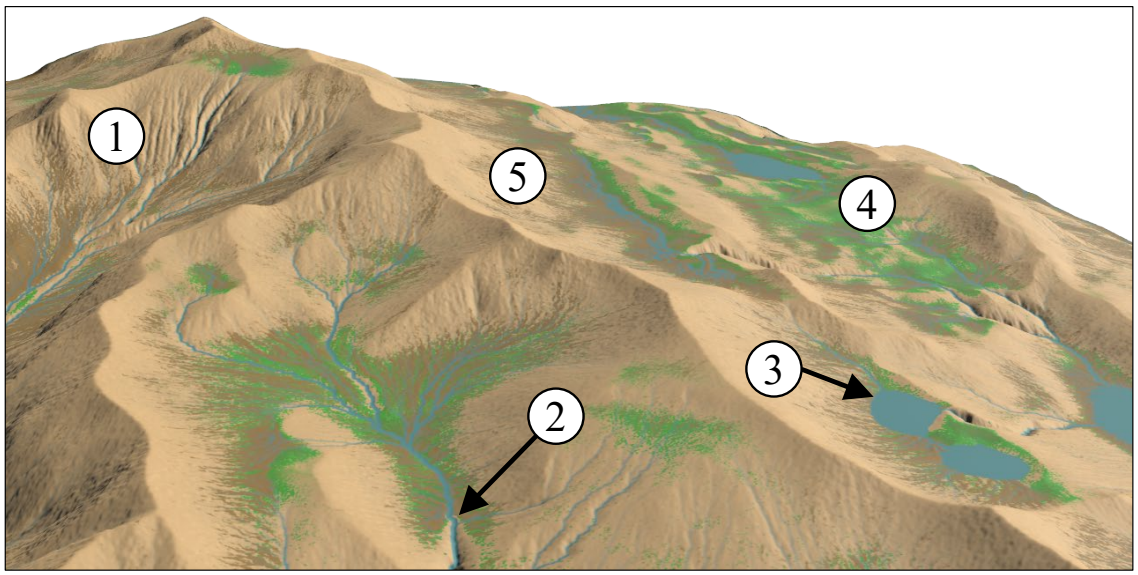


Figure 4.1: *Our GPU flow and depression routing algorithms can be applied to accelerate multiple aspects of landscape simulation, including 1) fluvial erosion, 2) rivers, 3) lakes, 4) ecosystems, and 5) sediment deposition.*

water to be routed through chains of depressions before it can reach an outlet. Approximate GPU methods can maintain interactive performance, but as seen in Chapter 3, their iterative corrections limit the use of large timesteps and strong erosion parameters. To make such simulations both fast and robust, flow and depression routing must be solved exactly while remaining efficient on massively parallel hardware.

In this chapter, we address this problem by presenting GPU algorithms for exact flow and depression routing. Although we describe the problem in specific terms of water flow [Cordonnier et al., 2016], the same routing operations apply to other transported quantities such as sediment [Yuan et al., 2019]. These algorithms therefore act as reusable computational primitives for a range of landscape simulation tasks, including rivers, lakes, erosion, deposition, and ecosystem modeling (Figure 4.1).

Instead of treating water explicitly (for instance by solving the Shallow Water Equations [Benes, 2007] or simulating particle hydrodynamics [Krištof et al., 2009]), researchers in computer graphics [Cordonnier et al., 2016], hydrology [Garbrecht and Martz, 1997] and geomorphology [Braun and Willett, 2013] have proposed methods for computing water discharge (the volumetric flow rate of water). This is based on the observation that discharge is the upstream integral of precipitation. Typically, a single simulated blanket of rain is applied and runoff is then progressively accumulated from higher elevations downward to minima along the bounding edges of the terrain. This is complicated due to the presence of local minima (or depressions) in the terrain interior, which trap the flow. This necessitates a global computation of routing through chains of depressions toward the outflow boundary.

For clarity, we define *flow routing* as the computation of the discharge – or any other material flux – over the terrain, and *depression routing* as the computation of the water flow path out of depressions.

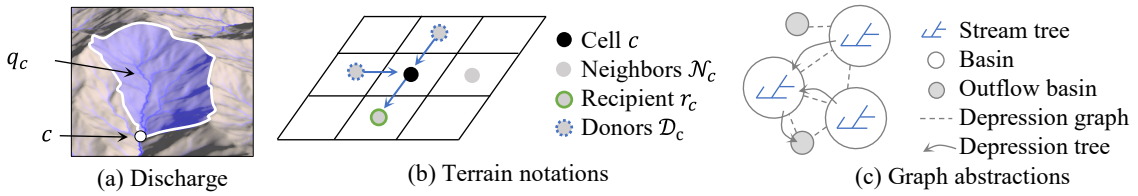
While previous work has addressed both flow and depression routing, existing methods struggle to maintain efficiency at scale. For depression routing, algorithms have evolved from priority-queue flooding [Barnes et al., 2014] to distributed computing [Barnes, 2016; Barnes et al., 2020] and optimal CPU solutions [Cordonnier et al., 2019]. However, existing solutions for the GPU [Barnes, 2019; Schott et al., 2023] are inherently inefficient for flow routing–requiring $\mathcal{O}(l)$ iterations where l is the length of the longest channel [Barnes, 2017]–and typically do not consider depression routing at all, despite local minima being prevalent in natural topography [Wang et al., 2019].

In this chapter, we present a GPU algorithm to solve *both* flow and depression routing within the same framework. In the former case, we express flow routing as an accumulation (or scan) across a tree. We use a rake-compress algorithm [Sevilgen et al., 2005], which combines pointer jumping and tree pruning, to perform this accumulation efficiently. In the latter case, we cast depression routing as an instance of searching for a minimum spanning tree and are thus able to adapt Boruvka's algorithm [Boruvka, 1926; Vineet et al., 2009]. Our algorithms for flow and depression routing over a terrain with n nodes complete, respectively, in $\mathcal{O}(\log n)$ and $\mathcal{O}(\log^2 n)$ iterations, assuming n processors. Furthermore, we provide optimized implementations of our algorithms in PyTorch and TensorFlow with custom CUDA kernels, which are well-suited for integration into existing terrain modeling pipelines.

We demonstrate their practicality through three example applications: river generation, terrain erosion, and ecosystem simulation. These use cases illustrate that with minimal modification, our algorithms are applicable to terrain modeling, geospatial analysis, and simulation models in computer graphics, geomorphology, and ecology. For instance, computing moisture proxies like the Topographic Wetness Index [Riihimäki et al., 2021] creates severe bottlenecks for large-scale ecological simulations; our fast routing directly alleviates this. Furthermore, we show how to adapt our GPU algorithm to an implicit time-stepping scheme for erosion simulation using the Stream Power Law. This overcomes the constraints of previous explicit schemes [Schott et al., 2023] that require stable river networks and many timesteps, significantly enhancing interactivity. We also present a new strategy to account for sediment deposition.

Finally, we benchmark our solution against CPU and distributed computing variants, as well as previous GPU solutions. Our GPU implementation for flow routing provides a $5\times$ speedup on a 1024×1024 resolution terrain over competing GPU implementations, while depression routing gains $34\times$ to $52\times$ speedup compared to parallel CPU approaches, depending on the variant of our algorithm. This improvement in performance enables applications in natural phenomena including river and lake modeling, terrain erosion, and sediment deposition, to cross the threshold and achieve interactive response times, especially when flow and depression routing need to be recomputed over many iterations.

To summarize, the contributions of this chapter are: 1) an efficient parallel algorithm for flow and depression routing, 2) an accompanying implementation on the GPU, and 3) demonstrations of its suitability for typical application areas.



(a) Discharge (b) Terrain notations (c) Graph abstractions

Figure 4.2: *(a) The discharge q_c corresponds to the integral of the precipitation p upstream of a cell c (Image courtesy of [Cordonnier et al., 2016]). (b) Notation for the connectivity between terrain cells. (c) Two levels of graph abstraction. The stream trees cover the terrain grid and abstract the connections from each node to its recipient. At a higher level, the depression graph connects basins, which are regions of the terrain sharing a common local minimum. This depression graph is integral to the depression routing algorithm, which removes local minima: we update the recipients guided by the depression tree rooted in an outflow basin, defined as the minimum spanning tree of the depression graph.*

4.2 Overview

Our stated aim is to efficiently calculate the direction and volume of water flow over an input terrain with due consideration of local depressions. We choose to represent the terrain as a 2D regular grid of elevations z, often referred to as a Digital Elevation Model (DEM). This is an implementation decision and adaptation of our algorithms to other heightfield structures, such as Triangular Irregular Networks [Banninger, 2007], is straightforward. Note, however, that z is a unique mapping $h(x, y) = z$ over the x, y plane and so we do not support true 3D terrains with caves and overhangs [Galin et al., 2019].

There are three layers of terrain input to our algorithms. The first is the elevation grid. We refer to each element of the grid as a *terrain cell* c, with n being the total number of cells. We prefer to use the term "cell" as opposed to the more familiar "vertex" or "node" in this context because we want to make clear that this represents a rectangular area on the terrain surrounding the sample. The neighbors of c, denoted by $\mathcal{N}_c$, are the 4 cells directly adjacent to c in the x and y directions (Figure 4.2 (b)). The second input layer is the amount of water (or any loose material) initially deposited on the terrain and subject to flow. This can be interpreted as rainfall or precipitation and is denoted as p_c. The final layer is a user-defined boolean mask $b_c \mapsto \{0, 1\}$ indicating the presence (1) or absence (0) of outflow cells. These serve as boundary conditions and act to trap water flow. Typically, outflow cells would line the edges of the grid, but there are cases, such as groundwater sinks, estuaries, and sea shores where they would be placed within the

domain.

We output two maps: one for the per-cell discharge, and another for per-cell recipients corrected by depression routing to yield uninterrupted water paths.

Given these preliminaries, we now provide an overview of the flow and depression routing algorithms.

4.2.1 Flow Routing

Rain falling on a landscape flows downstream and progressively accumulates into streams and rivers. The volume of water flowing through a river cross-section per unit of time is called the discharge. Equivalently, the discharge is the upstream integral of the precipitation map. *Flow routing* involves the computation of this discharge for each cell of the terrain grid.

The direction of the water flow is discretized on the basis of a per-cell *recipient*, r_c, which is chosen among the neighboring cells of the terrain and obeys the condition $z_c > z_{r_c}$. This allows several mechanisms for the choice of neighbor. While the different choices have their place and are compatible with our algorithm, we prefer a random selection, with a probability based on the elevation difference between the source and potential recipient [Tzathas et al., 2024].

Let us consider the cells of a terrain as nodes in a graph, and interpret the connection between a cell and its recipient as a directed edge. The strict monotonically decreasing elevation of recipients guarantees that the graph does not contain any cycles. Since each cell has a single recipient (equivalently, a single edge leaves each node), this graph represents a set of trees (an algorithmic *forest*).

We use the term *stream tree* for each connected subgraph, since it represents a set of water channels. This allows flow routing to be recast as an accumulation (or *scan*). For a given stream tree, the scan proceeds from its leaves (ridges of the terrain) to a single root (an outflow cell). We propose employing *rake and compress* [Sevilgen et al., 2005] algorithms to perform this task in parallel (Section 4.3).

We handle the user-defined outflow mask by excluding recipient edges for its cells so that they become root nodes in the stream forest. However, other cells may also lack recipients if none of their neighbors meet the $z_c > z_{r_c}$ condition, i.e., local minima and flat areas. In reality, instead of interrupting flow, these areas would fill and overflow. This necessitates a solution for depression routing.

4.2.2 Depression Routing

We are now in a position to define what a *depression* means in our context. As a preliminary, we use the term *basin* for the set of terrain cells that share the same stream tree, effectively all belonging to the same catchment area and channeling water to a single root cell. If this root cell is an outflow, then the basin is classified as an outflow basin, otherwise, it constitutes a depression (Figure 4.2 (c)).

Depression routing is the task of finding reasonable water paths out of all depression basins. Unfortunately, this problem is non-local and therefore not amenable to trivial parallelization. For instance, an obvious first thought would be to connect a depression to an adjacent basin at the point of the lowest neighboring elevation, but this may give rise to a chain of connections that form a cycle, and therefore is unable to route the water toward outflow cells.

Two families of solutions have been proposed to tackle this issue. One option is to progressively flood the terrain from the outflow cells toward the interior, keeping track of cells to be processed through a priority queue [Barnes et al., 2014]. While this enables a simple and efficient single-threaded solution, it is not suited to parallelization on the GPU, because it imposes a sequential order, with each processed cell potentially changing the subsequent state of the queue. The other option is to build a *depression graph* by representing depressions as nodes and their adjacency relationships as edges. There are two subtleties. First, all outflow basins are grouped into a single node marked as the start. Second, edges are weighted according to the lowest altitude along their shared boundary. The path out of all depressions is then given by the Minimum Spanning Tree (MST) of the depression graph, which connects all depressions to the collection of outflow basins.

In Section 4.4, we detail our GPU implementation of depression routing and propose two strategies for correcting the flow path: either creating a jump connection between a local minimum in the current depression and a cell in the next basin or preserving the continuity by inverting flow for a subset of cells in the depression.

4.3 Flow Routing

As a reminder, *flow routing* involves computing the *discharge* q_c for a cell c resulting from the accumulation of precipitation p over a stream tree. This tree is implicitly defined by the relationship between a cell c and its downstream recipient r_c. We can also travel

upstream by defining a *donor* of c as a node d, such that $r_d = c$. We write the set of up to 4 donors of c as $\mathcal{D}_c$. In practice, we store the donors in an $n \times 4$ matrix, with an accompanying list of the number of donors per cell. This matrix is derived by parsing the cells in parallel. Each cell increases the donor count of its recipient r with atomic operations and adds its position to the donors of r.

Water flows downstream from donor cells to recipients, and therefore the *flow routing* problem consists of accumulating all of the upstream precipitation:

$$q_c = p_c + \sum_{d \in \mathcal{D}(c)} q_d. \tag{4.1}$$

Note that this is a recursive definition so that the discharge at a cell is the sum of the precipitation falling on the cell and the discharge from all donor cells.

With the exception of some cells marked by the outflow mask, stream trees cover all of the terrain, which makes the design of an efficient parallel solution to Eq. 4.1 non-trivial.

Fortunately, this problem can be cast as an instance of Parallel Tree Accumulation, which aims to compute the cumulative sum of quantities in tree nodes, from leaf to root. This can be achieved with straightforward pointer jumping, but this may result in a $\mathcal{O}(n)$ span due to the required atomic operations. An alternative is to use an Euler tour [Sevilgen et al., 2005] along the tree edges combined with pointer jumping, which reduces the span to $\mathcal{O}(\log n)$, since the number of edges is less than the number of nodes. However, this method typically requires $2 \log_2 n$ iterations, as each edge is traversed twice. A still more efficient approach is rake-compress, which combines pointer jumping with leaf pruning to achieve a span of $\mathcal{O}(\log n)$. This method typically needs only $\log_2 n$ iterations, making it faster in practice than the Euler tour. Sevilgen *et al.* [Sevilgen et al., 2005] provide a starting point in the form of a general rake-compress solution to graph accumulation problems for distributed architectures. This relies on the observation that leaves constitute half the nodes in a balanced binary tree. So, if we iteratively transfer accumulated discharge from a leaf to its recipient and then prune the leaf and do this in parallel across all leaves, we only need to iterate $\log_2 n$ times where n is the number of nodes in the terrain.

In general, river trees might be neither binary nor balanced. In the worst case, where a river tree is composed entirely of single donors, the iterations degrade to $\mathcal{O}(n)$. To compensate, we use pointer jumping by re-routing the edges of cells whose donor itself has a single donor (parent). Effectively, the parent (donor) is skipped and an edge is formed

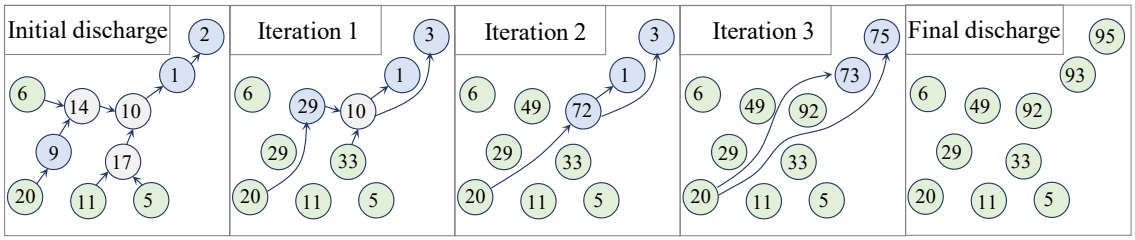


Figure 4.3: *A few iterations of a parallel rake-compress solution to flow routing. On each iteration, leaf nodes (in green) are pruned and recipients of cells with a single donor (in blue) are rerouted.*

directly from the grandparent (donor of the donor). Thereby, worst-case performance is restored to $\mathcal{O}(\log_2(n))$ parallel iterations.

As shown in Algorithm 1, with an accompanying example in Figure 4.3, on each iteration and for each terrain cell c, we parse each donor d in parallel, with up to 4 donors in our implementation. If d is a leaf, we add its discharge to c and remove it from the donor matrix. Alternatively, if d has only a single donor, we again add its discharge to c but in this case reconfigure the donor matrix so that c instead points at the donor of d. Distinguishing between these cases requires a single lookup in the donor count list, *e.g.*, , leaves have a donor count of zero. Straightforward extensions of Algorithm 1 can be used for other flow accumulation functions, including those involving maxima or weighted sums of upstream values.

Note that some care must be taken in the implementation of Algorithm 1, which exhibits potential read-after-write (RAW) hazards when updating the discharge and recipients. An immediate solution is to write the updated q and r to temporary arrays and copy them back at the end of each iteration. Unfortunately, this forces the unnecessary transfer on each iteration of a large number of cells that are unmodified. Instead, we propose a *per-cell ping pong scheme*. We first allocate sufficient space to hold the state of two full terrains, T_A and T_B. We then introduce an additional *source* buffer with one 8-bit integer per cell, initialized to 0. The sign of the integer at cell c in the *source* indicates whether data should be read from T_A and written to T_B, or vice-versa. When cells are updated, the sign is flipped. To avoid RAW hazards within the same iteration, the remaining bytes of each *source* entry store the iteration at which c was last updated: a thread will check the *source* to negate the read sign if the stored iteration is the current one. At the end of each iteration, the latest state of each cell is either in T_A or T_B, as indicated by $source$. Hence, a final merge step is required before returning the result as

a single tensor. Incorporating this optimization leads to a $14\times$ speedup on a 1024×1024 terrain.

Algorithm 1: Flow Routing: downstream accumulation

Input : Terrain cells $\mathcal{T}$ with initial discharge (precipitation) $q = p$ and their donors

Output : Updated accumulated discharge

```
for i ← 1 to log2(|T|) do
    foreach cell c ∈ T in parallel do
        foreach donor d of cell c do
            if d is a leaf then
                q_c ← q_c + q_d
                remove edge d → c
            else if d has a single donor then
                q_c ← q_c + q_d
                donor of c ← donor of d
        if all donors are removed then
            tag c as leaf
```

4.4 Depression Routing

To match the expected natural behavior, water needs to be routed out of local depressions. Our strategy is to create new donor-recipient connections and reroute others to ensure an uninterrupted, minimum-cost path from any node to one of the designated outflows. In this context, we define the minimum cost to be the smallest increase in altitude needed to move water out of a depression.

Broadly speaking, this involves connecting depression basins with outflow basins through saddle points and then re-orienting a path of edges within each depression to flow uphill and link to the corresponding outlet. The challenge lies in preventing the formation of cycles as this will lead to highly unnatural flow behavior. Once all stream trees have been corrected, we only need to run flow routing once to calculate discharge volumes.

Depression routing consists of building a depression graph, in which the depression and outflow basins represent nodes, and edges are formed by the adjacent cells between

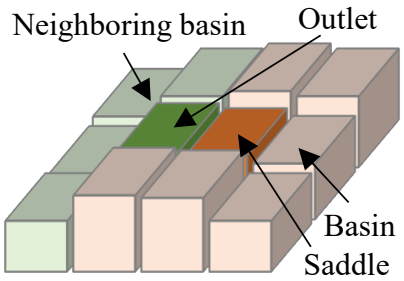
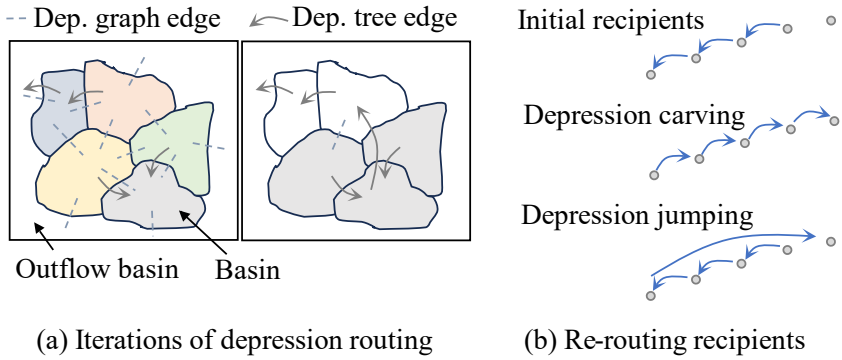


Figure 4.4: *A saddle between basins: With respect to a source basin (in light red) and its neighbor (light green), the saddle and outlet together form the lowest elevation bridge.*

different basins. It is likely that there will be multiple potential candidates between any two basin nodes formed by cells paired along their common border. We choose the one with the lowest maximum altitude and call this the *saddle* (see Figure 4.4). The saddle altitude is the weight assigned to the new edge in the depression graph. From a physical perspective, choosing the path with the lowest altitude ensures that the potential energy cost of water routed out of the basin is kept to a minimum.

Following Cordonnier *et al.* [Cordonnier et al., 2019], the minimum cost route out of depressions can now be cast as a minimum spanning tree problem. Among the several MST algorithms (*e.g.*, , Prim's, Kruskal's), Boruvka's [Boruvka, 1926] is perhaps the most amenable to parallelization [Vineet et al., 2009]. Boruvka's Algorithm constructs a minimum spanning tree by iteratively identifying and merging the minimum-weight edges of each component in parallel until only one component remains. We note that the optimal theoretical complexity for parallel MST algorithms is $\mathcal{O}(\log n)$ [Chong et al., 2001], but the specific algorithmic improvements needed to achieve this are not portable to the GPU, as they involve complex branching and scheduling operations. Therefore, we build our algorithm upon the more usual $\mathcal{O}(\log^2 n)$ variant.

Boruvka's algorithm consists of greedily forming sub-trees by treating each as a single collapsed node. It begins by treating all nodes in the initial graph as separate sub-trees. Then, each of these is grown by incorporating the incident edge of minimal weight (and its attached sub-tree). The two sub-trees are joined, their shared edge is collapsed, and the algorithm repeats until a single tree results. The parallel variant of Boruvka's algorithm [Vineet et al., 2009] applies edge selection in parallel and replaces the edge collapse with a structure that maintains the connected components of the tree under construction. This requires at most $\log_2(V)$ iterations, where V is the number of vertices



(a) Iterations of depression routing (b) Re-routing recipients

Figure 4.5: *(a) Two iterations of depression routing. The gray basin on the left does not have an outgoing edge, likely because it would have created a loop with a basin of a higher id. In the next iteration, basins are recolored based on their new local minima, and only one edge remains. (b) From the initial configuration of the recipients (top), we apply either depression carving, which reverses the path of water until it reaches the outlet, or depression jumping, which directly connects the local minimum with the outlet.*

in the graph (depressions in our case) (Figure 4.5 (a)). Each of these iterations parses the structure for connected components, adding an extra $\log_2(V)$ iterations at worst, and resulting in an overall complexity of $\mathcal{O}(\log^2(V))$.

While we could build an explicit depression graph, compute a minimum spanning tree, and parse the tree to update recipients, such a construction would be costly as it involves the detection of pairs of adjacent depressions. Furthermore, while Boruvka's algorithm outputs an undirected tree, we require directed edges to orient water toward outflow basins. Instead, we introduce a hybrid structure where terrain data is used in conjunction with the depression graph to progressively detect the path of water and update recipients. The algorithm proceeds through the stages of basin identification, saddle edge selection, and donor-recipient re-routing.

Basin identification. Our algorithm begins by segmenting out each stream tree and associating with its cells a unique basin identifier (hereafter referred to as a *basin-id*, or *bid* in the pseudo-code of Algorithm 2). To this end, we assign to each non-outflow local minima a unique identifier – leaving a common outflow identifier for all outflow basins – and use pointer jumping to copy this identifier in parallel to all upstream nodes of its stream tree (Algorithm 2).

Algorithm 2: Propagation of basin identifiers

Input : Terrain cells $\mathcal{T}$, non-outflow local minima $\mathcal{L}$
Output: Per-cell basin identifiers bid

```
for i ← 0 to |L| − 1 in parallel do
    bid of L[i] ← i
for i ← 1 to log2(|T|) do
    foreach cell c ∈ T in parallel do
        bid of c ← bid of recipient of c
        recipient of c ← recipient of recipient of c
```

Saddle edge selection. We use the per-cell basin annotation to isolate cells lying along the border with neighboring basins. To do so, we tag as a border cell, any cell in the current basin with at least one adjacent cell belonging to another basin. Ultimately, one of these border cells is chosen as a saddle edge in the depression graph. We do not need to construct these edges explicitly as Boruvka's algorithm only requires the edge with the lowest weight.

For each border cell, this weight is set as the lowest elevation that will permit an overflow from it into an adjacent basin. Specifically, this is the maximum of the border cell's altitude and the lowest altitude among its neighboring cells. Then we choose the border cell with minimum weight per basin.

Although we could compute the minimum altitude using a variant of rake-compress in $\mathcal{O}(\log n)$, we found it more efficient to instead compute it in a single step with atomic operations. We also store the *saddle*, which is the source cell in the current basin with minimum weight, and the *outlet*, which is its lowest neighboring cell across the basin border (see Figure 4.4).

Close-coupled loops can occur when two basins have saddle edges pointing at each other. We resolve this by simply deleting the edge with the higher saddle basin-id. We also apply a similar strategy to choose among saddles of identical elevation, which could lead to larger cycles. This is prevented by selecting saddles in lexicographic order (elevation concatenated with the basin-id of the outlet). This algorithm is summarized in Algorithm 3.

Algorithm 3: Finding saddles and outlets

Input : Terrain cells $\mathcal{T}$ with elevation z, set of depression basins $\mathcal{D}$
Output : Per-depression saddle and outlet

// Compute border cells and store the border altitude in z'
foreach *cell* $c \in \mathcal{T}$ *in parallel* **do**
  if $bid_c \neq bid_n$ *of any 4-neighbor* n *of* c *(*$n \in \mathcal{N}_c$*)* **then**
    tag c as a border cell
    $znb \leftarrow \min(z_n \text{ for } n \in \mathcal{N}_c \text{ such that } bid_c \neq bid_n)$
    $z'_c \leftarrow \max(z_c, znb)$

// Saddles and outlets (with atomic lexicographic argmin)
foreach *basin* $d \in \mathcal{D}$ *in parallel* **do**
  $saddle$ of $d \leftarrow$ argmin$((z'_c, bid_n)$ for each border cell c and $n \in \mathcal{N}_c$ such that $bid_c = d$ and $bid_n \neq d$)
  $outlet$ of $d \leftarrow$ argmin(z_n for each neighbor n of the $saddle$ such that $bid_n \neq d$)

// Remove cycles
foreach *basin* $d \in \mathcal{D}$ *in parallel* **do**
  $d' \leftarrow$ basin of outlet of d
  if bid *of outlet of* d' *=* bid *of saddle of* d **then**
    if bid *of outlet of* $d <$ bid *of saddle of* d **then**
      Delete the saddle and outlet of d

Re-routing recipients. Ordinarily, the next step in Boruvka's algorithm would be to collapse the newly incorporated edges. Due to our two-layer structure, with an implicit depression graph and explicit flow routing, this needs to be interpreted differently. Instead, we merge stream trees by adding and re-routing recipient links, using one of two strategies (Figure 4.5 (b)).

The first strategy, *depression jumping*, involves adding a recipient link directly from the depression's local minima to the newly established outlet cell. This is straightforward and efficient but leads to a jump discontinuity in the flow path, which, depending on the application, may be undesirable.

The second strategy, *depression carving* [Rieger, 1998], mimics the course of water channeling down from the saddle. We first compute the flow path that connects a saddle and outlet to the associated local minimum and then reverse the direction of all recipient links. In this way, we allow water to flow upstream from the local minimum to the outlet.

The pseudo-code for these two strategies is presented in Algorithm 4.

Algorithm 4: Re-routing flow paths

Input : Terrain cells $\mathcal{T}$, set of depression basins $\mathcal{D}$
Output: Updated recipients

if *Depression jumping variant* **then**
  foreach *depression* $d \in \mathcal{D}$ *in parallel* **do**
    $l \leftarrow$ local minimum of d
    recipient of $l \leftarrow$ outlet of d
else
  // Depression Carving Variant
  tag $\leftarrow False$ for all cells
  foreach *depression* $d \in \mathcal{D}$ *in parallel* **do**
    tag[saddle of d] $\leftarrow True$
  // recipient is local to next two loops
  for $i \leftarrow 1$ **to** $\log_2(|\mathcal{T}|)$ **do**
    foreach *cell* $c \in \mathcal{T}$ *in parallel* **do**
      if $tag[c]$ **then**
        tag[recipient of c] $\leftarrow True$
        recipient of $c \leftarrow$ recipient of recipient of c
  foreach *cell* $c \in \mathcal{T}$ *in parallel* **do**
    if *tag[recipient of* c*] and* c *is not a local minimum* **then**
      recipient of recipient of $c \leftarrow$ c
  foreach *depression* $d \in \mathcal{D}$ *in parallel* **do**
    recipient of saddle of $d \leftarrow$ outlet of d

A single iteration of basin identification, saddle edge selection, and re-routing of recipients ends up being insufficient. It will not necessarily connect all depressions to the set of outflow basins. However, on any given iteration this process removes at least half of the remaining depressions, meaning that a total of $\log(L)$ iterations are required, where L is the number of local minima (Algorithm 5). Each iteration, in turn, requires $\log_2(n)$ sub-iterations, where n is the number of cells, yielding an $\mathcal{O}(\log(L)\log(n))$ algorithm.

Implementation. As with flow routing, we prevent read-after-write hazards while minimizing the extent of copying. For basin identification, we store the basin-id only for

Algorithm 5: Depression Routing

Input : Terrain nodes $\mathcal{T}$ with depressions; non-outflow local minima $\mathcal{L}$
Output: Updated terrain graph with rerouted depressions

for $i \leftarrow 1$ **to** $\log_2(|\mathcal{L}|)$ **do**
    Construct basins $\mathcal{D}$ and annotate the cells (Algorithm 2)
    Find min edges, saddles and outlets (Algorithm 3)
    Reroute recipients (Algorithm 4)

the local minima and not for all cells, and use the updated recipients at the end of Algorithm 2 as a per-cell pointer to the downstream local minima, which we interrogate to obtain the basin-id. This change relaxes the need for any copy during basin identification, as read-after-write no longer prevents the correct convergence of the algorithm. Furthermore, we observe that this operation is required only for the first iteration. In subsequent iterations, we only need to update the pointers to reflect the new connections out of the local minima, yielding an overall complexity of $\mathcal{O}(\log(n) + \log^2(L))$.

Another RAW hazard can occur in the depression carving variant for re-routing flow paths. To prevent this, we use a localized ping-pong strategy (similar to the one presented in Section 4.3), which is simplified by using the tag variable as a condition for copying.

4.5 Application to Landscape Simulation

Here we show how we apply flow and depression routing to accelerate landscape simulation, with use cases in river and lake modeling, terrain erosion, and ecosystem simulation. In particular, our GPU algorithms together provide the discharge necessary for erosion simulation. Furthermore, they enable a novel GPU-based solution for implicit time-stepping of erosion using the Stream Power Law, which we combine with a simulation of sediment deposition.

River and lake modeling. To identify the surface of rivers and lakes we begin by filling depressions in the terrain. This is sometimes referred to as *pit removal* and has value as an algorithm in its own right. Our input is the terrain elevation grid, and a set of recipients identified during depression routing to provide the flow path with minimum energy cost. We call the *water surface* the elevations obtained after filling the identified depressions. This surface should obey two conditions: it should be as close as possible to

Algorithm 6: Extracting a depression-free water surface

Input : Terrain cells $\mathcal{T}$ with elevation z, small slope ϵ
Output : Elevation of the water surface w

$w \leftarrow$ copy of z
for $i \leftarrow 1$ **to** $\log_2(|\mathcal{T}|)$ **do**
  foreach *cell* $c \in \mathcal{T}$ *in parallel* **do**
    $w[c] \leftarrow \max(z[c], w[\text{recipient of } c] + 2^{i-1}\epsilon)$
    recipient of $c \leftarrow$ recipient of recipient of c

the terrain, and at the same time be monotonically non-decreasing along all stream trees, which we intuitively expect since water flows downhill. To achieve this, in Algorithm 6 we introduce a variant of the basin identification algorithm. We follow the water path of each stream tree upward from destination to sources, retaining as the water surface the maximum elevation previously reached. For completeness, we optionally allow a small slope ϵ on the water surface as this is sometimes required in hydrology applications. Note that this variant is only possible with the *depression carving* strategy, as it requires a continuous path. Next, cells with water discharge above a specified threshold can be tagged as belonging to bodies of water, as demonstrated on real terrains in Section 4.6.2.

Ecosystem simulation. The water routes derived through river and lake modeling have direct application to ecosystem simulation. Water, along with sunlight and warmth, is a key requirement for plant growth, with different species having varied adaptation to water availability. Modeling water discharge is therefore a crucial element in ecosystem simulations. To demonstrate applicability, Figure 4.6 shows the outcome of an ecosystem simulation for a biome in the Pyrenees mountains [Peytavie et al., 2024] at the 50 year mark, with water accumulation provided by our flow and depression routing. Since plants draw on soil moisture rather than on channel discharge alone, the ecosystem model spreads the moisture of each river and lake over an area that grows with discharge and decays with distance and with elevation above the water. Our routing only replaces the flow accumulation that this pipeline takes as input. For clarity, pioneer species with strong drought tolerance have been removed and what remains are species, such as Sessile Oak and European Beech, that require the greater moisture found in riparian areas. Most ecosystem simulations use a month as the timestep granularity, due in part to the overhead of calculating water flow. The acceleration provided here creates an

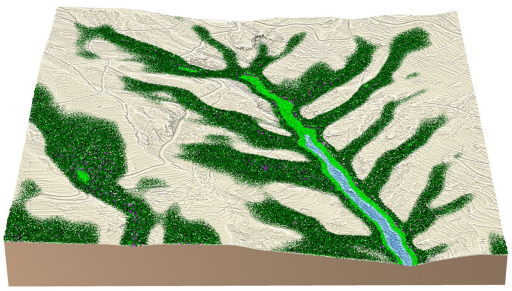


Figure 4.6: *Ecosystem simulation with water provided by flow routing. Certain species (colored in shades of green, black, and purple) favor river banks due to higher levels of available water.*

opportunity for shorter, weekly, or even daily, timesteps with non-uniform rainfall across the landscape [Pałubicki et al., 2022] and hence greater simulation accuracy.

Terrain erosion. Flow routing is a critical component of terrain erosion, as it provides the mechanism by which material is worn away and transported, ultimately over geological time shaping the mountains themselves. Furthermore, even the ordering of recipient cells plays a role in the implicit integration of the stream power law erosion equation. This is expressed as [Cordonnier et al., 2016]:

$$\frac{\partial z}{\partial t} = u - kQ^m \left|\frac{\partial z}{\partial x}\right|^n, \tag{4.2}$$

where the first term u is the tectonically induced uplift (or growth rate of the mountain), and the second term is the erosion, dependent on the discharge Q, the slope $\partial z/\partial x$, an erosion coefficient k, and exponents m and n, whose ratio m/n, typically between 0.4 and 0.6, governs the concavity of river profiles [Whipple and Tucker, 1999]. We set $n = 1$, which makes the implicit update linear (Eq. 4.4) and thus compatible with our parallel traversal, and $m = 0.4$ as in Chapter 3. The discharge provided by flow routing is scale-independent, so to compensate we scale the discharge Q by accumulating $\Delta x^2 p$, where Δx is the cell size and p is the precipitation. The uplift is usually considered time-independent and therefore applied as a pre-process. This leaves an implicit solution to the second erosive part of Eq. 4.2 as:

$$z_t[c] = z_{t-\Delta t}[c] - K(z_t[c] - z_t[\text{recipient of } c]), \tag{4.3}$$

where Δt is the timestep and $K = kQ^m \, \Delta t / \Delta x$. Eq. 4.3 can be further simplified to:

$$z_t[c] = \alpha z_t[\text{recipient of } c] + \beta, \tag{4.4}$$

where $\alpha = K/(1+K)$ and $\beta = z_{t-\Delta t}[c]/(1+K)$.

As with water surface extraction, we can compute z_t using a simple modification of basin identification (see Algorithm 7). The outcome of applying our accelerated method of fluvial erosion is illustrated in Figure 4.8. Further research is required to adapt our solution to the general case, for instance with a global Newton-Raphson's algorithm.

Algorithm 7: Implicit fluvial erosion

Input : Terrain cells $\mathcal{T}$ with elevation z, per cell coefficients α and β
Output: Eroded terrain z'

```
1 z' ← copy of z
2 for i ← 1 to log2(|T|) do
3     foreach cell c ∈ T in parallel do
4         z'[c] ← z'[c] + α[c] z'[recipient of c]
5         α[c] ← α[c] α[recipient of c]
6         recipient of c ← recipient of recipient of c
```

Fluvial erosion is not the only erosive process responsible for shaping terrain. *Hillslope processes* model the gradual accumulation of solid material at the base of mountains and hills [Braun and Sambridge, 1997] and is usually expressed as a diffusion equation. We follow Tzathas et al. [2024] and approximate it by including additional terms in the Stream Power Equation, changing kQ^m to $kQ^m + k_t + k_h A^{-h}$, where k_h and $h = 0.6$ are hillslope erosion parameters, A is the drainage area (obtained via flow routing with precipitation set uniformly to $p = \Delta x^2$). The parameter k_t accounts for slope-dependent effects (landslides, debris-flow), regrouped in computer graphics under the catch-all term *thermal erosion* [Musgrave et al., 1989].

Sediment deposition. Finally, flow routing can more generally propagate any type of transportable material. A case in point is sediment deposition, which acts as a complement to hydraulic erosion. Incorporating sediment deposition requires an additional term in the Stream Power Law [Yuan et al., 2019]:

$$\frac{\partial z}{\partial t} = -kQ^m \frac{\partial z}{\partial x} + k_d \frac{Q_s}{Q}, \tag{4.5}$$

where k_d is the sediment deposition coefficient and Q_s is sediment flux, obtained by accumulating the negative elevation balance $-\Delta x^2 \frac{\partial z}{\partial t}$ downstream with flow routing. While Yuan *et al.* [Yuan et al., 2019] suggest a fully implicit solution that requires iterating between Eq. 4.5 and the accumulation of Q_s, we propose a semi-implicit variant in which Q_s is computed from the elevation balance of the previous timestep. While our solution introduces a time dependency into mass conservation, it does not require iterating, which leads to a faster algorithm with no visual difference.

4.6 Results

In this section, we show the results of our method applied to the use cases detailed in section 4.5, and then demonstrate this performance against competitor methods.

4.6.1 Implementation

We prototyped our algorithms in Python with PyTorch and optimized our implementation with custom CUDA kernels. In particular, we seek to strike a balance by executing high-level functions in PyTorch or TensorFlow and handling compute-intensive kernels with CUDA. The code for this chapter is available at: `https://gitlab.inria.fr/landscapes/fastflow`. For ease of use, this includes our standalone unoptimized PyTorch implementation and a TensorFlow port, as well as our CUDA optimizations. The repository also includes examples of how to use our code to edit terrain with *Houdini* [SideFX, 2023].

In terms of optimization, we reduce memory allocations and copying and prevent CPU-GPU communication in the control flow. We found that allowing the GPU to run for the maximum number of iterations (*e.g.*, , $\log_2 n$ for flow routing) was more efficient than checking for early convergence. The only exception is in depression routing, where we halt the algorithm when no depressions remain.

4.6.2 Interactive Landscape Simulation

In this section, we showcase the application of our methods to interactive landscape simulation. In doing so, we compare against a recent approximate GPU solution for flow routing [Schott et al., 2023] and evaluate the impact of our new approach to sediment deposition.

As a starting point, Figure 4.1 shows a real digitized terrain, subsequently altered by

fluvial erosion and sediment deposition, and overlaid with flow-dependent vegetation, lakes, and rivers, in under 1.5 seconds.

Interactive landscape authoring. To demonstrate the general applicability of our algorithms we provide extracts from an interactive landscape authoring session in Figure 4.7 and the accompanying video. Here, user-guided simulation is applied to a 512×512 terrain, which, thanks to GPU acceleration combined with implicit time-stepping, requires only 10 iterations and 0.1s to capture $700,000$ years of geomorphological evolution. For comparison, a CPU implementation [Cordonnier et al., 2016] requires 2.6s. The user first defines the primary mountains by progressively painting on an uplift map, and can then freely change simulation parameters (here the deposition constant). Finally, the user advances the age of the mountain by increasing the number of iterations to 100, producing the result on the far right in 0.7 seconds. Note that lakes disappear over time as a consequence of filling by deposition and uplift, combined with erosion, which gradually removes the obstructions between them.

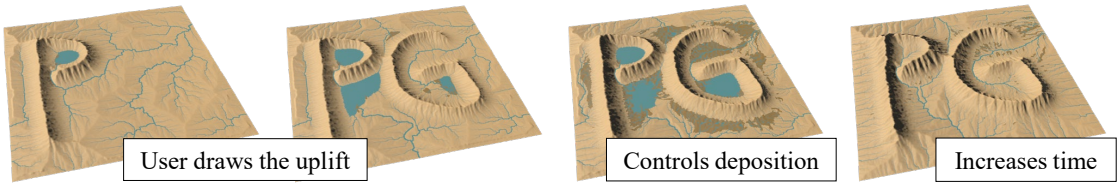


Figure 4.7: *Different stages in an interactive authoring session. The user paints uplift to provide an initial mountain structure, then changes the deposition parameters (sediment in dark brown and lakes and rivers in blue), and, finally, increases the overall simulation duration.*

Our method scales to higher-resolution landscapes, as illustrated in Figure 4.8 for a 2048×2048 terrain with $300,000$ simulated years (200 iterations) generated in under 10 seconds.

Rivers and lakes In Figure 4.9, we demonstrate how our algorithms can be used on complex large-scale terrains to demarcate lakes and rivers that are registered correctly with existing erosion lines and depressions. On the left in Figure 4.9, a mountainous region in the Alps transitions into an urbanized flat glacial valley. On the right, a tidal estuary around Mont Saint-Michel, France, leads into the sea. They contain $383,918$ and

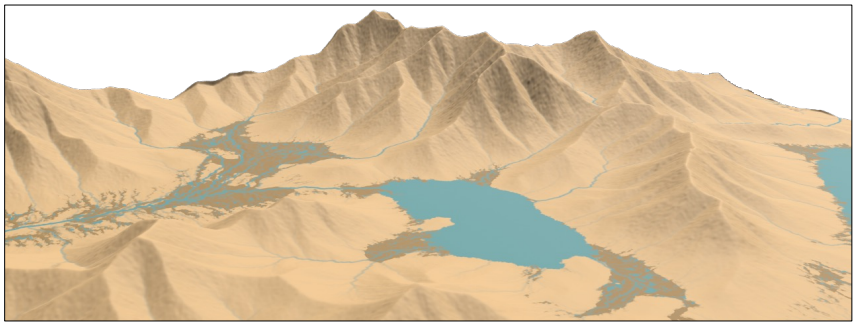


Figure 4.8: *A* $300,000$ *year-old mountain, generated on a* 2048×2048 *grid in under* 10 *seconds, incorporating fluvial erosion, sediment deposition, and river and lake formation.*

$8,445,644$ basins, respectively. In both cases, the elevation data was obtained from the IGN RGE ALTI Digital Elevation 1m dataset [IGN, 2022].

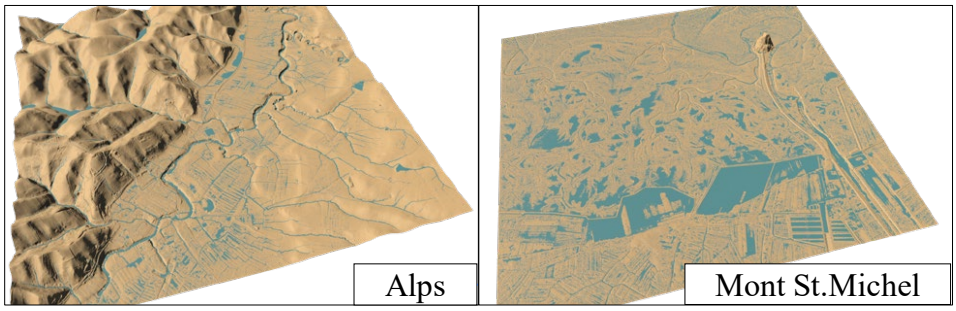


Figure 4.9: *River and lake identification on an alpine terrain dominated by a flat glacial valley (left), and a complex estuary (right).*

Sediment deposition. Figure 4.10 highlights the impact of sediment deposition. This increases the diversity of landscape patterns. In particular, sediment deposition, where present, dampens erosion, and leads to steeper slopes. Finally, we show in Figure 4.11 the visual impact of our semi-implicit deposition strategy. Our results are similar to a fully implicit variant [Yuan et al., 2019]. We note, however, that our gain in performance comes at the cost of some material loss.

Comparative evaluation. An alternative strategy for landscape simulation on the GPU [Schott et al., 2023] is to use explicit time-stepping for erosion combined with approximate flow routing, where the discharge is propagated by a single cell on each simulation step (Chapter 3). This method requires many more iterations due to the stability

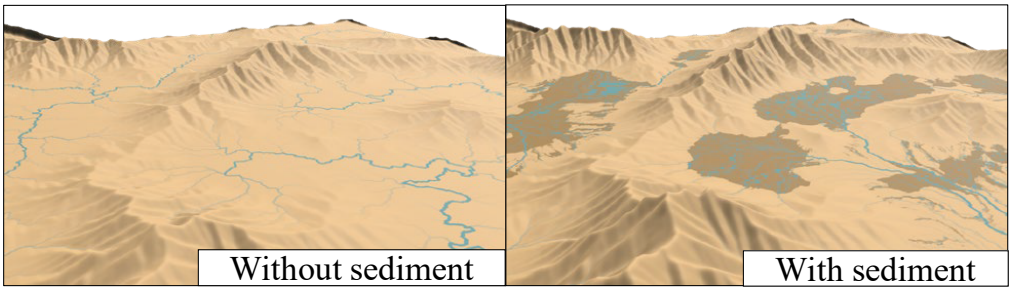


Figure 4.10: *Sediment deposition acts as a shield against erosion leading to more pronounced slopes.*

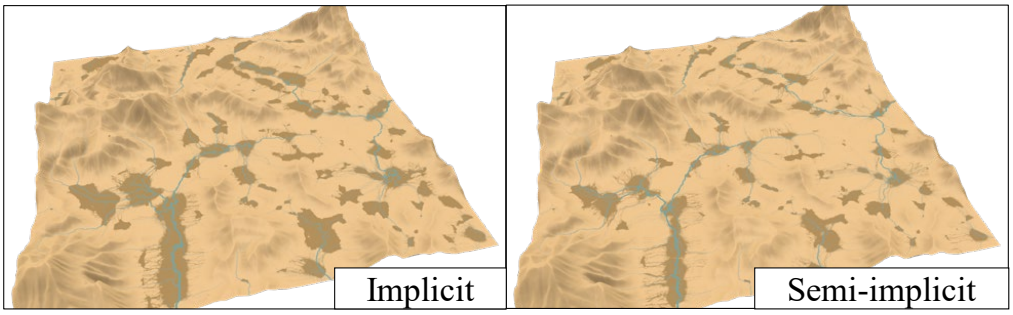


Figure 4.11: *The visual results of implicit and semi-implicit deposition are on par (with* 100 *iterations at* $\Delta t = 2000$*,* $\Delta x = 16$*, and a resolution of* 2048^2*).*

conditions of the explicit scheme but does eventually converge to a solution qualitatively similar to ours. Crucially, there are several situations where this strategy cannot be applied. One case is the erosion of an existing terrain containing depressions. Another failure case (see Figure 4.12) occurs when noise is added to uplift to increase the diversity. The approximate solution relies on the assumption that the flow path weakly varies over time, which allows the approximation of the discharge to progressively improve. Adding noise invalidates this assumption and leads to a significant underestimation of drainage and, consequently, an underestimation of erosion. This is visible in Figure 4.12, where the maximal elevation depends on the amount of noise. In contrast, our combination of implicit-time-stepping and exact-flow does not suffer from this shortcoming.

Another consideration is the acceleration afforded by the larger timesteps of our implicit solution. We show in Figure 4.13 that a 512×512 terrain with $\Delta x = 32$m obtained with an explicit scheme ($\Delta t = 1,000$ years) is visually similar to the result of our implicit solution ($\Delta t = 20,000$ years). Our implicit scheme allows an increase of the timestep by a factor of 20, which reduces, by the same factor, the iterations needed

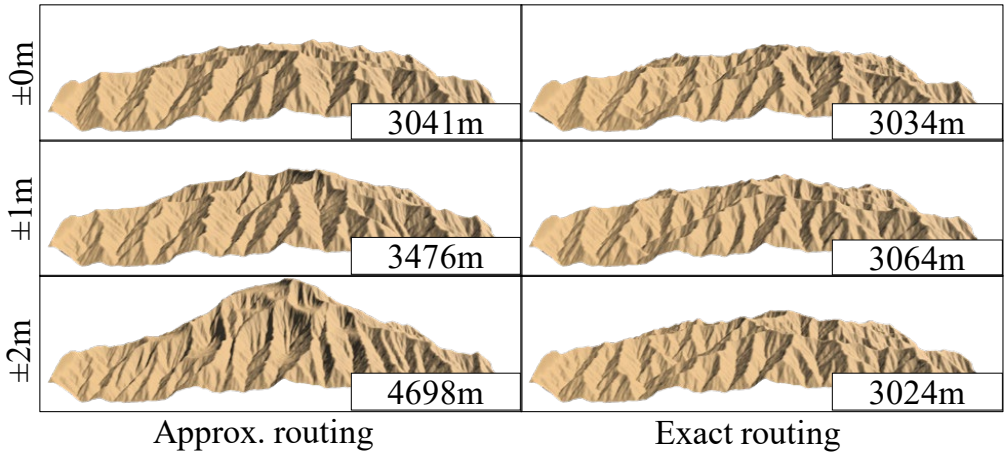


Figure 4.12: *Approximate flow routing [Schott et al., 2023] (left) versus ours (right), subject to noisy uplift with varying amplitude (top to bottom: ± 0, 1, and 2m per timestep). Compared to ours, the elevation with approximate routing depends on the magnitude of that noise.*

to achieve the same total geological timespan. With such timesteps, generating a 10 million year-old landscape requires 7.2s and 0.5s, for the explicit scheme and our implicit scheme, respectively. Note that larger implicit timesteps are unconditionally stable, and only diverge slightly from the explicit solution. Note that explicit and implicit schemes in general do not yield identical solutions as they accumulate discretization error differently.

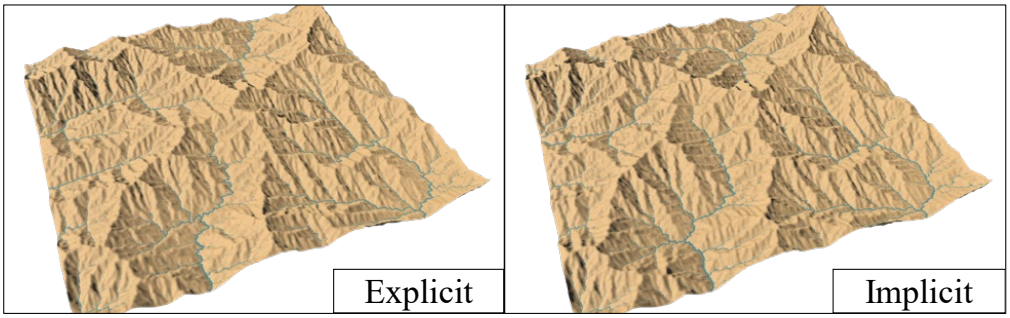


Figure 4.13: *Our implicit solution with $\Delta t = 20,000$ (right) provides a result similar to the explicit scheme with $\Delta t = 1,000$ (left). The latter is at its upper stability limit.*

We further motivate the need for depression routing by recreating the *escarpment* scenario from Cordonnier et al. [2019]. This begins with a flat terrain in which the boundary edges are lowered by 100m. Figure 4.14 shows the results without uplift after 250 iterations of $5,000$ years, both with and without depression routing. Depression routing prevents water loss, and therefore leads to a much deeper carving of the valleys.

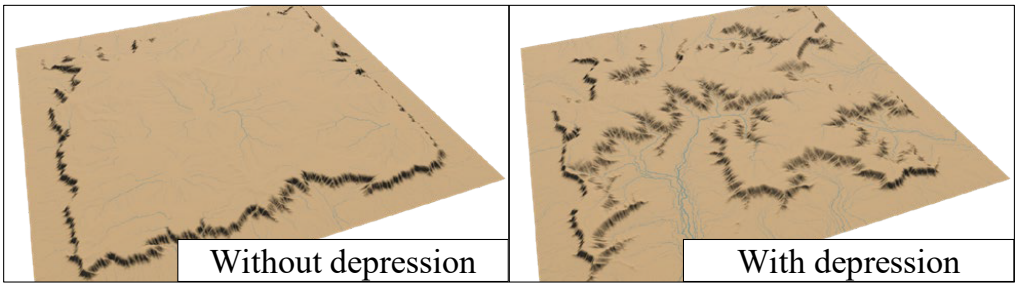


Figure 4.14: *Erosion carving on an escarpment, without (left) and with (right) depression routing.*

Limitations. For efficiency, our flow routing relies on parallel operations on a tree data structure, which in turn requires a single recipient per cell (Single Flow Direction, or SFD). As a consequence, we cannot handle multiple recipients (Multiple Flow Directions, or MFD), unlike some other schemes [Barnes, 2019; Schott et al., 2023]. Figure 4.15 illustrates the differences in appearance between rivers formed by SFD and MFD, using a CPU variant of Barnes's approach [Barnes, 2019]. From a visual standpoint, MFD is more blurred and SFD more sharply defined. From an application standpoint, SFD is favored for terrain erosion and river modeling, while MFD is more used in ecosystem simulation. Furthermore, Algorithm 7 for implicit fluvial erosion assumes a linear correlation with slope, meaning it cannot be applied to study non-linear behavior in geomorphology.

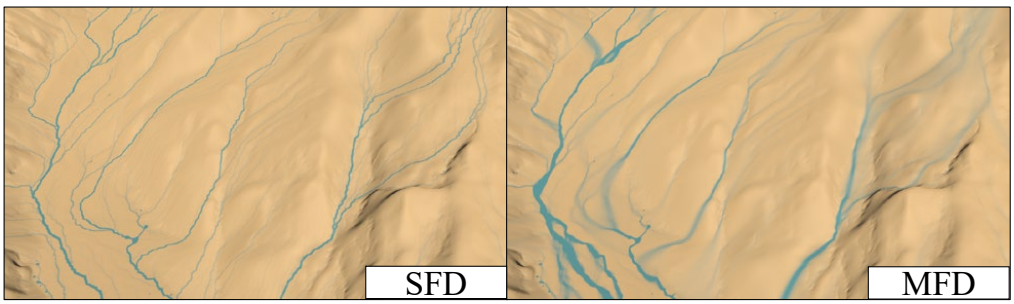


Figure 4.15: *The visual differences between our Single Flow Direction (SFD) algorithm and Multiple Flow Directions (MFD), which we are unable to handle.*

4.6.3 Performance

In this section, we compare our approach against other parallel GPU and CPU implementations of flow and depression routing.

Our performance tests were executed on a computer equipped with an Nvidia RTX A6000 GPU with 48GB of memory and used 20 cores of an Intel Xeon Gold CPU clocked at 2.10GHz with 128GB RAM.

For our test dataset, we incorporated a mix of real and synthetic terrains. We began by extracting a set of 10 real terrains from the IGN RGE ALTI Digital Elevation 1m dataset [IGN, 2022] chosen on the basis of topographic variety. These 10 terrains were sampled at cell resolutions of 512×512, 1024×1024, 2048×2048 and 4096×4096 to support scaling experiments.

We also included 5 synthetic terrains at the same resolution levels, and each with a controlled proportion of depression coverage (at *synth*-1%, *synth*-5% and *synth*-15% levels). These were generated by erosion simulation (Section 4.5), followed by layering different amplitudes of uniformly distributed noise. Note that it is common practice to add this type of noise during erosion simulation to mimic natural stochastic processes.

Finally, we included a uniformly flat terrain as an extreme test case, since in this instance all cells are local minima and hence coded as depressions.

4.6.3.1 Spatial Scaling

We show in Figure 4.16 the relationship between performance and terrain size, for resolutions ranging (logarithmically) from 64×64 to 8192×8192. We observe that parallelism is limited by the maximum simultaneous threads in our GPU model at around 8192×8192, where performance follows a near-linear trend.

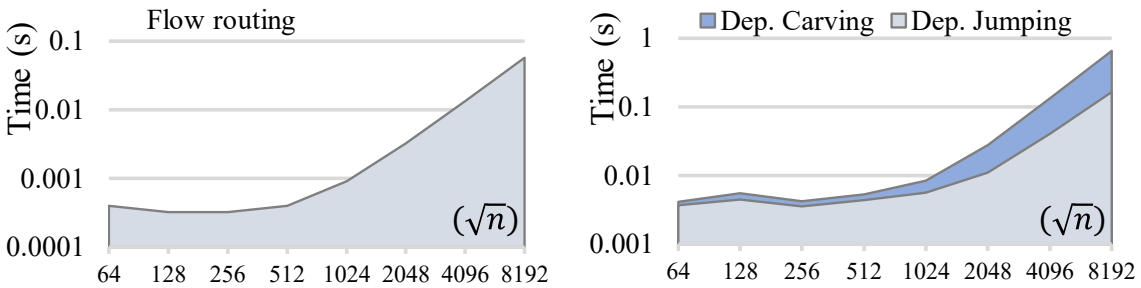


Figure 4.16: *Scaling experiments: (1) flow routing, (2) depression jumping, and (3) depression carving.*

4.6.3.2 Flow Routing

Topographic variation does not have a significant impact on the performance of flow routing, so we benchmark flow routing solely on real terrains. We compare our algorithm against:

- Barnes [Barnes, 2019], who provides a GPU algorithm for flow routing based on a pre-ordering of the computation graph.
- Schott *et al.* [Schott et al., 2023], who use a GPU algorithm similar to ours, but without pointer jumping, which results in as many iterations as the node length of the longest river in the terrain ($\sim \sqrt{n}$).

Note that neither of these algorithms provides a solution for depression routing on the GPU. The results are illustrated in Figure 4.17, with an average speedup of $10\times$, $5\times$, $3\times$ and $3\times$ against [Barnes, 2019] for each target resolution, respectively.

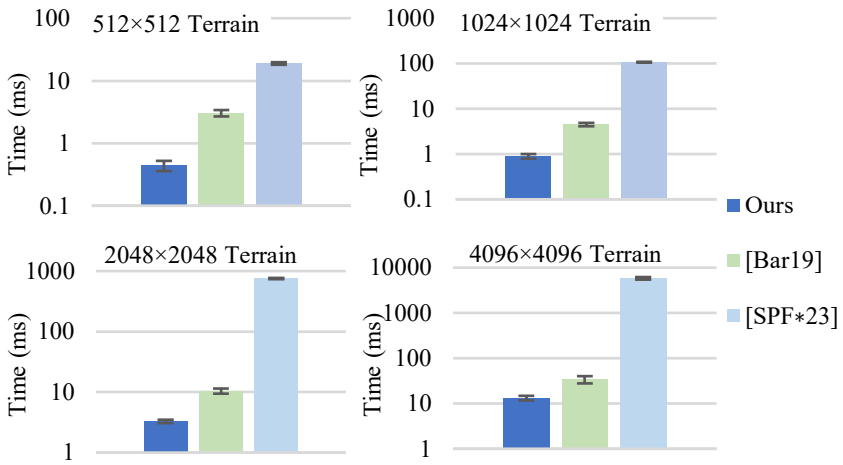


Figure 4.17: *Performance of our flow routing against [Barnes, 2019] and [Schott et al., 2023].*

4.6.3.3 Depression Routing

To test the performance of depression routing we include all terrain categories (synthetic, real, and flat) from our benchmarking suite since the distribution of local minima can have a significant impact in this case. We compare two variants of our algorithm and previous work, as follows:

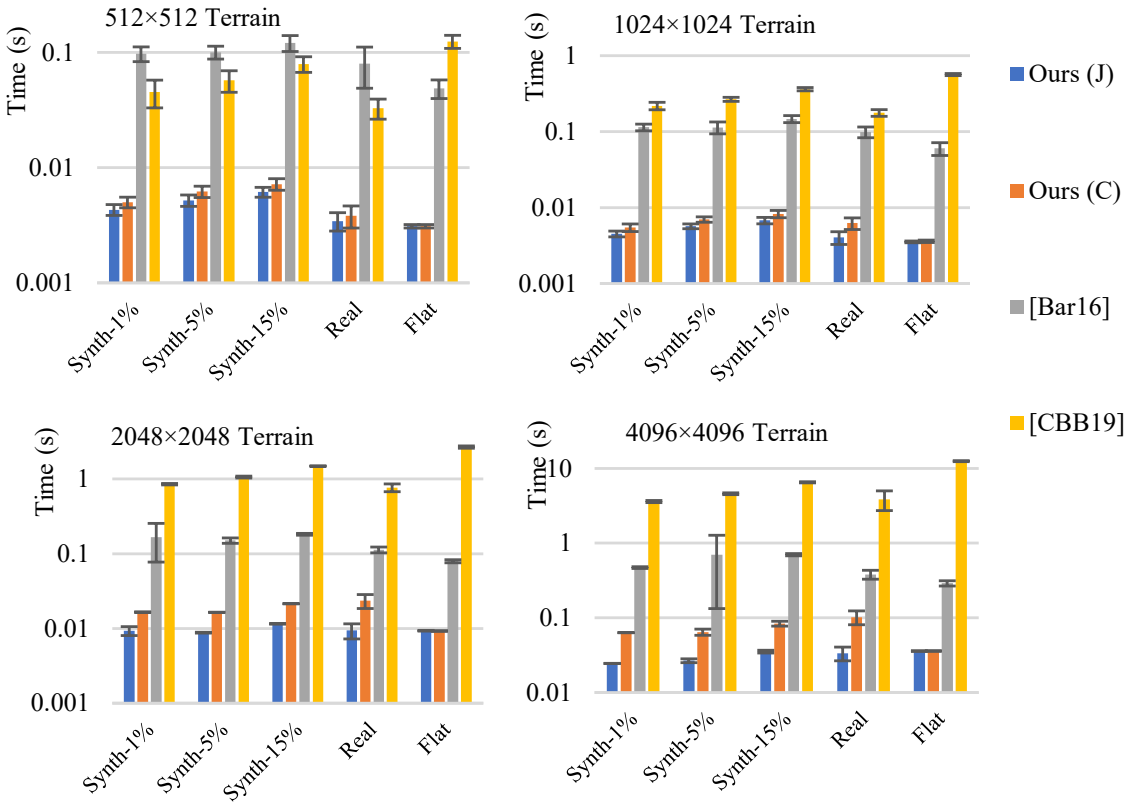


Figure 4.18: *Performance of depression jumping (Ours (J)) and carving (Ours (C)) against a parallel CPU implementation [Barnes, 2016] and a single-core CPU implementation [Cordonnier et al., 2019].*

- **Ours (J)** Our algorithm with depression jumping.
- **Ours (C)** Our algorithm with depression carving.
- Barnes [Barnes, 2016], who proposed an algorithm optimized for CPU-based parallel or distributed computing.
- Cordonnier *et al.* [Cordonnier et al., 2019], who provided a $\mathcal{O}(n)$ solution for flow routing on the CPU with a single-core implementation.

Note that [Barnes, 2016] does not solve the depression routing problem as posed here since their goal is instead to fill depressions. Extending this to a more general solution that enables flow routing for discharge computation is far from trivial.

The comparative performance of these depression routing algorithms is shown in Figure 4.18. In particular, we note that our algorithms outperform previous work by one order of magnitude for terrains up to a resolution of 2048×2048. However, this gap narrows somewhat for the largest terrains, which indicates unfavorable scaling at the upper end. Also worth noting is that, with the exception of flat terrains, the comparative advantage of using depression jumping increases for larger terrains.

4.7 Conclusion

Algorithms for evaluating water flow over terrains are a staple of geo-analysis, with applications in computer graphics and beyond. Consequently, any improvement in their run-time performance is worth consideration. In this chapter, we provide improved algorithms for solving both flow and depression routing problems, with, respectively, $\mathcal{O}(\log n)$ and $\mathcal{O}(\log^2 n)$ complexity for a terrain with n nodes. This is an improvement on previous methods, which usually require as many iterations as the length of the longest river ($\sim \sqrt{n}$). Most importantly, we are the first to propose a GPU solution for both flow and depression routing.

In comparative terms, our GPU implementation for depression routing outperforms an optimized parallel CPU algorithm by $34\times$ to $52\times$ on a 1024^2 resolution terrain, depending on the strategy for recipient correction. We also improve on previous GPU methods for flow routing [Barnes, 2019; Schott et al., 2023] by a factor of $5\times$.

In terms of raw performance, our GPU implementation executes in under 55ms on terrains up to 4096^2 sample resolution. This opens up new opportunities for using flow and depression routing in interactive terrain simulation. Further work is still required to broaden the algorithm, for instance by allowing multiple recipients through Multiple Flow Directions. With exact and efficient routing on heightfields in place, the remaining limitation is the representation itself. Heightfields cannot describe overhangs, arches, or caves, even though water and wind carve some of their most distinctive structures there. The next chapter therefore extends flow and depression routing to volumetric, particle-based terrain, where the notions of surface, recipient, and depression must be redefined, and uses them to simulate sandstone landforms.

Chapter 5

Arenite: A Physics-based Sandstone Simulator

5.1 Introduction

The previous chapter developed exact GPU routing methods for heightfield terrains, which represent elevation as a single-valued surface: each horizontal position corresponds to exactly one height. These methods make it possible to recompute flow and depression structure efficiently during interactive simulations. However, many landscapes cannot be represented adequately by a heightfield. Sandstone formations, in particular, include dramatic cliffs, mesas, natural bridges, canyons, caves, arches, and pinnacles, whose geometry often contains overhangs or cavities. Extending flow-based terrain simulation to such landforms therefore requires moving beyond heightfields and representing terrain as an erodible volume.

Sandstone landscapes are visually striking structures, frequently used in films to evoke grandeur, isolation, adventure, or otherworldly environments. Yet, despite their prominence in natural scenery and visual culture, they remain rare in digitally generated



Figure 5.1: *A large sandstone structure generated by Arenite. The user defines simple initial conditions of each object (layer hardness and vegetation), and the physics-based simulation generates the results.*

landscapes. This absence comes from the difficulty of simulating the layered erosion that shapes them over millennia through the combined action of wind, water, material strength, and climatic variation.

Sandstone is a sedimentary rock made of sand grains cemented together by a matrix of minerals such as quartz, feldspar, or calcite [Pettijohn et al., 2012]. In desert or semi-arid regions, the wind carries abrasive particles that wear down the surface of the sandstone [Loope and Mason, 2009; Ostanin et al., 2017], while episodic flash floods channel debris-laden water through narrow fissures, acting as a powerful abrasive force that carves prominent landmarks such as the Antelope Canyon. The impact of these eroding factors is reduced in the case of local stress, thanks to a unique process called *fabric interlocking* that makes compressed sandstone material much stronger [Bruthans et al., 2014; Safonov et al., 2020; Turkington and Paradise, 2005; Migoń, 2021]. Differential erosion, where softer layers erode faster than harder ones, and irregular stress distribution create unique features such as pinnacles, natural bridges, and arches. Over millions of years, water flows, dissolving cemented layers, and digs steep canyons and gorges.

Despite these well-documented geological mechanisms, simulating these unique 3D structures remains an open challenge in computer graphics. Traditional terrain modeling largely relies on heightfields to simulate hydraulic and thermal erosion [Musgrave et al., 1989; Benes and Forsbach, 2001; Štava et al., 2008], which inherently cannot capture complex overhanging volumetric features, while data-driven terrain synthesis methods [Guérin et al., 2017; Huang and Yuan, 2023] also struggle to replicate these formations due to the lack of comprehensive 3D sandstone datasets. Some methods have successfully introduced 3D medium-scale terrain generation via interactive layered carving (such as Arches [Peytavie et al., 2009]), volumetric construction [Paris et al., 2019a], directable spherical and wind erosion [Jones et al., 2010; Krs et al., 2020], or particle-based erosion models [Hartley et al., 2024], yet fast and accurate computation of 3D flow paths for water and sediment transport over complex volumetric shapes remains a key limitation.

The key insight of this chapter is to use a combination of stress simulation and multi-factor erosion mechanisms to address the specificity of sandstone erosion. We carefully isolate the key shape-forming phenomena: multi-physics fabric interlocking, wind, fluvial erosion, and particle-based deposition processes. We show how their interplay can be used to simulate a wide variety of observable phenomena [Härtel et al., 2007]. One common challenge of physics-based algorithms is controlling the simulation to allow the authoring

of the resulting landscapes. Our interactive authoring system allows users to guide the evolution of physically plausible landscapes by interactively painting key parameters such as erosion rates, material distribution, and stress feedback that directly affect crack formation. An example in Figure 5.1 shows a 3D sandstone structure generated by *Arenite*. The user positions several 3D blocks, paints vegetation and erodibility on the surfaces, and the physics simulation then generates the results for each object. The scene definition takes less than a minute, and the overall simulation time was less than five minutes per sandstone structure.

The main contributions of this chapter are 1) extending flow and depression routing to 3D shapes represented by particles, surpassing the limitations of existing methods that only support heightfields, also called 2.5D terrains, as detailed in Chapter 4; 2) a fluvial erosion model for carving gullies and canyon-like channels; and 3) a deposition method that calculates how eroded particles are deposited in 3D environments, incorporating key findings in geomorphology.

The stress computation and wind erosion components were contributed by Zhanyu Yang, and are summarized briefly in Section 5.2.4. The contributions listed above were my individual contributions, which we detail in the following sections.

5.2 Overview

Sandstone formation in CG is an understudied topic due to several specific challenges: such mid-scale objects (tens to hundreds of meters) must be fully represented in 3D, which adds complexity to the physics simulation. Moreover, sandstones undergo a complex interplay of erosion and internal stresses, requiring novel algorithms that need to operate on dedicated data representations.

5.2.1 Fabric Interlocking Controls Sandstone Erosion

Sandstones are sedimentary rocks composed of sand grains (quartz or feldspar) glued by a cementing material (e.g., silica) [Pettijohn et al., 2012; Okada, 1971]. These rocks are formed by the accumulation and compaction of sand deposits at the outlet of rivers, beaches, deserts, or shallow continental shelves. After millions of years, changes in geological forces exhume the sandstones and subject them to erosion, which leads to intricate patterns, especially in arid environments where chemical weathering is minimal and vegetation does not conceal the underlying rocks.

Iconic structures stemming from sandstone erosion, such as mesas, canyons, arches, alcoves, buttes, or caves, have been widely studied in geomorphology. Yet, only recently, a consensus emerged on the key process in their formation, namely a negative feedback between erosion and internal stress [Bruthans et al., 2014]. Stress impacts erosion through a process called *fabric interlocking*, where high loads favor the chemical dissolution and re-precipitation of sand grains. The rearranged internal structure is then characterized by a lower porosity and higher grain contact area, which increases resistance to erosion [Migoń, 2021]. A typical example is load-bearing areas such as pillars, where the local stress increases when weathering reduces the cross-section. When a critical threshold is reached, fabric interlocking transforms granular sediment into a strong, rock-like material that can resist further erosion.

5.2.2 *Arenite*

Our sandstone simulator *Arenite* (see Figure 5.2) tackles the efficient simulation of the fabric interlocking process, enabling the emergence of most features observed in sandstone landscapes. Our solution relies on a hybrid data structure that captures the interplay between stress and erosion, as well as a unique way to simulate the progressive change of rock state through the notions of *erodibility* (capacity to erode) and *viability* (current state), both locally updated over time.

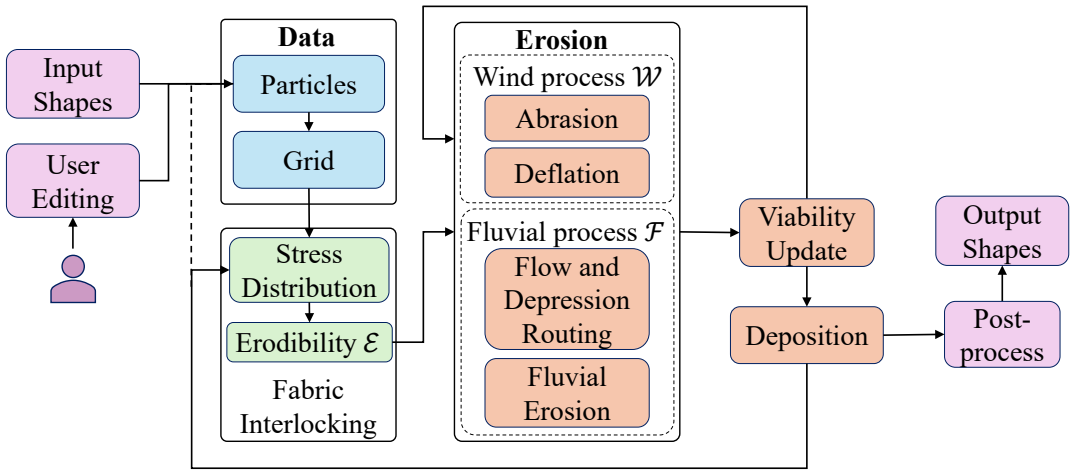


Figure 5.2: *Arenite overview. The rough input shape has some areas painted by the user (viability, vegetation) and is converted into particles in a grid. Stress distribution is calculated, and erodibility is determined. The wind and fluvial erosion then modify the shape, and some material is deposited, which results in the final shape.*

Stress and Erosion. We model the interplay between stress and erosion as follows. Considering a small volume of rock at the surface of the sandstone, we estimate its *viability* b that describes the expectancy that it has not yet been eroded. The viability is initialized as $b(t = 0) = 1$, and evolves as:

$$\frac{\partial b}{\partial t} = -\mathcal{E}(\sigma_c)\,(\mathcal{W} + \mathcal{F}) \tag{5.1}$$

where $\mathcal{W}$ and $\mathcal{F}$ are scalars representing local erosion effects caused by the wind and fluvial processes.

When b is small, the local eroded rock is transported and deposited. This decreases the rock volume, which in turn increases its Cauchy stress tensor σ_c. When stress exceeds a threshold I, erodibility $\mathcal{E}$ switches to the fabric interlocking state:

$$\mathcal{E}(\sigma_c) = \begin{cases} k_s & \text{if trace}(\sigma_c) > I \\ k_w, & \text{otherwise.} \end{cases} \tag{5.2}$$

The parameters k_w and k_s, where $k_w \gg k_s$ stand for weak versus strong material, and respectively model the erodibility of the local rock layer before and after the interlocking phenomenon.

Hybrid Data Structure. Modeling such negative feedback between stress and erosion and the evolution of the 3D rock shape requires a data structure capable of capturing the stress distribution within a 3D volume and its temporal evolution. We adopt a hybrid representation that combines a regular grid with particles. In contrast to similar hybrid structures in fluid simulation and continuum dynamics [Brackbill et al., 1988; Sulsky et al., 1995], we change the purpose of particles: Instead of tracking dynamic motion – which is not needed to model static geomorphological features – they capture the local evolution of weathering, and by being moved, model both rock erosion and deposition. The benefit is that topological changes can be allowed at low costs while the grid is used for efficiently computing the internal stress forces within the remaining rock structure.

5.2.3 Erosion Algorithm

The *input* to our erosion simulation algorithm is the initial shape of the rock structure and a set of local erodibility parameters. The former can be built using any geometric representation compatible with subsequent 3D sampling with particles, such as voxels,

manifold meshes, or implicit surfaces. In our implementation, we use a constructive solid geometry (CSG) tree made of the union and intersection of simple cubes and spheres. Note that rough initial shapes have little influence on the visual quality of results, as even a cube may erode into a realistic sandstone pillar (see Section 5.5). Due to the sedimentary nature of sandstone, we specify the local erodibility parameters through a stack of altitude values defining the limits of sediment layers and a set of values (k_w, k_s) in each of these layers. This vertical stacking enables us to mimic successive layers of sandstone sediments, whose composition and, therefore, resistance to erosion depend on the climatic and geological conditions during their formation.

To complement this input, we sample the particles uniformly within the initial shape and assign them the erodibility parameters corresponding to their respective sediment layers. The user can paint specific erodibility values directly on certain particles for more expressive control and/or perturb them with Perlin noise (see Section 5.5.2).

We simulate Eq. (5.1) iteratively, splitting the equation and successively evolving each term over time with forward Euler in the following steps, as outlined in Algorithm 8:

Algorithm 8: Sandstone simulation

Data: A constructive solid geometry representing the initial shape
Result: Output sandstone shapes

Initialization
- Sample particles
- Extract grid representation

forall *Frames* **do**
- Update stress distribution using MLS-MPM (Section 5.2.4.1)
- Compute wind deflation and abrasion (Section 5.2.4.2)
- **Fluvial Process**
  - Compute receivers (Section 5.3.2)
  - Reroute depressions (Section 5.3.2)
  - Compute water discharge (Section 5.3.2)
  - Compute fluvial erosion using Eq. (5.4)
- Compute overall erosion using Eq. (5.1)
- **Deposition**
  - Update deposition pointers (Section 5.4)
  - Deposit eroded particles

1. *Stress computation:* We rely on the Material Point Method on the grid to efficiently compute σ_c and deduce $\mathcal{E}$.

2. *Wind erosion:* We model wind erosion $\mathcal{W}$ through two mechanisms: (1) a background effect that simulates global, cumulative erosion, and (2) a local process that simulates localized, non-uniform erosion via Smoothed Particle Hydrodynamics (SPH), capturing particle interactions between wind and sandstone.

3. *Fluvial erosion:* We introduce a 3D flow transport algorithm to compute the fluxes of water running over the sandstone and deduce the associated fluvial erosion $\mathcal{F}$ (Eq. (5.3)).

4. *Deposition:* We transport all the eroded particles to a stable location to model deposition.

The user interactively controls the simulation process, allowing parameter changes. Finally, for rendering purposes, we convert the final set of particles into a mesh by first extracting normals and then using Screened Poisson Reconstruction [Kazhdan and Hoppe, 2013], complemented by Least Squares Subdivision Surfaces [Boyé et al., 2010].

5.2.4 Overview of Stress and Wind Erosion Models

While the primary focus of this chapter is on fluvial and depositional processes, the complete *Arenite* framework relies on a coupled simulation of internal stress and wind erosion to fully capture sandstone geomorphology. This subsection briefly outlines these supporting mechanisms to provide necessary context for the subsequent results.

5.2.4.1 Stress Computation

Internal stress critically impacts the response of sandstone to erosion by altering the microscopic arrangement of sand grains through a process known as fabric interlocking. To efficiently evaluate the Cauchy stress tensor, the simulation employs the Moving Least Squares formulation of the Material Point Method (MLS-MPM) [Hu et al., 2018]. Sandstone is modeled as a stiff elastic material subject to the conservation of mass and momentum [Stomakhin et al., 2012; Jiang et al., 2016]. The MLS-MPM discretization utilizes a hybrid data structure where particles act as material points that transfer their mass and velocity to a background grid [Jiang et al., 2015]. The grid updates its velocity

based on internal and external forces, and these velocities are subsequently mapped back to advect the particles. Ultimately, the calculated stress is used to update the local erodibility of the particles, ensuring that compressed, load-bearing regions successfully resist erosion.

5.2.4.2 Wind Erosion

Wind erosion, which is paramount for shaping arid sandstone landscapes into complex 3D structures, is decoupled into two scalar components: abrasion and deflation. Wind abrasion simulates the directional, non-uniform mechanical weathering caused by the impact of wind-blown particles [Loope and Mason, 2009; Greeley and Iversen, 1985]. This is computed by treating the wind as an incompressible fluid via SPH [Bender and Koschier, 2015], allowing for smooth rigid and fluid coupling against the static sandstone particles to evaluate impact velocities and angles [Akinci et al., 2012]. Conversely, wind deflation models the continuous, uniform detachment of fine-grained particles from the rock surface over long geological periods [Ostanin et al., 2017]. Deflation calculates the probability of particle detachment using the Mohr-Coulomb law for dry friction, establishing a critical normal drag force threshold [Hennessey Jr, 1978]. The combined effects of abrasion and deflation progressively decrease the viability coefficient of the surface particles. Once a surface particle's viability reaches zero, it is considered fully eroded, removed from the rock structure, and deposited (Section 5.4).

5.3 Fluvial Erosion on Sandstones

Even in the arid landscape surrounding sandstones, water occasionally surges from episodic storms and acts as an additional process for sandstone erosion. Fluvial patterns are mainly visible as narrow gullies that converge to form dendritic networks. We discuss how to compute the fluvial erosion coefficient $\mathcal{F}$ to update a sandstone particle's viability via Eq. (5.1). Similar to wind erosion, fluvial erosion affects only the surface particles.

5.3.1 Choice of an Erosion Model

We borrow the shear stress erosion law from geomorphology [Howard, 1994], which erodes proportionally to the shear stress of water τ, when it exceeds a constant critical

threshold τ_c:

$$\mathcal{F} = -k_t(\tau - \tau_c)^+. \tag{5.3}$$

Re-expressing τ as a function of the terrain slope S, flow discharge Q, and fluvial erosion coefficients k_t and k_f in [Howard, 1994], yields:

$$\mathcal{F} = k_t(k_f \cdot Q^{0.6} S^{0.7} - \tau_c)^+. \tag{5.4}$$

Note that while Eq. (5.4) is similar to the Stream Power Law used to erode large-scale landscapes [Cordonnier et al., 2016], this expression has the advantage of explicitly using the critical stress threshold. This is particularly relevant for smaller-scale geomorphological features, where discharge and stress are low, as in our case.

In practice, the critical shear stress τ_c controls the distance from higher elevations at which fluvial erosion patterns emerge, as our results show (see Figure 5.13).

5.3.2 Computation of the Fluvial Erosion Terms

We now detail how we evaluate the different terms in Eq. 5.4.

The slope S of our 3D representation of sandstone is derived from the normal vector to the sandstone surface $\mathbf{n}$ as $S = \sqrt{n_x^2 + n_y^2}/|n_z|$. The normal vectors are calculated by identifying surface particles on a grid and deriving their normals from the covariance of their nearest neighbors.

Discharge Q, defined as the upstream integral of precipitation and water sources [Cordonnier et al., 2016] can be computed for a given point p as the sum of the precipitation of all the points q, so there is a path of strictly decreasing elevations from q to p. While this definition was proposed for 2.5D terrains, it generalizes to 3D with the constraints that all paths are either located on the surface of an object if this surface is oriented upward ($n_z > 0$, this is not a ceiling), or traverse empty space vertically (waterfalls).

Computing Q is challenging as water transfers are non-local. At the geological scales of erosion, water flows over long distances during a timestep. Approximating the problem as a local one (where water is only transferred between neighboring elements) [Schott et al., 2023] stabilizes if the erosion is slow, but cannot handle *depressions* i.e., local minima that interrupt the flow path. Instead, we follow *FastFlow* (Chapter 4), which presents the estimation of discharge (flow routing) along with a solution to the problem of depression. Flow paths are updated to route the flow out of the local minima as if the depression fills and outflows.

FastFlow was designed to run at the surface of a height-field terrain. To generalize it to 3D, we use a 3D regular grid aligned to the one used for the MPM stress evaluation.

Receivers. *FastFlow* uses a tree-like structure for flow paths, where each element has a single *receiver* of strictly lower elevation. To extend this concept to 3D, we first tag grid cells as empty if they do not contain particles and as surface cells if their upper neighbor is empty. We find the altitude of the highest particle in each surface cell c and assign it as a proxy elevation z_c. This elevation provides a total ordering of the surface cells, preventing any cycle in the flow paths.

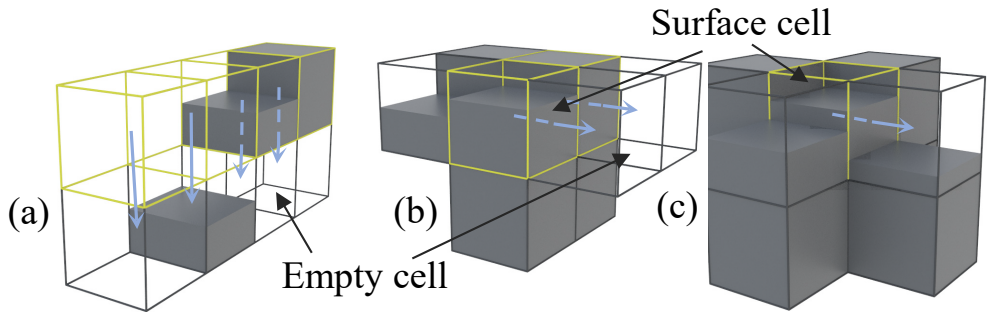


Figure 5.3: *Choice of the receiver (blue arrow) of the yellow cells. (a) All types of cells point to the cell below if empty. (b) Otherwise, cells point to a horizontal neighbor if empty. (c) Otherwise, surface cells give to a lower surface cell.*

We assign receivers to 3D cells as follows: (see Figure 5.3). If the cell below c is empty, we choose it as a receiver. Otherwise, we randomly select one of the four horizontal neighbors of c that are empty, if any. If none of the above applies but c is a surface cell, we select one of these horizontal neighbors n if n is a surface cell and $z_n < z_c$ (randomly with a probability proportional to $z_c - z_n$ if there is more than one choice). Otherwise, no valid receiver exists. If c is a surface cell, it is tagged as a local minimum; otherwise, it is tagged as solid and masked out of the subsequent steps of the algorithm.

Note that cells on the walls and ceiling of the sandstone have a specific status: they are not surface cells but have a lower or lateral receiver. Even if water cannot reach these cells, this choice allows the evacuation of sediments eroded by wind.

Depression Routing. The *FastFlow* algorithm is applied to the new set of receivers, with modifications arising from our extension to 3D. In particular, depression routing removes local minima by connecting them to external nodes. *FastFlow* tags a set of cells

as a basin if they flow toward the same local minima and then computes a minimum spanning tree to connect these basins. This can fail in 3D if solid surfaces entirely surround a cavity. We detect this situation during the minimum spanning tree computation (a connected component does not have a valid neighbor) and remove all the nodes of this basin from the computation.

Flow Routing. We now have all the components to compute the discharge Q. We assign to each cell at the upper boundary of the domain a constant precipitation value p and set the precipitation to zero everywhere else. We use the rake-compress algorithm from *FastFlow* to efficiently integrate precipitation along the flow paths, which yields the discharge Q.

5.4 Deposition

Our model simulates the gravitational settling of rock and debris particles dislodged by wind and water erosion. We neglect smaller sand grains and their transport by wind, as they are carried away from our domain size. As for fluvial erosion, the challenge of deposition lies in the multiple-cell displacement of falling debris over erosion timescales. We propose an algorithm to estimate the deposited location of the particles directly without explicitly modeling sand motion. This algorithm builds upon the 3D flow routing method detailed in Section 5.3. In particular, we reuse the 3D grid that embeds the receiver information, updated to account for depression routing.

Granular materials deposit where the surface slope falls below an angle of repose, also known as the talus slope [Musgrave et al., 1989]. We define a *stable cell* as a grid cell where the average slope of surface particles is less than this critical angle. For each surface cell c, we compute a *deposition pointer* $p(c)$ that indicates the next closest downward stable cell. This pointer predicts where an eroded particle originating from cell c would be deposited. Initially, $p(c)$ is set to the receiver of cell c. To efficiently determine the final deposition location, we iteratively update these pointers using a parallel pointer jumping technique: $p(c) = p(p(c))$. This process is repeated for a maximum of $\log(n)$ iterations, where n is the total number of surface cells. Figure 5.4 illustrates this algorithm, converging in three iterations in this simple 2D case.

Each cell containing eroded particles uses p to increase the count of the number of particles that should be deposited in each stable cell. We deposit particles at random 2D

positions within the cell and at the cell proxy elevation z, increasing after each deposited particle to maintain a target particle density. This approach, however, could introduce staircase artifacts. To mitigate this, we move the grid by a random offset for particles at each deposition step, ensuring particles are not grouped in the same grid cell each iteration.

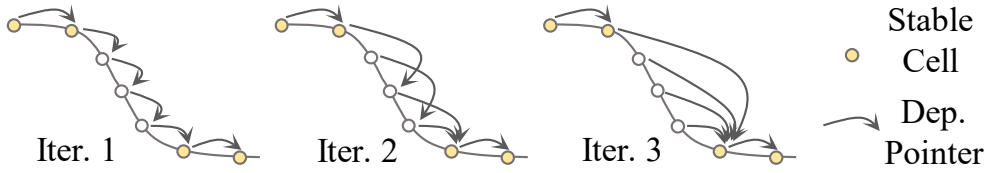


Figure 5.4: *Deposition Algorithm: each cell points to its deposition position after three iterations of pointer jumping.*

5.5 Implementation, Results and Validations

We implemented our method using Python with PyTorch, Taichi [Hu et al., 2019] and custom CUDA kernels on the GPU. We generated all results on a desktop computer with an Intel Xeon Gold CPU clocked at 2.10GHz, 128GB RAM, and an Nvidia RTX A6000 with 48GB of memory. By default, the simulations use a weak rock erodibility of $k_w = 10^{-4}\,\mathrm{year}^{-1}$, a strong rock erodibility of $k_s = 10^{-6}\,\mathrm{year}^{-1}$, a discharge multiplier of $k_f = 8 \times 10^{-5}\,\mathrm{Pa\,m^{-3}s^{-1}}$, and a timestep of $\Delta t = 1000\,\mathrm{year}$. The meshes were textured using tri-planar mapping, vegetation was generated procedurally, and the results were rendered using Houdini (and Terragen for Figure 5.6).

5.5.1 Emergent Sandstone Structures

We claim that the key ingredient for the emergence of sandstone structures is the negative feedback between erosion and stress-driven fabric interlocking. We validate this assertion by showing that sandstone structures emerge from our simulation from simple initial conditions in Figure 5.5. For these experiments, we show a photograph of a real rock formation, the simulation's initial conditions, the algorithm's output as a set of particles, and the final rendering. We then turn toward structures where fluvial erosion is a primary driver. At large scales, water can have a more significant impact, for instance, on the formation of canyons, and cave networks.

Alcoves are large empty recess areas that form arcs in sandstones (Figure 5.5 top). We

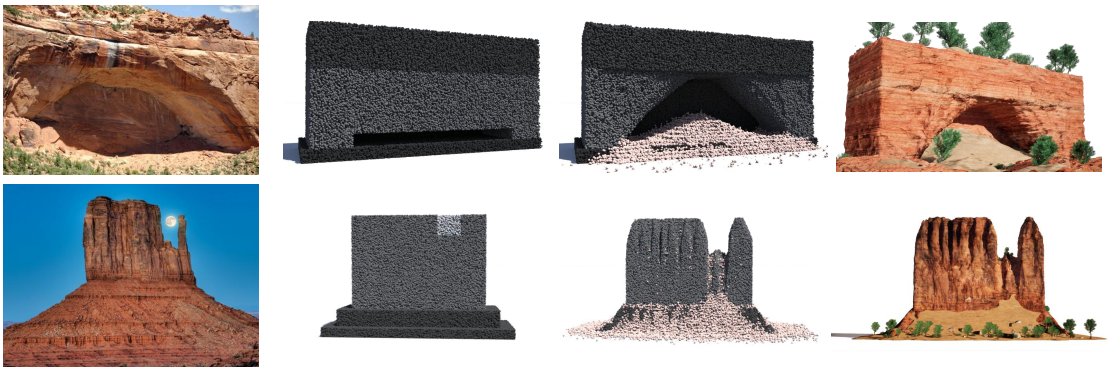


Figure 5.5: *A real sandstone structure (column a), the input to our simulator (column b) where the rock is composed of layers with varied erodibility (darker is stronger), particle representation (column c), and realistic rendering (column d). (top) An alcove is a large recess in sandstone. An example from Zion National Park, Utah (courtesy James St. John, Flickr) is replicated by a large block with an erodible block at its foot. A small hole at its foot initiates the erosion process and deposits are piled below (particles in lighter shade). (bottom) The butte is a large vertical block eroded by fluvial erosion from the top. The Monument Valley butte from Monument Valley, Utah (courtesy John Fowler, Flickr) is replicated using three blocks and interactively marking the top one with several instabilities. The fluvial erosion then forms the resulting object. Both simulations use 1.5M particles. The alcove uses fluvial shear stress threshold* $\tau_c = 5\,\mathrm{Pa}$*, fluvial erosion coefficient* $k_t = 10^2\,\mathrm{Pa}^{-1}$*, and deposit viability* $b = 0.2$*; the butte uses* $\tau_c = 1\,\mathrm{Pa}$*,* $k_t = 4 \times 10^2$ *to* $8 \times 10^2\,\mathrm{Pa}^{-1}$*, and* $b = 0.1$*.*

use a cuboid as our initial shape, with varied rock erodibility (darker is stronger). We disable erosion at the back, left, and right faces, and increase the hardness of the top layer. The process is initialized by a small hole at the foot of the cave, and the simulation erodes the structure, with eroded particles being deposited below.

Buttes are isolated hills with vertical walls and flat tops (Figure. 5.5 bottom). The initial pillars formed by stream erosion as the area was submerged. After it drained, fluvial erosion caused the pillar to split into vertical columns. The hard top layer, or caprock, is slowly eroded by wind and rain, causing mesa-like deposition at the butte's foot. We reproduce the butte by defining a cubical block standing on two larger ones, forming a pyramidal structure. We carve a small crevice on the caprock to create an instability, which causes fluvial erosion to converge on the right side of the butte.

Canyons are primarily made of sedimentary rocks, such as limestones and sandstones [Hereford et al., 1993]. Sandstones are subject to fluvial erosion for high water discharge while capable of maintaining large cliffs when little or no water is present

because of the critical shear stress term in Eq. 5.4. Figure 5.6 (right) shows an example of a canyon simulated by fluvial erosion. We input an initial water path on a rock block capped with a gentle slope upon a hard layer of rock and impose a high discharge at the water source. Fluvial erosion carves a deep canyon, which is widened by wind erosion.

Cave networks are mainly a feature of limestone but can also occur for specific combinations of permeable and impermeable sandstones [Veress, 2020]. These conditions allow the resurgence of small water streams along underground natural cracks, which amplify to form tunnels and natural cavities. We paint the initial water course inside a sandstone block with layers of rocks with different erodibility, impose a strong water discharge at the river's source, and let fluvial erosion amplify the network (Figure 5.6 left). We enable wind deflation with a small coefficient: while the wind is generally absent from these caves, this process accounts for any other internal disturbance due to gravity, internal flooding, and earthquakes.

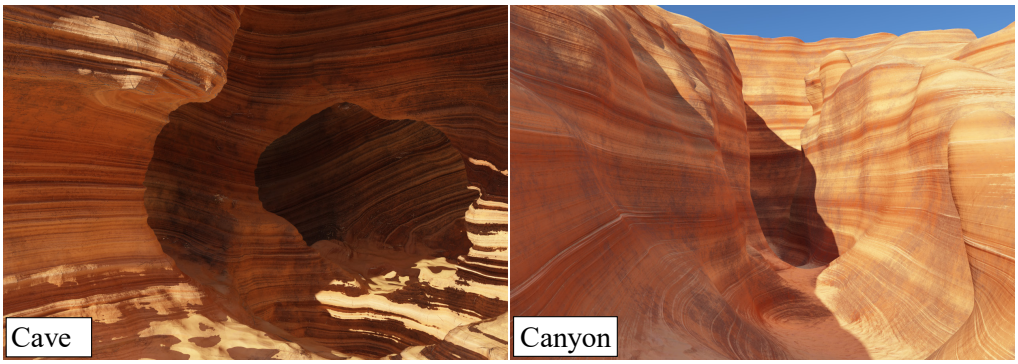


Figure 5.6: *Fluvial erosion: The cave network (left) was carved by fluvial erosion from an initial underground water path and high discharge value at the source. Setting this path on the surface results in the formation of a canyon (right).*

5.5.2 Scale, Performance and Interactive User Control

Scale and Performance To enable interactive simulation and control, our simulations employed 1-3 million particles within a 128^3 grid. On average, a single frame took around 200ms, with individual components: MPM, wind erosion and 3D flow routing (fluvial erosion + deposition) requiring 20ms, 5ms and 180ms, respectively. By increasing the particle count and grid resolution, our simulation can generate sandstone structures with greater detail (Figure 5.7). This improvement, however, incurs additional computational time and GPU memory requirements, as shown in Figure 5.8.

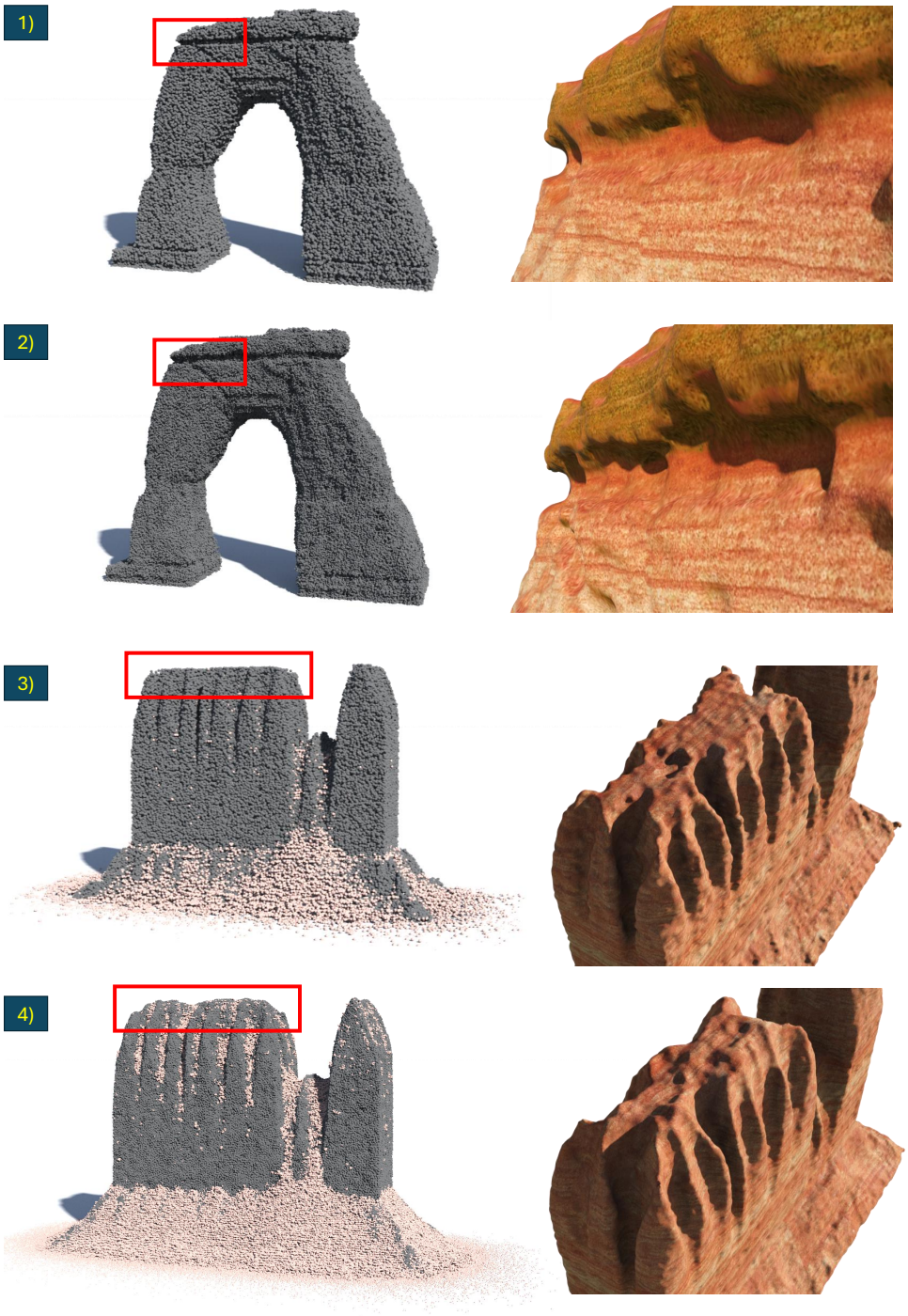


Figure 5.7: *Increasing the particle count and grid resolution leads to structures with greater detail. For the arch case, we compare 1 million particles on a 128^3 grid (row 1) and 9 million particles on a 256^3 grid (row 2). The higher-resolution simulation exhibits greater detail with more pronounced pillar-like formations in the weaker layers on the top, which are characteristic of sandstone structures. For the butte, comparisons between 1.5 million particles on a 128^3 grid (row 3) and 27 million particles on a 320^3 grid (row 4) reveal enhanced surface irregularity and a more natural appearance.*

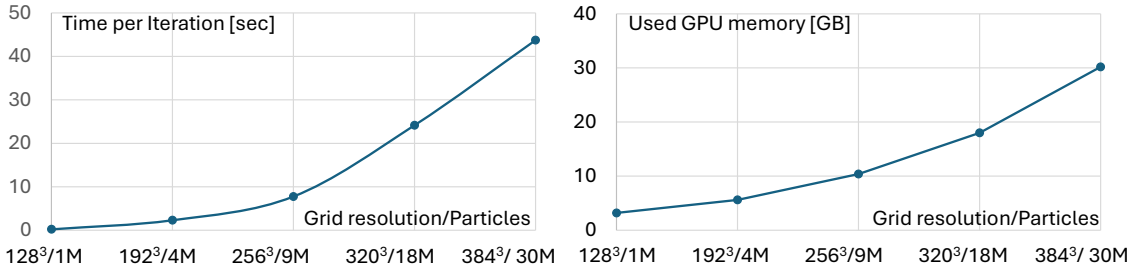


Figure 5.8: *GPU memory usage and computational time for different particle numbers and scaled grid size.*

Visualization and Interaction Our method allows users to interactively visualize the current state during the simulation stage and dynamically adjust the direction and speed of the global wind. At any point in the simulation, users may also consider the current shape as part of a new input, enabling its editing with the brush tool and adding new shapes. At around 200 ms per frame, this allows for interactive editing sessions, as illustrated in Figure 5.9.



Figure 5.9: *User interaction: A lightly eroded rock cube (left) is marked at three locations by the user and eroded. Then the user added two cylinders on the top and eroded again.*

The user can paint layers of *vegetation* as shown in Figure 5.10. The plants adjust the erodibility and provide a shielding effect of erosion of plants [Pawlik et al., 2023].

5.5.3 Comparisons and Ablation Studies

5.5.3.1 Comparisons

We compare our results with lab experiments from [Bruthans et al., 2014], where weakly cemented sand was eroded by water for several months until an arch emerges. Figure 5.11 shows that *Arenite* can generate a similar arch under similar conditions.

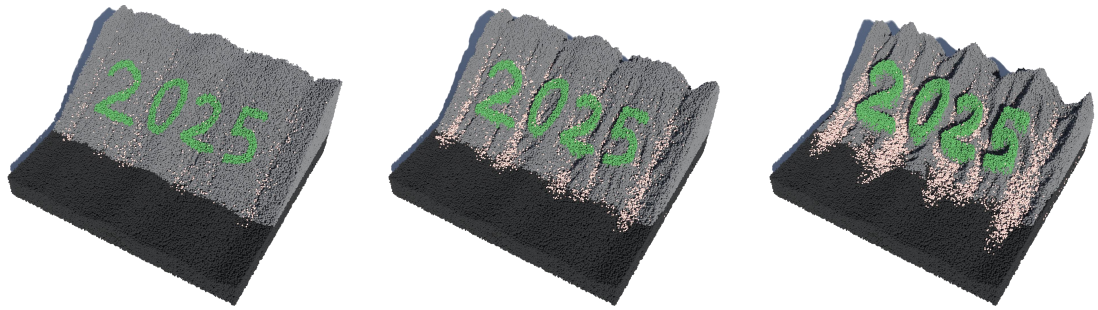


Figure 5.10: *Plants: The plants hold soil and protect sandstone from wind and water erosion, playing an important role in shaping channels for sandstone. New plants grow at the deposit areas at the foot of the hill.*

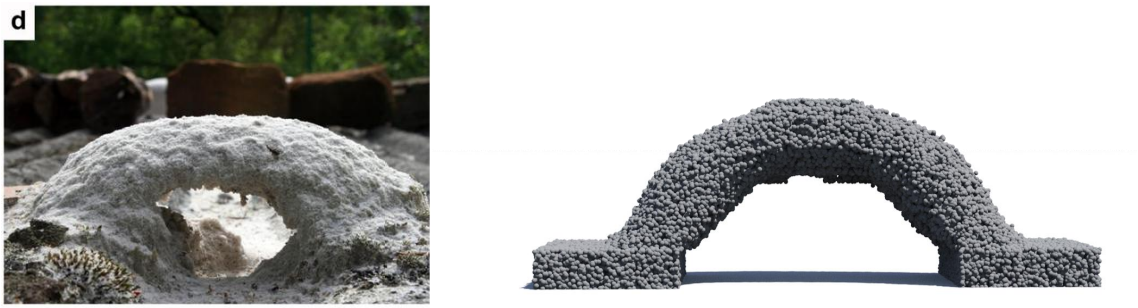


Figure 5.11: *Comparison to [Bruthans et al., 2014] (left). The contours of both shapes show that Arenite can generate shapes similar to those formed under natural erosion.*

5.5.3.2 Ablation Studies

We show the impact of our different phenomena and parameters on the formation of sandstone structures.

Erosions Figure 5.12 shows that both wind and fluvial erosion are critical to forming hoodoos. Their pillars are separated by fluvial erosion, but their characteristic shape is directed by wind erosion. When the fluvial erosion is missing, the pillars are not separated (left). When wind erosion is absent, the pillars do not have their typical form.

Fluvial erosion critical stress Another threshold is the τ_c variable in Eq. 5.4 that controls the erosion rate. Figure 5.13 shows that while a high threshold keeps a significant part of the shape unaffected and separates the erosion patterns, a low threshold results in joined ridges similar to large-scale terrain models [Cordonnier et al., 2016].

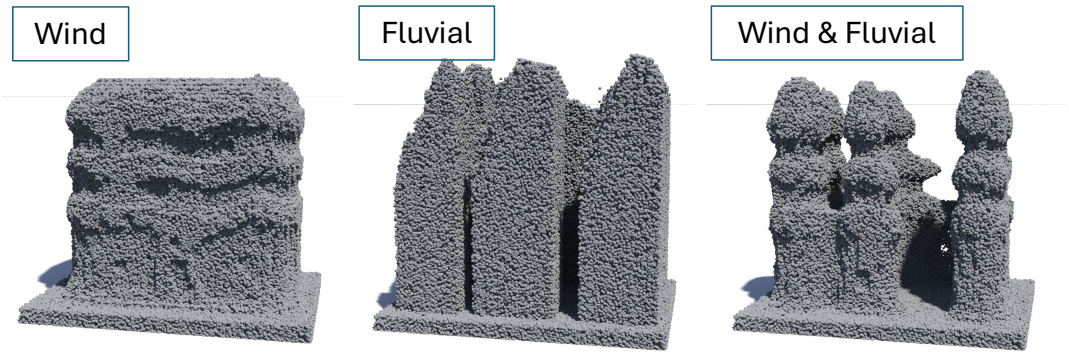


Figure 5.12: *Hoodoos are formed by fluvial and wind erosion (right). Without fluvial erosion (left), the initial sandstone cannot be separated into multiple pillars. Without the wind erosion (center), fluvial erosion cannot shape the distinctive stratification.*

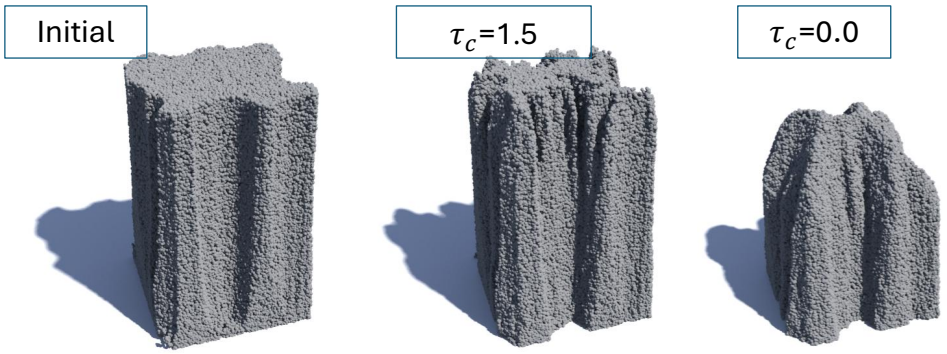


Figure 5.13: *Fluvial erosion critical stress: initial shape (left),* $\tau_c = 1.5$ *(middle), and* $\tau_c = 0.0$*. Small values lead to ridges and saddle points.*

Viability of eroded particles We assign an initial value for the viability of the deposit lower than the initial sandstone viability to account for the lack of cementing material, which makes them more prone to erosion. We show in Figure 5.14 the impact of the choice of the deposit viability. A higher viability (left) increases the stability of the deposit slopes and self-shielding of the structure against erosion. A lower value further increases the slope's spread.

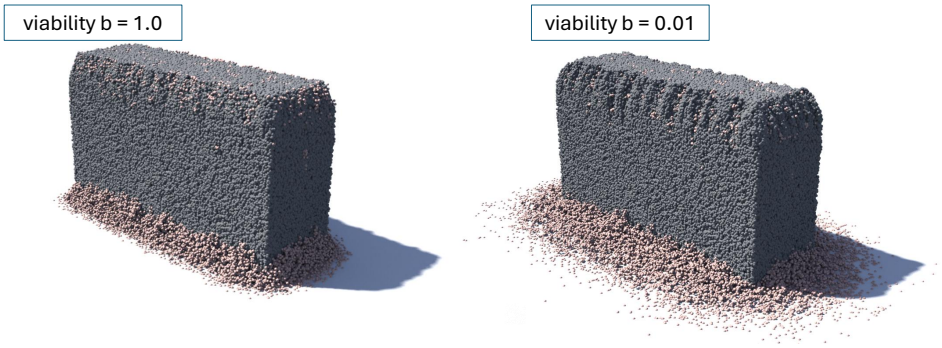


Figure 5.14: *Viability allows the user to control 1) spread of deposit 2) conformity to slope and 3) self-shielding for erosion.*

5.6 Conclusion and Future Work

This chapter extended the flow-network view developed in Chapter 4 beyond heightfields and into volumetric terrain. Sandstone landforms are a useful stress test for this idea because their characteristic structures, such as arches, alcoves, caves, and buttes, cannot be represented as a single-valued surface. *Arenite* addresses this by combining a particle/grid representation with stress-dependent erosion, wind erosion, fluvial erosion, and deposition. In this setting, particles store the evolving material state, while the grid supports both stress computation and the routing operations needed to move water and eroded material through a changing 3D shape.

The main contribution of the chapter is therefore not only a simulator for sandstone, but a generalization of terrain transport mechanisms to erodible volumes. We adapted flow and depression routing to exposed 3D surfaces, used the resulting discharge to drive fluvial incision, and introduced a deposition procedure that transports eroded particles toward stable locations. Coupled with fabric interlocking and wind erosion, these mechanisms allow recognizable sandstone structures to emerge from simple initial blocks with painted material properties, and vegetation masks. The results show that the same transport abstraction used for large heightfield landscapes can organize erosion and deposition in domains with overhangs or cavities.

The limitations of this approach also indicate where the abstraction becomes insufficient. *Arenite* captures the long-term visual effect of several dominant processes, but it does not model the full geological history of sandstone. In particular, crack propagation and block failure depend on pre-existing defects, anisotropy, sedimentary structure, and

tectonic history that are not represented in our material model. Vegetation is treated as a homogeneous protective layer, whereas real roots can both stabilize loose grains and mechanically or chemically weaken the rock. Finally, scaling the method to larger scenes remains limited by GPU memory and computation time, especially when higher particle counts and grid resolutions are required.

Future work could address these limits by modeling fracture mechanics, crack propagation, generalizing the method to different types of rocks, and accounting for the effect of vegetation at varying scales. Future work also includes the study of multi-scale and distributed algorithms to scale our method to larger scenes, and its coupling with large-scale terrain generation. More broadly, this chapter continues the thesis theme of using flow networks to make terrain generation more plausible and controllable. The next chapter follows the same principle in a different setting, where flow networks support an inverse process that reconstructs a complete terrain from a single image rather than simulating a landscape forward from physical processes.

Chapter 6

Pixels2Peaks: Converting Terrain Images to Heightmaps

6.1 Introduction

The previous chapters generated terrain by modeling the forward processes, from debris-flow erosion to volumetric sandstone erosion. This chapter turns to the inverse problem. Instead of starting from an initial terrain and evolving it through simulation, we start from a single image and reconstruct a complete landscape that remains consistent with geomorphological and hydrological principles, while being guided by flow networks from Chapter 4.

Mountainous scenes form an iconic backdrop in many films, video games, and virtual worlds. A reference photograph or concept art image is often the first step toward such a landscape, but turning this 2D inspiration into a complete navigable 3D world remains a laborious manual process. An artist must interpret the visible terrain, infer hidden geometry beyond the image, model the matching 3D structure, and ensure that the final asset is both aesthetically pleasing and physically plausible. This workflow is time-consuming, requires significant expertise, and presents a major bottleneck in



Figure 6.1: *From an input image (left), Pixels2Peaks generates a complete heightmap of both the visible and occluded landforms (center), ready for offline rendering (right). [Input image from the GeoPose3K dataset [Brejcha and Čadík, 2017]]*

production pipelines.

We address the challenge of automating this process by reconstructing a complete and detailed 3D terrain heightmap, the standard representation used by artists, from a single 2D image (Figure 6.1). This provides an immediate base for the scene, allowing artists to focus on creative refinement rather than routine modeling. This represents an ill-posed inverse problem, since a single photograph captures only a partial, view-dependent snapshot of the terrain. Detail may be obscured by distance, but, more importantly, large sections are likely to be occluded from the viewpoint of the camera by folds in the landscape, particularly in mountainous terrain.

We believe that three consistency attributes are key to a viable solution for single-image terrain reconstruction:

1. Geomorphological consistency — the generated landscape follows the geomorphological features of the visible regions.

2. Hydrological consistency — the river network is locally and globally coherent, with unbroken channels that flow downhill to form proper drainage patterns.

3. View consistency — the reconstructed heightfield matches the reference image when rendered from the original viewpoint.

Existing methods fail to achieve one or more of these key attributes. Photogrammetric techniques for 3D terrain reconstruction, such as Structure from Motion and Multi-View Stereo [Westoby et al., 2012], rely on constraints across multiple viewpoints and cannot be directly applied in a single-image scenario. Early first-person terrain authoring systems [Cohen et al., 2000; Gain et al., 2009] provide view control from a 2D perspective, but typically rely on simpler inputs and do not guarantee realistic geomorphological or hydrological structure. Procedural modeling and simulation [Galin et al., 2019] can generate convincing terrains with physically plausible features, but they are difficult to constrain tightly enough to match a specific photograph while preserving global coherence.

Recent example-based and diffusion methods [Lochner et al., 2023; Hu et al., 2024] demonstrate that they can synthesize visually plausible terrains. However, visual plausibility alone is insufficient for editable terrain assets, as the generated heightfields may contain hydrological artifacts such as closed basins, often requiring post-processing or localized optimization such as depression breaching [Scott and Dodgson, 2021]. Some

recent data-driven approaches do report good hydrological consistency, but in our experiments this consistency did not extend to large regions. At the same time, general-purpose single-image 3D reconstruction models have advanced rapidly [Poole et al., 2022; Tencent, 2025], but they are trained primarily on object-centric data and do not enforce the physical constraints needed for natural landscapes.

The closest prior work to our setting is the image-to-terrain pipeline of Takahashi et al. [2022]. Their method first estimates a depth map and shadow-free color image from the camera view, rasterizes the result into a heightfield through a top-down orthographic projection, and then completes missing elevation and color using a modified CGAN. While the general progression from depth estimation, through projection, to generative model completion is similar, our architecture diverges significantly in other respects. We focus specifically on elevation, infer camera parameters rather than requiring them as input, support variable terrain resolution, and train on a curated 1m-resolution terrain dataset rather than 30m or 90m data. Most crucially, we incorporate specific modifications into our model to address geomorphological, hydrological, and view consistency.

For our process, we first infer a 3D pointmap and camera parameters from the input image, and project them into an incomplete heightmap. This heightmap provides constraints for a guided diffusion model that adds coherent detail to the visible areas and inpaints the occluded areas. Our diffusion model operates on shifting tiles to allow large resolution outputs.

We achieve visual plausibility by training our model on a large-scale curated dataset of scanned real-world terrains. We enforce geomorphological, hydrological and view consistency through guidance terms incorporated into the iterative denoising of the diffusion process. These guidance terms favor coherent drainage networks, align slope and curvature statistics between visible and occluded regions, and ensure the alignment of the terrain ridges to the silhouettes from the original image. Furthermore, the heightfield resolution is automatically adapted to the scale of the terrain captured by the input image.

Our results demonstrate the generation of diverse and plausible terrains that are consistent with the input photograph. Our approach for estimating camera tilt and terrain scale outperforms horizon-line detectors and monocular geometry models. Ablation studies validate our guidance terms; for instance, view guidance reduces the reconstruction error, while hydrological guidance successfully resolves common artifacts, like unrealistic water basins. Compared to a generalist image-to-3D model, our method robustly gener-

ates physically-consistent terrains. We further demonstrate our method's versatility by creating text-to-terrain and sketch-to-terrain pipelines. The entire process executes in approximately 5 minutes on a H100 GPU.

In summary, the key technical contributions of this chapter are:

1. A multi-stage inference process that converts a single source image into a terrain heightmap of appropriate resolution.

2. A terrain-aware diffusion model with guidance terms designed to promote hydrological, geomorphological, and view consistency.

3. A large 1.2 TB heightmap dataset curated to isolate landscapes with noticeable relief (the elevation difference, ranging from large hills to mountains) at 1m resolution.

6.2 Overview

In the absence of constraints, synthesizing a landscape from a single perspective photograph is inherently ill-posed. Therefore, we first constrain the process by introducing a set of consistency attributes that act to enforce the plausibility of the resulting terrain and its alignment with the input, as follows:

Geomorphological consistency: While natural landscapes can exhibit marked diversity, this is less the case for mountain ranges, where features are shaped by a common tectonic, climatic, and lithologic history. In our context, this means that occluded regions should be structurally consistent with visible regions in the input photograph. We choose to interpret this through a statistical alignment of slope and curvature between visible and occluded regions, as these metrics have proven to be an effective and efficient basis for landform characterization [Jasiewicz and Stepinski, 2013; Argudo et al., 2025].

Hydrological consistency: Large-scale mountain structures are subject to fluvial erosion, which prevents the formation of significant isolated valley structures, since these basins would otherwise fill to become massive lakes subject to rapid erosion through spillage. An overabundance of unrealistic pits and basins is a well-known shortcoming of localized terrain synthesis [Scott and Dodgson, 2021]. To enforce hydrological consistency, there should always be a continuous downhill flow path

for water to flow away from any given point [Martz and Garbrecht, 1998]. The literature primarily offers two operations to modify a heightfield to respect this property: filling and breaching. Since filling produces unrealistically flat regions, we chose to rely on breaching. This can be recast as a minimization of the breaching volume, which is the volume of terrain that must be removed to create a continuous flow path through the barriers that enclose pits.

View consistency: There should be a close pixel-wise match between the placement of silhouettes (ridge lines) in the input photograph and a rendering of the generated terrain with the same view parameters. Furthermore, no additional spurious silhouettes should be introduced. We chose silhouettes as our basis, in preference to measures of texture similarity, because of their perceptual salience [Todd, 2004].

Our goal is thus to synthesize a terrain heightmap from an input image while conforming to these constraints. We structure our approach (see Figure 6.2) into two primary phases: the construction of an incomplete terrain from a photograph, and subsequent completion of the full landscape through guided diffusion.

First, in order to generate a sparse heightmap covering the topographic areas visible in the photograph (Section 6.3), we estimate a sparse 3D pointmap and then segment out foreground objects and the background sky of the image. To resolve camera orientation and scene scale ambiguities, we query a Vision-Language Model (VLM) for semantic priors, such as plausible slope and elevation. These priors constrain the geometric projection, yielding a partial but appropriately scaled and oriented heightmap.

Second, we complete the occluded portions of the terrain (Section 6.4) by tiling the heightmap using a guided diffusion model. The tiling process is more expensive than a single inpainting step, but adapts well to a variety of sampling scales and heightmap resolutions. We guide the diffusion process with differentiable formulations of our three identified consistency constraints. Specifically, we minimize the difference in slope and curvature statistics between visible and occluded areas (geomorphological consistency), penalize sinks by comparing against a depression breached terrain (hydrological consistency), and prevent known silhouettes in the photograph from being altered (view consistency). Diffusion is initialized with multi-scale, view- and hydrologically-consistent noise and is followed by an iterative polishing loop to produce a complete, coherent, high-resolution output terrain.

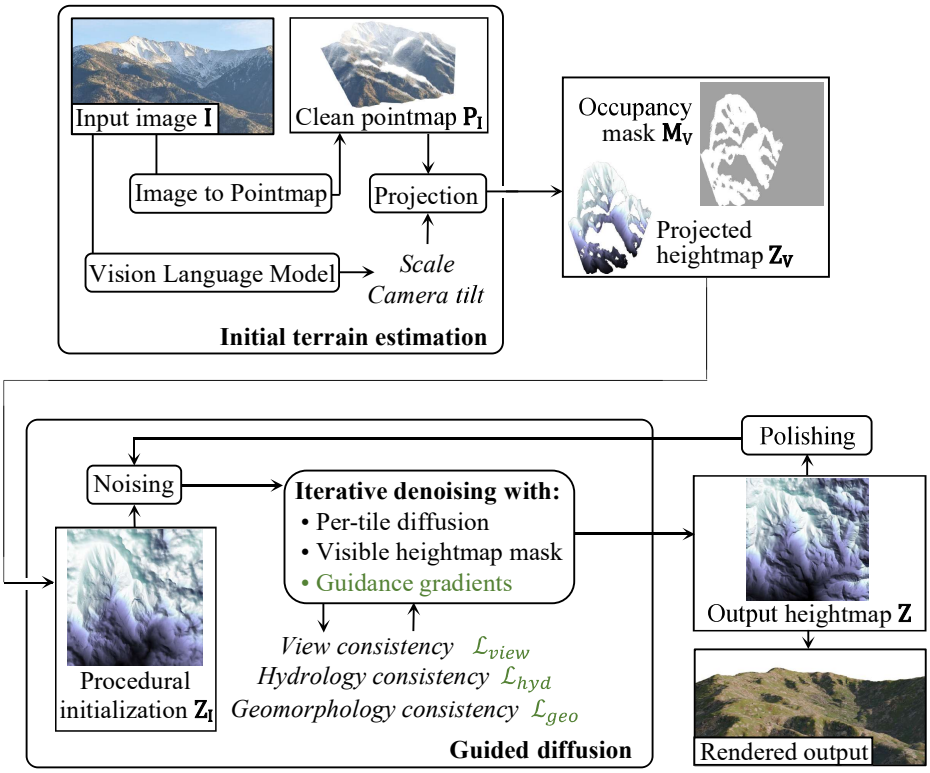


Figure 6.2: *Method overview. We start from an input photograph (©Pixabay), which is processed into a cleaned pointmap accompanied by scale and camera tilt parameters. This allows us to output an occupancy mask and an initial heightmap of the visible region. We complete the initial heightmap procedurally, and then refine it with a diffusion model, guided by hydrological, geomorphological, and view consistency. This leads to the final heightmap, after an optional polishing step.*

The efficacy of our terrain completion phase is heavily reliant on sufficient high-resolution real-world terrain data, since this is needed to train the underlying diffusion model. To address this, we employ a multi-stage curation pipeline (Section 6.5) to filter source U.S. Geological Survey data [U.S. Geological Survey, 2024] and assemble a 1.2TB collection of suitable terrain with relief.

6.3 Initial Terrain Estimation from a Single Image

In this section, we describe our method for deriving an initial, incomplete terrain heightmap covering the visible mountainous regions in a single input image (see Figure 6.3). We treat our photographic input as in-the-wild and make no assumptions as to

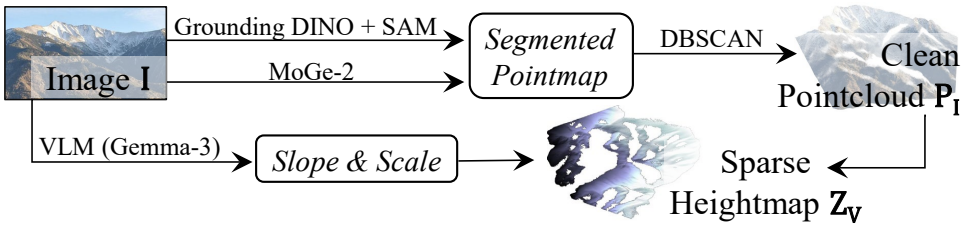


Figure 6.3: *Initial terrain estimation. Grounding DINO and SAM isolate mountains in the input image, while MoGe-2 extracts a pointmap. After DBSCAN cleans this pointmap, it is projected into a sparse heightmap using slope and scale parameters estimated by the Gemma-3 VLM.*

the availability of either intrinsic or extrinsic camera parameters, such as focal length and camera orientation. This first estimation phase must thus overcome two obstacles: the local inherent geometric ambiguity of individual pixels and the unknown global scale and orientation of the scene. The former is addressed by applying a pre-trained geometry estimation model (MoGe-2 [Wang et al., 2025]), which provides a 3D pointmap (a per-pixel 3D coordinate map). The latter is solved by querying a Vision-Language Model (VLM) [Google, 2025] to deduce approximate camera pitch and scene extent. This allows us to calibrate the base plane and derive a partial heightfield from the pointmap.

6.3.1 Sparse 3D Pointcloud Reconstruction

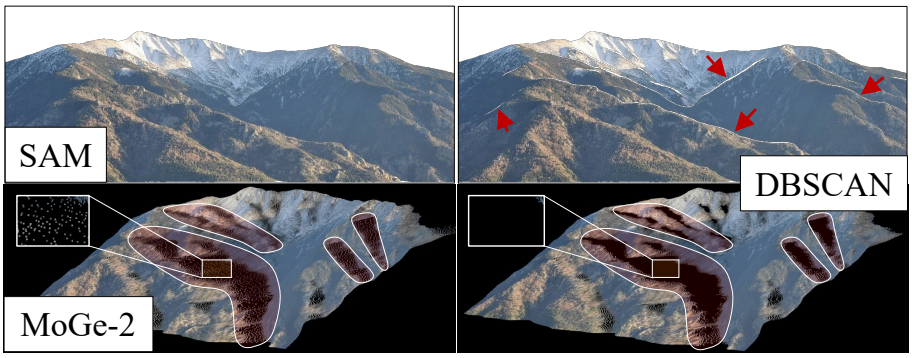


Figure 6.4: *Pointcloud extraction. The input image (as in Figure 6.2) is processed with Segment Anything Model (SAM) guided by Grounding-DINO to remove the foreground and sky (top-left). The input is then converted to a pointmap using MoGe-2 and masked by the SAM output, which, however, contains noise along the structure boundaries (noise highlighted; bottom-left). DBSCAN is applied to clean these noisy points (location indicated by red arrows in top-right and highlighted in bottom-right).*

Our pipeline accepts a single input image $\mathbf{I}$, which can take the form of a real-world photograph, landscape painting, or AI-generated scene (see Section 6.6). From this, our initial goal is to derive a 3D pointcloud covering the terrain visible in the image.

We first process the input image to mask out all foreground objects and background sky using the Segment Anything Model (SAM) [Kirillov et al., 2023]. This segmentation is additionally guided using bounding boxes provided by the Grounding-DINO model [Liu et al., 2023b] instructed with the prompt "all mountains" (Figure 6.4, top-left).

At the same time, we employ the MoGe-2 model [Wang et al., 2025] to map each image pixel to a 3D point. The resulting 3D pointmap obviates the need for explicit management of the camera intrinsics. This pointmap is masked out on the basis of the segmentation so that foreground and background pixel-points are excluded and only terrain coverage remains (Figure 6.4, bottom-left).

Finally, we filter the resulting pointmap using the DBSCAN [Ester et al., 1996] density-based clustering algorithm. This step is necessary to remove sparse noisy artifacts, a known issue of the pointmap prediction model that arises along object boundaries due to noisy training data [Wang et al., 2025]. As shown in Figure 6.4 (right), this filtering discards outliers that would otherwise corrupt the final heightmap, yielding the clean pointcloud $\mathbf{P_I}$.

6.3.2 Projection onto an Incomplete Heightmap

The standard toolchain in digital terrain processing relies heavily on a heightmap representation and so we must convert our cleaned 3D pointcloud to this format. For this purpose, we define a sparse heightmap $\mathbf{Z_V}$ on a 2D base plane with an accompanying binary mask $\mathbf{M_V}$ to indicate which heightmap pixels contain projected points from $\mathbf{P_I}$. This heightmap is characterized by its resolution (the number of elevation samples), the sampling density (the separation between samples in *meters / pixel*), and, consequently, the scene extent (the landscape area covered by $\mathbf{Z_V}$ in *meters*). Unfortunately, this conversion requires knowledge of the camera's extrinsic parameters in order to orient the 3D point cloud. One option is to make use of a horizon line detector. However, it is more effective (as demonstrated in Section 6.6.2) to exploit landscape-specific priors.

The problem can be divided into two parts: determining camera orientation and the scene scale. We begin by reducing the degrees of orientation freedom by assuming no camera roll. In practice, photographers tend to naturally align their camera with

the horizon. This is borne out by an examination of the GeoPose3K dataset [Brejcha and Čadík, 2017], comprising ∼3k in-the-wild terrain photographs with known camera parameters, where the roll variation was found to be minimal ($\mu = 0.01°$, $\sigma = 1.62°$) compared to the wider distribution in pitch ($\mu = -0.04°$, $\sigma = 8.82°$).

Unfortunately, even a scalar estimation of the camera's pitch (or tilt), critical as it is to correctly orient the ground plane, remains ill-conditioned. Standard approaches, such as detecting vanishing points and horizon lines, are ill-suited to mountainous terrain lacking geometric priors like architectural straight lines.

Instead, we address this ambiguity by leveraging the semantic text-based interpretation afforded by Vision-Language Models (VLMs), which combine the best aspects of computer-vision encoders and large language models. We posit that a VLM can infer plausible physical properties of a depicted scene, which can then serve as priors to constrain the geometric reconstruction. In particular, we seek estimates for the mean slope s_{VLM} and elevation range $[z_{\min}, z_{\max}]_{\text{VLM}}$ to provide targets for the unknown pitch and scene scale, respectively. To obtain these priors, we query the Gemma-3 VLM [Google, 2025] with the input image $\mathbf{I}$ and a structured text prompt requesting estimates of mean slope and minimum and maximum elevation across all visible mountains. To elicit a more reasoned and stable response, we employ Chain-of-Thought reasoning [Wei et al., 2022], prompted by an instruction to think step-by-step [Kojima et al., 2022]. To enhance robustness and mitigate prompt sensitivity, we then apply self-consistency [Wang et al., 2022] by taking an aggregate over several prompt variations, leading to final estimates for s_{VLM}, and $[z_{\min}, z_{\max}]_{\text{VLM}}$.

The VLM-estimate of slope s_{VLM} provides a prior for resolving the camera's optimal pitch angle, θ^*. To find θ^*, we undertake a grid search over a discretized range of candidate pitch angles $\{\theta_i\}$. For each θ_i, the point cloud $\mathbf{P_I}$ is rotated by the corresponding rotation matrix $\mathbf{R}(\theta_i)$ and orthographically projected onto the global ground plane to generate a temporary heightmap $\mathbf{Z}_i$ and occupancy mask $\mathbf{M}_i$. Finally, the optimal pitch θ^* is obtained by minimizing the absolute difference between the average rotation-invariant slope s_i (Chapter 3) of the unmasked pixels in $\mathbf{Z}_i$ and the VLM prediction:

$$\theta^* = \arg\min_{\theta_i} |s_i - s_{\text{VLM}}| \tag{6.1}$$

With the optimal pitch θ^* and orientation $\mathbf{R}(\theta^*)$ now established, what remains is a fit of the scene scale. This is achieved by uniformly scaling and translating the pointcloud associated with θ^*, so that it fits the elevation range $[z_{\min}, z_{\max}]_{\text{VLM}}$ predicted by the

VLM. The scaled pointcloud is then projected onto the heightmap using a user-selected sampling density dx.

This process yields a final output for this phase: a *visible heightmap* consisting of an incomplete heightfield $\mathbf{Z_V}$ at a user-specified sampling density dx, and a corresponding binary occupancy mask $\mathbf{M_V}$ indicating the cells containing valid elevation data.

6.4 Guided Diffusion for Terrain Completion

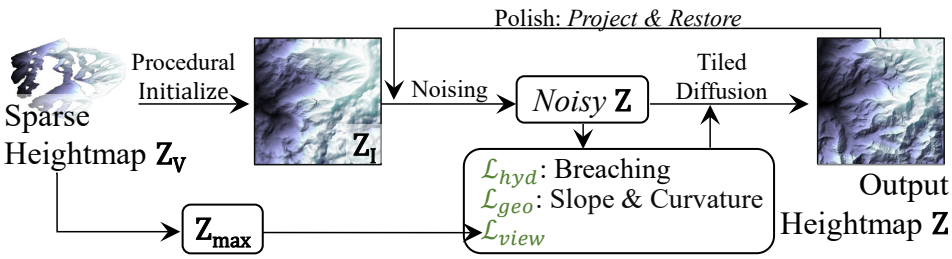


Figure 6.5: *Guided Diffusion Overview. The sparse heightmap is procedurally initialized and noised. Tiled reverse diffusion is then applied, guided by hydrological, geomorphological, and view consistency, followed by a final polishing step to produce the output heightmap.*

Having derived an initial heightmap for the visible terrain, our next task is to synthesize the occluded regions in a physically and visually consistent manner (see Figure 6.5). We frame this as a completion problem, addressed with a Diffusion Model adapted for inpainting (Section 6.4.1) and trained on a curated terrain dataset (Section 6.5). The core of our method is a novel guidance mechanism that steers the reverse diffusion process toward hydrological, geomorphological, and view consistency (Section 6.4.2). For this guided process to converge effectively, we employ a procedural initialization scheme (Section 6.4.3). We also use a resolution-adaptive tiled approach (Section 6.4.4) to ensure scalability to large terrains. Finally, an additional iterative polishing step (Section 6.4.5) refines the results to achieve high-quality outputs.

6.4.1 Terrain Completion

Our method builds upon Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020]. These combine a forward diffusion process, which gradually corrupts data by applying noise, with a learned reverse denoising process.

In our case, the forward process $q(\mathbf{Z}_0, t)$ successively injects Gaussian noise into an

initially clean heightmap data sample $\mathbf{Z}_0$ over time $t \in [0, 1]$. Note that we can directly sample a noisy version $\mathbf{Z}_t$ for any arbitrary timestep t from $\mathbf{Z}_0$ in closed form:

$$\mathbf{Z}_t = q(\mathbf{Z}_0, t) = \sqrt{\bar{\alpha}_t}\mathbf{Z}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \tag{6.2}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a random Gaussian noise sample and the schedule $\bar{\alpha}_t$ is a monotonically decreasing function that controls the noise variance over time, such that $t = 0$ represents the clean data and $t = 1$ represents pure noise.

The reverse process learns to recover structure through denoising. It relies on a U-Net [Ronneberger et al., 2015], $\boldsymbol{\epsilon}_\theta(\mathbf{Z}_t, t)$, which is trained to predict the noise component $\boldsymbol{\epsilon}$ in a noisy sample $\mathbf{Z}_t$. During reverse diffusion, the model iteratively refines a noisy heightfield $\mathbf{Z}_t$ to produce a cleaner version $\mathbf{Z}_{t-\Delta t}$, culminating in $\mathbf{Z}_0$.

This is designed as a completion process, so the original visible regions in $\mathbf{Z_V}$ must be recovered intact. To achieve this, we introduce the correctly noised version of the visible heightmap $\mathbf{Z_V}$ masked according to the occupancy mask $\mathbf{M_V}$, following the RePaint [Lugmayr et al., 2022] strategy:

$$\mathbf{Z}_t \leftarrow (1 - \mathbf{M_V}) \odot \mathbf{Z}_t + \mathbf{M_V} \odot q(\mathbf{Z_V}, t) \tag{6.3}$$

This step anchors the regions visible in the photograph, while allowing the rest of the terrain to be synthesized freely. This reinjection is disabled during the final denoising steps ($t < 0.05$) to enable smooth blending between known and synthesized regions and refine away any inaccuracies in the initial heightmap estimation.

6.4.2 Guided Reverse Process

The core of our contribution is a mechanism for steering the reverse diffusion process toward hydrological, geomorphological, and view consistency. This is built on the foundation of gradient-based guidance [Dhariwal and Nichol, 2021], in which a guidance term enforces desired image properties.

First, we use the network's noise prediction $\boldsymbol{\epsilon}_\theta(\mathbf{Z}_t, t)$ to estimate the expected final heightmap $\hat{\mathbf{Z}}_0$ [Ho et al., 2020]:

$$\hat{\mathbf{Z}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{Z}_t - \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{Z}_t, t)\right) \tag{6.4}$$

From the predicted heightmap $\hat{\mathbf{Z}}_0$, we extract three penalty terms that measure the degree of hydrological $\mathcal{L}_{\text{hyd}}$, geomorphological $\mathcal{L}_{\text{geo}}$, and view $\mathcal{L}_{\text{view}}$ inconsistency. These are

combined into a single weighted composite guidance loss $\mathcal{L}_{\text{guide}}$, as follows:

$$\mathcal{L}_{\text{guide}} = \lambda_{\text{hyd}}\mathcal{L}_{\text{hyd}} + \lambda_{\text{geo}}\mathcal{L}_{\text{geo}} + \lambda_{\text{view}}\mathcal{L}_{\text{view}} \tag{6.5}$$

where λ_{hyd}, λ_{geo}, and λ_{view} are scalar weights controlling the contribution of each respective consistency loss. The gradient of the loss with respect to the current noisy state $\mathbf{Z}_t$ is then used to perturb the original noise prediction:

$$\boldsymbol{\epsilon}'_t = \boldsymbol{\epsilon}_\theta(\mathbf{Z}_t, t) - s \cdot \sqrt{1-\bar{\alpha}_t} \cdot \nabla_{\mathbf{Z}_t}\mathcal{L}_{\text{guide}} \tag{6.6}$$

where s is the guidance scale. The modified noise estimate $\boldsymbol{\epsilon}'_t$ is then used in the scheduler's denoising step to compute $\mathbf{Z}_{t-\Delta t}$. We now define the individual consistency guidance terms.

Hydrological Guidance This loss term improves flow consistency across the generated surface by penalizing depressions, which are local minima or sinks where water would pool erroneously. The loss uses a novel depression-breaching function H, which algorithmically carves drainage channels from these sinks to lower ground, ensuring every point on the surface has a continuous downhill path. This process creates a breached or hydrologically conditioned surface, essential for realistic erosion channels.

Formally, hydrological loss is defined as the difference between a smoothed and downsampled version of the prediction, with and without breaching:

$$\mathcal{L}_{\text{hyd}} = \mathbb{E}[(D_\sigma(\hat{\mathbf{Z}}_0) - \text{sg}[H(D_\sigma(\hat{\mathbf{Z}}_0))])^2] \tag{6.7}$$

Here, D_σ is a Gaussian smoothing and downsampling operator. Smoothing removes high-frequency noise that can manifest as spurious depressions during denoising, while downsampling lets breaching act at the scale of valleys, carving wider instead of artificially steep ones. The stop-gradient operator $\text{sg}[\cdot]$ detaches the breached output from the computational graph, effectively treating it as a fixed target that is not subject to backpropagation of its gradients.

Since depression breaching is performed on every iteration, it can potentially degrade the performance of the denoising process. To overcome this hurdle, we extend the *FastFlow* framework (Chapter 4) to develop a novel GPU-parallel implementation of depression breaching H (see Algorithm 9) that uses pointer jumping to reduce the number of iterations from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$. The *FastFlow* framework provides a set

of directed flow trees over the heightfield, where pointers are directed from a cell (the source) to a lower neighbor (the recipient). Depression breaching ensures that a recipient is always lower than its source by some ϵ. This is performed in parallel through a pointer jumping (or doubling) scheme.

Algorithm 9: GPU-Breach

Input : Elevation grid $\mathbf{Z}$, Slope differential ϵ
Output : Breached elevation grid $\mathbf{Z}$

Compute *FastFlow* recipients for all grid cells
$N \leftarrow$ number of grid cells in $\mathbf{Z}$
for $i \leftarrow 1$ **to** $\log_2(N)$ **do**
  foreach *grid cell* c *in* $\mathbf{Z}$ *in parallel* **do**
    $\mathbf{Z}[\text{recipient of } c] \leftarrow \text{atomic-min}(\mathbf{Z}[c] - 2^{i-1}\epsilon, \mathbf{Z}[\text{recipient of } c])$
    recipient of $c \leftarrow$ recipient of recipient of c

Geomorphological Guidance Our goal with this loss term is to encourage geomorphological stationarity, in which the geomorphological characteristics of the visible (known) regions are replicated in the occluded (unknown) regions, thus creating a unified seamless landscape. To achieve this, our geomorphological guidance loss $\mathcal{L}_{\text{geo}}$ minimizes the difference between the mean slope and curvature of the known and unknown regions, a concept analogous to style loss in neural style transfer [Gatys et al., 2015] and weak stationarity in texture synthesis [Efros and Leung, 1999]. While numerous terrain descriptors are available [Argudo et al., 2025], we chose slope and curvature as being among the most fundamental [Moore et al., 1991].

Our geomorphological loss is derived from the same smoothed and downsampled prediction $D_\sigma(\hat{\mathbf{Z}}_0)$ used for hydrological guidance. Now, let $S(\cdot)$ and $C(\cdot)$ be operators that compute local slope and curvature (where $C(\cdot)$ computes the Laplacian using a standard 5-point stencil, which serves as a linear approximation of mean curvature), and let $\mathbf{M}_\sigma$ be the binary mask identifying the known visible regions at the downsampled resolution. The loss $\mathcal{L}_{\text{geo}}$ is then defined as:

$$\begin{aligned}\mathcal{L}_{\text{geo}} = \lambda_{\text{s}} \left(\mathbb{E}_{1-\mathbf{M}_\sigma}[S(D_\sigma(\hat{\mathbf{Z}}_0))] - \text{sg}[\mathbb{E}_{\mathbf{M}_\sigma}[S(D_\sigma(\hat{\mathbf{Z}}_0))]] \right)^2 + \\ \lambda_{\text{c}} \left(\mathbb{E}_{1-\mathbf{M}_\sigma}[C(D_\sigma(\hat{\mathbf{Z}}_0))] - \text{sg}[\mathbb{E}_{\mathbf{M}_\sigma}[C(D_\sigma(\hat{\mathbf{Z}}_0))]] \right)^2\end{aligned} \tag{6.8}$$

where $\mathbb{E}_{\mathbf{M}_\sigma}[\cdot]$ and $\mathbb{E}_{1-\mathbf{M}_\sigma}[\cdot]$ denote the expectation over the known and unknown regions, respectively. As before, the stop-gradient operator $\mathrm{sg}(\cdot)$ ensures that the guidance only affects the unknown regions. The scalars λ_s and λ_c control the relative weighting of the slope and curvature terms.

View Guidance This loss term mitigates against any of the newly synthesized terrain obscuring the known terrain when viewed from the perspective of the original camera. Consequently, it also preserves the salience of the internal and external landscape silhouettes in the input image. We enforce this constraint by pre-computing a per-cell upper elevation bound over the visible heightmap $\mathbf{Z_V}$. This takes the form of a bounding heightmap $\mathbf{Z}_{\max}$, whose entries prescribe the maximum allowable elevation for cells in the regions being inferred. If an elevation in the inferred region breaches this cap, it will become visible from the camera perspective. It is computed (see Algorithm 10 and Figure 6.6) by casting rays from the camera viewpoint and matching elevations in the unknown regions to those of surface skimming rays. The resulting bounding heightmap $\mathbf{Z}_{\max}$ is used as a soft constraint via the view-guidance loss:

$$\mathcal{L}_{\text{view}} = \mathbb{E}_{1-\mathbf{M}}[(\text{ReLU}(\hat{\mathbf{Z}}_0 - \text{sg}[\mathbf{Z}_{\max}]))^2] \tag{6.9}$$

This loss penalizes values in $\hat{\mathbf{Z}}_0$ that exceed $\mathbf{Z}_{\max}$. The one-sided ReLU function ensures that penalization only applies for elevations causing an occlusion (i.e., positive values), while the stop-gradient on $\mathbf{Z}_{\max}$ prevents backpropagation.

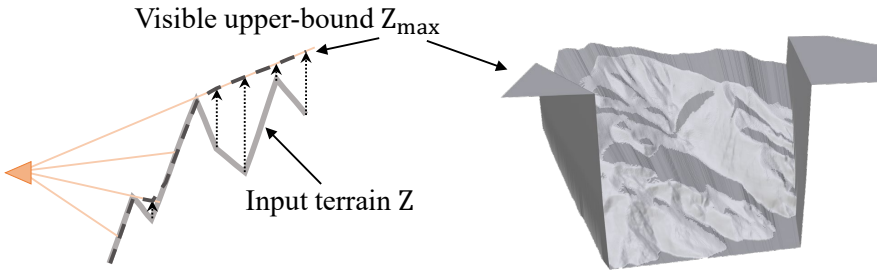


Figure 6.6: *We estimate the maximum terrain elevation envelope $\mathbf{Z}_{max}$ that preserves view consistency, by casting rays into the scene and intersecting them with the terrain. Hidden parts of the terrain are projected to the elevation of the lowest skimming ray immediately above them. We show a 3D view of $\mathbf{Z}_{max}$, with light gray for the visible elevations, and darker gray (dashed in 2D) for the projections of the hidden regions onto the envelope.*

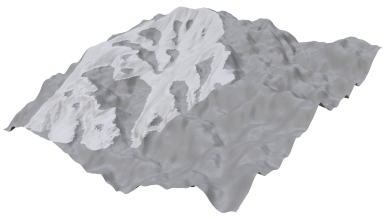
Algorithm 10: Ray-trace View Bound

Input : Elevation grid $\mathbf{Z}$, Camera parameters $\mathbf{V}$
Output: View bound $\mathbf{Z}_{\text{max}}$

$\mathcal{R} \leftarrow \text{Get-Rays}(\mathbf{V})$
$\mathbf{Z}_{\text{max}} \leftarrow \infty$
while *any ray in* $\mathcal{R}$ *exists* **do**
  foreach *ray* $r \in \mathcal{R}$ *in parallel* **do**
    $r_{xyz} \leftarrow r_{xyz} + \delta r_{direction}$
    if r_x, r_y *out of grid* $\mathbf{Z}$ *bounds* **then**
      $\text{Terminate-Ray}(r, \mathcal{R})$
    else if $r_z \leq \mathbf{Z}[\lfloor r_x \rfloor, \lfloor r_y \rfloor]$ **then**
      $\text{Terminate-Ray}(r, \mathcal{R})$ ▷ Ray hits $\mathbf{Z}$
    else
      $\mathbf{Z}_{\text{max}}[\lfloor r_x \rfloor, \lfloor r_y \rfloor] \leftarrow \text{atomic-min}(\mathbf{Z}_{\text{max}}[\lfloor r_x \rfloor, \lfloor r_y \rfloor], r_z)$

6.4.3 Initialization for Guided Inpainting

In practice, our guidance mechanism struggles to produce plausible results when starting from a pure noise state ($t = 1$), for two reasons. First, the initial noise represents a highly variable initial topography that is usually too far from the input view to be correctable via the view consistency term. Second, the breaching operation within the hydrological consistency term requires the terrain to exhibit at least some structure and has little meaning for pure noise.



We address this by starting the reverse diffusion process not from pure noise at $t = 1$, corresponding to a fully learned approach, but from an intermediate timestep $t \approx 0.5$. The initial state at this timestep is based on noising a procedural initial guess (dark gray in the inset figure). This initial guess $\mathbf{Z}_{\text{I}}$ is constructed (see Algorithm 11) using an iterative procedural noise process, inspired by multi-scale value noise [Ebert et al., 2002], with added hydrological breaching and view clamping.

From a coarse starting scale, we add uniform noise to the unmasked (occluded) regions ($1 - \mathbf{M}_{\mathbf{V}}$), smooth with Laplacian diffusion, correct for hydrological consistency with breaching, clamp to the maximum elevation allowed for by view consistency, and finally

upsample and repeat the process at the next finer scale. As observed by Scott and Dodgson [2021], applying breaching at every scale also serves to prevent the formation of unnaturally narrow corrective valleys. The pass of Laplacian diffusion is used to reduce high-frequency artifacts and sharp discontinuities.

Algorithm 11: Noise Inpainting for Diffusion Initialization

Input : Elevation grid $\mathbf{Z_V}$, Mask $\mathbf{M_V}$, View bound $\mathbf{Z}_{\text{max}}$, Num octaves $\mathcal{O}$
Output : Inpainted elevation $\mathbf{Z}_{\text{I}}$

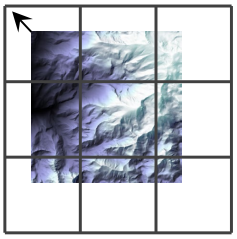
Let $\mathcal{U}_k$, $\mathcal{D}_k$ be up/down-sampling operators by factor k
Function $\text{Constrain}(\mathbf{Z}, \mathbf{M}, \mathbf{Z}_{max})$
    $\mathbf{Z} \leftarrow \text{Smooth}(\mathbf{Z})$ ▷ *e.g.*, Laplacian Diffusion
    ▷ Breach unmasked regions
    $\mathbf{Z}' \leftarrow (1 - \mathbf{M}) \odot \mathbf{Z} + \mathbf{M} \odot \infty$
    $\mathbf{Z} \leftarrow (1 - \mathbf{M}) \odot \text{GPU-Breach}(\mathbf{Z}') + \mathbf{M} \odot \mathbf{Z}$
    return $\text{Minimum}(\mathbf{Z}, \mathbf{Z}_{\text{max}})$ ▷ View constraint
▷ Initialize at the coarsest level
$\mathbf{Z}_{\text{I}} \leftarrow \text{Constrain}(\mathcal{D}_{2^{\mathcal{O}}}(\mathbf{Z_V}, \mathbf{M_V}, \mathbf{Z}_{\text{max}}))$
▷ Iterate from coarse to fine
for $k \leftarrow \mathcal{O} - 1$ **to** 0 **do**
    $\mathbf{Z}^k, \mathbf{M}^k, \mathbf{Z}^k_{\text{max}} \leftarrow \mathcal{D}_{2^k}(\mathbf{Z_V}, \mathbf{M_V}, \mathbf{Z}_{\text{max}})$
    $\boldsymbol{\epsilon} \leftarrow$ scale-dependent noise
    $\mathbf{Z}_{\text{up}} \leftarrow \mathcal{U}_2(\mathbf{Z}_{\text{I}} + \boldsymbol{\epsilon})$
    $\mathbf{Z}_{\text{merged}} \leftarrow (1 - \mathbf{M}^k) \odot \mathbf{Z}_{\text{up}} + \mathbf{M}^k \odot \mathbf{Z}^k$
    $\mathbf{Z}_{\text{I}} \leftarrow \text{Constrain}(\mathbf{Z}_{\text{merged}}, \mathbf{M}^k, \mathbf{Z}^k_{\text{max}})$

This process produces a procedural multi-scale heightfield $\mathbf{Z}_{\text{I}}$. Following a process akin to SDEdit [Meng et al., 2021], we then derive a starting state for our diffusion model by adding Gaussian noise to $\mathbf{Z}_{\text{I}}$, consistent with the noise schedule at timestep $t \approx 0.5$. This noisy state then serves as the starting point for the guided denoising process.

6.4.4 Resolution Adaptive Tiled Diffusion

Depending on the scale of the scene, landscape images may range in extent anywhere from a few hundred meters in the case of small hills shot in close up to tens of kilometers for a panorama of a mountain range. With a fixed sampling density (*e.g.*, 4 m/pixel), this variation in scene extent results in a wide range of output heightmap resolutions.

To accommodate this variability, we adopt a tiled denoising approach, based on the SpotDiffusion sliding window strategy [Frolov et al., 2025].



In general terms, tiled denoising operates by performing multiple diffusion processes in parallel over different portions of the heightmap and then combining the outputs. The details differ as to how tiles are positioned and blended. We choose SpotDiffusion [Frolov et al., 2025], which places tiles contiguously without overlap and removes seams by performing a global random translation of the tile set after each iteration, because it balances quality and computation cost. The original work of SpotDiffusion is targeted at panoramic images with a long strip of tiles that are shifted horizontally between iterations. We extend this to the 2D domain and perform random shifts along both the x- and y-axes. The inset illustrates this with a grid of parallel denoising processes and an arrow indicating the terrain offset toward the top-left.

We further improve scalability by automatically selecting the sampling density dx based on the scene extent, as predicted in Section 6.3. For instance, rather than denoising a 16km^2 terrain at $dx = 1$m, which produces a heightfield with 16000^2 grid cells, we can denoise it with a model trained at $dx = 16$m, resulting in a 1000^2 grid, and significantly reduce computation cost. To enable this, we train a collection of diffusion models to produce 512^2 heightfields at $dx = 1$m, 2m, 4m, 8m, and 16m by cropping and downsampling the tiles in our 1m dataset (Section 6.5).

6.4.5 Final Polishing

At this juncture, our model synthesizes a terrain that is view-, geomorphologically- and hydrologically-consistent. Nevertheless, some spurious local inconsistencies remain (see Figure 6.16d). We therefore incorporate a final iterative super-resolution and refinement step to correct these remaining issues and elevate the quality of the result. Unlike the main diffusion phase, which incorporates consistency as soft guidance terms, this polishing step enforces compliance through hard projection. This iterative refinement loop is documented in Algorithm 12, and summarized below.

To begin, the heightmap is upsampled via bicubic interpolation to reduce the sampling interval dx. Each subsequent refinement iteration consists of two phases:

1. **Constraint Projection:** The heightmap $\mathbf{Z}$ is projected back onto the space of

valid solutions. The view constraint is enforced via element-wise clamping against the bounding heightmap $\mathbf{Z}_{\text{max}}$, while the flow constraint is enforced by breaching after downsampling (as before preventing the formation of unnatural steep-sided valleys).

2. **Realism Restoration:** To reduce artifacts arising from the projection, an SDEdit-style diffusion step is applied to the projected output after noising it as the initial state. Starting from a timestep close to completion (initially ~ 0.05) facilitates a short trajectory, sufficient to smooth artifacts and restore high-frequency details without significantly altering the now constraint-compliant structure. On each iteration, the timestep is brought successively closer to completion.

This strategy successively refines the heightmap, improving constraint adherence while enhancing realism.

Algorithm 12: Iterative Heightmap Polishing

Input : Elevation grid $\mathbf{Z}$, View bound $\mathbf{Z}_{\text{max}}$, Upfactor $\mathbf{u}$
Output : Polished heightmap $\mathbf{Z}$

Let $\mathcal{U}_k$, $\mathcal{D}_k$ be up/down-sampling operators by fixed factor k
$\mathbf{Z} \leftarrow \mathcal{U}_{\mathbf{u}}(\mathbf{Z})$
$\mathcal{T} \leftarrow [t_1, t_2, \ldots, t_n] \triangleright\ t_i > t_{i+1},\ t_1 \sim 0.05,\ n \sim 3$
for $t \in \mathcal{T}$ **do**
    $\mathbf{Z} \leftarrow \mathcal{U}_k(\text{GPU-Breach}(\mathcal{D}_k(\mathbf{Z})))$
    $\mathbf{Z} \leftarrow \text{Minimum}(\mathbf{Z}, \mathbf{Z}_{\text{max}})$
    $\mathbf{Z}_t \leftarrow q(\mathbf{Z}, t) \triangleright$ Add noise
    $\mathbf{Z} \leftarrow \text{Guided-Denoise}(\mathbf{Z}_t, t)$

6.5 Terrain Dataset Curation

Training our diffusion model is data intensive and its inference quality relies on a dataset of real-world mountainous terrain data that is large-scale, high-resolution, freely available, with noticeable relief and cleaned. Unfortunately, as shown in Table 6.1, no existing dataset meets all these requirements, necessitating our own curation process.

Our training is predicated on both a short sampling distance ($dx = 1$m) and high tile resolution ($10{,}012^2$). This combination captures fine detail for close-up views and supports downsampling to multiple sampling distances (1m, 2m, 4m, 8m, 16m), while maintaining

Table 6.1: *A comparison of terrain datasets. Our dataset provides a unique combination of open access, overall size, high individual tile resolution, small sampling distance (dx), and data cleaning.*

Dataset	Open	Size	Tile Res.	dx(m)	Cleaned
[Guérin et al., 2017]	No	NA	3600^2	30	No
[Argudo et al., 2018]	**Yes**	3.5GB	Variable	2	No
[Perche et al., 2023]	No*	NA	1000^2	5, 30	**Yes**
[Lochner et al., 2023]	No	NA	256^2	2.39	**Yes**
[Czerkawski et al., 2025]	**Yes**	973GB	356^2	30	No
Ours	**Yes**	**1.2TB**	$\mathbf{10012}^2$	**1**	**Yes**

* Provide links to raw source but not a cleaned dataset.

a viable tile resolution. Furthermore, raw elevation data often contains artifacts, missing regions, or non-mountainous features. A dedicated cleaning and curation process is therefore necessary to ensure the model learns from high-quality, relevant terrain.

Our starting point is the USGS 3DEP 1m dataset [U.S. Geological Survey, 2024], which contains over 100,000 high-resolution ($10{,}012^2$) tiles at a 1-meter sampling distance. This is fed into a multi-step filtering pipeline, as follows:

Heuristic thumbnail culling The entire collection of raw TIFF files of the USGS 3DEP is unmanageably large. Instead, we perform a first round of data reduction based on the 300×385 JPEG thumbnail images accompanying each TIFF. Absolute elevation is not available because these are 8-bit normalized grayscale images. However, this is sufficient for culling tiles with data voids covering more than half the image, because this manifests as blacked out pixels (Figure 6.7a). This is followed by filtering out relatively flat tiles employing a texture-based heuristic. Each thumbnail is segmented into a grid of 32×32 patches and we calculate the mean of the per patch standard deviation in grayscale value. Below a mean threshold of 5, the terrain is considered flat (Figure 6.7b) and the image is culled. Together, these two heuristics effectively reduce our candidate pool by 70%, to approximately 30,000 images.

Iterative learning-based filtering Some relatively flat regions, particularly those with complex non-mountainous river systems and erosion gullies, slip past our variance-based culling. To address this, we iteratively fine-tune a ResNet-18 [He et al., 2016] image classifier. We begin with a manually classified seed dataset of 1000 negative (e.g.,

meandering river systems, as in Figure 6.7c) and 1500 positive (e.g., dendritic ridges and valleys, as in Figure 6.7d) cases. In each cycle, the trained model classifies a new subset of data, which we then manually review for misclassifications, adding the corrected samples back to the training set. After three such refinement iterations, our dataset is filtered down to approximately 9,000 high-confidence thumbnails.

Final curation of full-resolution tiles We download the corresponding full-resolution TIFF files for tiles that pass the initial rounds of culling. Then, on the basis of the detailed data newly available, a final two-step verification is performed: tiles containing any data voids are removed, as are those without significant vertical relief (quantified as an elevation range of less than 200 meters). This yields a final curated dataset of 4,002 high-quality TIFF files, each with a $10{,}012^2$ resolution at a 1-meter sampling density, totaling 1.2 TB.

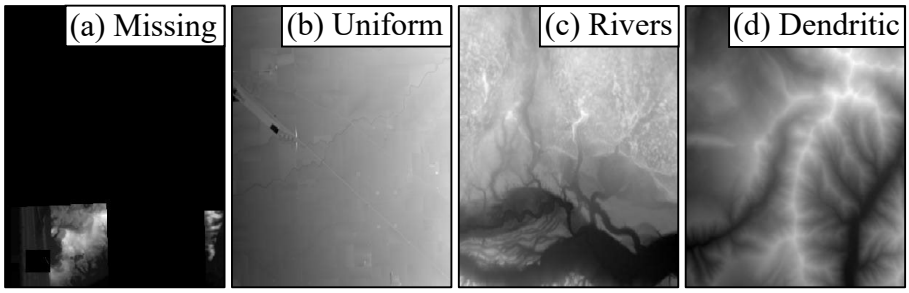


Figure 6.7: *Our multi-stage filtering pipeline for dataset curation, illustrated with thumbnail examples. Heuristic filters first discard thumbnails with large data voids (a) or low texture variance, indicative of flat terrain (b). Subsequently, a trained classifier distinguishes unwanted features (c) from high-relief terrain (d).*

6.6 Results and Evaluation

We implement our method using Python and PyTorch, and optimize using custom CUDA kernels, with Google Gemma-3-27b-it [Google, 2025] applied to VLM queries. The terrain heightmaps depicted in the chapter have been rendered in Blender. Our code, trained models, and dataset are available at: `https://gitlab.inria.fr/landscapes/pixels2peaks`.

Training and inference were performed on a node equipped with an Intel Xeon 6710E CPU clocked at 3.2GHz, with 256GB RAM and an Nvidia H100 NVL GPU with 94GB of

memory.

We trained five separate diffusion models to generate 512^2 terrain heightmaps at 1m, 2m, 4m, 8m, and 16m resolutions. Each 6-level U-Net model (71M parameters, 2 ResNet layers per block, with self-attention at lower resolution scales) uses DDPM (T=1000) and was trained from scratch for 500,000 steps (10 days per model) using cropped and resized data. Training employed the AdamW optimizer [Loshchilov and Hutter, 2017] with batch size 16 and a cosine annealing learning rate schedule decaying from 10^{-4} to 10^{-6}.

For inference, we set the default guidance terms to the values shown in Table 6.2.

Table 6.2: *Guidance parameters*

Parameter	s	λ_{hyd}	λ_{geo}	λ_{view}	λ_{s}	λ_{c}
Value	10^0	10^8	10^6	10^5	10^0	10^0

Breaking down performance, the entire image-to-heightmap pipeline completes within roughly 5 minutes. The pre-diffusion steps require $\approx$ 80s, of which VLM inference accounts for $\approx$ 60s. The guided diffusion process generates a 1024^2 terrain using a 512^2 model in $\approx$ 150s. The subsequent polishing step, which includes minor corrections and upsampling to 2048^2 takes $\approx$ 70s.

The main factor that scales computation cost is the number of tiles in the denoising grid. By default, the sampling density dx is chosen so as to produce a 1024^2 resolution heightfield before final upsampling, with an associated 3×3 grid of 9 tiles. Each tile measures 512^2, though an enlarged border is needed to accommodate random shifts of the heightfield in x and y. Occasionally, a higher resolution heightfield and larger tile grid may be required for particularly expansive and distant panoramas.

To extract the elevation and slope necessary for calibration, we supplied the VLM [Google, 2025] with an ensemble of prompt inputs, which were subsequently followed by an output cleaning step. For estimating elevation, the model was queried with variations such as: "Estimate the approximate height of this terrain in meters. Also give the approximate elevation/altitude of the lowest and the highest visible point in this image. Think step by step.", "Based on the provided image, please estimate the approx elevation above sea level (in meters) for: 1) The lowest point clearly visible. 2) The highest peak visible within the frame.", "Analyze this terrain image. Provide an estimate for the elevation above sea level, from the lowest point visible and the highest peak visible (in meters).", "Examine the landscape shown in this image. What are your best estimates

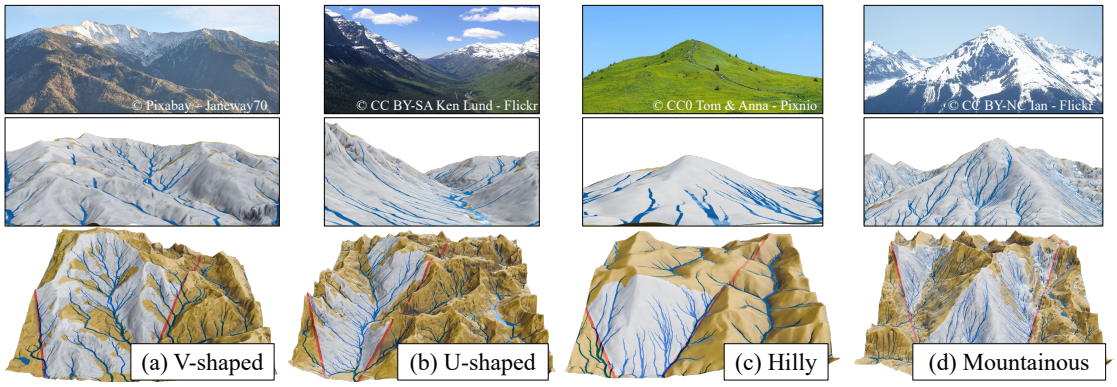


Figure 6.8: *Diversity in landscape forms (from left to right): a V-shaped valley, a U-shaped glacial valley, a nearby hill, and a high-elevation mountain. Given the input photograph (top row), we show the reconstructed landscape from the same camera viewpoint (middle) and a pulled-back view (bottom row). To provide a better visual handle on geomorphological and view consistency, we shade regions of the terrain visible from the camera viewpoint in gray, and generated occluded regions in taupe. The lateral limits of the camera frustum are indicated with red lines.*

for the minimum and maximum altitudes (in meters) visible within the picture?", and "Can you identify the approximate geographical region or mountain range depicted in this image? Based on that (or general terrain analysis if unidentified), please estimate the elevation above sea level (in meters) for the lowest and highest points shown.".

Similarly, to estimate the terrain's average slope, we prompted the model with the following texts: "Based on the provided image, estimate the approximate average slope (in degrees) for the terrain. Think step-by-step.", "Analyze this terrain image and quantify the approximate average incline angle in degrees. Think step-by-step.", "Using visual cues in this landscape photo, what would you estimate the mean gradient (in degrees) to be? Think step-by-step.", "From the visible features in this terrain image, calculate the approximate mean incline in degrees. Think step-by-step.", and "Looking at this landscape photo, provide your estimate of the terrain's average slope angle. Think step-by-step.".

6.6.1 Qualitative Results: Terrain Diversity

Our terrain synthesis is capable of matching a variety of landforms. This is demonstrated in Figure 6.8(top) using source photographs of landscapes incorporating: a) V-shaped valleys formed by fluvial processes, b) U-shaped valleys resulting from glacial erosion, c) low-lying hills, and d) high-elevation mountain peaks.

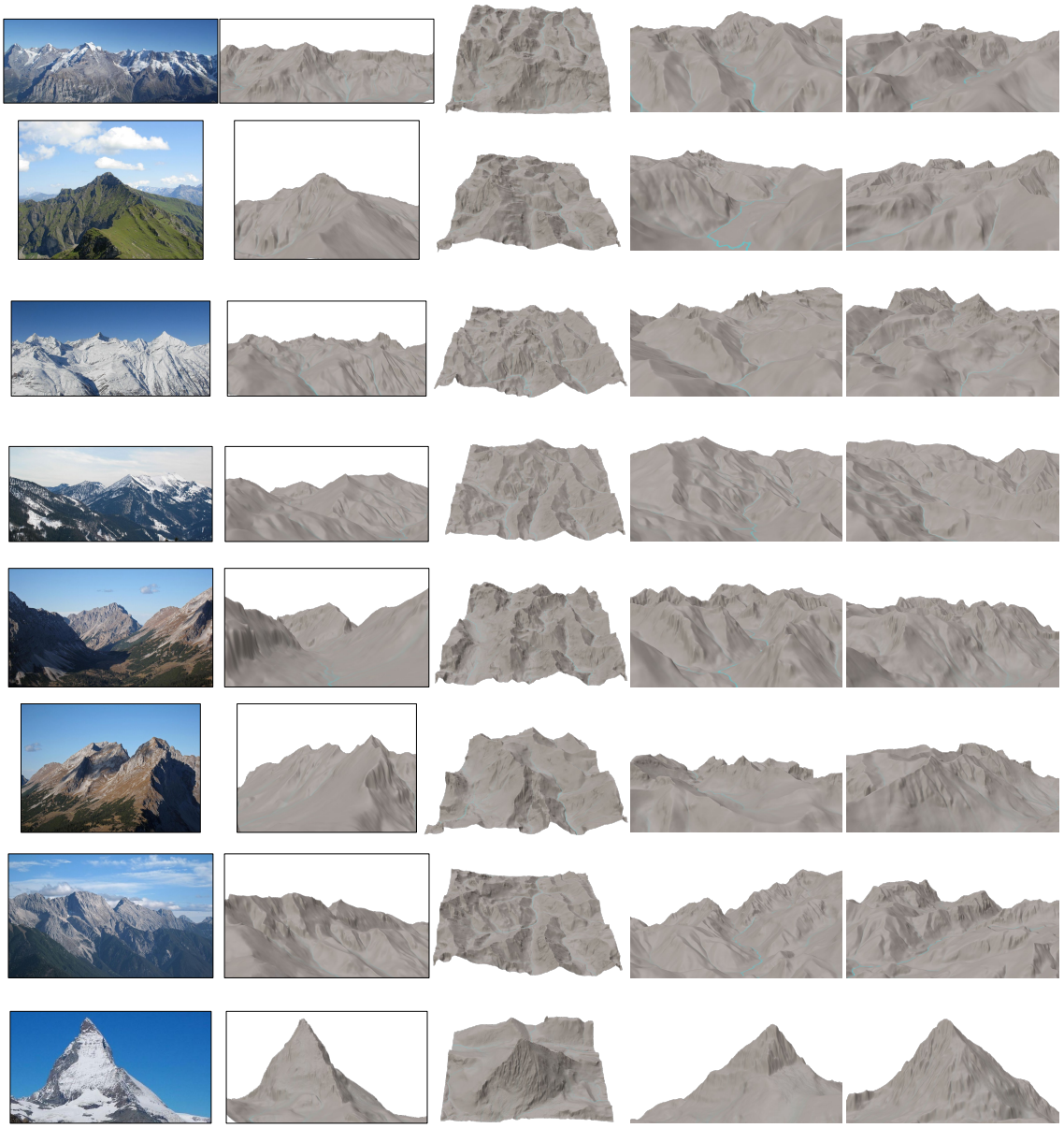


Figure 6.9: *Qualitative results of Pixels2Peaks applied to eight sample images from the GeoPose3K dataset. From left to right, the columns display: the original input photograph, the reconstructed heightmap rendered from the same camera viewpoint, a pulled-back view revealing the complete terrain, and two novel random viewpoints of occluded regions.*

We observe that the generated heightmaps (middle and bottom rows) consistently match the landforms in the photographs, that the mountain silhouettes are preserved, and that our results are free of large water bodies typical of depressions in hydrologically inconsistent terrains. Qualitatively, from a geomorphological-consistency perspective, there are no abrupt transitions in the type of landforms between visible and occluded areas. Likewise, violations in view consistency are rare, as demonstrated by the scarcity of taupe (occluded) pixels in the camera perspective (middle row).

Figure 6.9 provides additional qualitative results for a selection of input images from the GeoPose3K dataset, demonstrating robustness and the generation of coherent occluded regions from novel viewpoints.

6.6.2 Validation: Initial Terrain Estimation

We now turn to a validation of the individual components of our algorithm, beginning with initial terrain estimation. First, we evaluate the pointmap estimation provided by MoGe-2 to establish its quantitative accuracy and qualitative robustness on terrain data. Following this, we evaluate the necessity of using a Visual Language Model (VLM) to estimate the tilt of the camera and the elevation range captured in the image.

MoGe-2 quantitative validation We evaluated MoGe-2's terrain depth estimation using 512 Phong-shaded synthetic images and ground-truth depth maps, rendered from manually arranged viewpoints across our heightmap dataset. After aligning MoGe-2's predictions to the ground truth via least-squares scale and shift optimization, the model achieved an Absolute Relative Error (AbsRel) of 0.082 ± 0.034 and a threshold accuracy ($\delta < 1.25$) of $92.06\% \pm 6.58\%$.

MoGe-2 Robustness Validation. To assess robustness to MoGe-2 estimation errors, we perturb the estimated pointmaps with multiplicative Gaussian noise. Specifically, spatial coordinates are scaled by a noise map sampled from $\mathcal{N}(1, \sigma)$ using standard deviations $\sigma \in \{0.00, 0.05, 0.10\}$ (Figure 6.10) to simulate depth-proportional sensor measurement errors. While this noise inherently degrades high-frequency details, the diffusion model effectively smooths the perturbations to synthesize plausible landscapes that preserve the low-frequency structure of the input image.

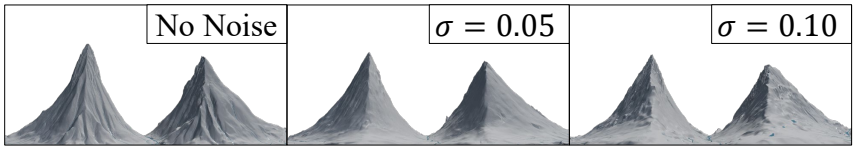


Figure 6.10: *MoGe-2 robustness to noise. Pointmaps are perturbed with multiplicative Gaussian noise: (Left)* $\sigma = 0$*, (Center)* $\sigma = 0.05$*, and (Right)* $\sigma = 0.10$*. The input image is from Figure 6.18(b).*

VLM tilt validation As a baseline against which to evaluate camera tilt prediction, we make use of the GeoPose3K dataset [Brejcha and Čadík, 2017]. This consists of 3,111 photographs of the Alps, with accurate camera position and orientation metadata derived by refining rough initial camera parameters using alignment with scanned real-world heightmaps. We reserve this dataset for evaluation, since with 3,111 images confined to the Alps it is too small and regionally narrow to train a tilt estimator that generalizes across landscapes. This baseline is compared against three alternatives (see Figure 6.11): a state-of-the-art horizon-line detector (SOFI) [Janampa and Pattichis, 2024], directly querying a VLM for tilt (VLM(t)), and our search procedure for inferring tilt from VLM-predicted slope (VLM(s)). Our VLM(s) (median error = 7.18°) significantly ($p < 0.01$) outperforms both VLM(t) (median error = 9.07°) and SOFI (median error = 13.16°) based on a Kruskal-Wallis test (H(2) = 610.4, $p < 0.001$, ε^2 = 0.066) followed by post-hoc Dunn's pairwise testing. The relatively poor performance of SOFI is explained by its reliance on geometric cues, which are often absent in mountainous scenery. Less clear is why slope prediction (VLM(s)) outperforms direct tilt prediction (VLM(t)). We hypothesize that this may be due to a higher frequency of terrain slope labeling in the internet-sourced training set of the VLM.

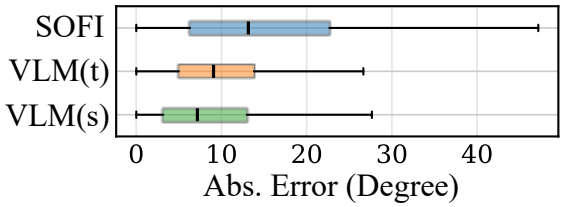


Figure 6.11: *To evaluate the options for camera tilt prediction, we compare the absolute error in degrees of a horizon line detector [Janampa and Pattichis, 2024] (SOFI) against a VLM, instructed to predict tilt directly (VLM(t)) or the terrain slope followed by an optimization search (VLM(s)).*

VLM elevation range validation For estimating the difference between the minimum and maximum elevation in an image, we compared the accuracy of our VLM approach against MoGe-2 [Wang et al., 2025], a state-of-the-art metric monocular geometry model. On a 16-image dataset derived from Google Earth with known viewpoints and elevation ranges, our VLM (mean error: 27.59% $\pm$ 19.08%) proves considerably more accurate than MoGe-2 (91.35% $\pm$ 4.57%). For an explanation, it is illustrative to consider the two cases in Figure 6.12. The Mt. Everest image (left) has a ground truth elevation range of $\sim$ 3,000m (from 5,700m to 8,700m) accurately inferred by the VLM (a 3,484m range) compared to MoGe-2 (a 77m range). The VLM's reasoning confirmed that it correctly identified the mountain itself along with the names of the visible glaciers.

A similar pattern holds for less immediately identifiable cases, such as the mountain in Idaho (right) with a $\sim$1,400m ground truth range, where the VLM (1,605m range) again outperforms MoGe-2 (203m range). While the VLM fails to precisely geolocate this image, it is able to reason by analogy using similar scenery and vegetation found in California's San Gabriel range.

The VLM's reasoning traces highlight that it establishes scale by semantically recognizing the scene, either directly or by analogy, and then accessing its stored geospatial knowledge. Monocular predictors, which infer scale from visual cues, such as atmospheric perspective, are far less reliable for large-scale natural landscapes.

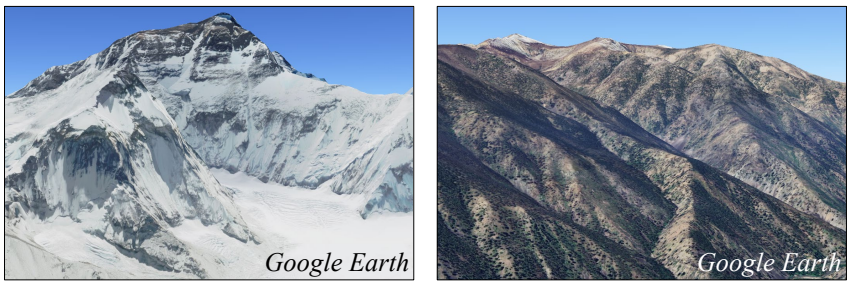


Figure 6.12: *Two illustrative cases of the landscape images selected for validation of our elevation range prediction against that of a metric geometry model (MoGe-2).*

6.6.3 Validation: Guided Diffusion

Next, we consider the guided diffusion process, including the pre-process of initialization and the post-process of polishing, from the perspective of our three key consistency attributes (see Section 6.2).

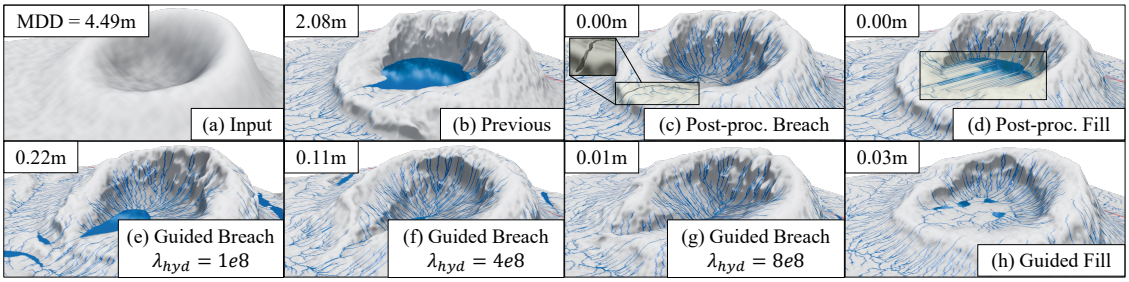


Figure 6.13: *An evaluation of flow guidance. We apply our diffusion model to a crater-shaped terrain (a). Diffusion without guidance [Hu et al., 2024] leaves a large endorheic basin (b). Post-processed breaching (c) or filling (d) of the depression results in an unnaturally deep and narrow trench or a flat sheet. In contrast, our method (bottom row) forms a valley to accommodate water outflow, controlled by the guidance weight (bottom left and center). As an option, the user can instead apply guidance with filling (bottom right).*

Hydrological consistency Endorheic basins with inflow but no outflow are a flaw common to many example-based terrain generation schemes. This gives rise to hydrologically unrealistic features, such as downward sloping valleys that are walled off at the end. In Figure 6.13, we explore guidance-based correction of one such challenging case: a prominent crater, with a mean depression depth (MDD) of 4.49m (Figure 6.13(a)). Here, MDD is defined as the average depth of water trapped on the terrain after flow simulation.

In the top row, we consider the performance of an unguided baseline and traditional post-processing. Unfortunately, these strategies are suboptimal: unguided diffusion [Hu et al., 2024] using SDEdit-style partial noising ($t \approx 0.5$) followed by denoising reduces the crater wall somewhat but still leaves an MDD = 2.08m basin behind. Post-hoc corrections fix the endorheic basin completely, but at the expense of realism. Breaching (c) introduces artificially narrow channels, while filling (d) produces unnaturally flat planar surfaces.

In the bottom row, we integrate guided breaching (e-g) and filling (h) into the diffusion loop, yielding more natural results. Guided breaching is as described in Section 6.4.2, while for guided depression filling we replace Algorithm 9 with a filling algorithm (Chapter 4).

Geomorphological consistency To evaluate geomorphological guidance, we perform a terrain completion task on a 2048^2 grid of which the top-left 512^2 tile is provided as a fixed prior (see Figure 6.14(left)). In the absence of guidance, a clear visual distinction exists between the fixed and synthesized areas. This qualitative observation is substantiated by a quantitative analysis of slope and curvature distributions, which reveals a

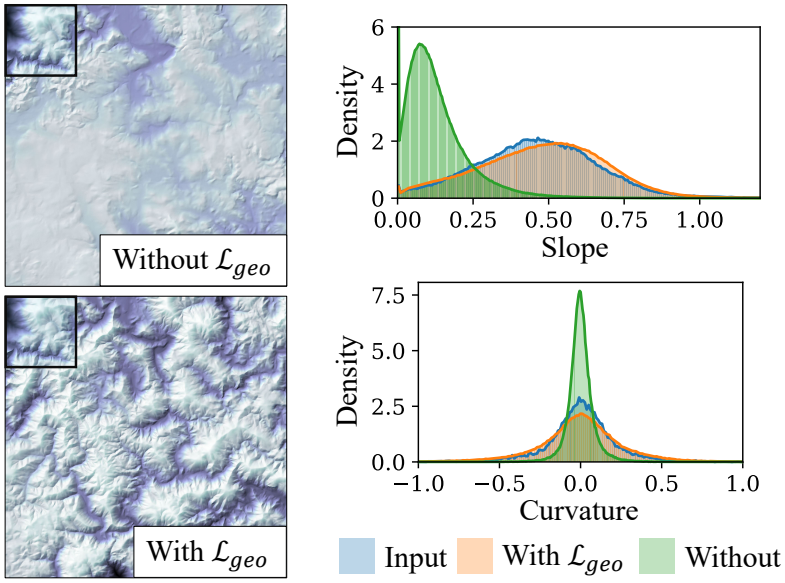


Figure 6.14: *An evaluation of geomorphological guidance. Better completion of a terrain (inset) is achieved with (bottom left) than without (top left) the $\mathcal{L}_{geo}$ guidance term. The improvement is further supported by the slope and curvature histograms (right); these compare the input terrain (blue) to completions with (orange) and without (green) guidance.*

much stronger statistical match between the two regions when guidance is applied (see Figure 6.14(right)).

View consistency In Figure 6.15, we ablate the view guidance term using the mountain range photograph from Figure 6.8(d). The impact of view guidance is particularly evident in the foreground and near silhouettes, both of which are visually salient. In quantitative terms, we define the Visible View Error (VVE) as the average over the heightfield of $\mathbf{M}_V \odot |\mathbf{Z} - \mathbf{Z}_V|$, and Occluded View Error (OVE) as the average over the heightfield of $(1 - \mathbf{M}_V) \odot \max(\mathbf{Z} - \mathbf{Z}_{\max}, 0)$. We observe that without view guidance, VVE is 2.36m and OVE is 8.52m, whereas with view guidance, these errors are 2.32m and 0.09m, respectively.

Initialization and polishing validation We demonstrate the efficacy of the initialization (Algorithm 11) and polishing steps (Algorithm 12) in Figure 6.16, using the text-generated image from Figure 6.18(b) as input. We consider four configurations: (a) initializing with a pure noise state (diffusion starts at $t = 1$) for occluded regions; (b)

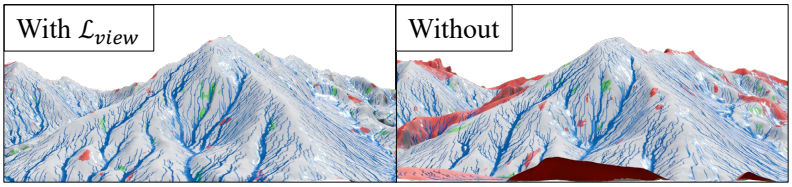


Figure 6.15: *An evaluation of view guidance. The peak from Figure 6.8(d), synthesized with (left) and without (right) the $\mathcal{L}_{view}$ guidance term. Red indicates an over-prediction of the elevation ($\mathbf{Z} > \mathbf{Z}_{max} + 5$), which does not respect terrain silhouettes, and green shows an under-estimation ($\mathbf{Z} < \mathbf{Z}_V - 5$) of the elevation compared to the heightmap projected from the input image.*

initializing with Laplacian diffusion to interpolate visible into occluded regions; (c) using Algorithm 11 for initialization but omitting polishing, and (d) leveraging the complete pipeline.

As with the previous experiment, overestimated and underestimated view errors are coded in red and green, respectively. Hydrological inconsistencies are highlighted as depression maps in the inset plan views. For each configuration, we estimate hydrological consistency with the mean depression depth (MDD), and view consistency with Visible View Error (VVE) and Occluded View Error (OVE) metrics defined earlier. We observe that omitting initialization entirely (a) prevents convergence (MDD = 0.85m, VVE = 2.26, OVE = 186.34m). Initialization with Laplacian diffusion (b-left) produces degraded results even after guidance (b-right, MDD = 0.49m, VVE = 2.74m, OVE = 88.64m). In contrast, a consistent initialization (c) yields coherent terrain with minor inconsistencies (MDD = 0.09m, VVE = 2.20m, OVE = 1.86m), which polishing (d) effectively resolves (MDD = 0.01m, VVE = 2.06m, OVE = 0.14m).

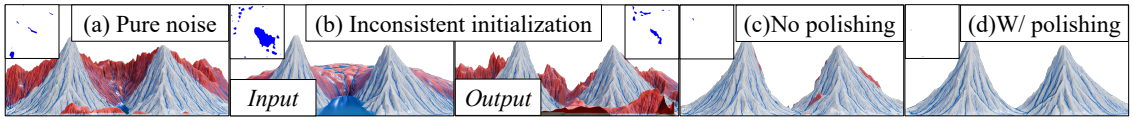


Figure 6.16: *The impact of initialization and polishing. Even with guidance, inconsistent initialization with noise (a) or Laplacian interpolation of the visible topography (b) fails to preserve view and hydrological consistency. Our initialization leads to a consistent terrain (c), with the resolution of marginal artifacts through polishing (d). The insets show depression maps. Input image from Figure 6.18(b).*

Setting hydrological guidance weighting To evaluate the impact of the hydrological guidance weight λ_{hyd} on synthesis quality, we undertook a comparison of real and generated terrains at various scales $\lambda_{\text{hyd}} = 0, 10^0, 10^1, \ldots, 10^{11}$ on the basis of Fréchet Inception Distance (FID) [Heusel et al., 2017], Kernel Inception Distance (KID) [Bińkowski et al., 2018], and mean depression depth (MDD) metrics (see Figure 6.17). FID measures the distance between Gaussian approximations of real and generated Inception features, whereas KID is an unbiased alternative that measures their Maximum Mean Discrepancy without distributional assumptions. We generated 10,000 unconditional diffusion samples for each λ_{hyd} scale and collected 10,000 real-world samples as a baseline. Results reveal a clear optimal weighting range of $10^8 < \lambda_{\text{hyd}} < 10^9$ with the lowest FID (11.01–12.99) and KID (0.009–0.011) scores, and MDD stabilized below 0.01m. A plateau of stable but weaker scores appears initially ($\lambda_{\text{hyd}} < 10^6$), likely arising from disconnected water flow paths in independently generated tiles, an artifact not present in real terrain. Past the optimum, FID and KID degrade sharply as the denoising diffusion process is suppressed in favor of flow routing, causing a drift away from the real data distribution. Note that the U-shape is in line with the application of guidance in other image diffusion contexts [Dhariwal and Nichol, 2021; Ho and Salimans, 2022].

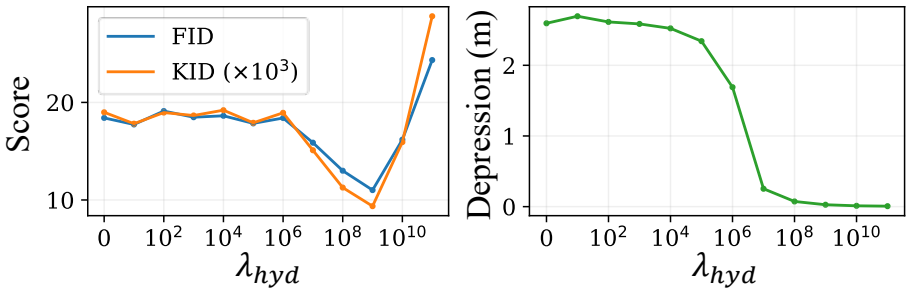


Figure 6.17: *The influence of hydrology guidance on quality and depression depth. We plot the FID and KID (left), and mean depression depth (right) of generated samples with increasing levels of guidance. Quality increases with sufficient hydrological guidance, reducing unnatural flow patterns, but then decreases when the guidance is overly strict.*

6.6.4 Applications

Beyond validating hydrological, geomorphological, and view consistency, we further demonstrate that our method extends effectively to non-photographic inputs, in the form of text, paintings, and sketches. This can be achieved by coupling a pre-trained

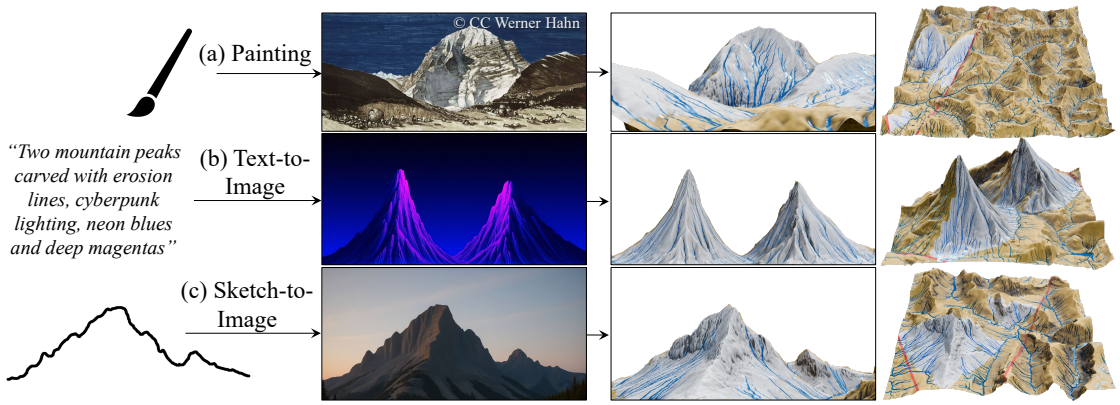


Figure 6.18: *Diverse input modalities, including painting, text, and sketching (column 1), are used to obtain images (column 2). From these images, we synthesize the corresponding landscapes (column 3: same camera viewpoint; column 4: pulled-back view).*

X-to-image model with our image-to-terrain model, effectively bolting on the former as an upstream module to our pipeline, creating an X-to-terrain model, where X is a placeholder for a range of input media.

Figure 6.18 demonstrates terrain synthesis from various forms of non-photographic input: (1) a painting, Thron der Götter by Werner Hahn (CC), which can be supplied directly to our existing pipeline; (2) an image produced by a text-to-image model [Labs et al., 2025], which we adapt for text-to-terrain synthesis; and (3) an image generated by a sketch-to-image model [Labs et al., 2025], extended for sketch-to-terrain.

Sketch-to-Terrain Figure 6.19 provides a side-by-side comparison of our sketch-to-terrain pipeline with the sketching system of Gain et al. [2009]. To enable a direct comparison, we traced a sketch from their teaser and processed it through our pipeline (Figure 6.18(c)). The sketch was first converted to a landscape image using FLUX.1 Kontext [Labs et al., 2025], which was in turn converted to a heightfield terrain by our pipeline. Our method's use of diffusion priors and hydrological guidance leads to a higher-fidelity terrain, with more consistent water flow paths.

Text-to-Terrain We compare and contrast our extended text-to-terrain pipeline against MESA [Czerkawski et al., 2025], which uses a Stable Diffusion model trained on terrain data with synthetic prompts. Unfortunately, since MESA's outputs are not on a metric scale, we cannot directly compare quantitative metrics, such as MDD. Instead, we

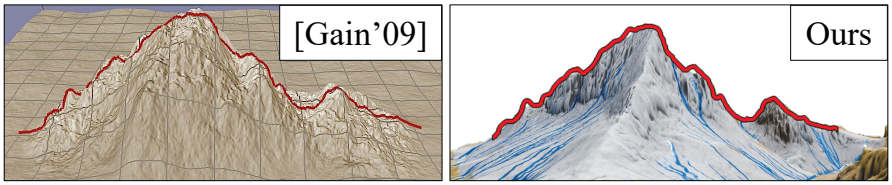


Figure 6.19: *Comparison with the sketch-based terrain synthesis of Gain et al. [2009]. (Left) Terrain generated using their system, with the input sketch shown in red. (Right) The same sketch is replicated and processed through our sketch-to-terrain pipeline (also in Figure 6.18(c)).*

visualize the derived depression maps, which show regions where water accumulates. Figure 6.20 presents 8 of the 9 samples available on the MESA project page, excluding only the depiction of plains, and should be contrasted with the right-most depression map derived from Figure 6.18(b). Because MESA does not incorporate an explicit hydrological mechanism, water tends to accumulate. Notice the presence of larger depressions compared to our previous examples.

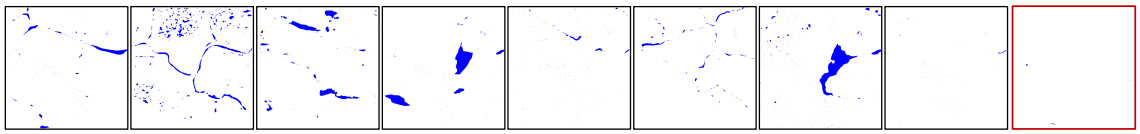


Figure 6.20: *Depression maps from MESA [Czerkawski et al., 2025] outputs show extensive water accumulation from unrealistic depressions in contrast to the text-to-image sample from Figure 6.18 (red outline). We used the following prompts from the MESA project page to derive the depression maps: "Mountains in the Alps in January", "Broadleaf forests and hills in Germany in August", "Norway fjords", "Puna and mountains in Bolivia in May", "Shrublands and mountains in United States of America in May", "Dry woodlands steppe and mountains in Tunisia in July", "Temperate forests and mountains in New Zealand in November", and "Tundra and mountains in United States of America in November".*



Furthermore, MESA relies on structured prompts and struggles with free-form inputs. As shown in Figure 6.18(b), our method successfully generates "two mountain peaks", while MESA produces an out-of-distribution canyon-like structure for the same prompt (see inset). Conversely, prompts like *"broadleaf forests and hills in Germany in August"* (their second sample) are a challenge for our pipeline, as text-to-image models often introduce large, undesired foreground elements in such cases.

Image-to-3D To demonstrate the necessity of a domain-specific approach, we compare our method against a general-purpose image-to-3D model, Hunyuan3D-2.1 [Tencent, 2025], using 667 low-tilt ($< 2°$) landscape images from the GeoPose3K dataset [Brejcha and Čadík, 2017]. Of these, we excluded 246 samples (37%) where the baseline generated geomorphologically implausible artifacts, such as floating islands or hollow cubes. To enable direct quantitative comparison of the remaining 421 samples, we converted Hunyuan3D-2.1's meshes to heightmaps via top-down orthographic projection and then applied a common VLM-derived scaling factor.

To assess hydrology, we computed mean depression depth (MDD) averaged over all generated samples. Because Hunyuan3D-2.1 is a generalist model lacking terrain-specific adaptation, it introduces unnatural sinks (MDD = 9.64m) not present in our model (MDD = 0.11m).

To assess realism, we measure the feature space distance between generated terrains and a reference set of 167,370 real-world multi-scale (dx from 1m to 42m) terrain feature patches from our dataset. Using DINOv2 [Oquab et al., 2023] embeddings, we compute the mean cosine distance to the 5 nearest real-world neighbors, where a lower score indicates higher realism. By this metric, our pipeline (mean = 0.221, SD = 0.052) outperforms Hunyuan3D-2.1 (mean = 0.312, SD = 0.104). We further confirmed this using Kernel Inception Distance (KID) against 16,000 ground-truth samples spanning dx from 1m to 42m, where our method achieved a better KID (0.0187) than the baseline (0.1304).

Qualitatively, Hunyuan3D-2.1's lack of terrain-specific training results in outputs lacking natural landscape features, as demonstrated by the Hunyuan3D-2.1 generated samples in Figure 6.21.

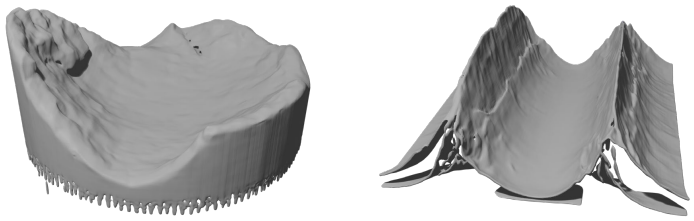


Figure 6.21: *Terrain samples generated by Hunyuan3D-2.1, lacking natural landscape features. For a comparison with the input images and our outputs, contrast the left sample against Figure 6.8(b) and the right sample against Figure 6.18(b).*

6.6.5 Limitations

Our method has several limitations that open up avenues for future research. The workflow is specialized for high-relief landscapes and struggles with out-of-distribution planar topographies (Figure 6.22, left), such as flatlands, desertscapes and meandering rivers. Performance also degrades in challenging photographic conditions, such as heavy fog, which can corrupt the initial geometry.

Furthermore, our sequential pipeline is susceptible to error propagation. For instance, an incorrect initial slope estimation can cascade into an implausibly steep heightmap (Figure 6.22, right), or undesirable foreground elements may be introduced during the text-to-terrain process. Tiled diffusion remains computationally expensive, limiting scalability for very large terrains. Tiling can also introduce minor but visible seams, though the tile offsetting strategy reduces this significantly. Finally, small inconsistencies such as unanticipated depressions can stem from inherent trade-offs between guidance terms in the diffusion denoising process.

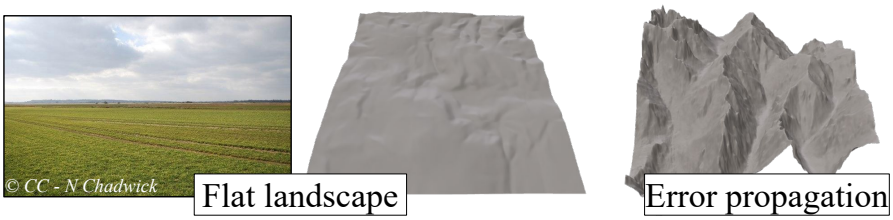


Figure 6.22: *Failure cases. Left: Out-of-distribution flat landscapes fail to reconstruct as perfectly planar, instead producing unintended elevation artifacts. Right: Incorrect VLM slope estimation cascades into inaccurate tilt and unnaturally steep terrain that mismatches the input (Figure 6.9, row 5). To demonstrate this, the VLM slope was manually forced to 60°.*

6.7 Conclusion

In this chapter, we presented *Pixels2Peaks*, a novel method for converting a photographic image of a landscape into a physically plausible 3D bare-earth heightmap. Our work addresses the fundamentally ill-posed problem of reconstructing unseen geometry by introducing a framework guided by three critical principles: geomorphological, hydrological, and view consistency. This ensures that the generated terrain not only conforms visually to the input image but also adheres to the natural laws that govern landscape formation.

Our technical contribution is a multi-stage pipeline that first resolves camera and scale ambiguities, producing a sparse but correctly-oriented initial heightmap of the visible terrain. We then employ a guided diffusion model, trained on a new, large-scale 1.2 TB dataset of high-resolution real-world terrain, to generate the occluded regions. The core of our method lies in the guidance terms that steer the diffusion process, ensuring the preservation of ridge-line silhouettes, the formation of coherent drainage networks, and the statistical alignment of landform features between visible and synthesized areas.

Through qualitative and quantitative evaluation, we have demonstrated that our approach reliably generates diverse landscapes that are consistent with their source images. Our ablation studies validate the necessity of each guidance component, and our comparisons show a significant improvement in physical plausibility and hydrological correctness over general-purpose models. Furthermore, we have demonstrated the flexibility of our core method by incorporating it into text-to-terrain and sketch-to-terrain pipelines, allowing additional forms of creative control.

While *Pixels2Peaks* represents a step forward, limitations remain that suggest promising directions for future research, such as exploring end-to-end architectures to improve robustness, expanding the model to cater to a wider range of topographies, and investigating more efficient synthesis strategies. More broadly, this chapter closes the thesis by showing that the same principle of flow networks used in the preceding chapters can also guide inverse terrain generation. Ultimately, our work helps bridge the gap between 2D inspiration and navigable 3D worlds, providing a tool for automating a key bottleneck in the creation of digital environments.

Chapter 7

Conclusion

7.1 Physically Guided Terrain Generation

Digital terrains act as the structural foundation of natural scenes. In computer graphics workflows, they must support exploration, editing, rendering from novel viewpoints, and downstream simulation. This makes terrain generation difficult, as the visible structure of a landscape is the macroscopic trace of water, sediment, rock, debris, wind, material strength, and time. Reproducing that full geological history, however, exceeds the iteration speed and controllability required by artists and technical users.

This thesis has argued that the tension between realism, efficiency, and control can be addressed through *physically guided* terrain generation. The methods developed here preserve selected process-level structures that explain the dominant visible organization of terrain, while avoiding the cost of simulating every microscopic event. The central abstraction throughout the thesis is the *flow network*: a macroscopic representation of how material moves through a landscape. Across the four contribution chapters, this abstraction was used to model debris-flow and fluvial erosion, accelerate exact hydrological routing, extend transport to volumetric sandstone structures, and constrain learned inverse terrain generation from images.

7.2 Main Contributions

The first contribution, presented in Chapter 3, introduced a debris-flow erosion and deposition model for valley-scale mountainous terrain. Existing fluvial models capture the organization of high-discharge river systems, while steep, low-drainage regions often remain too smooth or uniform. By incorporating geomorphological observations about debris flows, the chapter showed that steep slopes can be shaped by debris-flow erosion, producing scars near ridges and debris cones at the foot of cliffs. The model was coupled with fluvial erosion and deposition in a unified GPU simulation using approximate flow routing for water, sediment, and debris transport. Because deposits can create new local

minima that trap water and sediment, the chapter also introduced an approximate GPU depression-routing algorithm to progressively reconnect flow paths through the evolving terrain.

Chapter 3 also exposed a broader computational issue. The approximate flow and depression routing used remained a bottleneck for robustness and extreme parameters. Chapter 4 addressed this bottleneck directly by introducing GPU algorithms for exact flow routing and depression routing. Flow accumulation was formulated as a parallel computation over stream trees, while depression routing was formulated as a minimum spanning tree problem over the graph of terrain basins. The resulting algorithms run in $\mathcal{O}(\log n)$ and $\mathcal{O}(\log^2 n)$ iterations, respectively, for a terrain with n nodes. In practice, *FastFlow* achieved a $5\times$ speedup for flow routing compared with previous GPU approaches and a $34\times$ to $52\times$ speedup for depression routing compared with optimized parallel CPU methods on 1024^2 terrains. This made flow networks usable as interactive computational primitives for rivers, lakes, erosion, deposition, and ecosystem simulation.

Chapter 5 then moved beyond heightfields. Many characteristic sandstone landforms, such as arches, alcoves, caves, and buttes, extend beyond a single-valued surface through overhangs and internal cavities. *Arenite* showed that physically guided terrain modeling can be extended to erodible volumes by combining stress-dependent material resistance, wind erosion, fluvial erosion, and particle-based deposition. In a broader context, this chapter extended flow and depression routing to 3D, together with volumetric fluvial erosion and deposition mechanisms. The resulting simulator generated complex sandstone structures from simple initial blocks, painted erodibility, and vegetation masks. Typical interactive simulations used millions of particles, and the most complex examples ran in minutes on a desktop GPU, demonstrating that flow networks can efficiently organize transport in volumetric domains.

The final contribution, *Pixels2Peaks* in Chapter 6, turned from forward simulation to inverse generation. A single terrain image observes only part of the landscape from one camera, leaving the hidden geometry and metric scale underdetermined. *Pixels2Peaks* reduced this ambiguity by first estimating the visible terrain geometry, then using a terrain diffusion model to complete the unobserved regions under physical and view-based guidance. The method first recovers a partial, scaled heightmap from the visible terrain, then completes the occluded regions with guided tiled diffusion trained on a curated 1.2 TB dataset of real terrain. Hydrological guidance, derived from flow-network

conditioning, prevents the generated terrain from forming unrealistic closed basins, while view guidance preserves the silhouettes visible in the input image. The chapter showed that the same physical abstraction used in simulation can also guide learned generation, improving single-image terrain reconstruction and enabling image-, sketch-, and text-driven terrain workflows.

Taken together, these chapters support physical guidance in a concrete way. Long-term debris-flow effects shape mountain valleys without event-scale hazard simulation. Flow accumulation recovers the organizing effect of water without shallow-fluid dynamics. Stress, erosion, and deposition capture the characteristic forms of sandstone without simulating every individual geological event. Flow-based constraints supply hydrological structure to diffusion models when learned priors alone are ambiguous. In each case, the flow network acts as a compact process-level representation that makes terrain generation more realistic, efficient, and controllable.

7.3 Limitations and Future Work

The limits of these methods point to several directions for future terrain generation research. This thesis focuses primarily on medium- to large-scale terrain structure. While flow networks are well suited to valleys or drainage, close-up realism also depends on small-scale elements that require additional models. Pebbles, boulders, cracks, weathering marks, and local material heterogeneity each have their own spatial statistics and process histories. Modeling these features, and especially their distribution as a consequence of the larger terrain processes, remains an important direction for future work.

The flow-network abstraction itself also has limits, as it is primarily used to represent surface transport. This efficiently captures dominant drainage and sediment pathways, but it compresses several important degrees of freedom. More flexible representations are needed for lateral extent and multi-path motion, including river or debris-flow width, floodplains, landslide runout paths, and multiple-flow patterns such as braiding. They are also needed for transport that is not confined to the visible surface. Subsurface hydrology, seepage, and piping can erode or destabilize terrain from within, requiring coupled surface–subsurface networks with storage and permeability. Glacial landscapes pose a related challenge, where ice moves as a thick deforming body while transporting sediment, so adapting these methods to glaciers would require coupling routing-like abstractions with ice thickness, basal sliding, abrasion and quarrying.

More broadly, future work could ask how far physically guided transport abstractions extend beyond terrain. Atmospheric circulation, ocean currents, and mantle convection all organize natural structure through flow, but their dominant representations would not be downhill drainage trees. They would require time-dependent vector fields or circulation networks that can express a fully three-dimensional exchange. Extending the ideas of this thesis to such phenomena would preserve its central principle of identifying the process-level structure that matters at the target scale, then make that structure efficient and controllable enough for graphics workflows.

While the methods in this thesis offer various forms of user control, such as parameter tuning, masking, and image-based conditioning, there is room to make these tools more intuitive for artists. Raw simulation parameters are rarely legible; tweaking an erosion coefficient or a material threshold gives little immediate sense of the resulting landform, often forcing a tedious trial-and-error cycle just to see the consequences of an edit. Moving forward, authorial control can be improved through two complementary approaches: intent-driven forward authoring and observation-driven inverse generation.

On the forward authoring side, a promising direction is to elevate the physical proxy itself into a direct object of control. Rather than adjusting abstract coefficients around a simulation, artists could embed their intent directly into a flow network. By sketching drainage patterns, channel hierarchies, or ridge lines, users could guide the system to automatically solve for the compatible routing and boundary conditions. Machine learning could support this interface by translating sketches or text prompts into physically meaningful parameters like rainfall, uplift, or erodibility. Combined with multigrid techniques to rapidly propagate these high-level edits into detailed geometry, this approach transforms flow networks from rigid guiding structures into editable intermediates, blending physical consistency with artistic intent.

This goal of intuitive control naturally extends to inverse generation, which flips the paradigm: rather than sketching the underlying structure, artists can simply specify a desired final appearance, and the system works backward to infer the compatible terrain. *Pixels2Peaks* takes an initial step in this direction by reconstructing bare-earth heightmaps from observations. However, real production scenes are more complex, containing explicit river geometry, layered cliffs, vegetation, rocks, and snow. Rather than forcing these diverse elements into a single monolithic heightfield, future inverse methods must recover them as distinct assets. This points to a broader research horizon where image-to-terrain

reconstruction evolves into layer- and object-aware inference, utilizing a primary model to recover the base terrain, while specialized models independently extract rivers, rocks, sediment layers, and foliage.

7.4 Closing Perspective

Ultimately, the broader lesson of this thesis is that visual plausibility and physical structure are not opposing goals in terrain generation, but complementary ones. By identifying the specific process abstractions that govern landforms at a macroscopic scale, physics ceases to be a computational burden and instead becomes a practical, creative guide. Flow networks exemplify this principle: they distill the complex, organizing forces of a landscape into an efficient framework applicable to both simulation and learning. By embedding these transport routes directly into the generation pipeline, the resulting terrains do more than just look realistic. They inherit the deep, coherent patterns of drainage and deposition that define natural worlds, proving that true realism stems from a landscape's physical history, not just its visual form.

Bibliography

J. Adamovic, R. Mikulás, and V. Cílek. Sandstone districts of the bohemian paradise: emergence of a romantic landscape. *Geolines*, 21:5, 2006. 16

N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM TOG*, 31(4), July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185558. URL https://doi.org/10.1145/2185520.2185558. 84

N. H. Anh, A. Sourin, and P. Aswani. Physically based hydraulic erosion simulation on graphics processingunit. In *GRAPHITE '07*, pages 257–264, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1321261.1321308. 14

O. Argudo, A. Chica, and C. Andujar. Terrain super-resolution through aerial imagery and fully convolutional networks. *Computer Graphics Forum*, 37(2):101–110, 2018. 19, 115

O. Argudo, E. Guérin, H. Schott, and E. Galin. Terrain descriptors for landscape synthesis, analysis and simulation. *Computer Graphics Forum*, 44(2):e70080, 2025. 100, 109

D. Banninger. Technical Note: Water flow routing on irregular meshes. *Hydrology and Earth System Sciences*, 11(4):1243–1247, 2007. 17, 52

R. Barnes. Parallel priority-flood depression filling for trillion cell digital elevation models on desktops or clusters. *Computers & Geosciences*, 96:56–68, 2016. 19, 33, 50, 75

R. Barnes. Parallel non-divergent flow accumulation for trillion cell digital elevation models on desktops or clusters. *Environmental modelling & software*, 92:202–212, 2017. 18, 50

R. Barnes. Accelerating a fluvial incision and landscape evolution model with parallelism. *Geomorphology*, 330:28–39, 2019. ISSN 0169-555X. 18, 50, 72, 74, 76

R. Barnes, C. Lehman, and D. Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Comp. and Geosciences*, 62: 117–127, 2014. 18, 50, 54

R. Barnes, K. L. Callaghan, and A. D. Wickert. Computing water flow through complex landscapes–part 2: Finding hierarchies in depressions and morphological segmentations. *Earth Surface Dynamics*, 8(2):431–445, 2020. 19, 50

M. Becher, M. Krone, G. Reina, and T. Ertl. Feature-based volumetric terrain generation and decoration. *IEEE Transactions on Visualization and Computer Graphics*, 25(2): 1283–1296, 2019. doi: 10.1109/TVCG.2017.2762304. 11

J. Bender and D. Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings SCA*, SCA '15, page 147–155, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334969. doi: 10.1145/2786784.2786796. URL https://doi.org/10.1145/2786784.2786796. 84

B. Benes. Real-time erosion using shallow water simulation. In *VRIPHYS*, pages 43–50, 2007. 3, 14, 50

B. Benes and R. Forsbach. Layered data representation for visual simulation of terrain erosion. In *Proceedings of SCCG*, volume 25(4), pages 80–86. IEEE Computer Society, 2001. 14, 27, 78

B. Benes, V. Těšínský, J. Hornyš, and S. K. Bhatia. Hydraulic erosion. *Computer Animation and Virtual Worlds*, 17(2):99–108, 2006. doi: http://dx.doi.org/10.1002/cav.v17:2. 16

M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying mmd gans. *arXiv:1801.01401*, 2018. 126

O. Boruvka. O jistém problému minimálním. *Pràce Moravské přìrodovědecké společnosti*, 3 (3):37–58, 1926. 51, 58

S. Boyé, G. Guennebaud, and C. Schlick. Least squares subdivision surfaces. *Computer Graphics Forum*, 29(7):2021–2028, 2010. doi: https://doi.org/10.1111/j.1467-8659.2010.01788.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2010.01788.x. 83

J. U. Brackbill, D. B. Kothe, and H. M. Ruppel. Flip: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38, 1988. 81

J. Braun and M. Sambridge. Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. *Basin Research*, 9(1):27–52, 1997. 26, 66

J. Braun and S. D. Willett. A very efficient o(n), implicit and parallel method to solve the stream power equation governing fluvial incision and landscape evolution. *Geomorphology*, 180-181:170–179, 2013. ISSN 0169-555X. 14, 17, 31, 32, 50

J. Brejcha and M. Čadík. Geopose3k: Mountain landscape dataset for camera pose estimation in outdoor environments. *Image and Vision Computing*, 66:1–14, 2017. 97, 105, 121, 129

J. Bruthans, J. Soukup, J. Vaculikova, M. Filippi, J. Schweigstillová, A. L. Mayo, D. Mašín, G. Kletetschka, and J. Řihošek. Sandstone landforms shaped by negative feedback between stress and erosion. *Nature Geoscience*, 7:597–601, 2014. 16, 78, 80, 92, 93

N. Chiba, K. Muraoka, and K. Fujita. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation*, 9:185–194, 1998. 14

K. W. Chong, Y. Han, and T. W. Lam. Concurrent threads and optimal parallel minimum spanning trees algorithm. *Journal of the ACM (JACM)*, 48(2):297–323, 2001. 58

C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644, 2016. 21

V. Cílek, R. Williams, A. Osborne, P. Migoń, and R. Mikuláš. The origin and development of sandstone landforms. *Sandstone landscapes. Prague: Academia*, pages 34–43, 2007. 16

J. Coatléven. Some multiple flow direction algorithms for overland flow on general meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 54(6):1917–1949, 2020. 17

J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold: A world made of drawings. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 83–90, 2000. 20, 98

G. Cordonnier. *Layered Models for Large Scale Time-Evolving Landscapes*. Theses, Université Grenoble Alpes, Dec. 2018. URL https://theses.hal.science/tel-01971947. 10

G. Cordonnier, J. Braun, M.-P. Cani, B. Benes, E. Galin, A. Peytavie, and E. Guérin. Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum*, 35(2):165–175, 2016. 3, 12, 13, 14, 15, 17, 23, 25, 31, 32, 50, 52, 65, 68, 85, 93

G. Cordonnier, M.-P. Cani, B. Benes, J. Braun, and E. Galin. Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE transactions on visualization and computer graphics*, 24(5):1756–1769, 2017a. 15, 25

G. Cordonnier, E. Galin, J. Gain, B. Benes, E. Guérin, A. Peytavie, and M.-P. Cani. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM TOG*, 36(4):134:1–134:12, July 2017b. ISSN 0730-0301. doi: 10.1145/3072959.3073667. URL http://doi.acm.org/10.1145/3072959.3073667. 15

G. Cordonnier, B. Bovy, and J. Braun. A versatile, linear complexity algorithm for flow routing in topographies with depressions. *Earth Surface Dynamics*, 7(2):549–562, 2019. 19, 33, 34, 35, 45, 50, 58, 71, 75

G. Cordonnier, G. Jouvet, A. Peytavie, J. Braun, M.-P. Cani, B. Benes, E. Galin, E. Guérin, and J. Gain. Forming terrains by glacial erosion. *ACM Transactions on Graphics*, 42(4), 2023. 15, 24

M. Czerkawski, R. Martin, R. Rouffet, et al. Mesa: Text-driven terrain generation using latent diffusion and global copernicus data. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3067–3075, 2025. 115, 127, 128

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 107, 126

H.-T. Do, S. Limet, and E. Melin. Parallel computing flow accumulation in large digital elevation models. *Procedia Computer Science*, 4:2277–2286, 2011. 19

V. A. dos Passos and T. Igarashi. Landsketch: a first person point-of-view example-based terrain modeling approach. In *Proceedings of the International Symposium on*

Sketch-Based Interfaces and Modeling, page 61–68, 2013. doi: 10.1145/2487381.2487382. 20

D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and modeling: a procedural approach*. Elsevier, 2002. 12, 111

P. Ecormier-Nocca, G. Cordonnier, P. Carrez, A.-m. Moigne, P. Memari, B. Benes, and M.-P. Cani. Authoring Consistent Landscapes with Flora and Fauna. *ACM Trans. on Graph.*, 40(4), 2021. doi: 10.1145/3450626.3459952. 15

A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, 1999. doi: 10.1109/ICCV.1999.790383. 109

M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd*, 96(34):226–231, 1996. 104

H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 21

A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982. 3, 12

S. Frolov, B. B. Moser, and A. Dengel. Spotdiffusion: A fast approach for seamless panorama generation over time. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2073–2081, 2025. 113

J. Gain, P. Marais, and W. Strasser. Terrain sketching. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, I3D '09, pages 31––38, 2009. doi: 10.1145/1507149.1507155. 20, 21, 98, 127, 128

J. Gain, B. Merry, and P. Marais. Parallel, realistic and controllable terrain synthesis. *Computer Graphics Forum*, 34(2):105–116, 2015. doi: https://doi.org/10.1111/cgf.12545. 19

J. Gain, H. Long, G. Cordonnier, and M.-P. Cani. EcoBrush: Interactive Control of Visually Consistent Large-Scale Ecosystems. *Computer Graphics Forum*, 36(2):63–73, May 2017. 17

E. Galin, E. Guérin, A. Peytavie, G. Cordonnier, M.-P. Cani, B. Benes, and J. Gain. A review of digital terrain modeling. *Computer Graphics Forum*, 38(2):553–577, 2019. 3, 9, 11, 20, 52, 98

J. Garbrecht and L. W. Martz. The assignment of drainage direction over flat surfaces in raster digital elevation models. *Journal of Hydrology*, 193(1):204–213, 1997. ISSN 0022-1694. 17, 50

L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv:1508.06576*, 2015. 109

J.-D. Génevaux, É. Galin, E. Guérin, A. Peytavie, and B. Benes. Terrain generation using procedural models based on hydrology. *ACM TOG*, 32(4):1–13, 2013. 12, 15

J.-D. Génevaux, E. Galin, A. Peytavie, E. Guérin, C. Briquet, F. Grosbellet, and B. Benes. Terrain modelling from feature primitives. *Computer Graphics Forum*, 34(6):198–210, 2015. 12

Google. Gemma 3, 2025. URL https://goo.gle/Gemma3Report. 103, 105, 116, 117

R. Greeley and J. D. Iversen. *Wind as a Geological Process: On Earth, Mars, Venus and Titan*. Cambridge Planetary Science Old. Cambridge University Press, 1985. 16, 84

É. Guérin, J. Digne, E. Galin, A. Peytavie, C. Wolf, B. Benes, and B. Martinez. Interactive example-based terrain authoring with conditional generative adversarial networks. *Acm Transactions on Graphics*, 36(6):1–13, 2017. 3, 12, 13, 19, 20, 78, 115

H. Härtel, V. Cílek, T. Herben, A. Jackson, and R. Williams. *Sandstone landscapes*. Academia Praha, 2007. 78

M. Hartley, N. Mellado, C. Fiorio, and N. Faraj. Flexible terrain erosion. *The Visual Computer*, 40(7):4593–4607, Jul 2024. ISSN 1432-2315. doi: 10.1007/s00371-024-03444-w. URL https://doi.org/10.1007/s00371-024-03444-w. 16, 78

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 115

J. P. Hennessey Jr. A comparison of the weibull and rayleigh distributions for estimating wind power potential. *Wind Engineering*, pages 156–164, 1978. 84

R. Hereford, H. Fairlye, K. Thompson, and J. Balsom. Surficial geology, geomorphology, and erosion of archeologic sites along the colorado river, eastern grand canyon, grand canyon national park, arizona. *US Geological Survey, Open-File Report*, 93:517, 1993. 89

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 126

H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin. Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, 29(7):2179–2186, 2010. 12

J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 126

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 106, 107

P. Holmgren. Multiple flow direction algorithms for runoff modelling in grid based elevation models: An empirical evaluation. *Hydrological Processes*, 8(4):327–334, 1994. 17, 31

Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv:2311.04400*, 2023. 21

B. Horn. *Robot vision*. MIT press, 1986. 21

A. D. Howard. A detachment-limited model of drainage basin evolution. *Water Resources Research*, 30(7):2261–2285, 1994. doi: https://doi.org/10.1029/94WR00757. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/94WR00757. 84, 85

Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM TOG*, 37(4), July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201293. URL https://doi.org/10.1145/3197517.3201293. 83

Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM TOG*, 38(6),

Nov. 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356506. URL https://doi.org/10.1145/3355089.3356506. 88

Z. Hu, K. Hu, C. Mo, L. Pan, and Z. Wang. Terrain diffusion network: Climatic-aware terrain generation with geological sketch guidance. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(11):12565–12573, 2024. doi: 10.1609/aaai.v38i11.29150. 3, 13, 19, 98, 123

Y.-L. Huang and X.-F. Yuan. Styleterrain: A novel disentangled generative model for controllable high-quality procedural terrain generation. *Computers & Graphics*, 116: 373–382, 2023. 78

K. Hutter, B. Svendsen, and Rickenmann. Debris flow modeling: A review. *Continuum mechanics and thermodynamics*, 8:1–35, 1994. 15

IGN. Rge alti 1m [data set]. IGN, 2022. URL https://geoservices.ign.fr/rgealti. Accessed 2022-07-01. 69, 73

R. M. Iverson. The physics of debris flows. *Reviews of geophysics*, 35(3):245–296, 1997. 15, 24, 28

A. Jain, A. Sharma, and K. Rajan. Adaptive & multi-resolution procedural infinite terrain generation with diffusion models and perlin noise. In *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 1–9, 2022. 19

A. Jain, A. Sharma, and K. Rajan. Learning based infinite terrain generation with level of detailing. In *International Conference on 3D Vision (3DV) 2024*, 2024. 19

S. Janampa and M. Pattichis. SOFI: Multi-Scale Deformable Transformer for Camera Calibration with Enhanced Line Queries. In *35th British Machine Vision Conference 2025, BMVC 2025*, 2024. 121

J. Jasiewicz and T. F. Stepinski. Geomorphons—a pattern recognition approach to classification and mapping of landforms. *Geomorphology*, 182:147–156, 2013. 100

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. The affine particle-in-cell method. *ACM TOG*, 34(4), July 2015. ISSN 0730-0301. doi: 10.1145/2766996. URL https://doi.org/10.1145/2766996. 83

C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342896. doi: 10.1145/2897826.2927348. URL https://doi.org/10.1145/2897826.2927348. 83

M. D. Jones, M. Farley, J. Butler, and M. Beardall. Directable weathering of concave rock using curvature estimation. *IEEE TVCG*, 16(1):81–94, 2010. doi: 10.1109/TVCG.2009.39. 16, 78

M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM TOG*, 32(3), July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487237. URL https://doi.org/10.1145/2487228.2487237. 83

A. D. Kelley, M. C. Malin, and G. M. Nielson. Terrain simulation using a model of stream erosion. *Computer Graphics (Proceedings of SIGGRAPH)*, 22(4):263–268, 1988. 12

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023. 104

T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022. 105

B. Kotyra. High-performance watershed delineation algorithm for gpu using cuda and openmp. *Environmental Modelling & Software*, 160:105613, 2023. 34

P. Krištof, B. Benes, J. Křivánek, and O. Šťava. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum (Proceedings of Eurographics 2009)*, 28(2): 219–228, mar 2009. 3, 14, 16, 50

V. Krs, T. Haedrich, D. L. Michels, O. Deussen, S. Pirk, and B. Benes. Wind Erosion: Shape Modifications by Interactive Particle-based Erosion and Deposition. In *EG/ACM SIGGRAPH SCA*. The Eurographics Association, 2020. ISBN 978-3-03868-119-9. doi: 10.2312/sca.20201216. 16, 78

A. A. Kubade, A. Sharma, and K. S. Rajan. Feedback neural network based super-resolution of DEM for generating high fidelity features. In *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 1671–1674, 2020. 19

B. F. Labs, S. Batifol, A. Blattmann, F. Boesel, S. Consul, C. Diagne, T. Dockhorn, J. English, Z. English, P. Esser, S. Kulal, K. Lacey, Y. Levi, C. Li, D. Lorenz, J. Müller, D. Podell, R. Rombach, H. Saini, A. Sauer, and L. Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. 127

D. Lague and P. Davy. Constraints on the long-term colluvial erosion law by analyzing slope-area relationships at various tectonic uplift rates in the siwaliks hills (nepal). *Journal of Geophysical Research: Solid Earth*, 108(B2), 2003. 26

W. Li, J. Liu, H. Yan, R. Chen, Y. Liang, X. Chen, P. Tan, and X. Long. Craftsman3d: High-fidelity mesh generation with 3d native diffusion and interactive geometry refiner. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025. 21

R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023a. 21

S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv:2303.05499*, 2023b. 104

J. Lochner, J. Gain, S. Perche, A. Peytavie, E. Galin, and E. Guérin. Interactive authoring of terrain using diffusion models. *Computer Graphics Forum*, 42(7):e14941, 2023. doi: https://doi.org/10.1111/cgf.14941. 3, 13, 19, 20, 98, 115

D. B. Loope and J. A. Mason. Wind erosion of the wind-deposited navajo sandstone, usa. *Geology Today*, 25(2):65–70, 2009. doi: https://doi.org/10.1111/j.1365-2451.2009.00709.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2451.2009.00709.x. 16, 78, 84

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017. 117

A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022. 107

B. B. Mandelbrot. *The Fractal Geometry of Nature*. W.H. Freeman and Company, San Francisco, 1983. 12

L. W. Martz and J. Garbrecht. The treatment of flat areas and depressions in automated drainage analysis of raster digital elevation models. *Hydrological processes*, 12(6): 843–855, 1998. 101

L. A. McGuire, S. W. McCoy, O. Marc, W. Struble, and K. R. Barnhart. Steady-state forms of channel profiles shaped by debris-flow and fluvial processes. *Earth Surface Dynamics Discussions*, 2022:1–33, 2022. 15, 24, 28, 29, 43, 44

C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv:2108.01073*, 2021. 112

L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 21

P. Migoń. Sandstone geomorphology – recent advances. *Geomorphology*, 373:107484, 2021. ISSN 0169-555X. doi: https://doi.org/10.1016/j.geomorph.2020.107484. URL https://www.sciencedirect.com/science/article/pii/S0169555X20304578. 16, 78, 80

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2021. doi: 10.1145/3503250. 21

I. D. Moore, R. Grayson, and A. Ladson. Digital terrain modelling: a review of hydrological, geomorphological, and biological applications. *Hydrological processes*, 5(1):3–30, 1991. 109

F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. *ACM Siggraph Computer Graphics*, 23(3):41–50, 1989. 3, 12, 13, 14, 15, 24, 27, 66, 78, 87

S. Naik, A. Jain, A. Sharma, and K. Rajan. Deep generative framework for interactive 3d terrain authoring and manipulation. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 6410–6413. IEEE, 2022. 19

A. B. Neely and R. A. DiBiase. Sediment controls on the transition from debris flow to fluvial channels in steep mountain ranges. *Earth Surface Processes and Landforms*, 48 (7):1342–1361, 2023. 24

B. Neidhold, M. Wacker, and O. Deussen. Interactive physically based fluid and erosion simulation. In *Proceedings of Eurographics Workshop on Natural Phenomena*, volume 1, pages 25–32, 2005. 14

J. O'Callaghan and D. Mark. The extraction of drainage networks from digital elevation data. *CVGIP*, 28:323–344, 1984. 17

H. Okada. Classification of sandstone: analysis and proposal. *The Journal of Geology*, 79 (5):509–525, 1971. 79

M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023. 129

I. Ostanin, A. Safonov, and I. Oseledets. Natural erosion of sandstone as shape optimisation. *Scientific Reports*, 7, 12 2017. doi: 10.1038/s41598-017-17777-1. 16, 78, 84

W. Pałubicki, M. Makowski, W. Gajda, T. Hädrich, D. L. Michels, and S. Pirk. Ecoclimates: climate-response modeling of vegetation. *ACM Trans. Graph.*, 41(4), 2022. 65

A. Paris, E. Galin, A. Peytavie, E. Guérin, and J. Gain. Terrain amplification with implicit 3d features. *ACM Trans. Graph.*, 38(5), Sept. 2019a. ISSN 0730-0301. doi: 10.1145/3342765. URL https://doi.org/10.1145/3342765. 11, 78

A. Paris, A. Peytavie, E. Guérin, O. Argudo, and E. Galin. Desertscape simulation. *Computer Graphics Forum*, 38(7):47–55, 2019b. doi: https://doi.org/10.1111/cgf.13815. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13815. 16

L. Pawlik, P. Gruba, A. Galazka, A. Marzec-Grzadziel, D. Kupka, K. Szopa, B. Buma, and P. Samonil. Weathering and soil production under trees growing on sandstones - the role of tree roots in soil formation. *Science of The Total Environment*, 902:166002, 2023. ISSN 0048-9697. doi: https://doi.org/10.1016/j.scitotenv.2023.166002. 92

S. Perche, A. Peytavie, B. Benes, E. Galin, and E. Guérin. Authoring terrains with spatialised style. *Computer Graphics Forum*, 42(7):e14936, 2023. doi: https://doi.org/10.1111/cgf.14936. 19, 115

K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985. 3, 12

J. T. Perron, J. W. Kirchner, and W. E. Dietrich. Formation of evenly spaced ridges and valleys. *Nature*, 460(7254):502–505, 2009. 24

F. J. Pettijohn, P. E. Potter, and R. Siever. *Sand and sandstone*. Springer Science & Business Media, 2012. 78, 79

A. Peytavie, E. Galin, J. Grosjean, and S. Mérillou. Arches: a framework for modeling complex terrains. *Computer graphics forum*, 28(2):457–467, 2009. 11, 78

A. Peytavie, J. Gain, E. Guérin, O. Argudo, and E. Galin. Deadwood: Including disturbance and decay in the depiction of digital nature. *ACM Trans. Graph.*, 43(2), feb 2024. ISSN 0730-0301. 64

B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv:2209.14988*, 2022. 21, 99

P. Prusinkiewicz and M. Hammel. A fractal model of mountains with rivers. In *Proceedings of Graphics Interface*, pages 174–180, 1993. 12

G. Qian, J. Mai, A. Hamdi, J. Ren, A. Siarohin, B. Li, H.-Y. Lee, I. Skorokhodov, P. Wonka, S. Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv:2306.17843*, 2023. 21

P. Quinn, K. Beven, P. Chevallier, and O. Planchon. The prediction of hillslope flowpaths for distributed hydrological modeling using digital terrain models. *Hydrological Processes*, 5:59–80, 1991. 17

S. D. Rajasekaran, H. Kang, M. Čadík, E. Galin, E. Guérin, A. Peytavie, P. Slavík, and B. Benes. Ptrm: Perceived terrain realism metric. *ACM Trans. Appl. Percept.*, 19(2), July 2022. ISSN 1544-3558. doi: 10.1145/3514244. URL https://doi.org/10.1145/3514244. 1, 2, 12

D. Rickenmann. Empirical relationships for debris flows. *Natural hazards*, 19:47–77, 1999. 29

W. Rieger. A phenomenon-based approach to upslope contributing area and depressions in DEMs. *Hydrological Processes*, 12(6):857–872, 1998. ISSN 08856087. 61

H. Riihimäki, J. Kemppinen, M. Kopecký, and M. Luoto. Topographic wetness index as a proxy for soil moisture: The importance of flow-routing algorithm and grid resolution. *Water Resources Research*, 57(10), 2021. 17, 51

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015. 107

P. Roudier, B. Peroche, and M. Perrin. Landscapes synthesis achieved through erosion and deposition process simulation. *Computer Graphics Forum*, 12(3):375–383, 1993. 14

A. Safonov, M. Filippi, D. Mašín, and J. Bruthans. Numerical modeling of the evolution of arcades and rock pillars. *Geomorphology*, 365:107260, 2020. 16, 78

H. Schott, A. Paris, L. Fournier, E. Guérin, and E. Galin. Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, 42, 2023. 15, 17, 18, 23, 25, 26, 31, 32, 50, 51, 67, 69, 71, 72, 74, 76, 85

J. J. Scott and N. A. Dodgson. Example-based terrain synthesis with pit removal. *Computers & Graphics*, 99:43–53, 2021. 19, 20, 27, 33, 98, 100, 112

J. J. Scott and N. A. Dodgson. Evaluating realism in example-based terrain synthesis. *ACM Transactions on Applied Perceptions*, 2022. 1, 2, 12

F. E. Sevilgen, S. Aluru, and N. Futamura. Parallel algorithms for tree accumulations. *Journal of Parallel and Distributed Computing*, 65(1):85–93, 2005. 51, 53, 55

SideFX. Houdini, 2023. URL www.sidefx.com. 67

R. Smelik, T. Tutenel, R. Bidarra, and B. Benes. A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum*, 33(6), 2014. 11

P. Software. Terragen 4. http://planetside.co.uk/, 2024. 37

J. D. Stock and W. E. Dietrich. Erosion of steepland valleys by debris flows. *GSA Bulletin*, 118(9-10):1125–1148, 09 2006. 15, 24, 26, 27

A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran. Energetically consistent invertible elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, page 25–32, Goslar, DEU, 2012. Eurographics Association. ISBN 9783905674378. 83

D. Sulsky, S.-J. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. ISSN 0010-4655. doi: https://doi.org/10.1016/0010-4655(94)00170-7. URL https://www.sciencedirect.com/science/article/pii/0010465594001707. Particle Simulation Methods. 81

H. Takahashi, Y. Kanamori, and Y. Endo. 3d terrain estimation from a single landscape image. *Computer Animation and Virtual Worlds*, 33(6):e2119, 2022. doi: https://doi.org/10.1002/cav.2119. 21, 99

T. Takahashi. Mechanical characteristics of debris flow. *Journal of the Hydraulics Division*, 104(8):1153–1169, 1978. 15

T. Takahashi. Debris flow. *Annual review of fluid mechanics*, 13(1):57–77, 1981. 15

F. P. Tasse, A. Emilien, M.-P. Cani, S. Hahmann, and A. Bernhardt. First person sketch-based terrain editing. In *Graphics Interface 2014*, pages 217–224, 2014a. 20

F. P. Tasse, A. Emilien, M.-P. Cani, S. Hahmann, and N. Dodgson. Feature-based terrain editing from complex sketches. *Computers & Graphics*, 45:101–115, 2014b. doi: https://doi.org/10.1016/j.cag.2014.09.001. 20

Tencent. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material, 2025. 21, 99, 129

J. T. Todd. The visual perception of 3d shape. *Trends in cognitive sciences*, 8(3):115–121, 2004. 101

A. V. Turkington and T. R. Paradise. Sandstone weathering: a century of research and innovation. *Geomorphology*, 67(1):229–253, 2005. ISSN 0169-555X. doi: https:

//doi.org/10.1016/j.geomorph.2004.09.028. URL https://www.sciencedirect.com/science/article/pii/S0169555X0400282X. Weathering and landscape evolution. 16, 78

P. Tzathas, B. Gailleton, P. Steer, and G. Cordonnier. Physically-based Analytical Erosion for fast Terrain Generation. *Computer Graphics Forum*, 2024. ISSN 1467-8659. 17, 53, 66

U.S. Geological Survey. 3D Elevation Program (3DEP), 2024. Accessed: 2024-11-11. 1-meter resolution Digital Elevation Model tiles. 102, 115

L. O. Valencia-Rosado, Z. J. Guzman-Zavaleta, and O. Starostenko. Generation of synthetic elevation models and realistic surface images of river deltas and coastal terrains using cgans. *IEEE Access*, 9:2975–2985, 2020. 19

J. Vanek, B. Benes, A. Herout, and O. Stava. Large-scale physics-based terrain editing using adaptive tiles on the GPU. *IEEE Computer Graphics and Applications*, 31(6):35–44, 2011. 14, 17

M. Veress. Karst types and their karstification. *Journal of Earth Science*, 31(3):621–634, 2020. 90

V. Vineet, P. Harish, S. Patidar, and P. Narayanan. Fast minimum spanning tree for large graphs on the gpu. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 167–171, 2009. 36, 51, 58

O. Štava, B. Benes, M. Brisbin, and J. Křivánek. Interactive terrain modeling using hydraulic erosion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA'08, pages 201–210, Goslar, DEU, 2008. Eurographics Association. ISBN 9783905674101. doi: 10.5555/1632592.1632622. 14, 78

N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018. 21

R. Wang, S. Xu, Y. Dong, Y. Deng, J. Xiang, Z. Lv, G. Sun, X. Tong, and J. Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv:2507.02546*, 2025. 103, 104, 122

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv:2203.11171*, 2022. 105

Y.-J. Wang, C.-Z. Qin, and A.-X. Zhu. Review on algorithms of dealing with depressions in grid dem. *Annals of GIS*, 25(2):83–97, 2019. 18, 33, 50

H. Wei, G. Zhou, and S. Fu. Efficient priority-flood depression filling in raster digital elevation models. *International Journal of Digital Earth*, 0(0):1–13, 2018. 18

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 105

M. Westoby, J. Brasington, N. Glasser, M. Hambrey, and J. Reynolds. 'structure-from-motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179, 2012. doi: https://doi.org/10.1016/j.geomorph.2012.08.021. 98

K. X. Whipple and G. E. Tucker. Dynamics of the stream-power river incision model: Implications for height limits of mountain ranges, landscape response timescales, and research needs. *Journal of Geophysical Research: Solid Earth*, 104(B8):17661–17674, 1999. 14, 25, 65

S. Willett, C. Beaumont, and P. Fullsack. Mechanical model for the tectonics of doubly vergent compressional orogens. *Geology*, 21(4):371–374, 04 1993. 25

C. Wojtan, M. Carlson, P. J. Mucha, and G. Turk. Animating corrosion and erosion. In *Proceedings of the Third Eurographics Conference on Natural Phenomena*, NPH'07, page 15–22, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673494. 14

X. Xia, K. T. Jarsve, T. Dijkstra, Q. Liang, X. Meng, and G. Chen. An integrated hydrodynamic model for runoff-generated debris flows with novel formulation of bed erosion and deposition. *Engineering Geology*, 326:107310, 2023. 15

Z. Yang, G. Cordonnier, M.-P. Cani, C. Perrenoud, and B. Benes. Unerosion: Simulating terrain evolution back in time. *Computer Graphics Forum*, 43(8):e15182, 2024. doi: https://doi.org/10.1111/cgf.15182. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.15182. 15

A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4578–4587, 2021. 21

X. P. Yuan, J. Braun, L. Guerit, D. Rouby, and G. Cordonnier. A new efficient method to solve the stream power law model taking into account sediment deposition. *Journal of Geophysical Research: Earth Surface*, 124(6):1346–1365, 2019. 26, 39, 50, 66, 67, 69

L. Zhang, Z. Wang, Q. Zhang, Q. Qiu, A. Pang, H. Jiang, W. Yang, L. Xu, and J. Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)*, 43(4), 2024. 21

R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999. doi: 10.1109/34.784284. 21

Y. Zhao, H. Liu, I. Borovikov, A. Beirami, M. Sanjabi, and K. Zaman. Multi-theme generative adversarial terrain amplification. *ACM Transactions on Graphics*, 38(6), 2019. 19

G. Zhou, Z. Sun, and S. Fu. An efficient variant of the Priority-Flood algorithm for filling depressions in raster digital elevation models. *Computers and Geosciences*, 90:87–96, 2016. ISSN 00983004. 18

G. Zhou, X. Liu, S. Fu, and Z. Sun. Parallel identification and filling of depressions in raster digital elevation models. *International Journal of GIS*, 31(6):1061–1078, 2017. ISSN 13623087. 19

H. Zhou, J. Sun, G. Turk, and J. M. Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, 2007. doi: 10.1109/TVCG.2007.1027. 13, 19