Création d'écosystèmes cohérents et animés : Apprentissage efficace à partir de données partielles

Pierre Ecormier-Nocca

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAX078

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

Authoring consistent, animated ecosystems: Efficient learning from partial data

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École Polytechnique

École doctorale n°626
École Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique, Données et Intelligence Artificielle

Thèse présentée et soutenue à Palaiseau, le 03 décembre 2020, par

PIERRE ECORMIER-NOCCA

Composition du Jury :

Jean-Michel Dischler Professeur des universités Université de Strasbourg (ICUBE, UMR 7357)	Président
Joëlle Thollot Professeure des universités Grenoble INP (LJK, UMR 5216)	Rapporteure
Nuria Pelechano Associate Professor Universitat Politecnica de Catalunya (LSI)	Rapporteure
Eric Guérin Maitre de conférences INSA Lyon (LIRIS, UMR 5205)	Examinateur
Julien Pettré Directeur de recherche INRIA Rennes	Examinateur
Marie-Paule Cani Professeure des universités École Polytechnique (LIX, UMR 7161)	Directrice de thèse
Pooran Memari Chargée de recherche CNRS École Polytechnique (LIX, UMR 7161)	Co-directrice de thèse

Résumé

Grâce aux récentes améliorations de puissance de calcul, les mondes virtuels sont maintenant plus vastes et complexes que jamais. Alors que ce type de contenu se généralise dans de nombreux médias, les utilisateurs attendent une expérience de plus en plus réaliste. En conséquence, de nombreuses recherches ont été effectuées sur la modélisation et la génération de terrains et de végétation, et parfois leurs interactions. Néanmoins, les animaux ont reçu bien moins d'attention, et, comme les plantes, sont souvent étudiés en isolation. Avec le manque d'outils d'édition intuitive, ces problèmes font de la modélisation d'écosystèmes une tâche difficile pour les artistes, qui se retrouvent soit limités dans leur liberté créative, soit forcés d'ignorer le réalisme biologique.

Dans cette thèse, nous présentons des nouvelles méthodes adaptées au design et à l'édition d'écosystèmes virtuels, permettant la liberté créative sans pour autant renoncer à la plausibilité biologique. Notre approche a pour objectif de fournir des outils basés sur des données concrètes pour permettre une édition efficace des écosystèmes, tout en ne nécessitant qu'un nombre peu élevé de données. En incorporant les connaissances existantes sur la biologie à nos modèles, nous sommes capables de garantir à la fois la cohérence et la qualité des résultats.

Nous présentons des méthodes dédiées à l'instantiation précise et intuitive d'éléments statiques et animés. Pour prendre en compte le fait que les éléments statiques, tels que la végétation, peuvent présenter des interactions complexes, nous proposons une méthode précise basée sur l'exemple pour synthétiser des agencements arbitrairement complexes d'éléments statiques pouvant se recouvrir. Nous appliquons un concept similaire à l'édition de troupeaux, en utilisant des photographies ou courts segments vidéos comme entrée d'un algorithme de synthèse par l'exemple. À une échelle plus large, nous utilisons des données biologiques pour formuler un processus unifié gérant l'instantiation globale et les interactions de long terme entre la végétation et les animaux sur un terrain donné. En plus de garantir la cohérence biologique, ce modèle offre un contrôle sur le résultat en permettant l'édition manuelle des informations à n'importe quelle étape du processus.

Les méthodes proposées fournissent à l'utilisateur à la fois du contrôle et du réalisme tout au long du processus de création d'écosystèmes, couvrant les éléments statiques et dynamiques, ainsi que les interactions entre eux-mêmes et l'environnement. Différentes échelles sont également considérées, du placement et mouvement individuel à la gestion de l'écosystème complet. Nous montrons la validité de nos résultats à l'aide de plusieurs modes de validation, à savoir des études utilisateur, ainsi que des comparaisons avec des données réelles ou fournies par des experts.

Abstract

With recent increases in computing power, virtual worlds are now larger and more complex than ever before. As such content becomes widespread in many different media, the expectation of realism has also dramatically increased for the end user. As a result, a large body of work has been accomplished on the modeling and generation of terrains and vegetation, sometimes also considering their interactions. However, animals have received far less attention, and, just like plants, are often considered in isolation. Along with a lack of authoring tools, this makes the modeling of ecosystems an arduous task for artists, who are either limited in their creative freedom or are forced to break biological realism.

In this thesis, we present new methods suited to the design and authoring of virtual ecosystems, that allow for creative freedom without discarding biological plausibility. We focus on providing data-centered tools to allow efficient authoring of the ecosystem, while keeping a low data requirement. By taking advantage of existing knowledge regarding biology, we are able to guarantee both the consistency and quality of the results.

We present dedicated methods for precise and intuitive instantiation of static and animated elements. To account for the fact that static elements, such as vegetation, are able to display complex interactions, we propose an accurate example-based method to synthesize complex and potentially overlapping static arrangements. We apply a similar concept to the authoring of herds of animals, by using real photographs or short videos as input data for example-based synthesis. At a larger scale, we use biological data to formulate a unified pipeline handling the global instantiation and long-term interactions of vegetation and animals on a given terrain. While this model enforces biological consistency, we also provide control over the result by allowing manual editing of the data at any stage of the process.

Our methods provide both user control and realism over the entire ecosystem creation pipeline, covering static and dynamic elements, as well as interactions between themselves and their environment. We also cover different scales, from individual placement and movement of elements to management of the entire ecosystem. We demonstrate the validity of our results using different modes of validation such as user studies and comparisons with both real and expert data.

Thanks

I would like to first address a sincere thank you to my advisors for their support during my PhD. In particular, thank you Marie-Paule for your hard work and inspiring insights, both on my research and on Computer Graphics in general. Thank you Pooran for your mathematical and geometrical vision that has been a great help throughout my PhD, and a special thanks for bringing me into the lab in the first place.

Thank you to the jury who accepted to review this manuscript. I am looking forward to your comments, that will be a great help to continue my research.

I would also like to thank my co-authors Guillaume C., Julien, Bedrich, James and Baptiste. It was a pleasure working with you and I hope that we will have opportunities for more collaborations in the future.

Thank you to everybody related to the Tautavel project, since our team retreat in 2018. Thank you Anne-Marie, Philippe, Nicolas, Sophie, David, and all the others for your work.

A big thank you to all the researchers that I shared meals with, discussed with, interacted with in the day-to-day life in the lab. Thank you Pauline, Marie-Julie, Damien, Thibault, Corentin, Chloé, Maxime, Nicolas, Robin, Tashiv, Gowtham, Amal, Maks, Jean-Michel, Leo, Christophe, and everyone who has been a member of the GeoViC team. A special thank you to Thomas who accompanied me from the start of my PhD, for the problem-solving sessions, personal discussions, ping-pong matches, and everything else. Thank you Maud for being a great co-office, and for all the interesting discussions.

Thank you to Baptiste and Guillaume L., who I had the pleasure to advise during their respective internships. I hope you enjoyed your stay here and that your internship was useful to you. I would also like to thank the administrative and technical staff at the lab, including but not limited to Magali, Evelyne, Frédéric and Jordan, that keep the lab up and running.

A big thank you to my family, my parents and my sister for your continuous support over the years, and for giving me a great environment while growing up. You always pushed me towards science, curiosity and creativity, and for that I am forever grateful.

Finally, my deepest thanks are towards my wife Florence, who has been at the core of my life since I met her all those years ago. Thank you for sharing the best moments of my life with me, and for your unconditional support when I needed it most. Thank you for pushing me forward and always believing in me. Thank you for always bringing fun ideas, suggestions and activities to the table. Thank you for your help and your input, every time I needed it. For all of this, and so much more, this thesis is dedicated to you.

Table of Contents

Chapter 1

Introduction

Contents

Virtual worlds play an essential role in a growing number of multimedia content. While limited performances restricted them so far both in terms of quality and quantity of elements, new developments in hardware and algorithms have made virtual worlds a central part of games, movies, animations, and much more in recent years. To match this increased demand for virtual content, innovative methods have been created to boost the productivity of the artists designing them, while retaining fine control over the result. In this work, we specifically explore the living components of the world: ecosystems. We present new methods focused on improving ease of use and artistic control, and designed for the efficient authoring of complete virtual ecosystems.

1.1 Ecosystems in Computer Graphics

In this thesis, we use the term "ecosystem" to refer to the aggregate of all living species in a specific environment. While the word ecosystem has been extensively used in Computer Graphics literature as a shortcut for *plant* ecosystems, the term initially refers to all living entities including both fauna and flora. Although vegetation may have the overall highest impact on the visual appearance of an ecosystem, animals are vital for ecosystems to be considered truly complete. As the major dynamic entities in a landscape, animals can be used to naturally bring focus on specific areas of a landscape, provide additional information on the ecosystem, and give an overall much livelier feel to the scenery. Indirect information can also be used for artistic and visualization purposes: sounds of far away animals, trails and tracks, or a freshly grazed field can all convey plenty of information on the environment without directly showing the animals themselves.

1.1.1 Control

All the challenging problems considered in this thesis were tackled while keeping in mind the usability of the solution, *i.e.,* its ability to conform to the initial intention of the user. To best enforce usability and enable the combination of user control and help from the system towards consistency, we also made sure to restrict input information required from the user to *partial data*. We use this term to encompass all data that exhibit low requirements from the user: for example, data that can be incomplete or corrupted (low quality an quantity), that is widely available (*e.g.,* images on the internet), or that can easily be created manually. We did not consider *big data*, such as those used for deep learning, where very large quantities of potentially annotated data is required.

Originally, the absence of effective authoring tools meant that most of the creative work had to be done manually by artists, for every detail of a project. This places an important burden on creators, as a significant portion of their time is spent on simple tasks instead of being dedicated to the more creative aspects of their work. Since then, three main approaches have been introduced in Computer Graphics to reduce the workload of artists and help them in their creation.

The first, *inverse procedural methods*, attempt to reverse the usual procedural processes. Standard procedural methods start from a set of parameters that the user has to manually enter to generate a result. If the result is different from what the user expected, they update the parameters to bring the result closer to their initial vision. Unfortunately, this process quickly becomes tedious depending on the number of parameters, and linking a change of parameter to a specific impact on the result also becomes difficult as the complexity of the simulation increases. Inverse procedural methods reverse this problem, and try to provide mechanisms to automatically find the set of parameters that produce a given result when a specific generation technique is used.

In contrast, *example-based methods* start from an exemplar, but try to directly synthesize similar looking results instead providing the user with parameters for a specific algorithm.

Finally, *interactive modeling* relies on the user to interactively edit the generated content until a specific result is achieved. The edits can take the form of labels or constraints to guide the algorithm, or even direct modifications to the result to fit the needs of the user. This category of methods have the constraint of interactive performances for the synthesis algorithm, or the ability to continue the synthesis at specific points after edits from the user.

As the generation of ecosystems is a vast topic with many different facets, finding the right control systems for each task is a challenge in itself. We draw from both example-based methods and interactive modeling depending on the applications and type of available data, to steer towards the best possible balance between accuracy and user control.

1.1.2 Main challenges in populating landscapes

Instantiating objects, and possibly animating them over full virtual landscapes has been a long-standing problem in Computer Graphics. Even without considering animals, there has been a high demand for detailed virtual environments, populated with many objects such as plants and stones, for the background of media content.

While simulation methods have been developed for the particular case of growing and competing vegetation, they usually require long synthesis times and are rarely suited for high level user control. They may also not be compatible with other static elements that compose landscapes, such as rocks, branches, or man-made structures. Statistical methods no not depend on a specific type of content, and have been used for this purpose in the past. However, many fail to consider the spatial extent of objects, which in turn prevent them to accurately reproduce intricate relationships between elements. For example, the size of trees is the main variable needed to understand the distribution of resources between two plants, and is essential to model overlaps such as plants thriving in the shadow of others. Instantiation of animals and humans suffer from similar problems, but is made more complex by the dynamic nature of these entities. A moving object implies a direction of movement, which in turn makes an isotropic representation of such a distribution ill-suited. Dynamic elements also exhibit complex behaviors, that depend on a wide variety of variables related to the species and context, such as their speed, orientation, local density, vitality, and so on.

Taking all these attributes into account makes instantiation of elements in a landscape a difficult task, that is made even harder when control over the result is required. In this thesis, we use an example-based approach to bypass the cost of simulation and give an intuitive control over the result to the user.

1.1.3 Unified ecosystems

While Computer Graphics research on complete ecosystems is much less developed than local instancing, they can be used for the same applications. However, these two scopes present some key differences. For example, animals play a crucial role in such a context, as the expansion of plants would only be regulated by self competition without them.

While pure simulation approaches can in theory model a full ecosystem, they also present major drawbacks. Apart from the difficulty of coupling it with user control, stability is the main challenge for simulations. Indeed, the considerable number of parameters that need to be expressed in a complete ecosystem makes it nearly impossible to establish links between a single parameter and its impacts at the end of the simulation. Furthermore, the overwhelming majority of parameter configurations would lead to the extinction of one or more species, eventually bringing the ecosystem to a mostly dead resting state. The flexibility of the method is also important: modeling plants and animals as completely different systems could introduce inequalities in the framework, and

make it harder to naturally represent interactions between species.

An alternative approach to simulation methods can be devised to model ecosystems, in order to avoid their extensive computational costs. Large-scale solutions present such an alternative, where the main interactions considered are not between an element and its neighbors but between regions across the whole terrain. In this case, the precise instantiation of elements is not required during computations, as long as general constraints on the global population are satisfied. This can help simplify the computations, but requires a separate instantiation step to allow the user to visualize and explore the result afterwards.

If stability and computation issues can be solved, complete ecosystems techniques have the potential to open the way to new applications. For example, besides the expected use in entertainment and simulations, scientific study of inaccessible ecosystems can be made possible with such a system. It could also have future applications in museography, to allow visitors to visit unusual or now extinct ecosystems in real time.

1.2 General overview

This thesis presents novel methods dedicated to the authoring of various aspects of virtual ecosystems, from local instantiation of vegetation and animals to global control over a consistent, complete ecosystem.

1.2.1 Outline

After detailing the state of the art related to ecosystems in Computer Graphics (Chapter 2), we present our work in three chapters, each operating at different scales or on different subjects. The first two chapters of this thesis focus on direct instantiation of elements at a local scale, applied respectively to static and dynamic elements of the landscape. The third chapter targets ecosystems at a whole, and brings consistency between their different components at a global scale.

Static landscapes. We consider in Chapter 3 the placement of static objects in virtual worlds. This problem is critical for the creation of detailed virtual worlds, as it can be used for many different objects ranging from rocks and dead branches to trees and even man-made structures. We focus on providing a method adapted to objects with a variety of sizes, which may or may not exhibit specific constraints regarding their overlap, while relying on small quantities of data to facilitate user control. After detailing the necessary technical background on Pair Correlation Functions, used as a base for this chapter, we describe the different aspects that are necessary to the creation of such a method. We apply our approach to the analysis and synthesis of various arrangements of elements,

including direct applications to generation of vegetation. We validate our results by comparison with other methods with similar objectives.

Animated worlds. Chapter 4 discusses the extension of this framework to the analysis and synthesis of animated elements, such as animal herds. In comparison with static elements, the dynamic nature of herds makes efficient authoring a quite different problem. Because control over the result and ease of use remains our focus, we solve this issue by separating the control and animation in two separate parts of the approach. Following this logic, we extend previous methods to allow synthesis of static herds, where size, orientation, and density of animals are important. Standard methods in crowd simulation are then adapted to allow seamless interpolation between the generated static herds, producing a fully controlled and animated herd as a result. To further increase control, we allow the use of real photographs of animals as input through a semi-automatic method able to extract information from pictures. The output of the method is compared with the input photographs, and control over the different parameters are demonstrated with custom results. We finally show that an extension of this model to a fully animated method is possible, using video clips as input to gather additional information about the movements within the herd.

Ecosystems and interactions. In Chapter 5, we step away from local instancing to develop a global, unified framework for the authoring of complete ecosystems. After demonstrating the impact of dynamic elements on their environment with a case study of skiers on a snow-covered landscape, we detail the different interactions between the terrain, vegetation and animals in a full ecosystem. The work in this chapter was realized in collaboration with a team of paleontologists, who contributed through both data and insight about the interactions that take place within an ecosystem. We provide user control by allowing manual editing of data at any step of the pipeline, and resuming computations with the updated data. At the end of the process, our system outputs a stable ecosystem that can be interactively explored in 3D, showing vegetation, animals, and their impact in the form of trails and freshly grazed vegetation. We validate our results with a user study operated on both artists and scientists, and with comparisons with expected results that were hand-made by experts.

1.2.2 Publications

The research presented in this thesis have either been subject to publication, or is a work in progress expected to be submitted soon.

- Chapter 3: The main content of this chapter has been previously published in *Computer Graphics Forum* and presented at Eurographics:

  - Pierre Ecormier-Nocca, Pooran Memari, James Gain, and Marie-Paule Cani. Accurate synthesis of multi-class disk distributions. In *Computer Graphics Forum*, volume 38, pages 157–168. Wiley Online Library, 2019

  It has been extended and contextualized with content from another collaboration on a similar topic, presented as a short paper at Eurographics:

  - Baptiste Nicolet, Pierre Ecormier-Nocca, Pooran Memari, and Marie-Paule Cani. Pair correlation functions with free-form boundaries for distribution inpainting and decomposition. *Eurographics 2020 short paper proceedings*, page 4, 2020

- Chapter 4: This chapter has been published in *Computer Animation and Virtual Worlds* and presented at CASA:

  - Pierre Ecormier-Nocca, Julien Pettré, Pooran Memari, and Marie-Paule Cani. Image-based authoring of herd animations. *Computer Animation and Virtual Worlds*, 30(3-4):e1903, 2019

  It has been extended with work in progress demonstrating an avenue to extend such an approach to animated herds. As this part has not been completed, it does not include final results. A submission of this work is expected in the near future.

  - Pierre Ecormier-Nocca, Julien Pettré, Pooran Memari, and Marie-Paule Cani. Authoring animal herds through short video clips. *Work in progress, to be submitted.*

- Chapter 5: This chapter has been submitted before, but is currently in the process of being extended and resubmitted. It was realized in collaboration with other researchers, a team of paleontologists who provided data and knowledge, as well as engineers who helped on the visualization.

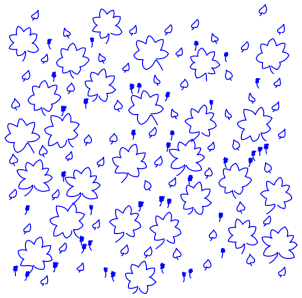
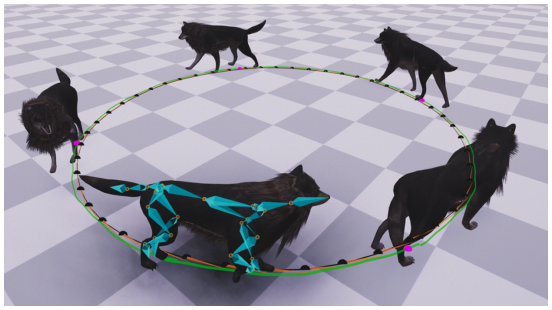
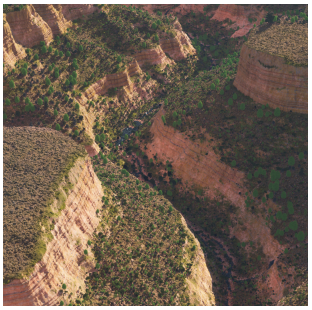
  - Pierre Ecormier-Nocca, Guillaume Cordonnier, Philippe Carrez, Anne-Marie Moigne, Pooran Memari, Bedrich Benes, and Marie-Paule Cani. Authoring Consistent Landscapes with Flora and Fauna. *Work in progress, to be submitted.*

  In the context of this thesis, a section from a different collaboration, published in *Computer Graphics Forum* and presented at Eurographics, has been added as a case study of an effect similar to the one seen in the main chapter:

  - Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani. Interactive generation of time-evolving, snow-covered landscapes with avalanches. *Computer Graphics Forum*, 37(2):497–509, May 2018

Chapter 2

State of the art

Contents

We present in this chapter an overview of previous research that has been used to place content such as objects, vegetation, and animals into virtual worlds. We first consider the creation of purely static landscapes. In this case, the process can be reduced to one of object placement in space, and has been extensively studied through the analysis and synthesis of point distributions (Section 2.1). While this approach can be used for learning and synthesizing the placement of all static objects, more specialized methods have been developed to tackle the generation of plants, from individual ones to ecosystems (Section 2.2). As virtual worlds are not limited to static elements, the movement and behaviors of both animals and humans have also been broadly studied in the past. These works will be detailed in Section 2.3. Finally, a few methods study the interaction between these different components, and will be presented in Section 2.4.

2.1 Distribution analysis and synthesis

Object placement in a scene, while used in a wide variety of Computer Graphics problems for decades, remains a difficult challenge. Although the synthesis of specific patterns has its own applications, we will focus on the task of learning these patterns from examples, in view of recreating them. We present in this section the state of the art and various applications of point distributions, before moving on to arrangements of shapes, where the extent and orientation of elements need to be considered.

2.1.1 Point distributions

The purpose of point sampling is to dynamically generate a set of points that embodies specific properties. It is widely used for stippling [MALI10, DSZ17, MAAI17], but also for rendering [Coo86, SJ13], remeshing [YLL+09, YW13, YGJ+14] and texture synthesis [DMLG02, IMIM08]. As one common requirement is to match the spectral profile of blue noise, there have consequently been many improvements to the state of the art in this area. For example, both [BWWM10] and [LNW+10] present improvements in computation time, by developing parallelized formulations of previous point synthesis algorithms. Blue noise synthesis methods have also been extended to new contexts such as anisotropic settings [LWSF10], or to problems where non spatial features also have an important place [CGW+13] by drawing inspiration from the bilateral filtering technique usually used for images. The analysis process has also seen similar improvements.

Point sampling synthesis methods can also be applied to general distributions instead of only predetermined ones. Example-based synthesis aims at deriving from a given representative input, an output that captures key visual aspects of the original, but which differs in certain specifics such as having larger extent or a constrained boundary. To achieve such a goal, different approaches are used depending on the application and constraints, and usually depend on the support used to encode information and the

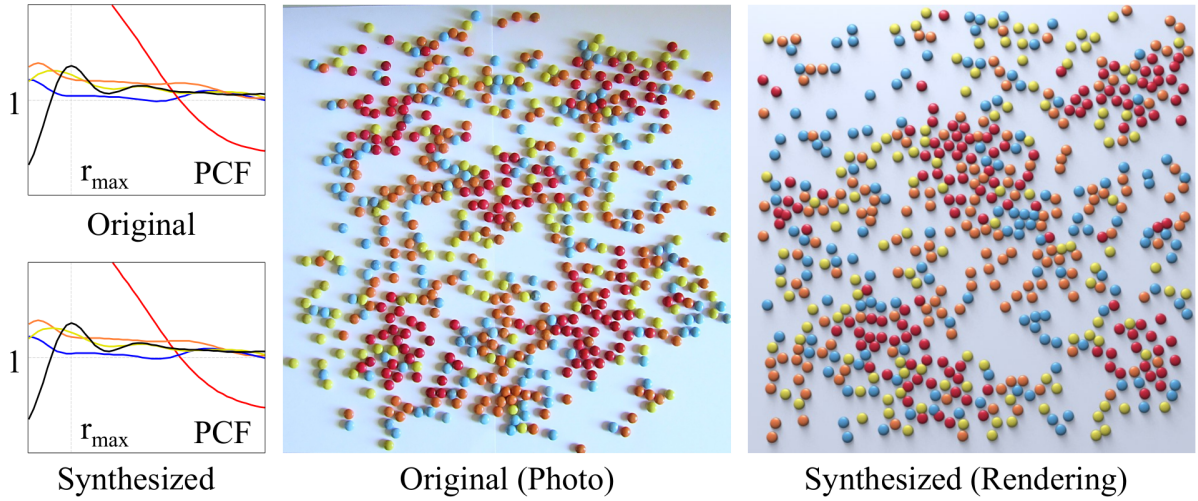


Fig. 2.1: After extracting Pair Correlation Functions from the input example (middle), a distribution with a similar appearance (right) can be synthesized by matching the curves [ÖG12].

selected synthesis algorithm. Discrete representations usually rely on histograms to count the number of samples appearing at different distances in the input exemplar. Since the histogram bins can have relatively large sizes, this allows for a fast analysis and low memory footprint, at the cost of precision in the output. Synthesis on such supports is usually done via the Metropolis-Hastings algorithm [HLT+09, EVC+15], where points are continuously added and removed at different probabilities based on convergence, or with dart throwing [GLCC17], by adding new points at random locations and either accepting or rejecting them to steer towards the target distribution.

In contrast, continuous representations for the distributions are heavier to store and compute, but offer higher precision and exhibit important properties to overcome sensitivity to noise and varying initial conditions. While the synthesis algorithms used for discrete supports can still be used, continuous representations support the computation of an analytic gradient, which in turn allows the use of robust solvers like gradient descent. Once a first estimation of the result is computed, this can be used to refine the solution by gradually moving the points towards their optimal position, resulting in a much more accurate solution.

Zhou *et al.* [ZHWW12] proposes a method using such a support, which relies on the spectral representation of a distribution to generate point distributions matching a user-defined spectrum. [ÖG12] uses an alternate 1D representation of the spectrum, namely Point Correlation Functions or PCFs, for the analysis and synthesis of a point distribution. They encode the point density depending on the distance between samples, and exhibit many interesting properties such as scale invariance, robustness to noise, and a strong characterization of the encoded distributions. As this approach has been used and extended for multiple applications throughout this thesis, the theoretical framework

on which it lies will be explained in detail in Chapter 3. PCFs have later been used for interpolation purposes between input exemplars by [RÖG17]. A functional sum-of-Gaussians was also used by Roveri *et al.* [RÖM+15] in the context of discrete, repetitive structure synthesis. While most work in this area focuses on distributions across a 2D plane or surface, one exception is the synthesis method of Lagae and Dutré [LD06], which supports Poisson sphere distributions in 3D space using an efficient tiling algorithm. More recent solutions such as [LSM+19] use deep learning techniques to provide efficient point sampling methods even in high dimensions. By transferring most of the computation time for a spectrum to the training phase, the method allows the synthesis of very high sample count at interactive rates. Recent neural network-based methods such as [TLH19] have also been used to accurately reproduce local details and regular patterns, a consistently challenging feature for previous methods.

2.1.2 Multi-class and shape aware solutions

One of the main problems with these approaches is that they do not effectively model interrelationships between classes of elements, and are mostly limited to distributions of points. As such, they are unable to reproduce more complex distributions where elements are placed differently based on their size, orientation, or display close interactions of overlap.

On the multi-class front, Hurtut *et al.* [HLT+09] automatically classify vector elements based on histograms of appearance that consider area, orientation, elongation, extremities and edge crossings. Arrangements among and between classes are then analyzed using multi-type point process statistics and synthesized with a variant of Metropolis-Hastings sampling. [EVC+15], also based on Metropolis-Hastings sampling, tackle multiple classes by iteratively computing descriptors of the distributions for each class, with respect to the previously instantiated classes. In a similar vein, multi-class variants of blue noise sampling techniques have been proposed by Wei [Wei10] based on extended dart throwing, and by Qin *et al.* [QCHC17] based on constrained Wasserstein barycenters. However, these enhancements are specific to blue noise distributions.

Shape issues are addressed by Ma *et al.* [MWT11] who places multiple point samples per element and uses an energy-based iterative solver that supports extra terms to capture user requirements (such as orientation fields). Similarly, Landes *et al.* [LGH13], uses relative orientation and the distance between the elements' geometry to reproduce shape-aware textures. [GLCC17] supports distributions of disks, and discretizes different interaction cases in bins depending on the proximity. This allows the system to learn and synthesize distributions where elements are completely inside, mostly inside, and mostly outside others, but does not provide a finer continuous control over their placement. All of these methods that handle distributions of shapes generally also consider correlations between elements across multiple classes.

Element packing can be seen as a specific type of arrangement synthesis, where

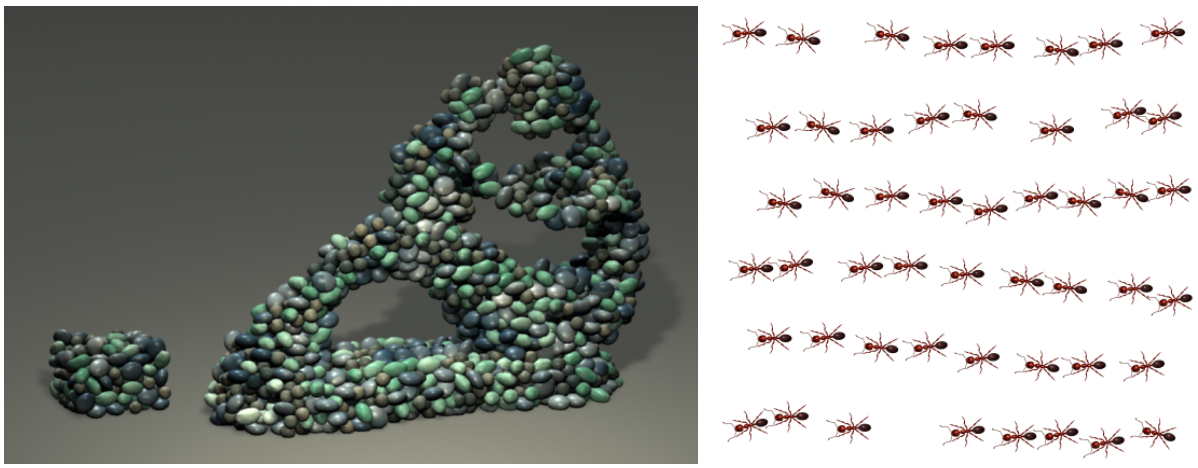


Fig. 2.2: [MWT11] uses multiple samples per element to produce complex shapes (left). [LGH13] instead takes into account relative position and orientation to synthesize distributions of shapes with a defined orientation (right).

elements are placed in a limited space with the objective of filling the space following predefined constraints. Taking shape and size into consideration is often an important feature of such methods due to their action on a constrained space. While some of the approaches to this problem are mostly automatic [GSP+07, XK07], other rely on user input to influence the result. For example, [RRS13] represents elements by their outline and asks the user to manually place three primitives to extrapolate the result, while [HWYZ20] represents shapes as a collection of disks and uses brush strokes from the user to compute the result.

2.1.3 Discussion

Example-based arrangement synthesis can be used to bring control over distributions for many applications. Considering points is not sufficient to analyze and synthesize complex shapes. Therefore, these methods have been extended to support different shapes and more parameters (*e.g.,* orientation). We use this idea, and more precisely the PCF formalism as a starting point for an efficient placement of objects in virtual worlds in Chapter 3. In particular, the smooth curves and their ability to generalize the encoded information make this approach very suitable for applications where a precise placement is required. However, the spatial extent of potentially large objects used in virtual worlds makes the use of standard techniques where only points are considered difficult. After extending PCFs for distributions of disks, we develop a similar approach for the case of distributions of animals in Chapter 4, where the size, orientation and overall shape of the distribution are all crucial for a convincing result.

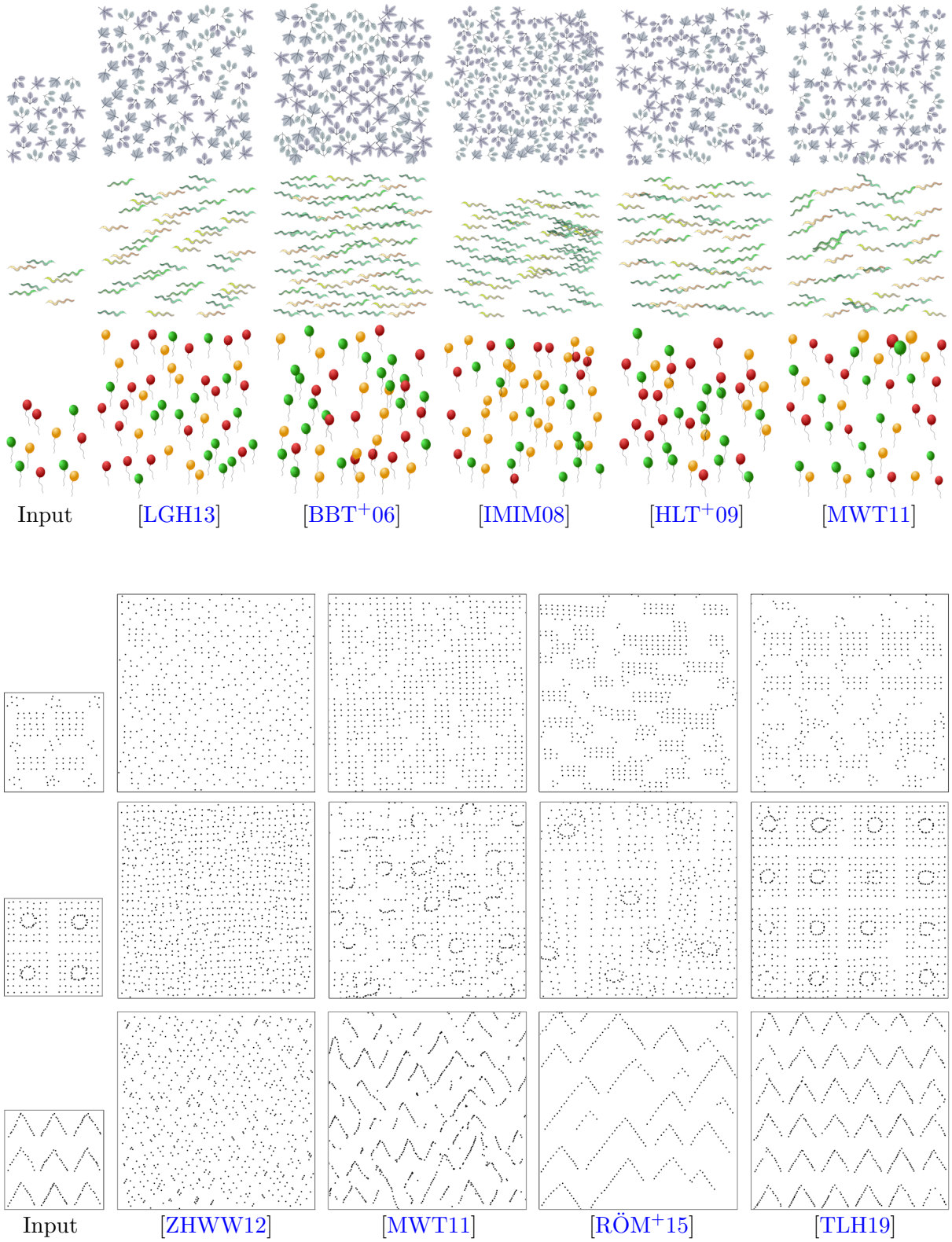


Fig. 2.3: Comparison of distribution synthesis methods for different shapes and regularity. Please see [LGH13] (top) and [TLH19] (bottom) for more examples and details.

2.2 Modeling vegetation

The generation and placement of vegetation has one of the most salient visual impact on natural landscapes. As such, is has been the subject of an important body of work ranging from the generation of geometry for individual plants to coordinated synthesis of entire plant ecosystems, with competition for resources and interactions with the environment.

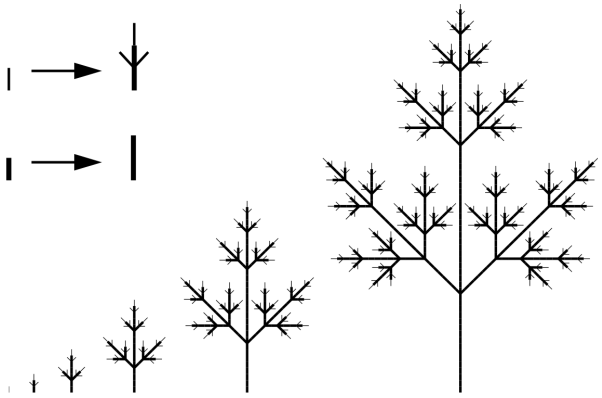


Fig. 2.4: Simple L-system demonstrating plant growth from a simple set of rules [PH95].

2.2.1 Generation of plants

Modeling the geometry of trees and individual plants, while not directly in the scope of this thesis, is a crucial step when bringing life to virtual worlds. While manually creating the models of plants is possible, generative methods can be used to bring diversity, and drastically reduce modeling time.

Fig. 2.5: Bush-like plant generated from L-systems [PLH+90].

One of the first and main approaches to plant generation are the Lindenmayer systems, also known as *L-systems* [Lin68, PLH+90]. This formalism, operating on recursive replacement of strings of symbols following a defined set of rules, has been used to generate wide variety of plants and has been at the foundation of decades of research in the modeling of vegetation. Figure 2.4 shows an example of a simple L-system used to simulate the growth of a plant. The two rules used to define this system are show in the top left of the figure. Starting from a simple stem, the plant is iteratively expanded using the rules until the final plant is generated. It is worth mentioning that while L-systems have been initially designed and used with vegetation in mind, their simplicity and versatility caused the system to be ported to many other areas of research, such as the generation of road networks [PM01] or buildings [MWH+06].

In order to allow interactions of plants with the environment during generation, and as a consequence produce more realistic results, Měch and Prusinkiewicz [MP96] extended this concept to *Open L-systems*. This version of the formalism is based around bi-directional communication between plants and the environment, allowing the simulation of competition for resources and responses to external events (*i.e.,* presence of obstacles or change of day length). Open L-systems have been used for entire plant ecosystems in the work of Deussen *et al.* [DHL+98], by simulating competition between plants and environmental constraints for large numbers of plants.

Different approaches more closely related to simulation have also been developed, particularly adapted to specific plant species. For the generation and growth of lichen, [DGA04] presents a method that first seeds lichen on a 3D object. The lichen is then propagated in a realistic fashion over the model, taking into account space limitations, moisture, lighting, etc. Hädrich *et al.* [HBDP17] model climbing plants as linked anisotropic particles. This allows plants to be physically simulated in real time, making the method prone to user interaction and authoring. By semi-random movement and duplication of the particles, the plants are progressively grown based on their environment and physical constraints.

Fig. 2.6: [HBDP17] uses a particle-based approach to physically simulate climbing plants.

While simulation and procedural methods are designed to produce realistic results, they can be harder to accurately control. L-systems for example, while flexible, require the user to manually enter a set of rules if a specific result is required. To mitigate this problem, some approaches have been designed with the issue of control in mind. Both [PMKL01] and [WBCG09] address this problem by allowing a user to control the output of the method using a sketch of the silhouette of the plant, respectively by treating it as additional constraints for the procedural rules or by a recursive manual design of the plant outline from the full species to individual leaves. Both methods are of course coupled with biological knowledge to ensure a plausible result. Beneš *et al.* [BAS09] propose a similar concept providing control over generated plants by constraining plant growth with 3D meshes instead of 2D sketches. Control and ease of use can also be provided in the form of example-based methods, where the system is tasked with an approximate reproduction of a plant provided by the user [SPK+14]. This concept allows a quick generation of diversity in a scene, provided that the user is able to contribute the initial exemplars to the system.

2.2.2 Simulation

Plant ecosystems have been addressed mostly through the simulation of competition for resources. The Eulerian approach to simulation encodes plant ecosystems in grids,

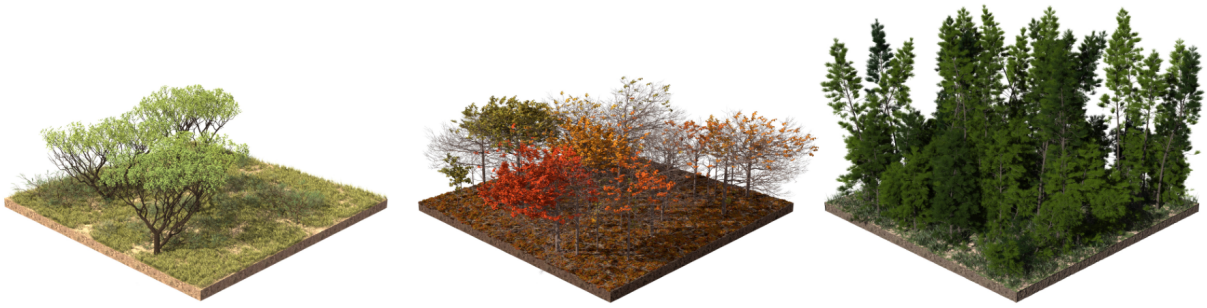


Fig. 2.7: Three plant ecosystems of different biomes generated by [MHS+19], respectively a deciduous forest, a boreal forest, and a rain forest.

and has been used for interaction with natural phenomena [CGG+17], or for modeling plant growth through cellular automata. In contrast, Lagrangian simulations simplifies individual plants as particles in order to compute relationships and interactions. This class of methods allows a precise instantiation of vegetation and will be summarized here.

After Deussen *et al.* [DHL+98] first used the competition of individual plants to generate plant distributions for large landscapes, the idea was extended to multilevel simulations by [LP02]. In this work, the concept of *multiset L-systems* is introduced, allowing the formalism to handle interactions of plants at different scales: the rules used operate on multiple sets of strings instead of one. This is used to represent multiple plants at once, and allows strings to be added or removed from the set, thus providing control over births and deaths in the population. The concept of mutli-scale plant ecosystems was recently further extended to layered ecosystems [MHS+19].

Competition for resources has been expanded to include effects where larger species limit the access to resources of the smaller plants in their vicinity. This process, known as asymmetric competition, is responsible for inhibiting the growth of smaller species when resources are shared. In practice, [AD05] models this phenomenon by expressing zones of influences for each plant, and considering overlaps between different zones as regions where resources are shared. The type and variety of resources considered for competition later increased to take into account elements such as sunlight, temperature, soil type and viability, and free space [Ch'11].

In an effort to combine the benefits of simulation with user control, Bradbury *et al.* [BSK+15] develops both local (plant editing, cut-copy-paste, spatial restriction of simulation, density control, etc.) and global (automatic mapping of manually created species to corresponding 3D models) operators that can be applied to plant ecosystems. [GLCC17] also offers some form of user control over the result, by providing the user with semantic brushes. After painting the desired features such as age, variability or density over the terrain, the user is able to smooth results with a healing brush that bring the different parameters back closer to the underlying terrain conditions.

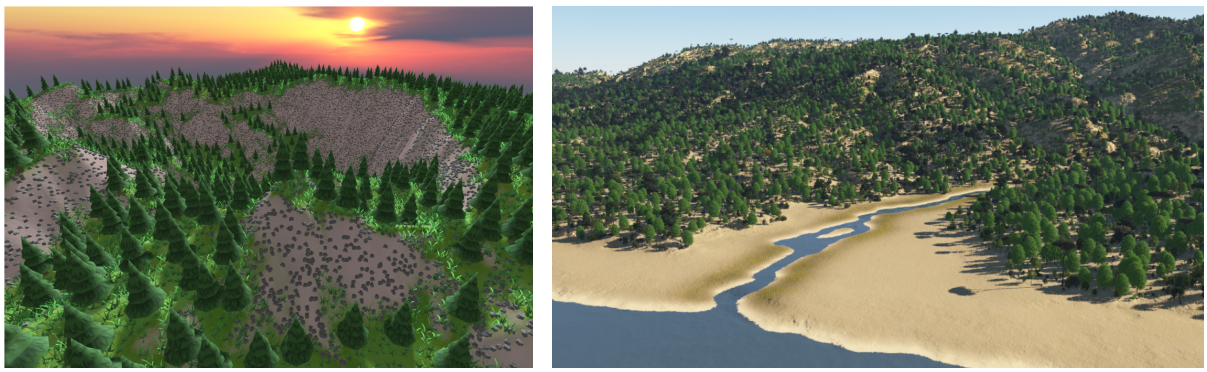


Fig. 2.8: Distribution synthesis applied to virtual worlds. [EVC+15] learns correlations between multiple objects (trees, grass, rocks) and provides a smart brush to allow the user to paint the terrain (left). [GLCC17] takes the size of trees into account for a more accurate placement (right).

2.2.3 Statistical approaches

Instead of running a potentially slow simulation when generating vegetation, some methods opt to directly synthesize the arrangement of plants based on a statistical representation of the target ecosystem. This representation is usually constructed from data or previous knowledge, and is encoded in such a way that makes the reconstruction of this data possible. These methods work on the assumption that it is possible to generate plausible plant configurations without intermediate steps by encoding and reproducing meaningful information about the relationship between plants from the input.

For example, a Wang Tile set, composed of multiple individual tiles with matching borders, is designed to be able to seamlessly tile a potentially infinite plane by successively placing tiles where their borders match. This concept has been adapted for plant ecosystem synthesis, by creating Wang Tiles corresponding to small groups of plants, and using them to efficiently tile a terrain [AD06]. For small details such as mushrooms, grass, twigs and rocks, Guérin *et al.* [GGG+16] encode collision information between elements in a custom structure named Ghost Tile. Once this structure is constructed, the method allows fast instantiation of heavily entangled details aware of collisions without requiring a full simulation.

Worldbrush [EVC+15] does not use an intermediate structure, and directly encodes the interrelationships within and between categories of scene elements (such as rocks, trees, roads, and buildings) as distributions of points. Artists can then use smart brushes to paint these distributions onto landscapes. In this framework, analysis is conducted using an adaptation of point process statistics to small input exemplars and a user-defined hierarchy of classes, and synthesis is achieved with a modified Metropolis Hastings algorithm. Ecobrush [GLCC17] extends this concept to address the problem of ecoplacement – populating landscapes with plants whose attributes (such as species, position and age) are ecologically sound. Here, input examples are automatically generated using sand-box

ecosystem simulations. The focus is on trees and shrubs with potentially overlapping canopies, which represents an instance of the more general problem of analysing and synthesising distributions of overlapping disks. However, Gain *et al.*'s solution is not general: overlap cases are modelled using three extra bins in the Metropolis Hastings distance histograms, to respectively represent complete inclusion, and more than and less than half inclusion. Moreover, since disk position and radii are not jointly analysed, young and mature trees of the same species are separated into different classes, which prevents any mechanism for continuous optimisation of tree radii at the synthesis stage.

2.2.4 Discussion

While the generation of individual plants has been a well studied problem for many years, generating consistent, large ecosystems while providing user control is still a challenge. The methods typically used for the generation of plant ecosystems can be placed on a spectrum starting from pure simulation techniques, which provides high realism but little control, to purely manual methods where an artist has perfect control over the result at the cost of biological consistency. Statistical approaches such as Worldbrush [EVC+15] and Ecobrush [GLCC17] provide a middle ground solution by giving control to the user, while giving them support to guarantee consistency. However, the constraint of user control and interactivity leads such methods to make approximations in their synthesis to reduce computation time. In Chapter 3, we attempt to alleviate this issue by using an example-based method for user control while focusing on providing results that remain as accurate as possible. In particular, we focus on close interactions and overlaps between plants, thus indirectly providing support for complex behaviors such as asymmetric competition and ecological niches. We later step back and focus on the interaction between vegetation and animals to consider a full ecosystem (Chapter 5), and model their long-term placement. At such large scales, we switch to an Eulerian approach, more adapted to the integration of vegetation in a larger, global system where a precise position of plants is not necessary.

2.3 Animation of creatures and crowds

The placement and animation of dynamic elements in a world is often considered a problem standing on its own, with little interaction with other constituents of their environment. While a good portion of the research in this field has been made with humans in mind, some of the approaches can also be applied to animals despite not being initially designed for it. A few methods have also been specifically designed for various types of animals.

2.3.1 Animation of individual creatures

While motion synthesis and control of individual animals is out of the scope of this thesis, it remains a critical point to consider during the process of creating lively and convincing worlds. In practice, this is generally either done manually by artists, or in contrast automated thanks to the use of motion capture tools. However, these approaches quickly become time and resource intensive as the number and complexity of models to animate increase. For this reason, we will only detail here specific automatic and semi-automatic methods that eases the creation of complex animations. For a more detailed overview geared towards quadrupeds, please see [SRH+09].

Fig. 2.9: Various activities can be performed: deadlift, cartwheel, kick [LPLL19].

In order to facilitate the animation process, many methods that operate at different stages of the pipeline and on different types of input have been developed. At the very first step, [RFDC05] present a method to semi-automatically construct a quadruped skeleton using a morphable model. The model, based on manually made reference skeletons, can be easily adapted to different species by selecting a few key points on a reference image of the animal and is then automatically adjusted and ready to be animated. If an animation needs to be directly generated, [FRDC06] and [XWL+08] provide methods to extract animations from videos and images, respectively. They operate on a similar principle of extracting silhouettes of animals with help from the user, and finding an walk or run cycle in the resulting images. [FRDC06] is also able to match a 3D model on the input, while [XWL+08] is geared towards animation on 2D images or billboards.

When learning from image or video example is not an option, animations can also be computed by knowledge or learning based models. [YLvdP07] provides a simple knowledge-based model for biped locomotion control. The approach makes use of a graph of poses in a normal walk cycle, where target angles for various joints are either expressed in world-space or relative-space depending on their role. This, coupled with feedback from the environment, allows the creation of a controller robust to slight variations of the terrain and disrupting forces. The initial publication of [Sim94], popularized the concept of automatically learning motions such as walking, swimming or jumping from the morphology and muscles of a creature. This diverse set of effects and creature was made possible with heavy use of genetic algorithms, used in conjunction with specific fitness values for each problem. Since then, physically and muscle-

based locomotion learning schemes have been continuously improved and now produce results of impressive quality and for many different activities [GvdPvdS13, LPLL19]. Please see the following survey [GPEO12] for a more detailed overview of the developed methods in this field.

Recent advances in deep learning techniques have also been put to use for creature animation. By learning animation cycles and transitions from motion-capture data, controllers that provide realistic real-time animations of humans [HKS17] and quadrupeds [ZSKS18] have been developed. On top of providing lifelike motion controllers that react to user input, these methods are also able to handle variations of the terrain thanks to the variety of input motion-data used as learning examples.

2.3.2 Crowd simulation

Many different approaches have been explored to allow crowds or groups of creatures to move in a realistic fashion. When based on a simulation, most methods developed for this task belong to the family of *microscopic simulations*, where each element is modeled individually using a fixed set of rules. This results into systems based on only a few customizable rules that rely on emergent behaviors to accurately reproduce the complex formations present in real life. However, due to their emergent and evolutionary nature, most of these algorithms tend to be difficult to control or author without being explicitly designed for it.

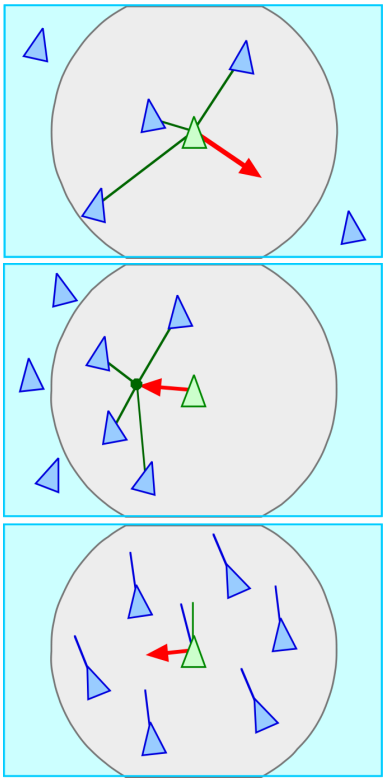


Fig. 2.10: Separation, cohesion, and alignment are the three rules used for a simple steering behavior in Reynolds' model [Rey99].

Force-based The pioneer work in microscopic simulations [Rey87], later expanded in [Rey99], was a force-based model at the origin of subsequent developments and improvements for decades. In this model, tailored to flocks of birds and schools of fishes, flocks are represented as a group of individual entities each governed by a set of rules. The rules are defined as simple computations based on the current disposition of neighbors around each entity, and output a force to be applied to individuals. For a simple coherent behavior, rules representing separation, cohesion, and alignment between individual elements are applied to each entity. Similar models have been specifically developed for pedestrians, representing intentions of individuals as forces [HM95], or more recently modeling individual personalities to increase the range of behaviors [PAB07, DAPB08].

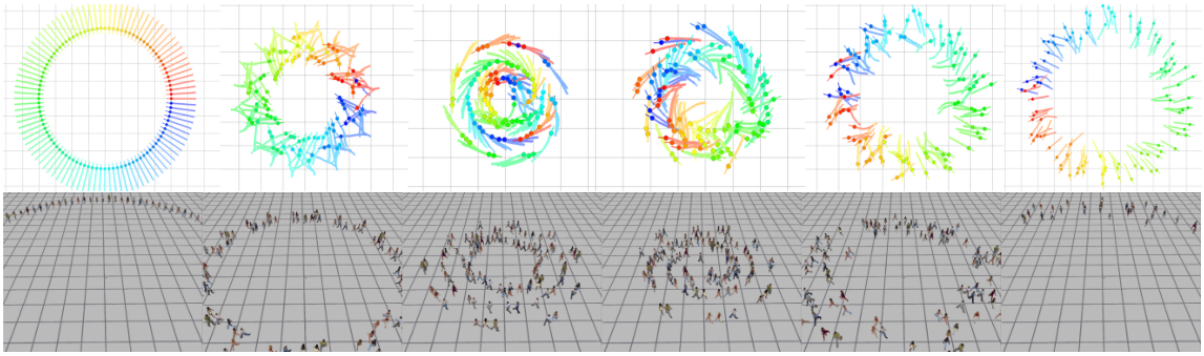


Fig. 2.11: Synthetic vision is simulated to allow agents to use information about their environment and find a path towards their target [OPOD10].

While force-based approaches are an efficient way to model coherent flocks at a reduced cost, they also suffer from limitations: the produced movements are characteristic of this kind of method, which can result in an unrealistic appearance depending on the creature represented. It can also introduce deadlocks and unnatural configuration when dealing with collision avoidance, both between entities and with the environment.

Velocity-based In order to improve the collision avoidance mechanisms of force-based approaches, velocity-based methods [PPD07, vdBGLM11] take into account both the position and movement of entities instead of just their position. From the current position and velocity of neighboring entities, their expected movement is compared against a map of reachable space for the current agent. This allows the computation of sets of parameters that result in a collision free motion, which are then scored and compared to return the best available solution.

Vision-based In an effort to get closer to the actual behavior of human crowds, vision-based approaches mimic the though process of humans navigating crowds or obstacles by providing the agents with synthetic vision [OPOD10, DMCN+17]. Information such as the perceived angle between trajectories or the time to collision are merged and used to detect potential obstacles and find alternate paths to avoid collisions. Alternate implementations of a similar system have been developed using the optical flow instead of a virtual camera [LCMP19], allowing the method to be used by robots and other entities where information about the other agents or environment is not available.

Compared with purely geometrical approaches, these methods tend to produce more realistic behaviors for crowds of humans and allow the formation of patterns such as queues of agents moving in the same direction. However, these results come at the cost of a much larger computational times.

Data-based Data-based methods use trajectory or position information, usually extracted from videos, to configure and calibrate their models. [LCF05] build a graph from clustered input video clips, which encodes the ability to smoothly transition from one formation to another. Simpler force-based models such as [Rey99] that was presented previously are then used to fill in the gaps between states. Other methods [LCHL07] also learn formations and trajectories from input video clips, but provide their own process to later reproduce the learned features. The different behaviors learned this way can also be blended together to produce smooth transitions between formations [JCP+10].

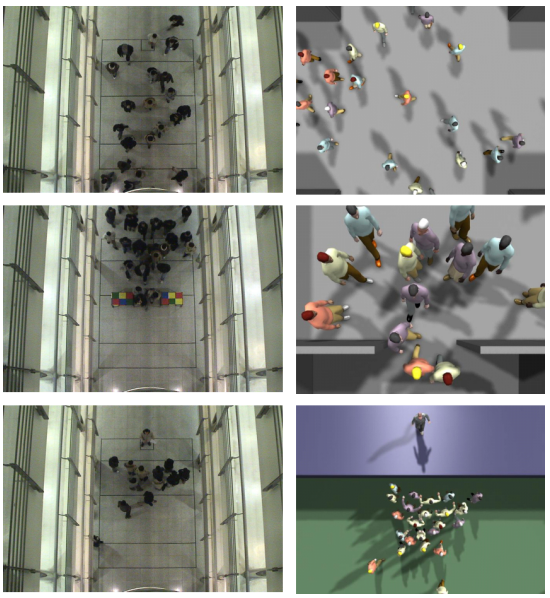


Fig. 2.12: Crowds in various situations (right) are reproduced from input video data (left) [LCHL07].

One major downsides of such methods is the difficulty to gather input data. Indeed, they are often manually produced by filming test spaces where humans interact in order to provide favorable and controlled conditions for later treatment. This largely limits the quantity and variety of data available, which, coupled with the difficulty such models have with handling previously unseen situations, reduces their usability.

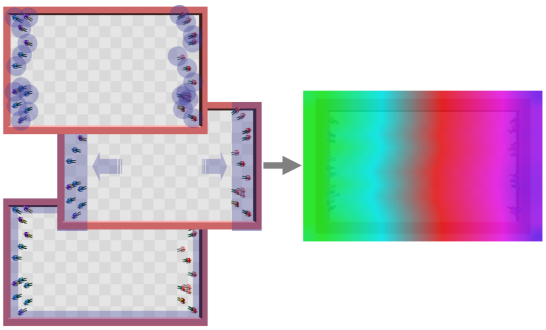


Fig. 2.13: Density, goal and boundary grids (left) are used to compute a potential field (right), in turn used to guide the crowd [TCP06].

Macroscopic simulation Contrary to microscopic simulations, a few projects take the opposite approach of considering crowds as a single, global phenomenon instead of solving constraints individually for each agent. Macroscopic simulations have modeled crowds as a "thinking fluid" [Hug03], or as dynamic potential fields [TCP06] that are then used to advect entities by using its gradient.

While these models are a good fit for highly dense crowds, they are not relevant for smaller individual groups or outlier behavior. Indeed, outliers break down the concept of considering the entire crowd as one, which also make it difficult to have precise control and recreate specific formations within the crowd.

2.3.3 Crowd animation

While the objective of crowd simulation is to replicate behavior as realistically or efficiently as possible, crowd animation instead aims at directly solving one specific crowd-related problem. The problems that can be solved this way can be related to the ease of use by a user, global efficiency of movement, and so on.

Animation patches have been widely used for this end in the field of crowd animation. They represent small clips of predefined animations created specifically for their interesting features such as an interaction between characters or an ability to be looped indefinitely. Patches are often used in part to offset most of the computations - the generation of the patches - as preprocessing, leaving only the need for them to be played back at runtime. This can drastically reduce the resources required for a scene, and allows the creation of arbitrarily large scenes at a small computational cost.

[SKSY08] uses patches to represent an interaction between two characters. These patches can be played simultaneously to emulate interactions between more characters, and chained to one another to create longer sequences of animations. Control of a character can also be given to the user, letting the system the freedom to place the patches needed for appropriate actions with other entities. The idea is developed further by [KHHL12], by giving patches a spatial presence in the virtual world, with entry points, interaction constraints, and a polygon representing the area of effect of the patch. This makes it possible to interactively deform patches by acting on the entry and exit points, effectively increasing the diversity of the system. The projection of the patches also allow for the detection and avoidance of collisions with the environment, for a more dynamic result.

Fig. 2.14: [JPCC14] uses patches of looping crowd animation to seamlessly animate large urban crowds at reduced costs.

Patches have also been used for the creation of collision free, large-scale environments such as cities. In the work of [JPCC14], patches are used to represent a looping animation of many pedestrians walking in and out of a limited section of space. Patches can be connected or swapped out if their exit and entry points match both in position and timing, preventing the end-user to see repetitions and obvious patterns. This formulation of patches also makes it possible to provide the user with an intuitive sculpting interface to expand, shrink, cut, connect and rotate patches. The patches are created and updated to fit the user constraints while retaining their original properties. This framework is adapted in [JCC+15] as a global tool, where the user instead paints the requested density and directions on maps. Here, the patches become part of an optimization problem where all constraints get progressively satisfied as optimization progresses. The result is an animation of a large environment populated

with pedestrians, with constraints completely editable by the user and satisfied at every moment of the animation.

The topology of a scene can also be used as a tool for crowd animation. By first computing its harmonic field, [BSK16] extract a Reeb Graph of the topology of the scene, as well as a collection of guide lines around obstacles. The Reeb Graph is then used to compute the maximal capacity of every available path, and deduce an optimal partition of pedestrians on different routes. This can be used to compute a perfect evacuation plan or routing system for complex environments.

2.3.4 Path planning

Path planning attempts to find a path as efficient as possible within the environment, allowing an agent or group to reach its goal. While the term encompasses more traditional methods used for pathfinding, path planning will be used here to refer to the narrower field focusing on challenges specific to the navigation of groups or the interactions with complex environments. We refer the reader to a survey on path planning, and more broadly human motion trajectory prediction for more details [RPH+20].

Many path planning algorithms rely on the use of roadmaps, or navigation meshes, which provide important information about the available paths in the environment. [BBJA02], for example, constructs a roadmap specifically designed to accommodate the passage of standard Reynolds-like [Rey99] agents. Multiple types of roadmaps can also be layered to achieve a convincing result on dynamic environments [KBG+13]. In this case, one navigation mesh is precomputed and only considers static geometry to provide a global approximated path towards the goal. A second mesh is focused on dynamic elements, and is updated as the simulation continues. It is used along with the static navigation mesh to compute a more accurate local version of the path, adapted to previously unseen obstacles. Navigation meshes can also be created with other attributes in mind, such as the local density of regions [PGT08, vTCG12] or their topology [vTP19].

For small groups or single individuals, an important focus is made on the interaction between the object of the path planning and the environment. Small groups can for example be represented as a deformable or hinged box [KO04] to simplify the computation

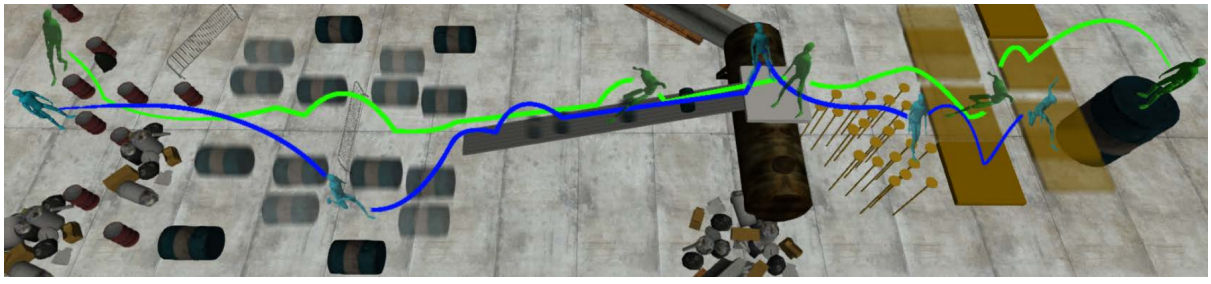


Fig. 2.15: Two agents finding a path in a complex, dynamic environment [KBG+13].

of valid paths. In this approach, the box can be either deformed and rotated, or bent to accommodate for less accessible zones of the terrain. Once a path is found using the simplified box, individual agents can be instantiated inside it, guaranteeing their ability to fit as a single group through the computed path. Other models [KO12] focus not only on the interaction with the environment, but also between different groups of humans.

2.3.5 User control and authoring

Specific group animation methods have been developed to address the problem of user control and authoring of animations. They often focus on the ability for an end-user to specify visual features that the group will exhibit. While the target visual requirements could in theory be varied, they are in practice often limited to the shape of a group, with a few approaches paying special attention to the transitions between shapes or their density. In most cases, this problem can be split in three main stages: sampling the position of entities on the target shape, finding correspondences between the source and the target shape, and finally animating the individual entities from their position in the source shape to their assigned position in the target shape.

A user-specified target shape can be populated with entities with the help of a geometrical representation. This is usually done by computing a Voronoi tessellation [ZZC+14], or a Delaunay triangulation [WZDJ14, XWY+15] of a given model. In these works, [ZZC+14] also provides control over the density of entities by taking into account a density function when computing the centroids of the Voronoi tessellation.

Other geometry-based approaches place a distinction between the inside of a shape and its outline, on the assumption that the entities lying on the edge of a formation are what ultimately define the shape, thus requiring special attention for optimal results. [XJY+08], designed to model flocks of small flying animals, only places entities on the surface on the 3D target mesh. After randomly sampling the surface of the mesh and computing final positions using an energy minimizing iterative algorithm, entities are matched by spherically projecting positions from one shape to the surface of the other, and matching closest points. The work presented in [GD13] fills the interior of shapes

Fig. 2.16: A crowd changes its shape from an ape to a human while following a curve [XWY+15]. An overview (top) and close-up (bottom) are shown.

instead of just their outline, but does so by first carefully placing entities along the edge to accurately represent the formation, before running a custom discrete flood-fill algorithm to place remaining entities.

Other methods also provide efficient tools for user control without following the traditional sample-match-animate pipeline. A spectral approach is used instead to handle this problem in [TYK+09]. Formation are represented, and reconstructed, from the Laplacian matrix of the Delaunay triangulation of the group. This formulation is used the encode the adjacency relationships between close entities, and can be interpolated later on to produce smooth transitions between formations. [HSK14] also stands out when compared with more standard pipelines, by using a mesh-based structure to represent crowds. The mesh can be interactively manipulated by the user with the help of a multi-touch device, allowing them to control the shape of the crowd and its pathway through the environment at runtime. The mesh, directly affecting the members of the crowd that it represents, is automatically distorted to interact with the obstacles while following the user-provided constraints as closely as possible.

2.3.6 Discussion

We once again observe a spectrum similar to that of vegetation synthesis methods. On one hand, agents in microscopic simulation methods have the potential to develop very realistic behaviors, but are very difficult to control because of their large set of parameters and reliance on emergent behavior. On the other hand, methods geared towards control provide artistic freedom to creators, but often present a realistic result in limited scenarios such as marching bands or military formations. In this thesis, we extend control oriented approaches to the case of large terrestrial animals and pay a particular attention to issues specific to animal herds, including preservation of the overall shape of the herd, relative orientation of animals and distribution of density (Chapter 4). We study the behavior of animals at a larger scale in Chapter 5, where we focus on computing the long-term presence of animals based on their environment and model interactions in the form of a simplified food chain. We also provide a visualization tool to interactively explore an ecosystem with plants and animals, and allow the user to observe the daily activities of herds.

2.4 Ecosystems and self-interacting models

Very little research have been dedicated so far to the modeling of interactions between a terrain, vegetation, animals, and other phenomena. For specific problems, this issue can be alleviated by visually displaying the result of the interactions without the cost of simulation. When this is not possible, full systems have been developed to simulate specific interactions. However, they are often restricted to feedback between vegetation and environment or vegetation and animals.

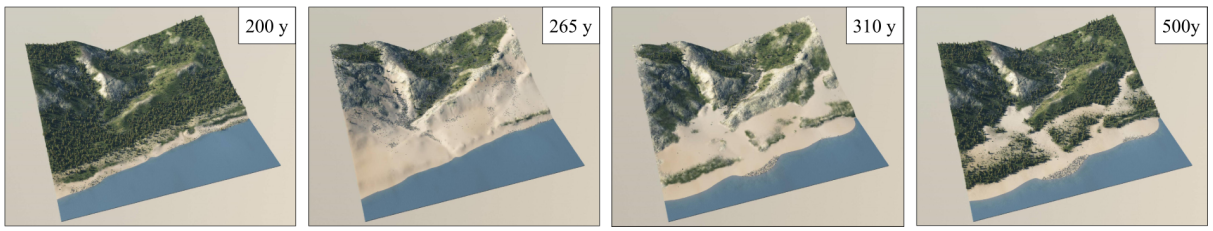


Fig. 2.17: [CGG+17] simulates the evolution of the terrain and vegetation. Here, the user successively adds sand and fires (200y), rainfall (265y), and humus (310y) before resuming the simulation.

2.4.1 Joint modeling of terrain and vegetation

Some methods attempt to go beyond raw landscapes, and emulate the result of complex interactions by adding specific details to the scenes. For example, [GPG+16] develops a framework for the synthesis of details such as snow cover, icicles, fallen leaves or grass tufts. This is achieved by the use of a database of environmental objects, able to act on its surroundings through effects (addition of geometrical details) and fields (modification of local variables, such as heat or humidity). As described in subsection 2.2.3, the addition of details was also studied by [GGG+16] through the use of a custom collision structure allowing efficient placement of entangled details.

At a larger scale, Benes *et al.* [BMJ+11] models the interaction of urban layouts with plant competing for resources, while considering human urban management. By computing manageability levels of city block, which quantify if a zone should be seeded in a wild or controlled manner, the system is able to generate a balanced urban ecosystem with plants and buildings.

A model handling the interaction between vegetation and other phenomena, including terrain erosion, was proposed by [CGG+17]. Cordonnier *et al.*makes use of a layered representation of the different resources present on the terrain, and introduce a stochastic simulation where individual events are simulated one at a time, many times. This formulation allows many different event types to be integrated in the system, such as rainfall, fires, lightning, or tree growth. Control is also provided to the user in the form of local editing of material quantities or properties when the simulation is paused.

2.4.2 Interactions between animals and vegetation

With the objective of improving realism and control over visual scenes, [BE03] proposed the use of agents to simulate the action of humans and animals over vegetation. By equipping each agent with sensors and a set of habits defined in a configuration file, they are able to reproduce the effects of actions such as watering, cutting or walking on plants. Using agents with different profiles, they show the impact of various actions on a resulting plant ecosystem. Ch'ng [Ch'13] also adopts an agent-based system to simulate

the interactions between populations including vegetation, herbivores and carnivores. If these agent-based approaches are able to model the variations of populations, they unfortunately require manual tuning of a large number of parameters to properly function. In their current state, they also lack means to take into account variations of the terrain and other more high level constraints needed to model real animals.

More specific models have also been developed to model the effects of insects on vegetation. In particular, [PH95] presents a model for plants based on L-systems that can progressively wither as insects eat their apical buds. Variations in the behavior of insects can be added to this system to model different or more complex behaviors. The concept is further developed by Hanan *et al.* [HPZS02], where both plants and insects are modeled using the same formalism of L-systems. This facilitates the expression of interactions between insects and vegetation. Specifically, this work takes into account the defense mechanisms of plants when modeling movement and interactions of insects.

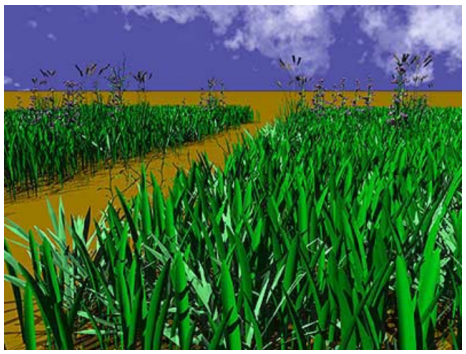


Fig. 2.18: A plant ecosystem after being affected by a herd of cows [BE03].

2.4.3 Biology-inspired models

Biologists modeling ecosystems designed predictive models for ecological niches of species, allowing decision making for global resource management. We refer to research on the impact of biodiversity on the workings of an ecosystem [Lor10], motivated by conservation biology for ecology and environmental analysis, and to [SHL+17] for a recent comprehensive survey on dynamic ecosystem models of a forest landscape. We also refer to the book [MNR+17] for comprehensive insights into mathematical modeling of ranging patterns and occupancy dynamics.

More abstract models, like the Dynamic Energy Budgets Theory (also known as DEB Theory) [Koo10], have also been developed based on biology. It is a unified framework used to describe and model the different aspects of metabolism, and has already seen many applications. Since this framework operates at the level of mass and energy requirements, it can be used indifferently to model plants and animals, as well as their interactions. However, the abstract nature of the theory makes it difficult to be used in Computer Graphics, where specific spatial information about the ecosystem is required. This has been attempted before [MLM+11], but, to our knowledge, has not led to a finalized product.

2.4.4 Discussion

While some methods include interactions between multiple elements of a complete ecosystem, they are unfortunately often limited to a single interaction type (*e.g.,* vegetation with environment [CGG+17] or vegetation with specific animals [BE03]). A tool allowing the authoring of a complete ecosystem with plants and animals, while taking into account their interactions and impact on their environment has, to our knowledge, never been created in Computer Graphics. We propose a unified framework designed for this purpose in Chapter 5, and with combined goals of user control, visualization, and the ability to exploit preexisting ecological data (detailed in Appendix A) to guarantee consistency.

2.5 Conclusion

Research in modeling of ecosystems has seen extensive advances on multiple fronts in the past decades. However, methods providing efficient authoring tools are scarce, and often rely on a trade-off between control and plausibility. We attempt to bridge the gap on this front by developing both easy to use and accurate methods to tackle the authoring of static arrangements of elements (Chapter 3), and dynamic groups such as herds of animals (Chapter 4). In an effort to unify different areas of research related to ecosystem generation, we also develop in Chapter 5 a general pipeline that can be used for the authoring of complete ecosystems. This includes the modeling of interactions between the environment, vegetation, and multiple species of animals along the food chain.

Chapter 3

Object placement in static landscapes

Contents

Creating arrangements of shapes in the plane (also known as 2D distributions) is a common requirement in different Computer Graphics problems, be it for vector-based texture synthesis of semi-regular patterns or for populating virtual worlds with varied elements, from rocks to vegetation. Building on recent developments in statistical analysis and synthesis, distributions can be learned from exemplars and seamlessly recreated over larger regions, either automatically or by painting with interactive brushes [HLT+09, EVC+15].

However, the application of point statistics to shape distributions has inherent limitations: the spatial extent of shapes cannot be expressed, so that undesired shape intersection may be inadvertently introduced in dense regions. Conversely, the presence of meaningful overlaps in the input (for instance, the occurrence of undercanopy plants beneath sheltering trees) simply cannot be captured using points distributions. This problem – where arrangements of shapes have different forms of 2D bounding circle intersection – was recently identified, leading to a first model for disk distributions [GLCC17]. Unfortunately, the handling of nested and overlapping cases was limited to a few extra bins in a distribution histogram, making it impossible to accurately describe the full range of overlapping configurations. Furthermore, while continuous representations such as Pair Correlation Functions (PCFs) [ÖG12] have been shown to improve the stability and robustness of point distribution synthesis, these methods have yet to be extended to either multi-class or extent-aware distributions.

In this chapter, we present an accurate and robust method for the analysis and synthesis of multi-class distributions of potentially overlapping disks, and demonstrate its applicability to populating virtual environments with 2D or 3D shapes. We also extend previous PCFs formulations to enable analysis in arbitrary domains, making them robust to noise or missing data.

A Pair Correlation Function (PCF) is a continuous curve that describes the density of neighboring points as a function of distance from any given reference point. Our method builds on these PCFs to achieve robustness in the multi-class setting. We first present a

License: Jim Champion, CC BY-SA 2.0

Fig. 3.1: Examples of natural scenes with complex interactions: fungi on the edge of a stump and in small clusters (left), lily pads with varying overlaps on a lake (right).

simple extension of PCF analysis and synthesis to allow for dependency graphs between multiple classes of disks, thereby representing objects with fundamentally different behavior in an arrangement. Since capturing spatial interactions within and between such classes requires an awareness of both disk location and radius, we also introduce a new metric for disks. This metric meets the differentiability requirements of PCF but is also tailored to distinguish between perceptually different key configurations, such as nested, tangent, or bordering disks. Finally, while retaining the global convergence of the mean PCF, we improve the visual quality of synthesized distributions by constraining each individual PCF to a validity region, based on the variance of input PCFs. As our results show, this framework leads to major improvements in the state of the art for capturing general disk distributions.

Our main technical contributions include:

- A novel formulation of PCFs suited for analysis of distributions with missing data or irregular domains. This broadens the range of exemplars that can be used as input, and is demonstrated by two applications to distribution inpainting and decomposition.

- An extension of state of the art methods for point distribution analysis and synthesis to multi-class data. This is a crucial consideration when synthesizing hierarchical or structured data, as it allows the user to model the relationship between different categories of objects, substantially improving visual fidelity.

- A new metric for evaluating pairs of disks, adapted to disk distribution synthesis and normalized to differentiate between key configurations in practice. In particular, our metric efficiently takes into account any existing overlap between shapes as well as nested configurations.

- The introduction of variance-aware PCFs and associated error handling routines. This improves synthesis quality, especially in highly-constrained cases with semi-regular patterns. Variance-aware PCFs limit the most prevalent problem in dealing with traditional PCFs, namely the loss of information that arises from solely considering the average curve.

We will discuss each of these contributions in a separate section, after a brief overview of the necessary technical background on analysis and synthesis of point distributions with PCFs. The final two sections are dedicated to our results and concluding remarks.

3.1 Technical background

In this section, we cover the technical background related to the Pair Correlation Function (PCF) framework as presented by Öztireli and Gross [ÖG12]. We first provide global overview of the method, discussing the input data and assumptions, and then detail the different equations and steps required for analysis and synthesis of distributions.

3.1.1 Data and assumptions

The PCF framework captures point-based distributions by considering every point in a stationary exemplar in turn and building an average density of surrounding points at different distances. This is encoded as a density function (the *Pair Correlation Function*) normalized according to the number of points in the exemplar, and which intuitively represents the variation in the number of neighbors around each point as a function of increasing distance. The main strength of PCFs is that they represent distributions as normalized continuous functions, which are amenable to derivative-based optimization, such as by gradient descent, at the synthesis stage.

A PCF also provides an easily interpretable visual signature that corresponds to the perceptual characteristics of the underlying distribution. Figure 3.2 illustrates this expressivity: while the PCF of a random distribution of points (top) is mostly flat, the one obtained from a blue noise distribution (middle) shows a very characteristic pattern of ripples of decreasing magnitude, with a global maximum located at the average distance between a point and its closest neighbor. With clustered distributions (bottom), the PCF features two dominant peaks: the first at the mean distance between points within a cluster, and the second, less pronounced, at the mean distance between points in neighboring clusters.

By design, PCFs operate on a set of assumptions related to the data that they represent. Firstly, they are in their original formulation aimed at stationary and anisotropic data. This is the case because PCFs average contributions from all points regardless of their location in space, apart from the distance between points. Since they average the

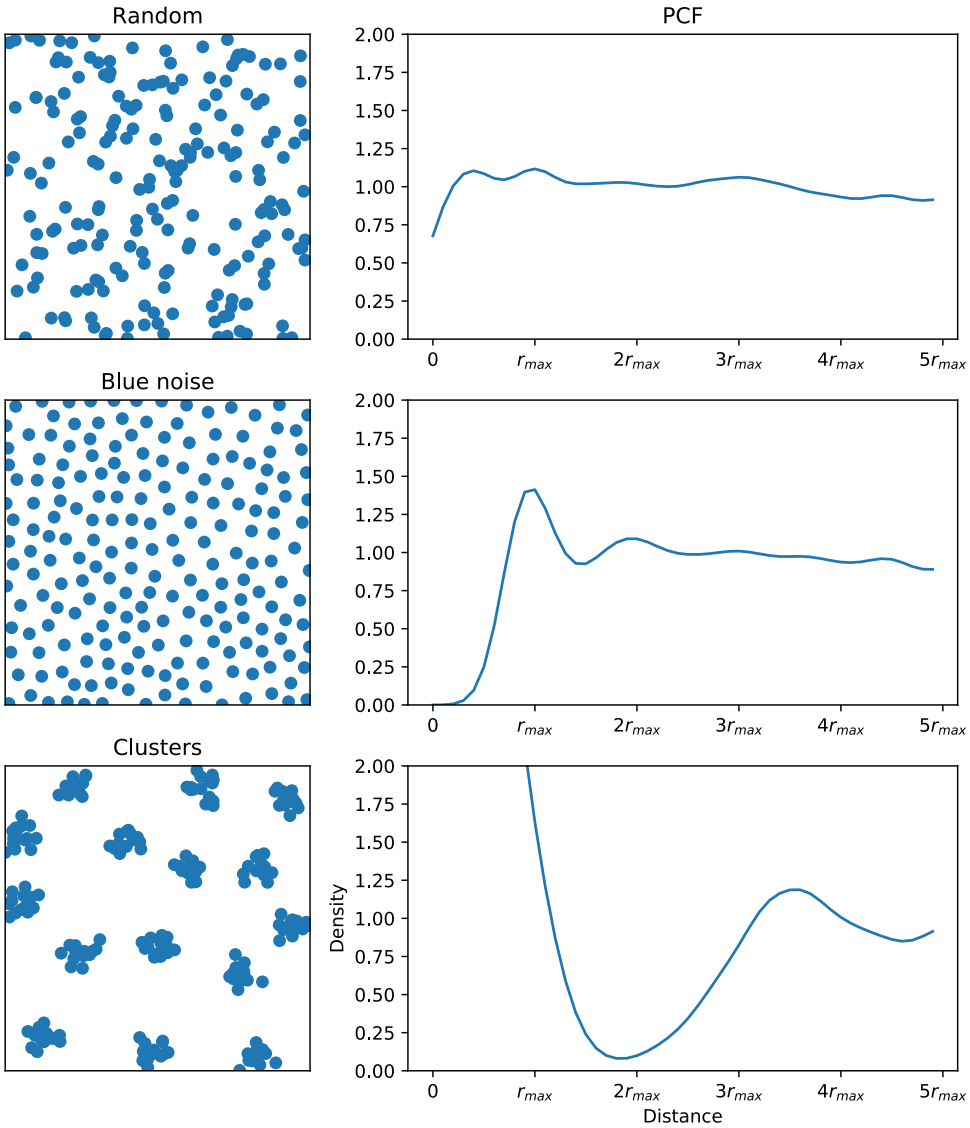


Fig. 3.2: Three typical point distribution exemplars (left) and their associated PCF representations (right).

contributions of nearby points, PCFs also rely on a few properties related to the number of input samples. A high sample count is not required to achieve a good result, but there is a minimum soft limit of a few dozen points needed to actually generalize the underlying distribution instead of overfitting the data. This tendency to generalize data can also become a problem with regular patterns such as grids. In such cases, it becomes very difficult for PCFs to actually produce accurate results, since they can smooth out important details or degenerate into overconstrainted problems at the synthesis stage.

3.1.2 Analysis and synthesis of point distributions with PCFs

Computing PCFs: As in standard point processes, the underlying principle of PCF derivation is to count neighbors within rings of increasing radius around each point. However, two modifications are required in order to generate density invariant, continuous curves:

First, distances are normalized according to overall exemplar density. Given an input exemplar with n points that, without loss of generality, occupy a unit square, distances are divided by a value r_{max} based on how far points would be placed if they were maximally spread to occupy the whole input domain. This value, which uniquely depends on the mean density of points in the exemplar, is given by (see [LD06]) :

$$r_{max} = 2\sqrt{\frac{1}{2\sqrt{3}n}} \tag{3.1}$$

Second, to ensure the function is continuous and robust to noise, the influence of a neighboring point is spread using a Gaussian Kernel $k_\sigma(x) = \frac{1}{\sqrt{\pi}\sigma} e^{-x^2/\sigma^2}$, centered at the distance r from the reference point. In our work, we use the following simplified version of the original PCF estimator [ÖG12], specialized for the 2D case and expressed over a unit square domain:

$$\mathrm{PCF}(r) = \frac{1}{A_r n^2} \sum_{i \neq j} k_\sigma(r - d_{ij}) \tag{3.2}$$

where A_r is the area of the ring with inner radius $r-1/2$ and outer radius $r+1/2$, and d_{ij} is the distance between reference P_i and neighbor P_j. As in [ÖG12], we compute PCFs in a relatively big neighborhood of P_i, depending on the density. In our experiments, r takes discrete values from zero to kr_{max}, where k and the discretization step are two prescribed constants.

We often refer to this PCF estimation as "mean PCF" in the remainder of this paper, to distinguish it from the so-called "individual PCF" associated to each P_i, and estimated as follows:

$$\mathrm{PCF}(P_i, r) = \frac{1}{A_r n} \sum_{j \neq i} k_\sigma(r - d_{ij}) \tag{3.3}$$

As we will see in Section 3.5, considering individual PCFs in the synthesis procedure results in a more accurate synthesized distribution.

Boundary handling: Since the input exemplars may have a limited number of points we cannot afford to simply discard those near the boundary. In the past, Emilien *et al.* [EVC+15] compensated this bias by weighting the contribution of every point, and for each ring radius r by the ratio between the length of the circle inside the domain and its full perimeter. The weighting factor w is computed as follows,

$$w_i(r) = \frac{2\pi r}{l_i(r)}, \tag{3.4}$$

where $l_i(r)$ is the length of the circle centered in P_i of radius r contained inside the domain. This weight is designed for square domains, and accounts for missing points on the portions of circles outside the domain. Effectively, this is used to compensate for the fact that points located near the edge of the domain have fewer neighbors than those further in. We present a more complete handling routing adapted to arbitrary domains in Section 3.2.

Distribution synthesis from PCFs: Following [ÖG12], synthesis is achieved in two steps:

An **initialization step** computes a first-pass solution using generalized dart throwing to set an initial position for n points in the target domain (with n computed from the exemplar mean density and target area). This step operates by successively creating and accepting or rejecting new random placements until the required number of points is reached.

Each candidate point is only retained if the revised PCF error remains below a threshold ϵ_m (set to increase with the number of accepted points m). The error is computed as $E_{init} = \max_r(PCF_{new}(r) - PCF(r))^+$ where x^+ indicates that negative values of x are truncated to 0. This forces the algorithm to either generate a curve that remains below the targeted PCF if possible, or exceed it by less than the tolerance threshold ϵ_m. In practice, this initialization step leads to a PCF reasonably close to the target one.

A **refinement step** is then applied, during which point positions are adjusted using gradient descent to better fit the current PCF to the target. We use a least squares cost function $E_{ref} = \sum_r(\mathrm{PCF}_{new}(r) - \mathrm{PCF}(r))^2$, leading to the following normalized gradient at point P_i:

$$\Delta_i = -\frac{\sum_{i\neq j} \mathbf{u}_{ij} w_{ij}}{\sum_{i\neq j} |w_{ij}|}, \tag{3.5}$$

$$\mathbf{u}_{ij} = \nabla P_i d_{ij}, \tag{3.6}$$

$$w_{ij} = \sum_r \frac{\mathrm{PCF}_{new}(r) - \mathrm{PCF}(r)}{r}(d_{ij} - r)k_\sigma(d_{ij} - r) \tag{3.7}$$

In turn, each sample P_i is moved along the associated Δ_i by a random distance in $\{10^{-1}, \ldots, 10^{-5}\}$ and only the single move that most reduces the error is retained. This scheme is repeated until convergence.

Let us also mention that, by definition, the PCF (and so the defined error) depend on the used metric which indicates the corresponding neighborhoods around each point of distance r. We will specify the metric we use in our method for disk distributions in Section 3.4.

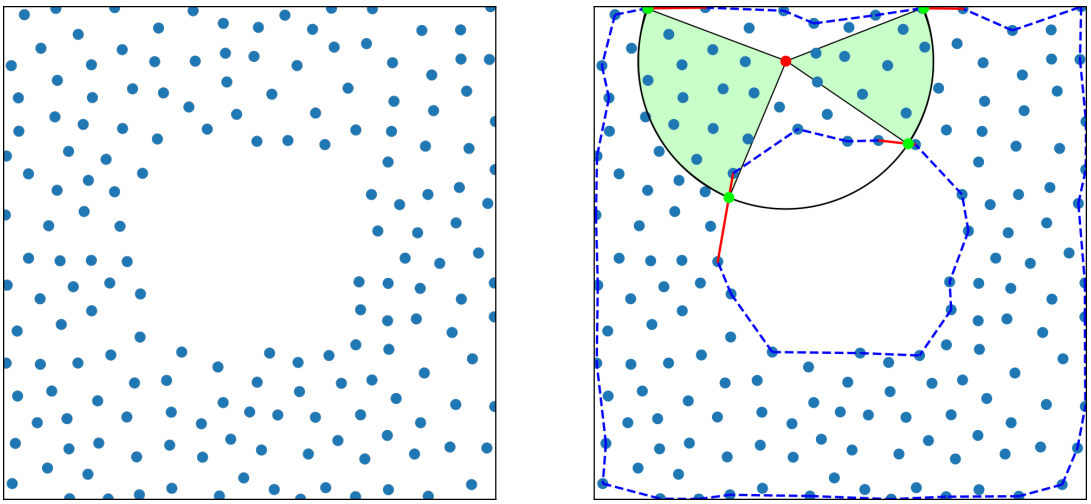


Fig. 3.3: Incomplete point-set (left) and the extracted free-form boundary (dotted lines, right). To compute the PCF of the red point, we sum contributions from areas of the circle with an arc inside the domain (light green). These areas are delimited by the intersection (green points) of the free-form boundary (red lines) with each circle.

3.2 Learning from arbitrary domains

In this section, we introduce a new simple and intuitive solution for PCF analysis with free boundaries. We rely on the PCF formulation in Equation (3.3), weighted to compensate the effects of a square domain with the term w presented in Equation (3.4). As this weight formulation was introduced for the specific case of square domains, our objective in this section is to extend it to domains with free-form boundaries.

3.2.1 Compensation of missing points

To generalize the weighing term in Equation (3.4), we first approximate the boundary of the input by computing its Delaunay triangulation and filtering the longest edges. We do so by going through every vertex and discarding the incident edges considered too long according to a local criterion: only the edges shorter than t times the shortest edge are kept, where t is a user-defined threshold. In our implementation, we found that $t = 3$ works well, even for non homogeneous point clouds (e.g. white noise). We then extract the boundary of the point set from the resulting triangulation, by stamping the edges contained in each triangle and filtering out those stamped more than once. Finally, we compute the intersection points between the circle of radius r and the estimated boundary, and the corresponding angles $\Delta\alpha_i$ (see Figure 3.3). We determine which angular portions lie inside the domain by selecting a point on the current circle with an angle only slightly larger than one of the computed angles, and determine whether it lies inside the boundary. The new weighing term is set to:

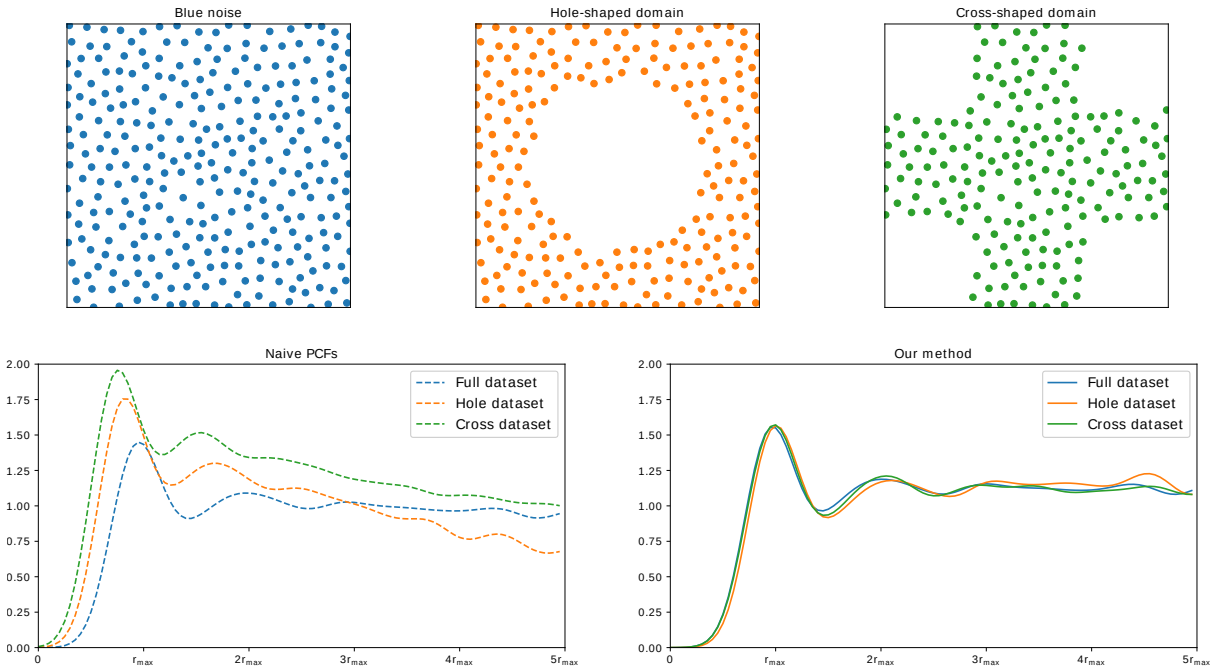


Fig. 3.5: Analysis of complete vs. incomplete data-sets. Contrary to standard PCFs (bottom left), we get consistent curves (bottom right). This is the desired result since the distribution did not change - the domain only was modified.

$$w_i(r) = \frac{2\pi}{\sum_i \Delta\alpha_i} \tag{3.8}$$

3.2.2 Quantitative results

Figure 3.5 compares our new PCFs with standard ones, on complete vs incomplete exemplars, showing that the mean PCF we extract does not depend on the boundary anymore. In addition, we computed the average standard deviation of individual point PCFs for these three data-sets, shown in Figure 3.4. The lower standard deviation for our method further confirms its relevance to handle arbitrary domains: While results are similar for the first exemplar (a), we significantly reduce variance in complex cases (25% decrease for cross-shape (c)).

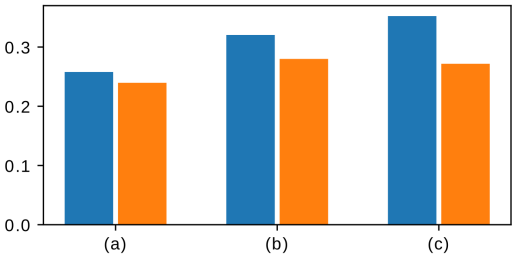


Fig. 3.4: Comparison of the average standard deviation of individual PCFs for regular (blue) vs. our PCFs (orange) on full (a), hole-shaped (b) and cross-shaped (c) domains.

In terms of efficiency, our method is indeed slower than regular PCFs due to the need

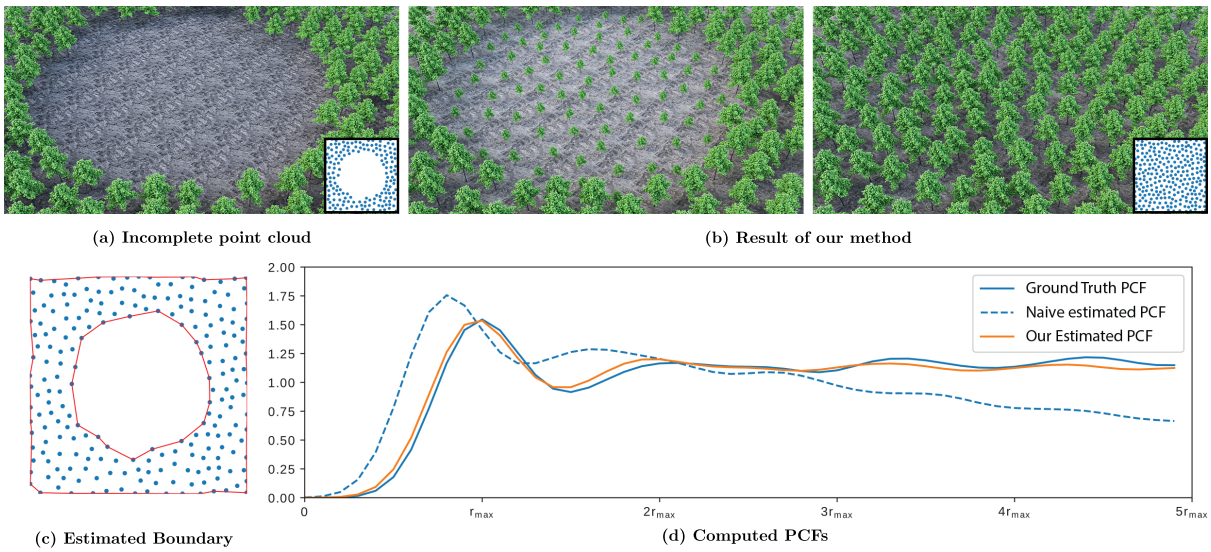


Fig. 3.6: Application of extended PCFs to the inpainting of distributions, illustrated by the re-planting of a partially destroyed forest (a,b). From an incomplete dataset (a), our method accurately estimates free-form boundaries (c) and compensates from missing points when learning PCFs (d). This allows us to restore the perceived initial distribution at the synthesis stage (b).

to compute a Delaunay triangulation, to filter triangles out, and to compute the weight of each point. In practice, we achieve around 9 seconds for 300 points in a unit square domain (a), 7 seconds for the domain with a hole (b), and 6 seconds for the cross-shaped domain (c), compared to between half and a quarter of a second for standard PCFs. Our algorithm was implemented on the CPU in Python and could be optimized. Timings were computed on an Intel® Core i5 clocked at 3.3GHz with 8GB of RAM.

3.2.3 Application to distribution inpainting

Filling holes in a distribution with missing data is an immediate application, and a good illustration of this extension. After computing boundary-independent PCFs, we simply add samples to the input point-set such that each added point approximately maintains the extracted PCF, and then fine-tune their position using gradient descent (similarly to the synthesis stage in [ÖG12]). Figure 3.6 shows that a dataset with significant missing data is sufficient to successfully recover an unbiased representation of the input, and then seamlessly reconstruct missing parts.

3.2.4 Application to distribution decomposition

Our method can also be used to decompose a complex, unlabeled point-set into coherent classes in terms of distribution, also enabling to analyze extra-class correlations. While

different methods have been developed for similar problems [PGMZ12, SP19], our framework brings a new and simple viewpoint to such decomposition problems. We use our boundary-handling method to compute the PCFs of individual points, and then cluster them by applying k-means in PCF space, where the number k of clusters is preset by the user. Two clustering results are shown on Figure 3.7. The use of our PCFs with free-form boundaries allows us to compute a relevant distribution whatever the point position, resulting in robust and consistent clustering results. We show results without the local triangle filtering step (Figure 3.7, center), to underline its importance in our framework.

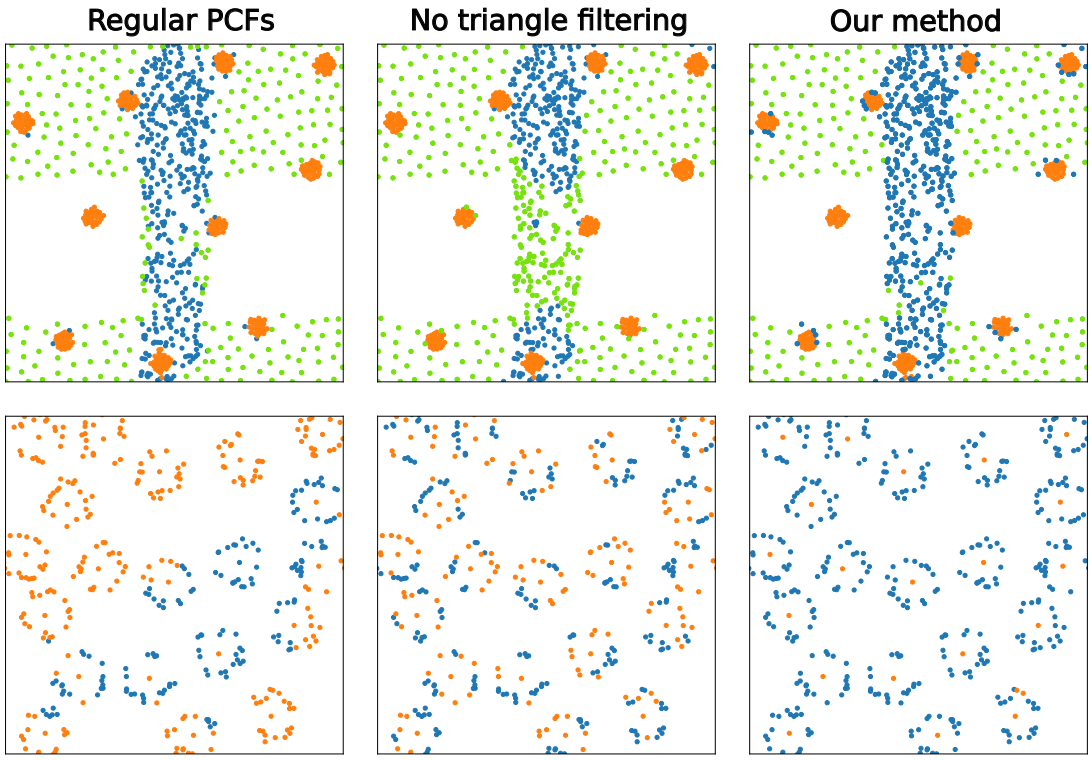


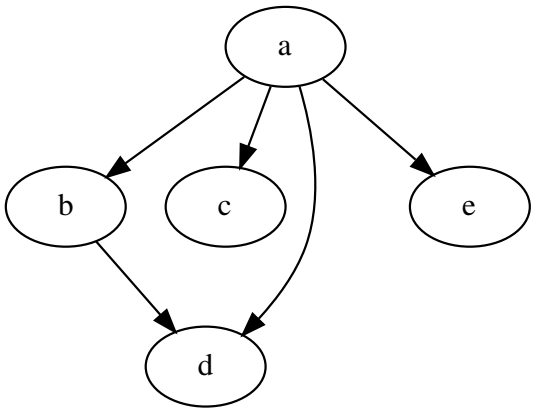
Fig. 3.7: Clustering results on 2 datasets, using regular PCFs, our method without, and with local triangle filtering. The colors represent the classes assigned to each sample.

3.3 Interactions between multiple classes

An extension of the PCF framework is required for it to cope with complex cases involving interdependent classes of objects. Although such an extension was mentioned in passing by Öztireli and Gross [ÖG12], they only add control over the overall distribution regardless of the point classes. This approach only works in simple cases and for largely uncorrelated data, since precise interactions between classes are not modeled. We demonstrate these limitations in the third example of Figure 3.17. A more complete approach was introduced in the context of Metropolis Hastings methods [EVC+15], but this only supports a strict

linear hierarchy of classes. Placement of points within a particular class is assumed to depend on all previously instantiated classes. While this approach works well for simpler cases, our early experiments showed that it struggles to produce adequate results when the number of classes increases, since later classes in the hierarchy are over-constrained and the refinement step fails to converge.

To tackle this issue, we express dependencies in multi-class data using a general *dependency graph*, stored in a Directed Acyclic Graph (DAG) data structure. Rather than insisting that a given subordinate class be fully connected to every previously instantiated class, we instead allow sparse parent-child connections. For example, in Figure 3.8 class d need not be connected to all of a, b, c, e, but can instead be more parsimoniously reliant on only a and b. In many cases capturing a few strong relationships directly and others transitively is sufficient to model a pattern. Crucially, this reduces the number of constraints, thus facilitating convergence to a significantly better solution.



	a	b	c	d	e
a	✓				
b	✓	✓			
c	✓		✓		
d	✓	✓		✓	
e	✓				✓

Fig. 3.8: An example of a dependency graph (left) and the corresponding adjacency matrix (right).

To incorporate such hierarchies, during analysis we derive a PCF for every pair of classes (C_i, C_j) connected by an edge in the dependency graph. This requires $n_i \times n_j$ additional pairs of points and thus terms in Equation (3.2), where n_i is the number of points in class C_i, in addition to the normal PCF computation within each class C_i (involving $n_i(n_i - 1)/2$ terms).

At the synthesis stage, classes are always processed in the topological sorting order of the dependency graph (e.g., a, b, c, e, then d in Figure 3.8). For each class, synthesis proceeds in two steps as before. We take care to recalculate the initialization error for each active PCF (the current class as well as its parent pairings in the dependency graph) when determining if a random point should be retained or discarded. The major change in the refinement step is that points are now moved in a direction computed using the sum of gradient vectors from all active PCFs, rather than a single PCF. Note that in our implementation, disk radii do not vary during refinement, which ensures that we retain the initial distribution of radii for each class.

3.3.1 Validation

To validate our multi-class synthesis method, we demonstrate its behavior for a case that exhibits strong interaction between classes. In Figure 3.9, red points cluster around isolated cyan points, while green points form around isolated purple points. The combined set of cyan and purple points are negatively correlated over the domain. It is impossible to reproduce such clustering behavior without addressing the inherent multi-class dependencies in evidence. Since clustering only occurs around cyan or purple centers, we construct the DAG in Figure 3.9 (left) to express this specific dependency: purple and cyan points influence each other, but clustered points only depend on the class they agglomerate around. This reduces the dependency constraints from 10 (since we have 4 classes, $\sum_{i=1}^{4} i = 10$) in the naïve approach to only 7 (10 minus 3 removed relations between the independent centers and clusters) in ours, which leads to a more efficient error minimization.

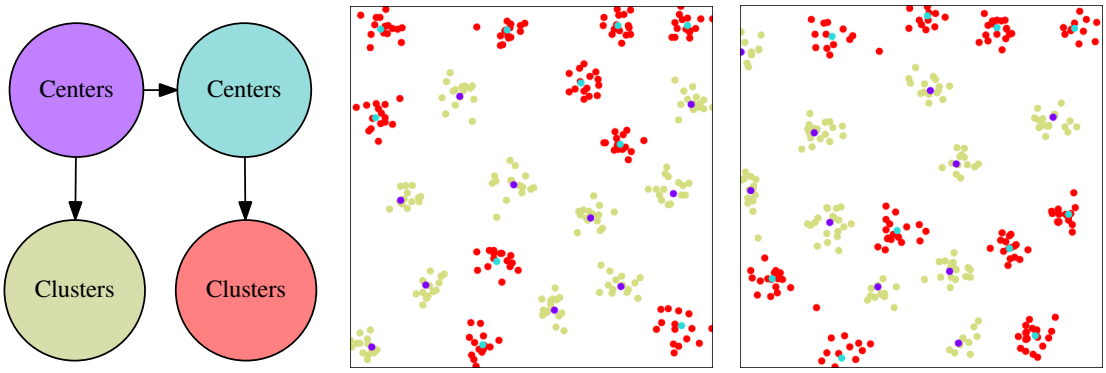
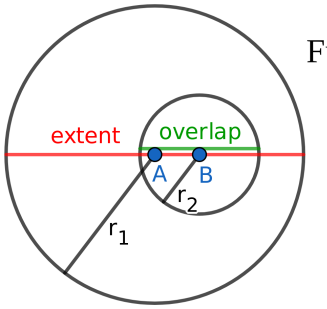


Fig. 3.9: Synthesis of a 4-class-dataset: dependency graph (left), exemplar input (middle), and synthesized output (right). Note that the chosen dependency direction between purple and cyan classes is arbitrary. Such a distribution is not reproducible without multi-class dependency.

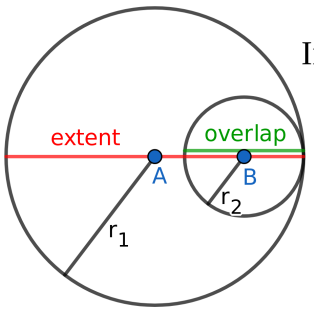
3.4 From points to disks

When placing collections of 3D objects on a 2D surface there are cases where considerations of relative size and potential intersection are vital. For instance, in ecosystem simulations (such as Figures 3.11 and 3.16) tree canopies often overlap in a botanically meaningful pattern. Such cases can be accommodated by treating them as distributions of potentially overlapping disks instead of points. In this section we consider the range of salient disk configurations, build a distance metric that distinguishes between them, and indicate how this metric can be incorporated into our framework to support disk distributions.



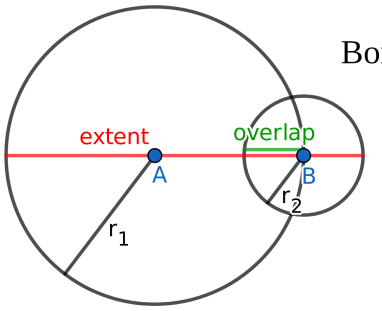
Fully Inside

$$d \leq r_1 - r_2$$
$$extent = 2r_1$$
$$overlap = 2r_2$$
$$f \leq 4r_1 - 4r_2$$
$$f_{norm} \leq 1$$



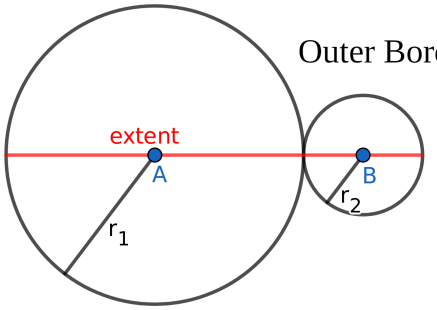
Inner Border

$$d = r_1 - r_2$$
$$extent = 2r_1$$
$$overlap = 2r_2$$
$$f = 4r_1 - 4r_2$$
$$f_{norm} = 1$$



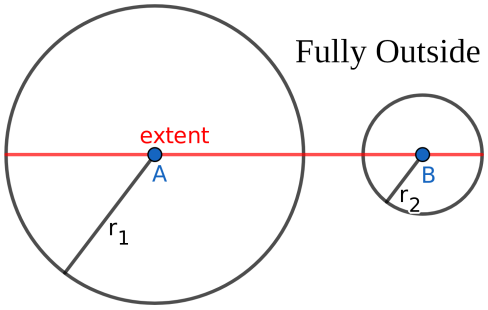
Border

$$d = r_1$$
$$extent = 2r_1 + r_2$$
$$overlap = r_2$$
$$f = 4r_1 - r_2$$
$$f_{norm} = 2$$



Outer Border

$$d = r_1 + r_2$$
$$extent = 2r_1 + 2r_2$$
$$overlap = 0$$
$$f = 4r_1 + 2r_2$$
$$f_{norm} = 3$$



Fully Outside

$$d \geq r_1 + r_2$$
$$extent = d + r_1 + r_2$$
$$overlap = 0$$
$$f \geq 4r_1 + 2r_2$$
$$f_{norm} \geq 3$$

Fig. 3.10: Principal salient configurations of disk interactions. Relevant notation is as follows: d is the distance between disk centers, *extent* and *overlap* are 1D measures linked to the union and intersection of the disks, f is the new disk distance metric that replaces r, and f_{norm} is a normalized version that distinguishes the salient cases.

3.4.1 Distinguishing important configurations

Our goal is to design a disk-aware distance metric suited to PCF processing, which distinguishes between perceptually salient disk interactions, as enumerated in Table 3.10. These configurations were derived from a study of several application scenarios, including layouts both organic (such as plant ecosystems) and structured (such as table settings and meshing cogwheels).

Unfortunately, existing disk metrics fail to disambiguate these salient configurations. The first naïve approach is to add disk radius as a third coordinate and calculate an $\mathbb{R}^3$ Euclidean distance, but this conflates position and radius so that visually distinct configurations can map to the same distance.

As a second standard measure classically used in computational geometry [Aur87], one can think of the sum of the power of each center with respect to the other circle (equal to $2d^2 - r_1^2 - r_2^2$ using the notation of Table 3.10), also known as the Laguerre distance. Although the sum of powers is strictly increasing and continuous, it becomes negative in certain overlapping cases. Moreover, since these negative values do not admit any general lower bound, a positive metric cannot be guaranteed by adding a constant term.

Lastly, as already mentioned, the EcoBrush measure for histogram binning of plant distributions [GLCC17] is discontinuous, which both prevents fine-grained analysis of overlapping cases and makes it fundamentally unsuited to the gradient-optimization that underpins the PCF framework.

Consequently, there is a need to develop a continuous metric that distinguishes between salient disks configurations, as identified in Table 3.10.

3.4.2 Saliency-based distance between disks

In addition to computational efficiency and meeting the standard requirements for metrics (namely, non-negativity, identity of indiscernibles, symmetry and triangle inequality), our main goal in designing a distance between disks is to ensure detection and distinction between perceptually-different disk configurations, such as nested, bordering or fully disconnected disks.

Let (A, r_1) and (B, r_2) be two disks, where radius $r_1 \geq r_2$, and d be the distance between their centers, A and B.

Our metric is based on what we term the *extent* of the disks, defined as the length of the intersection between the line that goes through A and B, and the convex envelope delimited by the circles (see the red lines in Table 3.10). This can easily be calculated using Equation (3.9). We also define the *overlap* (see Equation (3.10)), which is the length of the intersection between the line through A and B, and the intersection of the two circles (see the green lines in Table 3.10). Using these two measures, we construct our

new metric as the difference between the *extent* and the *overlap* (which can be regarded as a 1D approach to the union minus the intersection, or the length of the *extent* that is not common to both circles), to which we add the distance between centers d and the difference between radii, in order to differentiate all cases (see Equation (3.11)).

$$extent = \max(d + r_1 + r_2, 2r_1) \tag{3.9}$$
$$overlap = \text{clip}(r_1 + r_2 - d, 0, 2r_2) \tag{3.10}$$
$$f = extent - overlap + d + r_1 - r_2 \tag{3.11}$$

As it stands, our distance is a symmetric, continuous function that differentiates successfully between the three main cases of disk interaction (fully inside but abutting the border, centered on the border, and fully outside but against the border) for fixed r_1 and r_2. Unfortunately, it provides different values for the salient configurations when the radii change, as indicated by the radii-dependent values of f listed in Table 3.10. This makes it unsuitable for PCF computations, since merging values that are similar but carry a different meaning will destroy that meaning, thus making it impossible to synthesize a perceptually-correct disk distribution.

To circumvent this, we normalize our distance function based on these three special cases, transforming them into three fixed values. More precisely, we choose 1, 2 and 3 as the three normalized values for the specific cases "inner border", "border" and "outer border" (see Table 3.10), and define the normalized distance f_{norm} as a continuous piecewise linear function of f that maps the specific cases to these values:

$$f_{norm} = \begin{cases} f/(4r_1 - 4r_2), & \text{if } d \leq r_1 - r_2 \\ (f - 4r_1 + 7r_2)/(3r_2), & \text{if } r_1 - r_2 < d \leq r_1 + r_2 \\ f - 4r_1 - 2r_2 + 3, & \text{otherwise.} \end{cases} \tag{3.12}$$

3.4.3 Processing disk distributions

Analysis: Replacing the standard Euclidean distance with our new distance function for analyzing disk distributions is straightforward. The new distance f_{norm} is used as the horizontal axis of the PCF, and therefore both r and d in Equation (3.2) relate to this metric. However, the Euclidean distance between disk centers is retained for computations not directly related to disk interactions, such as the evaluation of r_{max} (Equation(3.1)) and compensation for border effects during analysis.

Synthesis: More care is required during synthesis, since both disk positions and consistent radii need to be generated.

During initialization, we match the original distribution of disk radii by sorting the radius values of the input exemplar within a class in descending order and simply picking

them sequentially during dart throwing. When the number of required output disks exceeds the input, we repeat the entire list of input radii as required and use a random non-duplicated sampling to make up any shortfall before sorting as before. This simple scheme has the virtue of being computationally efficient and allowing greater freedom of placement. Smaller disks are easier to fit as they generally have more valid locations, so we prefer to save them for later and focus on the largest and most difficult cases first.

At the refinement stage (Equations (3.5) – (3.7)), we replace point-wise distance (d_{ij}) with our new disk-wise distance (f_{norm}) and apply gradient descent to optimize for center positions. The radius distribution is handled in parallel where we take values from the input radius distribution in decreasing order. This way our final radius distribution matches exactly with the given distribution. Indeed, this strategy makes the best use of the given radius distribution, and seems practically more efficient than a gradient descent method optimizing on both position and radius simultaneously.

Figure 3.11, illustrates the use of this method for a distribution with three disk classes, including some overlapping and inner border cases. For comparison, the PCFs of our synthesized example are shown at the bottom right.

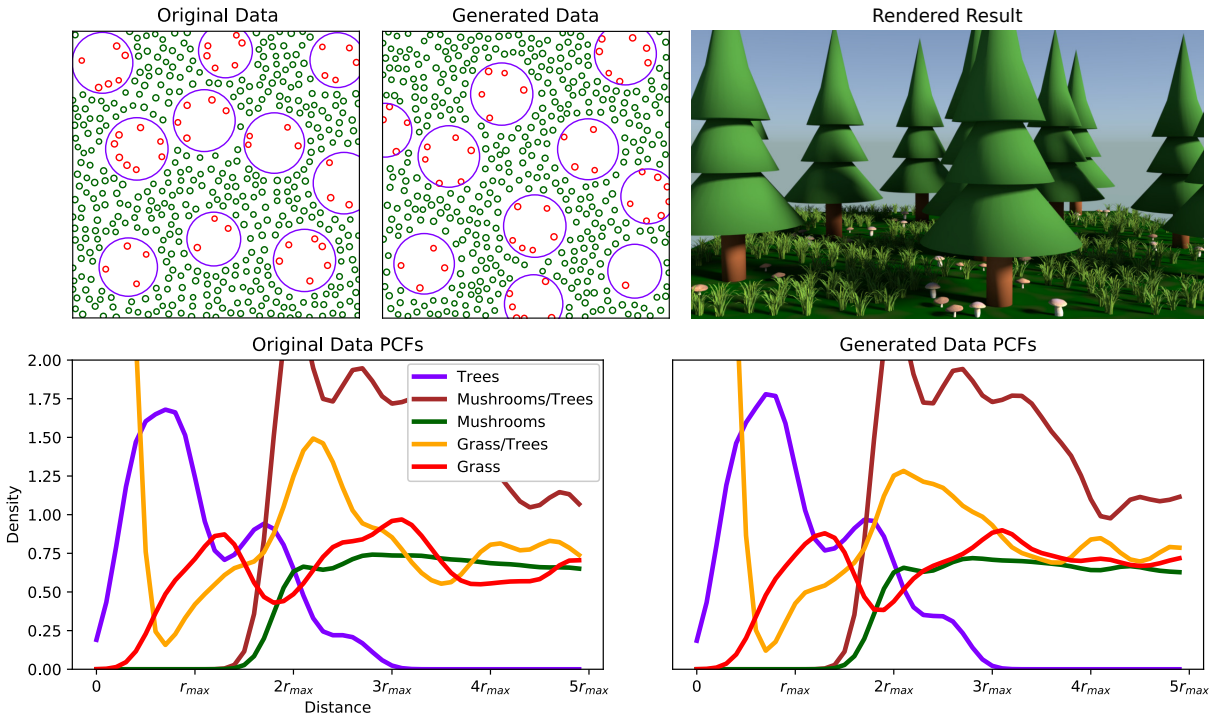


Fig. 3.11: Disk-based multi-class forest synthesis: mushrooms (in red) grow in the shade of trees (in purple), but grass (in green) is unshaded. Images include: the input exemplar (top left), output synthesis (top center), and corresponding preliminary rendering (top right), with analyzed input (bottom left) and synthesized Pair Correlation Functions (PCFs) (bottom right).

3.5 Improving convergence

Although putatively accurate in their convergence to the mean statistics for each class, PCFs methods can give rise to a clear mismatch between input and output. Furthermore, a few scattered disks may be introduced during initialization to compensate for inaccuracies in the positioning of other disks, thereby decreasing overall error in the moment. Once present, these outliers cannot be removed, since refinement only locally improves the radius and position of a disk, giving rise to objectionable artifacts.

In this section, we introduce a new strategy for validating inserted disks during synthesis. While retaining global convergence of the mean PCF, these additional constraints, based on the variance of input PCFs, significantly improve the visual quality of the final outcome.

3.5.1 Variance-aware PCFs

The method presented thus far creates a combined average over individual per-point PCFs. Predictably, this summarization leads to a loss of information compared to the original, individual PCFs. As Figure 3.12 shows, such a data reduction can cause visually inaccurate replication of the initial distribution in over-constrained cases, even when convergence to the mean PCFs (one for each class, and one per edge in the dependency graph) is achieved.

To tackle such challenging cases, we introduce a variance-aware extension of the PCF method (for typical outcomes see Figure 3.13). Our idea is to retain more data from the individual exemplar PCFs. This enables us to ensure during synthesis that the PCF of a candidate disk is never too divergent from individual PCFs in the input distribution. This is done without sacrificing convergence to the mean PCF.

In practice, rather than retaining individual PCFs, which would be both costly in memory and difficult to manipulate, during analysis we extract the lower and upper bound of the set of all the individual PCF curves and employ these bounds as constraints.

During synthesis, a new disk is accepted only if its PCF lies in the so-called *validity region* bounded below and above by the lower and upper envelopes of the union of the original PCFs (as illustrated in Figure 3.14, where gray curves represent the original PCFs from reference disks in the exemplar).

These bounds, and the difference between them, allow us to differentiate parts of the PCF that must be strictly enforced (an error is likely to be perceptually salient for human observers in regions of low variance) from those with greater leeway. An alternative would be to compute the standard deviation at points along the mean PCF curve, but this is not necessarily representative since the distribution of curves is often asymmetrically skewed about the mean. Moreover, bounds are a harder limit than the standard deviation,

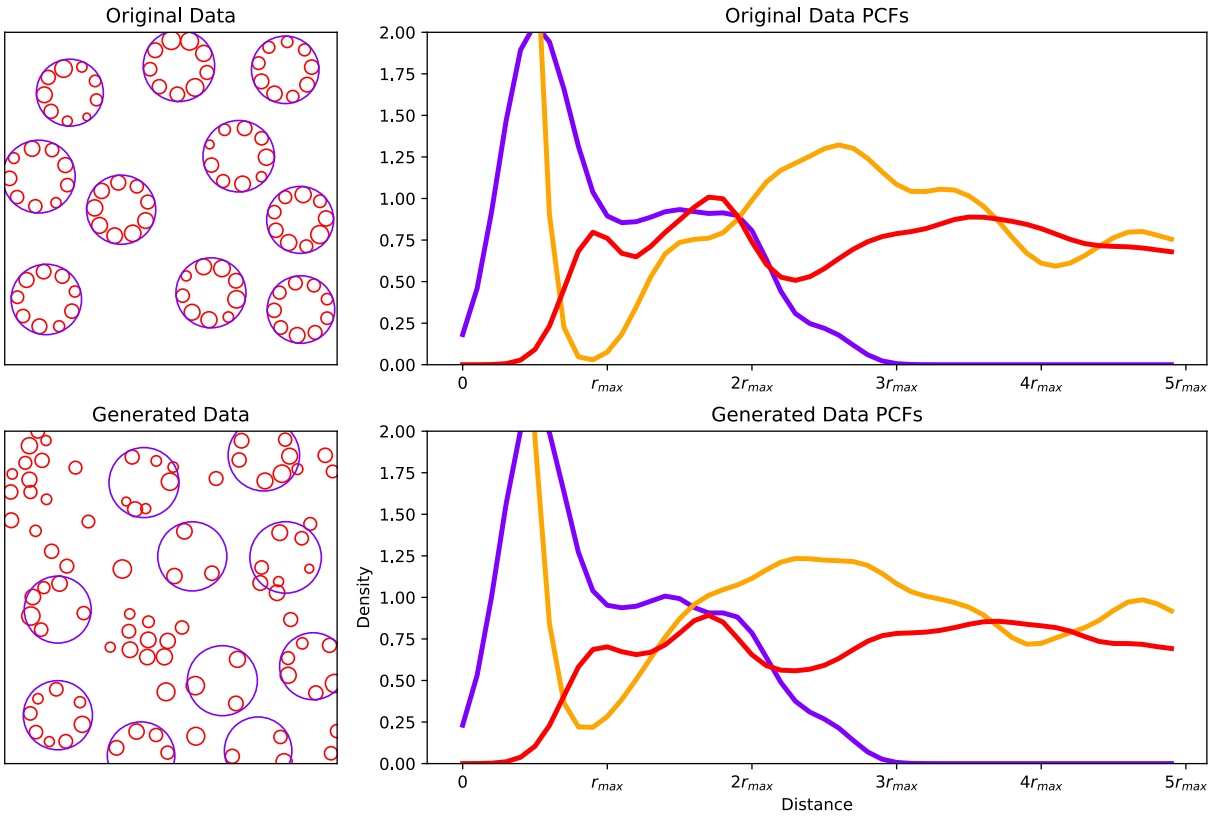


Fig. 3.12: Correctly matching the average PCF alone is not sufficient for perceptually convincing results in highly constrained cases. (Top) Input disk distribution and the corresponding mean PCFs, where the purple (respectively red) curves represent inter-relationships within the purple (respectively red) class of disks, and the yellow curve represents the dependency of red on purple. (Bottom) A perceptually incorrect synthesized distribution, despite convergence of all mean PCFs.

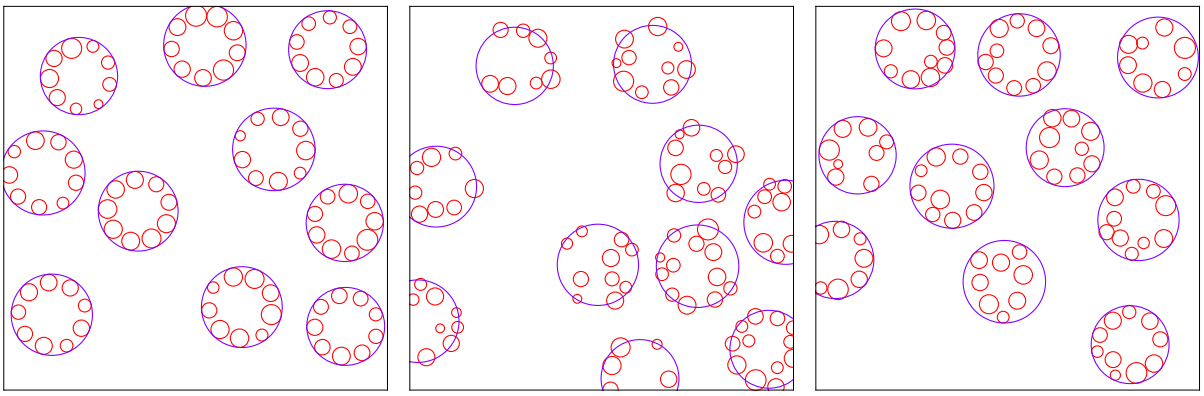


Fig. 3.13: A challenging over-constrained nested case (left). Synthesis result using Ecobrush [GLCC17] (center). Our variance-aware solution (right).

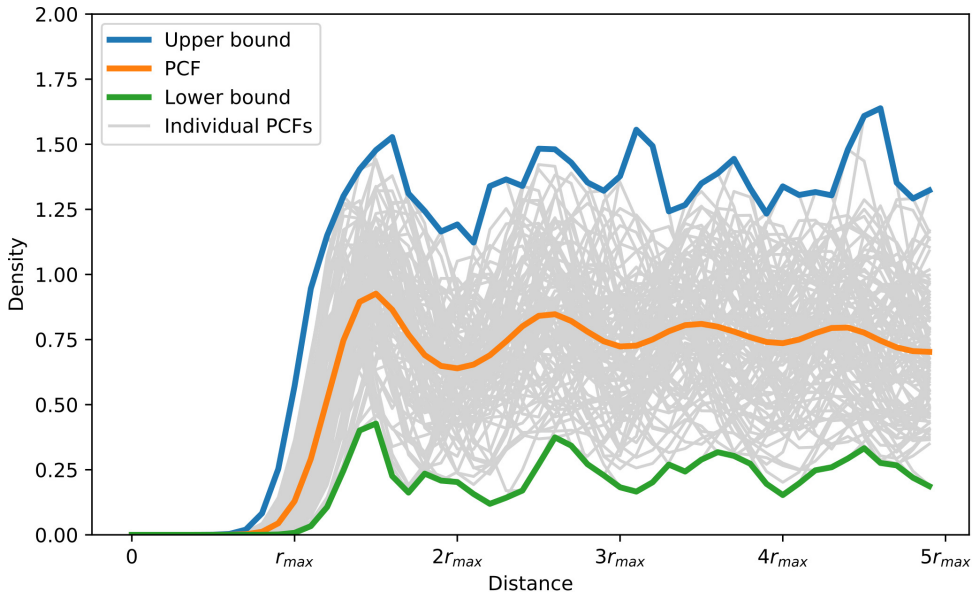


Fig. 3.14: Validity region: during synthesis the insertion of a new disk is only accepted if its PCF lies in the envelope formed by the union of all original PCFs.

allowing us a better fit for the more extreme cases that may have occurred in the source exemplar.

To make effective use of the information provided by our variance-aware PCFs, we add a new term to the error formula used during initialization. This term represents the largest distance between the new curve and the validity region. We also check that each individual error term remains positive, and divide the differences by the target value. This enables us to associate a higher error with variations where the target PCF value is close to 0. These represent visually salient errors, since they are equivalent to adding an outlying disk where none exists in the input data. Incorporating these changes, we end up with the following error function for synthesis initialization:

$$E = \left(\max_r \frac{new_{mean} - PCF_{mean}}{PCF_{mean}} \right)^+ + \max \left(\max_r \frac{new - PCF_{upper}}{PCF_{upper}}, \max_r \frac{PCF_{lower} - new}{PCF_{lower}} \right)^+ \tag{3.13}$$

where x^+ indicates that negative values of x are truncated to 0.

This change in the initialization proves effective in practice. In particular, we achieve the result in Figure 3.13 for the case with over-constrained nested disks.

3.5.2 Control of convergence

Since PCFs are by nature injective and not surjective (any layout of disks can be mapped to a PCF, but some PCFs cannot be realized as a corresponding disk layout), achieving initialization in reasonable time can be challenging.

Indeed, even with incremental relaxation of the error threshold ϵ, for complex distributions the error can sometimes become so large that it prevents ongoing point placement.

Further relaxing the error threshold could prevent the initialization from seizing up, but at the cost of lower quality in simpler cases. One solution is to adapt the relaxation factor according to the input configuration, but such automatic tuning in complex multi-class cases seems highly non-trivial.

Instead, we flag a lack of progression when the number of consecutive point rejections exceeds a certain threshold (we generally used a threshold of around 1000 rejections), and switch to a simple grid search. We divide the domain into a regular grid, and compute the expected error for each grid cell. We then sample and accept a random point in the cell with lowest error (or in a random cell among those with minimal error). While this algorithm is slower than dart throwing for easy cases, it runs in constant time regardless of the complexity of the distribution and always returns a solution close to optimal. It is also trivially parallelizable thanks to the independent error computation in each cell. Algorithm 1 sums up the initialization step of our pipeline with this grid search enhancement.

3.6 Results and applications

3.6.1 Parameters

In most of the examples in this paper, we used $\sigma = 0.25$ (smoothness of the Gaussian kernel), $\delta = 0.1$ (discretization step for radii analysis), and a neighborhood size around each point set to $5r_{max}$ for PCF computations. If the neighborhood radius is too small, the PCFs will not incorporate sufficient context area, reducing reproduction accuracy. Conversely, if the radius is too large, the method will overfit the data, damaging its ability to generalize. A similar trade-off exists for the smoothness parameter σ: reducing it improves the replication of detail and particular configurations, while increasing it speeds up convergence, with the drawback that details may be approximated. The termination test for refinement is when the accumulated adjustments to all disks during the last iteration drops below a distance threshold. Alternatively, since all PCFs are incrementally updated after each iteration, we could check the difference between the target and current PCF curves. We also set a maximal number of refinement iterations to obtain an approximate fallback solution when refinement fails to converge.

Input: Hierarchy of PCFs, number of elements n
Output: Initialized elements

foreach *class* **do**
  $fails \leftarrow 0$;
  repeat
    $\epsilon \leftarrow \epsilon_0 + \epsilon_\Delta fails$;
    Sample a random element from original distribution;
    Compute error E;
    if $E < \epsilon$ **then** Accept this element;
    else $fails \leftarrow fails + 1$;
    if *more than max_fails successive fails* **then**
      while *less than n elements accepted* **do**
        Grid-search the domain for the lowest error;
        Sample a random element in the best cell;
        Accept this element;
      end
    end
  until *n elements are accepted*;
end

Algorithm 1: Initialization algorithm incorporating grid search.

3.6.2 Comparison with previous methods

In WorldBrush [EVC+15], trees, grass and other scene elements are modeled as points for the purpose of placement. Since the varying spatial extents of objects are ignored, fine control over complex interactions, such as collisions, is unattainable. Figure 3.15 shows a typical failure case in close-up from their video results.

The example in Figure 3.11 provides a similar context for comparison. In this toy example of a fairytale forest, mushrooms are sun intolerant and must be slightly shaded by trees, while grass favors sunlight and cannot be shaded. In addition to avoiding collisions thanks to the use of disks, our method models such inter-class relationships effectively. We provided the exemplar from Figure 3.11 as input to an Ecobrush-style synthesis [GLCC17], to evaluate the benefits of our continuous distance function over a binned solution (see Figure 3.15 (right)). While their method respects coarse constraints (mushrooms in the shade, grass in the sun) the discrete distance function fails to reproduce the blue noise distribution of grass and the positioning of mushrooms near the border of tree shade.

Lastly, while the EcoBrush algorithm cannot handle over-constrained cases (as shown in Figure 3.13), our method proves to be robust to the types of dense ecosystems that is the focus of their work, even with the significant dependencies resulting from a full hierarchy of classes. Figure 3.16 demonstrates our results on a dense ecosystem with

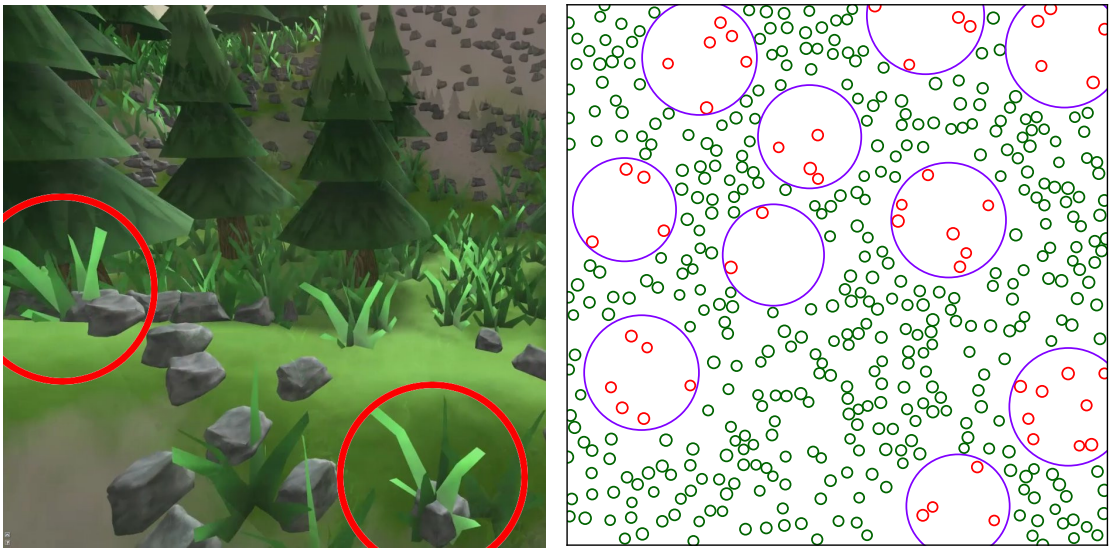


Fig. 3.15: Comparison with prior work: (Left) Modelling objects as points in Worldbrush makes collision handling impossible (e.g., grass growing through rocks). (Right) For the exemplar from Figure 3.11 Ecobrush instantiates some mushrooms (in red) too near tree centers (in purple) and fails to achieve an even distribution of grass (in green).

9 species and more than 2,300 individual plants (taken from Ecobrush, courtesy of the authors).

Figure 3.17 demonstrates that our method is able to reproduce point distributions found in previous work. We provide results that are perceptually similar to two examples from Ma *et al.* [MWT11], with the difference that we cannot capture the orientation of dominant lines, since our method is rotation invariant. We also include a comparison with a multi-class example from Öztireli and Gross [ÖG12]. Here, our results match the input far more closely, as indicated by the analyzed curves. For instance, the purple points are never close together in the input nor in our output. More generally, their approach is effective for relatively simple scenarios, but fails in more complex cases, since they only consider interactions within classes and for the dataset as a whole, but not between classes.

Unfortunately, as evident from Figure 3.19, our method is unable to reproduce Roveri *et al.*'s [RÖM+15] highly structured patterns, since one of our key goals is the ability to generalize an input pattern rather than perfectly replicate it. Similarly, non-stationary distributions (such as in the work of Roveri *et al.* [RÖG17], are beyond the scope of this paper. Importantly, however, our method is able to process arrangements of overlapping shapes, which is not the case for any of the more structured methods.

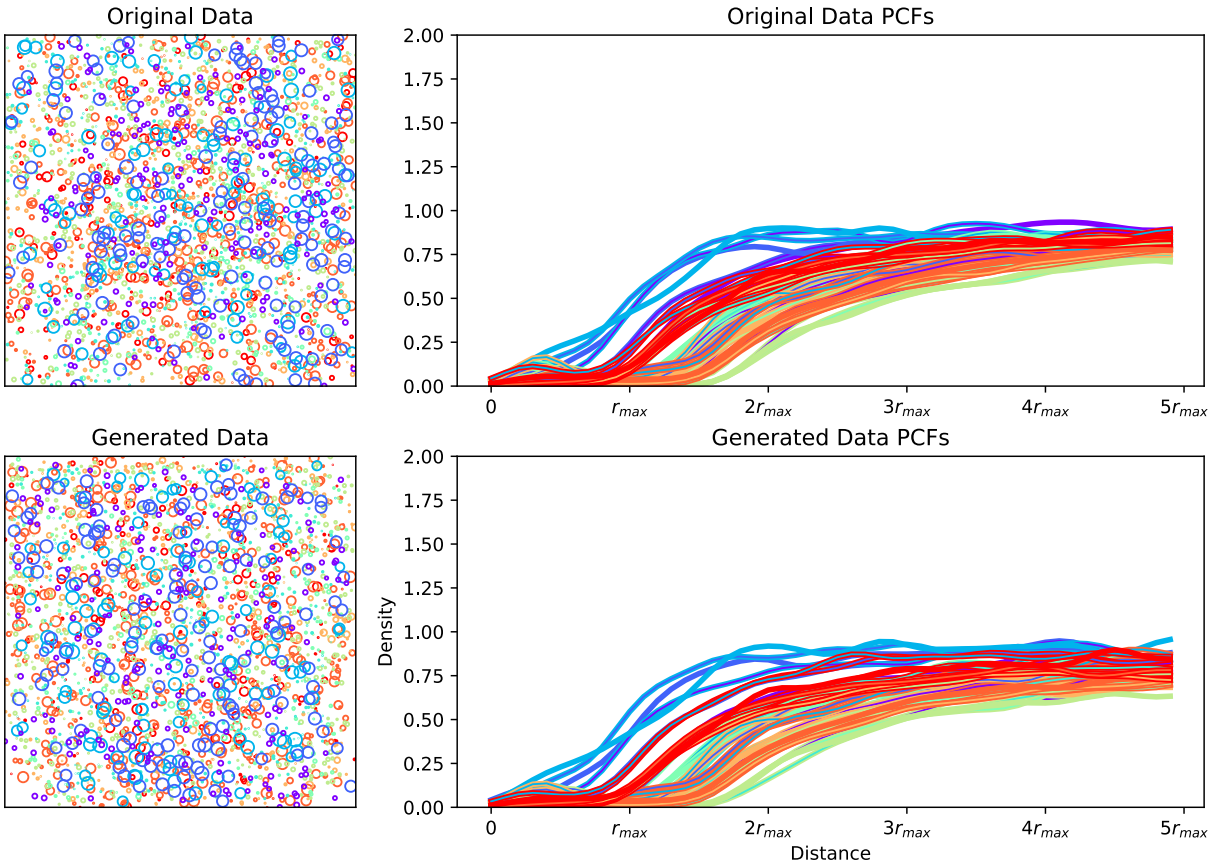


Fig. 3.16: Results for a dense ecosystem with 9 interdependent classes and more than 2,300 plants. (Top) input exemplar and analyzed PCFs. (Bottom) synthesized distribution and the corresponding PCFs.

3.6.3 Results

Figure 3.18 illustrates the use of our disk distribution synthesis in different application scenarios, that can be used in the context of virtual worlds.

The first scenario is derived from the behavior of scattered water droplets on a smooth surface, with applications to texture synthesis. Although input and output are rendered in the same style for ease of comparison, the input distribution has a physical analog and was extracted by hand from a photograph. To account for the interaction of size and placement, droplets are divided into 4 classes, with a sequential chain of dependency from largest to smallest.

The second example showcases general object placement, with a more complex distribution of plates, glasses, bowls and food items in class-specific configurations. The specific interactions that are encoded here include classes that may touch but should not overlap (glasses, bowls and plates), should overlap (food on plates), or may overlap (apples and candies).

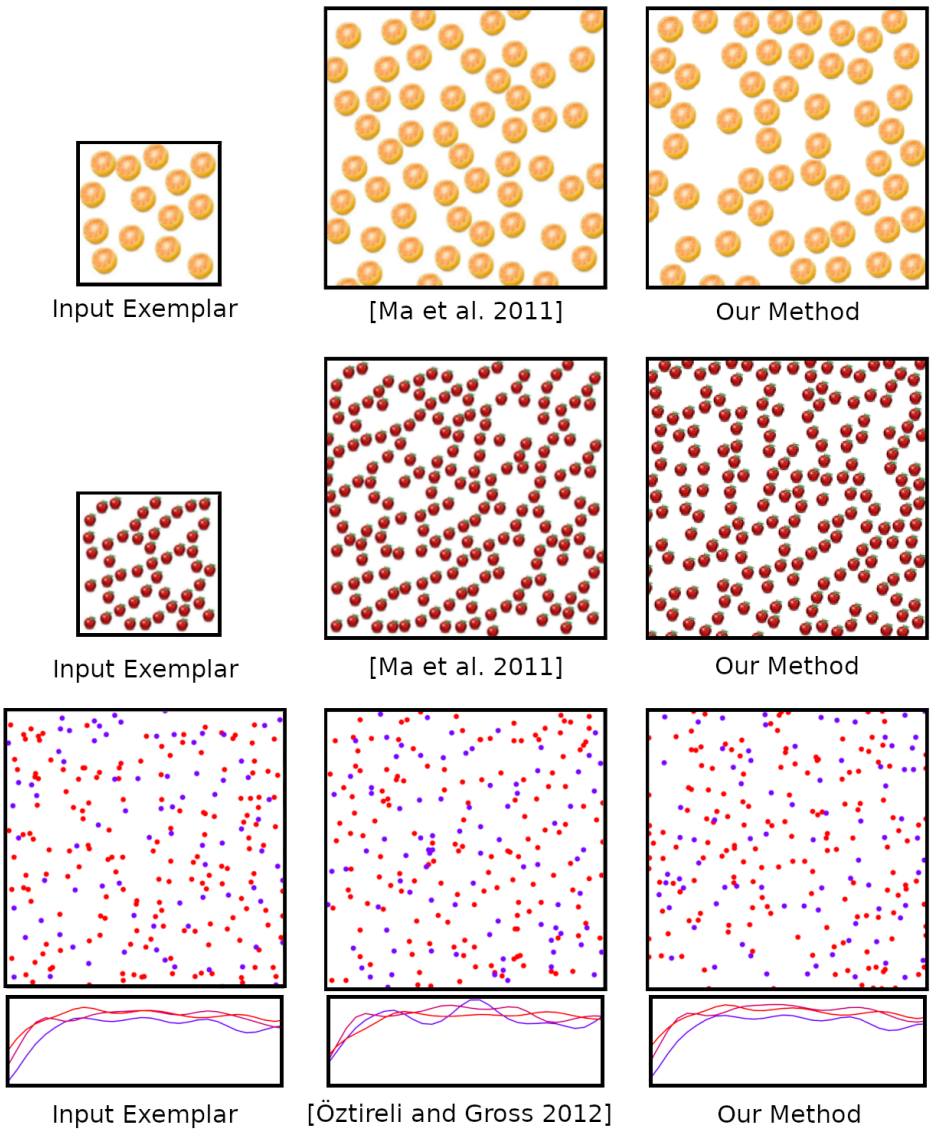


Fig. 3.17: Comparison with Ma *et al.* [MWT11] indicates that our method is capable of reproducing previous results, except for the specific orientation of the dominant lines, while a comparison with Öztireli and Gross [ÖG12] shows that we more accurately respect features in multi-class data (refer to the PCFs in the bottom row).

Lastly, we use our method to populate a Mediterranean landscape with plants, based on a distribution of 12 plant species with 78 dependencies, computed by an ecosystem simulator [GLCC17]. The output distribution is synthesized over a larger region, and used to instantiate 3D tree models (Figure 3.18 (bottom)). As demonstrated by this example and the corresponding rendered landscape in the teaser at the beginning of the chapter, our method is not limited to placing shapes with a strictly circular projection. Here, the disks used at the analysis and synthesis stages are only coarse bounds for the footprint of plant canopies.

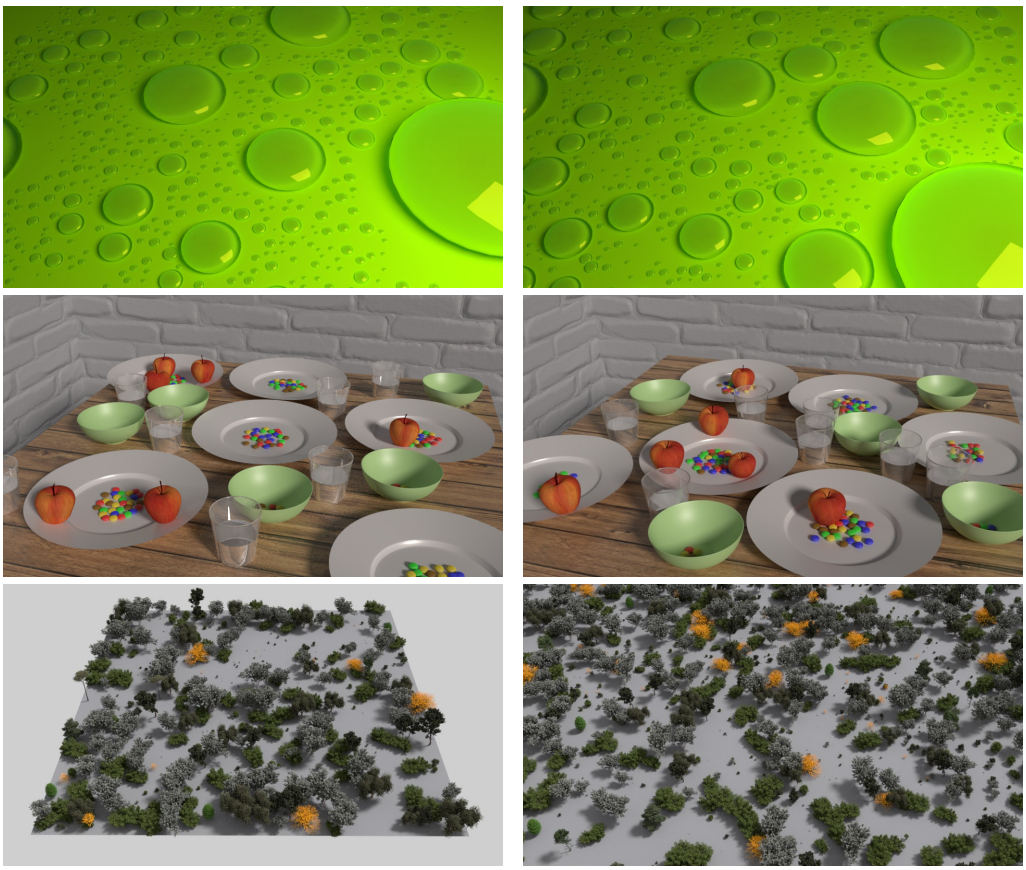


Fig. 3.18: Application of multi-class disk synthesis to the placement of 3D objects (input exemplars on the left, synthesized results on the right). From top to bottom: Water droplets (4 classes, no overlaps), table set with plates, bowls, glasses and food (5 classes, 6 interdependencies, nested distributions), a Mediterranean biome (12 classes, full dependencies, overlapping distributions). The domain size is increased during synthesis for the last example, to simulate the generation of vegetation from a sample.

3.6.4 Computation times

Table 3.1 provides the computation times and specifications for our synthesis examples. Timings were taken on an Intel® Core I5 with 4 cores, clocked at 3.3 GHz with 8GB of RAM, and an NVidia GeForce GTX 960 graphics card. It should be noted that the method was implemented in Python on the CPU without GPU acceleration and its runtime could be improved with further optimization.

3.6.5 Limitations and discussion

In this section, we discuss the limitations of our method. First, we have prioritized accuracy over speed. As it stands, the method is not suited to interactive design, where

Example	Disks	Classes	PCFs	Time
Toy forest (Fig. 3.11)	460	3	5	2min
Droplets (Fig. 3.18)	800+	4	10	7min
Food (Fig. 3.18)	224	5	11	1min
Mediterranean (Fig. 3.18)	1500+	12	78	12min
Dense ecosystem (Fig. 3.16)	2300+	9	45	30min

Table 3.1: Runtime and specifications for example scenes.

users create and edit distributions with cycles of rapid feedback. However, even in such situations there are options available. For instance, a placeholder distribution, derived from a partial initialization or by directly cutting and pasting the exemplar, could be displayed while refinement takes place in a background thread.

Another limitation, shared by previous point and disk synthesis methods, is an inability to reconstruct regular or otherwise highly constrained distributions, such as the case in Figure 3.19 (top). One issue is that the pressure of constraints tends to push disks to the same position during refinement. To address this we reuse our PCF validity regions to re-check the individual PCFs of generated points after refinement, and re-initialize outliers as required. This strategy, illustrated by Figure 3.19 (bottom), improves results locally – see for instance the leftmost bumps in Figure 3.19 (middle) that are consequently suppressed – but does not solve the issue more generally. An efficient handling of such regular configurations would likely require PCFs to be enriched with additional structural information. As a first pass, separating PCFs into directional arcs or quadrants, as in WorldBrush [EVC+15], would improve global alignment. Although the synthesis of structured data sets is quite far from the context we were considering in this thesis, it is worth noting that recent work such as [TLH19] have tried to tackle this particular problem with alternative approaches.

Since our method is specifically designed for disk distributions, a main limitation is that it does not cater for other shapes. Of course, any 2D shape could be approximated by a union of disks and therefore analyzed, but there would be no guarantees as to the quality of the synthesized result. Moreover, in such cases it is necessary to account for relative orientation as well as radial distance, and this is beyond our scope. While this is usually not an issue for trees and mostly round shapes, other objects in virtual worlds would benefit from a formulation that considers elongated shapes with a precise orientation. For example, particular rocks, animals, and man-made structures such as roads exhibit strong anisotropic features that cannot be replicated with this approach.

More generally, our method considers the input exemplar as a whole under the assumption that the distribution is homogeneous and isotropic. Although our analysis could be used to test homogeneity using cropped sub-regions we have not investigated spatially-varying distributions further. One possible avenue for catering to non-homogeneous distributions would be to extend Roveri *et al.*'s [RÖG17] method for handling two different distributions and densities.

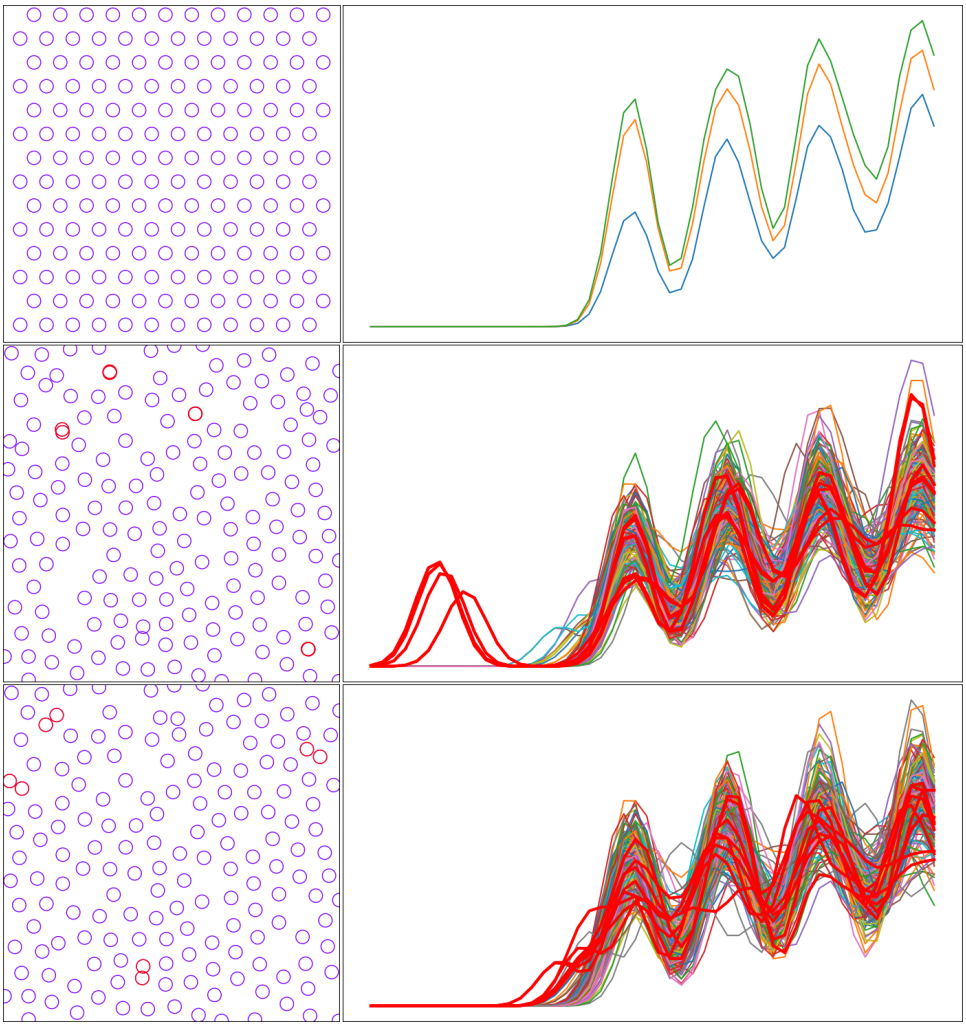


Fig. 3.19: Failure case on an hexagonal grid: (top) exemplar input, (center) standard disk synthesis, and (bottom) synthesis with outlier removal. The pattern and distance between points are preserved locally but the global layout does not converge to a visually acceptable solution. Disks with the most outlying PCFs are colored in red at each stage. In the center left pattern, these correspond to several superimposed red disks, inducing the initial PCF bumps (center right).

3.7 Conclusion

In this chapter, we presented a boundary handling technique for PCF computations which allows the analysis of 2D distributions in arbitrary domains. We also introduced for the first time an efficient PCF-based method capable of analyzing and synthesizing general two-dimensional distributions of disks, including those with nested and partially overlapping configurations. We tackled this from a statistical perspective, by defining a new distance metric adapted to disks and specifically designed to distinguish perceptually salient key configurations, as illustrated by a variety of applications to virtual worlds

throughout the chapter. Our approach handles multi-class distributions efficiently, and can be further optimized if provided with a user-specified inter-class dependency graph. Moreover, special attention was devoted to the issue of convergence, enabling completion of synthesis even in highly-constrained situations.

There are a number of viable avenues for future work. First, although our method has proven to be robust in converging to a visually convincing result for all our test cases, providing a general proof of convergence for every realizable PCF (i.e., those for which a disk distribution exists) remains as future work. Secondly, being able to distribute shapes in 3D as well as in 2D would open up many other applications, ranging from the placement of leaves, flowers and fruit in plants and trees to the instantiation of granular materials. From a theoretical point of view, our metric and the attendant framework are well suited to such a generalization.

Finally, one of the main limitations of the method is its restriction to isotropic distributions of disks. New metrics could be designed to handle more complex objects, but additional work outside of the metric will probably be necessary to handle arbitrary shapes. An alternative solution would be to represent arbitrarily complex objects as sets of points (similar to [RÖM+15]) and adapt our method to this new structure.

Focusing on lifting the constraint of isotropy, we propose in the next chapter an alternate method designed to handle anisotropic distributions, while still considering their spatial extent. In order to allow the concept of direction in the distribution, we replace disks with their oriented equivalent, namely ellipses. This allows us to properly model the distribution of animals in a herd, and in turn to propose an intuitive tool for the placement and animation of herds in virtual worlds.

Chapter 4

Towards animated worlds

Contents

While methods for easing the generation of complex virtual worlds widely spread in the last decade - leading to impressive results with detailed terrains as well as plausible distributions of rocks and vegetation - much less attention was paid so far to animal life. Yet, populating virtual worlds not only with vegetation blowing in the wind and a few birds, but also with moving groups of ground animals, is mandatory to get a lively result.

This chapter tackles the authoring of herd animations. The key idea is to enable intuitive authoring and enhanced realism through visual analogies with real photos. More precisely, we allow the user to key-frame the animation of a herd by copy-pasting a series of photos over the terrain using a pipette tool. In addition to the rough trajectories they define, each photo is used as a model for the local shape, distribution and density of the animals within the herd.

Fig. 4.1: A herd of cattle traveling on a road. We can see the emergence of lines, small groups, and a strong anisotropy.

The input photos may indeed show quite different numbers of animals. Therefore, our solution relies on a new statistical method to analyze the photos independently from animals count and then synthesize a visually similar herd while accounting for the target number of animals, their size and local obstacles avoidance. In addition to the statistical distributions governing the relative distance between animals (modeled as oriented ellipses), our model extracts and reproduces two higher level features from each photo, namely the herd's density map (also defining its global shape) and the local orientation map for animals within the herd. If desired, these features can be edited at high level by the user using brush strokes.

Lastly, we propose a simple method to generate the individual trajectories of the animals from a herd key-frame to the next. It includes an automatic best-pairing mechanism between animals in successive key-herds and a microscopic simulation enabling animals

to avoid collision while following the trajectory and having the desired relative motion in herd frame.

In summary, our three main contributions are:

- An intuitive authoring tool for key-framing herds, using the concept of pipette tool for copy-pasting the herd visual aspect from a photo and providing additional editing brushes;
- A new, two-levels method enabling to analyze both global herd features and local distributions of animals on images, and to synthesize visually similar ones with a given, target number of animals;
- A layered model using both the herd global trajectory and the desired relative motion within the herd to generate the animation between successive key-frames.

4.1 Herd animation from photos: overview

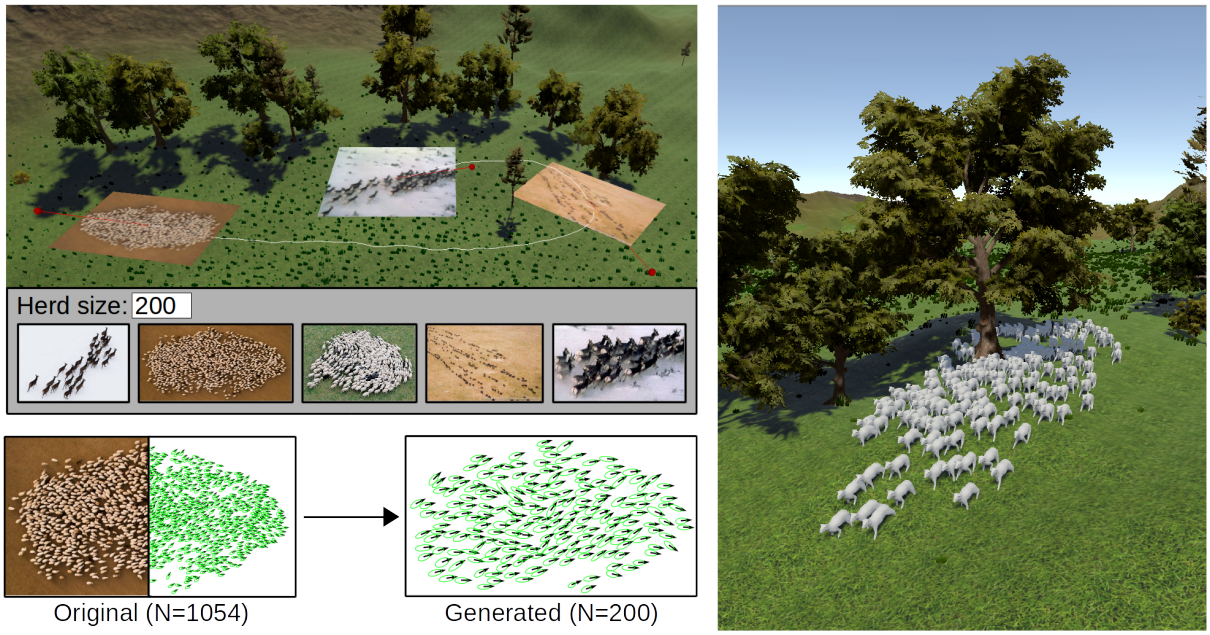


Fig. 4.2: To create a herd animation, the user selects the desired visual aspects over time on photos, by placing them as key-frames on the terrain (top left). The analysis-synthesis method generates, for each photo, a visually similar key-herd with the target number N of animals (bottom left). The animation is then generated (right), thanks to an automatic labeling process to pair the animals of successive key-herds.

We present a method for authoring herd animations based on data extracted from still photos.

4.1.1 Authoring interface

Our interactive tool enables the user to upload a virtual world which may contain a non-flat terrain with obstacles such as trees and rocks (top left of Figure 4.2). The user then choose a type of animal in a library, sets their maximal maximal speed V_{max} (which can be extracted if desired from zoological information) and their number N in the herd to be animated.

Our framework is inspired by traditional animation principles, namely the key-framing and interpolation workflow. In the reminder of this section, the key-frames correspond to important locations where the user requests a specific aspect for the herd, extracted from real world's pictures. In practice, the user can also add position-only key-frames to edit the trajectory without imposing a specific aspect of the herd.

Using an analogy with drawing software, we allow the user to upload reference photos of real herds in a palette window and to extract the herd aspect from them using a pipette tool to define key-frames. To provide an intuitive visual feedback, the selected photo is copy-pasted on the terrain at the position where the user would like to impose this specific aspect of the herd.

As in usual pipelines, the animation process is fully automatic once key-frames have been positioned on a timeline. Yet, the user can not only edit the timing and choice of reference images at key-frames, but also some high level parameters, namely the density and orientation maps at each key-frame, as detailed below.

4.1.2 Method and challenges

Key-herds from photos: From each key-frame position with an associated photo, the first step is to generate a key-position for a herd with the target number N of animals, which we call a *key-herd* (see bottom left of Figure 4.2). The latter should reproduce the visual aspect in terms of global shape, distribution and orientation of animals of the herd on the photo, while avoiding obstacles. The main challenges are to define and learn the right set of global and local descriptors from the photo, and then define an algorithm enabling to use them at the synthesis stage. Our solutions are detailed in the *Analysis and synthesis* section.

Animation: Once key-herds have been generated and projected onto the terrain, a global trajectory is computed between their centroids. Individual trajectories are then automatically generated for each animal. Achieving a fluid motion is a challenge, since it requires consistently labeling the animals within the successive key-herd. Moreover, the individual trajectories need to insure collision avoidance between animals and with obstacles on the terrain, while following the herd global path and matching the required change of position in herd frame. Our solutions to these problems are detailed in the *Herd animation* section.

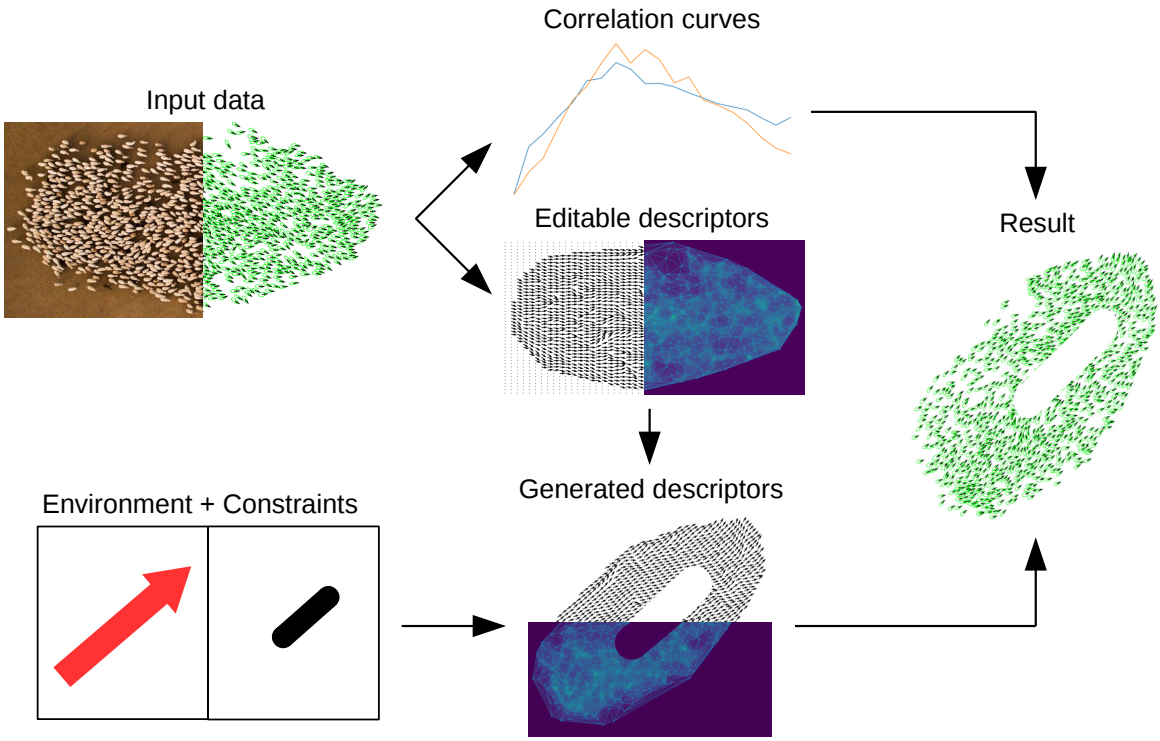
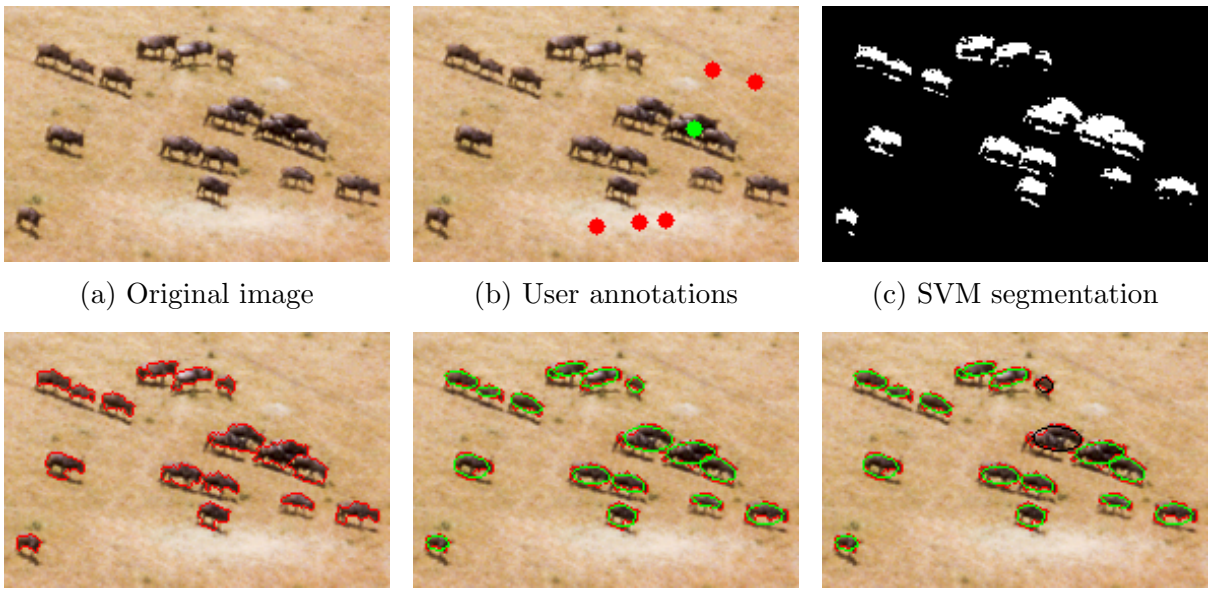


Fig. 4.3: Pipeline for computing a key-herd: From oriented ellipses extracted from the photo (top left), correlation curves and editable descriptors are computed. The descriptors are adapted to the trajectory orientation and obstacles (bottom), and then used in conjunction with the correlation curves to produce a key-herd with N animals (right).

4.2 Analysis and synthesis of static herds

While the key-herd synthesis process has a similar goal to the one presented in Chapter 3, the two use cases reveal in practice many differences that prevent a direct use of our previous method. Indeed, most animals have an inherent direction that has to be encoded, which is often not the case for static objects. This anisotropy constraint has consequences on the shape used to represent elements, as well as their computed distance. The fact that we focus on herds also has major implications: since the density and orientation of animals is not guaranteed to be uniform at different locations in the herd, a particular attention is required to faithfully reproduce such variations. The overall shape of the herd is also an important attribute that was not considered in the case of static objects, but is crucial to recreate the behavior of different species.

With these considerations in mind, our processing pipeline for generating a key-herd from a photo is detailed in Figure 4.3. The photo is first pre-processed to extract oriented ellipses giving the position and orientation of the animals. We analyze this resulting distribution to compute correlation curves modeling the local interactions between animals, as well as editable features in the form of density and orientation maps. We take the



(a) Original image (b) User annotations (c) SVM segmentation

(d) Extracted regions (e) Fitted ellipses (f) Outlier detection

Fig. 4.4: Semi-automatic data extraction process from user annotations. The method automatically extracts ellipses based on user input and detects wrongly assigned clusters.

global orientation of the herd trajectory, the possible obstacles on the terrains and the size of the target herd to modify these maps, which may also be interactively edited. Finally, we use both curves and features to synthesize a perceptually similar herd of the right size. We details these steps below.

4.2.1 Data extraction from a single image

We use a semi-automatic method to extract data from an image. It only requires manual tuning in ambiguous regions and greatly eases the extraction work for large images.

We ask the user to manually annotate a few spots on the image (see Figure 4.4b) as *foreground* (the animals) or *background* (for ground or other unwanted features). A specific Support Vector Machine (SVM) is then created and trained to discriminate foreground from background pixels based on their color in the user-defined annotations, and subsequently used to create a black and white mask (4.4c) of these features. Different regions, usually corresponding to different animals, are then extracted using a Watershed Transform (4.4d). Finally, we fit ellipses on each of these regions to retrieve their position, orientation and dimension (4.4e). The same arbitrarily chosen global direction is given to all ellipses, and can be flipped by the user if necessary.

Some animals can be wrongly captured as a single, large region if they were too close to each other. We detect these cases with an analysis on the distribution of extracted region sizes. In such situations, the created ellipses are too large, too small or of unusual

proportions. They are automatically detected and displayed in black (4.4f), allowing the user to manually draw new ellipses to replace them.

4.2.2 A PCF-based method for interactions

We model the local interactions between animals by extending the concept of the Pair Correlation Function (PCF) to distributions of ellipses within arbitrary convex polygons representing a herd.

Let us recall here the basic equation behind PCFs, previously explained in the last chapter. They are computed by weighting the contributions of neighbors at distance r with the Gaussian Kernel $k_\sigma(x) = \frac{1}{\sqrt{\pi}\sigma} e^{-x^2/\sigma^2}$. For the case of a 2D point distribution within the unit square domain, the PCF can be expressed by:

$$\mathrm{PCF}(r) = \frac{1}{A_r n^2} \sum_{i \neq j} k_\sigma(r - d_{ij}), \tag{4.1}$$

where A_r is a normalization factor defined as the area of the ring of radius r and thickness 1, and d_{ij} is the distance between points i and j.

From point to ellipse distributions: To extend the framework to distributions of ellipses, an appropriate distance encoding both the size and the orientation of each ellipse is required. We simplify the problem of computing an ellipse-to-ellipse distance by using two ellipse-to-point distance computations, one for each ellipse of every pair. This choice is consistent with the fact that PCFs compute and average the distances of each element with respect to the others in the distribution.

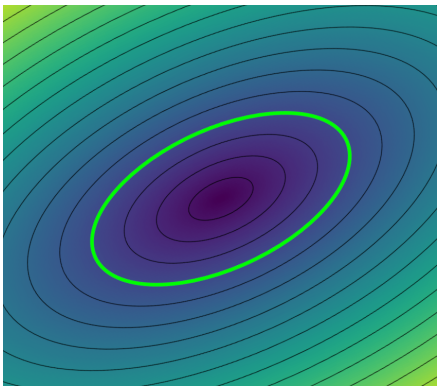


Fig. 4.5: Local distance field of an ellipse with isolines, based on orientation and size

In order to take into account the size and the orientation of the elements while computing ellipse-to-point distances, each point is expressed in local coordinates relative to the ellipse. More formally, for the point (x, y) in local coordinates, we compute its distance to the axis-aligned ellipse using:

$$d(x, y) = \sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2}}, \tag{4.2}$$

where x and y are the local coordinates of the point, and a, b re respectively the semi-major and semi-minor axes of the ellipse. More intuitively, this distance represents how

much larger the ellipse would have to be for the point to be lying on its contour. This scaling value is equal to 1 for all points on the ellipse (see Figure 4.5 for a visualization).

Generalizing the domain: Although the standard PCF formulation operates on square domains, herds can and usually do take arbitrary shapes and it is unreasonable to expect the user to search for input images that fit perfectly in a square, or to crop the input and lose precious information. While the free-form PCF formulation presented in the Section 3.2 of Chapter 3 could be used to compute an unbiased PCF from the herds, we will refrain from relying on it in this chapter. Instead, we can make a few assumptions and approximations about the input data in order to simplify the problem, and as a result reduce the computation time. The two assumptions that we make in this chapter is that the overall shape of herds can be reasonably approximated to a convex polygon, and that potential holes and variations of density that exist in this convex domain are meaningful and must be recreated by the model.

With this in mind, we choose to define the domain as the convex hull of the data points, which is an arbitrary convex polygon. Although accurately computing the area of an elliptic ring inside a convex polygon is complex, we can efficiently provide a good approximation of the solution: we consider an approximation of the ellipse as a collection of as many triangles as there are edges in the convex hull of the domain. Each of these triangles is composed of the origin of the ellipse, and two consecutive points on the hull polygon. We translate the points of the hull to the intersection between the ellipse and the convex hull. The area of the part of the ellipse in the domain is computed as the sum of areas of the triangles. The area of the inner ellipse is subtracted to get the area of the elliptic ring within the domain.

In practice, these computations are only performed for a few rings at regular distances around each point. The actual weighting coefficients, given by the ratio of the ring in the domain over the area of the full ring, can then be interpolated between these few computations to increase performances.

4.2.3 Editable descriptors

On top of the correlation curves used to keep track of the local relationship between objects, at least one other descriptor, the orientation field, is required to produce a result faithful to the original data. One last optional descriptor, the density map, can also be used in conjunction with the rest of the framework to replicate the density of regions in the same location in our output.

Orientation field: The orientation field is defined everywhere in the domain, and allows us to assign a plausible orientation at any hypothetical point within the original convex hull.

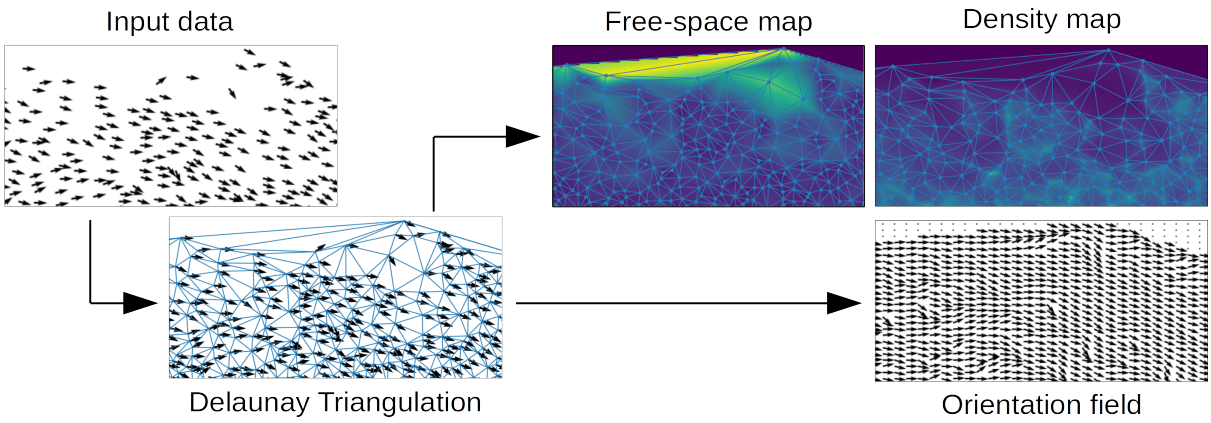


Fig. 4.6: The descriptors are computed by interpolation inside triangles from the Delaunay triangulation of the input data.

We compute this field by extracting the Delaunay triangulation of the centers of the extracted ellipses representing animals, and assigning the orientation of the ellipse to each vertex of the resulting mesh, as can be seen in Figure 4.6. When querying the field for the orientation of a new arbitrary point, we find the triangle that contains this location and interpolate the three angles of the vertices using barycentric coordinates.

Density map: Extracting density maps as well helps creating distributions as close to the original as possible by reproducing the position of the denser areas and empty regions.

Our approach to do so is similar to the one used for the orientation field. From the Delaunay triangulation of the ellipse centers, we use the area of the 1-ring, i.e. the collection of neighboring vertices, as an indicator of how much free space is available around at this location. We take the inverse of this area as an approximation of the local density and assign it to each point of the triangulation. We use barycentric coordinates interpolation withing each triangle to define density everywhere in the domain.

4.2.4 Synthesis algorithm

The editable descriptors computed from the input image are first transformed and modified to account for the general orientation of the herd trajectory at the key-frame, the obstacles in the environment and the size of the herd (each animal being of constant size, a herd of 100 will not take the same surface area on the terrain as a herd of 1000): after rotation and projection of the density map on the terrain, the part covered by obstacles is masked out. The map is then scaled in or out until it fits the requested number of animals. The same transformation is then applied to the orientation map.

Our synthesis method is based on the dart throwing algorithm (Algorithm 1) from the previous chapter. However, it has been modified to take into account the new features

added to the formulation, responsible for the handling of ellipses, herd shape, as well as non-uniform densities and orientations. As a result, our new synthesis algorithm takes as input the PCFs of the input distribution, the density and orientation maps, and the number of ellipses required in the target distribution, and outputs a new distribution matching the original. The method is summarized in Algorithm 2, and further detailed in this section.

Input: PCF, orientation field O, density map D, number of elements N
Output: Ellipse distribution

$fails \leftarrow 0$;
while $< N$ *elements accepted* **do**
    Sample ellipse from D;
    Sample orientation from O;
    Update PCF with new ellipse;
    if *error decreased or* $fails > max\_fails$ **then**
        Accept best ellipse;
        $fails \leftarrow 0$;
        continue;
    end
    $fails \leftarrow fails + 1$;
end

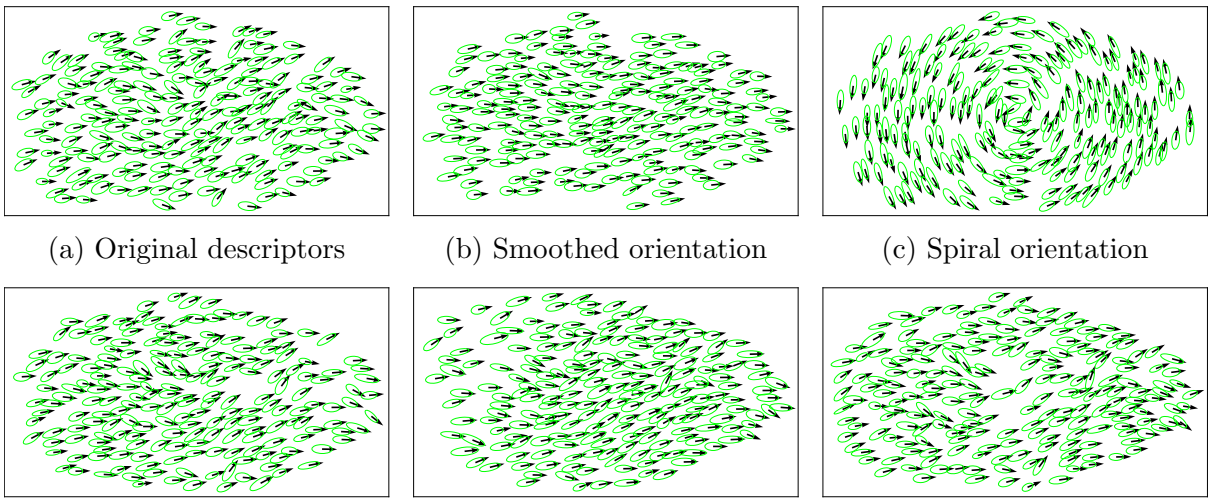
Algorithm 2: Synthesis algorithm

We first pick a random ellipse respecting the probability distribution of the density map. This sampling is done by computing running sums of densities per column of a discretized version of the density map, and then from one column to the next. This gives us the probability of having a point in each column, and for each column the probability of having a point at each pixel. We use these cumulative density functions to sample random ellipses according the density map. A rotation is then assigned to the ellipse using an orientation field query at its location. We update the current PCF with the impact of the new ellipse and compute the error E compared to the target PCF, after normalization to account for the difference in element count. We use:

$$E = \sum_r \frac{\big(PCF(r) - PCF_0(r)\big)^2}{PCF_0(r)} \tag{4.3}$$

where r spans distances to the ellipse.

The ellipse is accepted if adding it to the distribution reduces the error. Otherwise, to make sure that the algorithm does not get stuck in an infinite loop, we keep track of the best candidates while searching for ellipses. If the algorithm cannot find a good enough solution before reaching a threshold max_fails, the best candidate from the previous tries is accepted.



(a) Original descriptors (b) Smoothed orientation (c) Spiral orientation

(d) Uniform density (e) Left-to-right density (f) Radial density

Fig. 4.7: Effect of the editable descriptors on the synthesized result, with changes to the orientation field (top) and density map (bottom).

4.2.5 Descriptors as control tools

While it is possible to extract every component of our model from images and use them to create key-herds, the user can also edit the orientation map and density field to obtain a finer control over the result.

The density map is a completely optional parameter and removing it will yield an image that is similar but has local density extrema in different places. Manually painting a density map can be used for artistic purposes to alter the look of the resulting distribution.

Similarly, editing the orientation field leads to another form of control over the result. Indeed, replacing every orientation in the field by the same orientation produces a uniform flow, and smoothing or adding noise to the field can be used to control the level of detail in the emerging distribution.

The extent of this control is shown in Figure 4.7, with the top row showing the effects of orientation field changes while the bottom row showcases the impacts of different density maps. Sub-figure 4.7b shows the result after merging the orientation field with orientations pointing straight right, and figure 4.7c shows the effect of a pure rotational orientation field on this example. While changes in the density map are more discreet, its effects can still clearly be seen. Indeed, the most important empty spots are mainly located to the left of the herd in figure 4.7e while they are in the center in sub-figure 4.7f.

4.3 Herd animation

4.3.1 Global herd trajectory

We extend the idea of key-framing an animation to a group of animals. While traditional key-frames can be used to represent important poses of an individual character, our key-herds encode a full group of animals at a specific point in time.

From this input, a global trajectory for the herd, modeled using an interpolation spline between the centroids of key-herds and the extra position-only key-frames, is generated and projected onto the terrain. The herd can be seen as a local frame that moves along this global trajectory, and within which the animals will have some adequate relative motion.

4.3.2 Generating individual movement

We first compute a consistent labeling to pair animals within successive key-herds, and then use microscopic simulation to define individual trajectories, as described next.

Pairing based on optimal transport: Establishing a good correspondence between the animals of a key-herd and the following one is essential to generate fluid motion. In our work, we use optimal transport [BvdPPH11] for this computation, after virtually superposing the two successive key-herds to best align their centroids and orientations (by convention, the X axis in a local frame). More precisely, the mapping is computed as the optimal transport of one unit of mass for each animal located in local coordinates relative to the key-herd, to the local positions of the animals in the next key-herd. Depending on the type of animation required, the metric used for this computation can be the Euclidean distance, the Euclidean distance to the nth power to increase the penalty on distant mappings, or a metric that penalizes matching along one axis more than the other.

Herd-frame-guided simulation: The key idea to generate individual animal trajectories that both interpolate their assigned position in two successive key-herds, but also follow the herd trajectory in-between, is to use microscopic simulation following moving guides defined in the herd frame. This way, while the guides model the necessary relative motion within the herd, the steering behavior in world frame enables to avoid collisions with other animals and with obstacles.

We use a simple steering model based on five forces to generate individual motion. The first three forces are separation, alignment and cohesion [Rey87]. They are responsible for the general behavior of the individuals. They are completed by an extra force to handle collision with the environment: it steers the animals away from obstacles if their predicted future position given their current position and velocity gets too close.

The last force controls the animal movement by giving them guides to follow. The latter straightly move, in the local herd frame, from their previous position in a key-herd to the next. Guide positions are transformed to world frame at each time step, while the herd frame follows its global trajectory. In addition, to account for the actual speed of the associated animals, which can be slowed down by other interactions, the position of the herd frame along its trajectory is set to move forwards only when all the associated animals are about to reach their guide.

Note that in our current implementation, the orientation of animals can be computed either from their movement or by interpolating the orientations of successive key-herds.

4.4 Results and discussion

4.4.1 Results

Our framework is implemented in Python for herd analysis and synthesis and in C# on the Unity Engine for user interaction and rendering. Timings reported in Table 4.1 were run on an Intel Core i5 processor at 3.3GHz equipped with 8GB of memory.

Herd size	Ex.1 $N = 1054$	Ex.2 $N = 43$	Ex.3 $N = 220$
5	1.8s	0.5s	0.7s
25	1.9s	0.5s	0.8s
100	2.2s	5.1s	6.2s
200	3.5s	16.2s	15.4s
500	15.9s	73.4s	59.3s

Table 4.1: Benchmark of synthesis time

Figure 4.8 shows synthesis results for three different input images depicting herds of widely varying sizes and visual patterns. For each input, generation is performed with different target herd sizes (around 50, 100, 200, and 500+ animals) to illustrate the robustness of our method. We can see that the overall shape and orientations are preserved, and that the less dense spot of the herd near the bottom is also encoded by the model (leftmost example). In the middle example, where dense and sparse regions alternate, we demonstrate the capability of our approach to generalize to a larger sample size: the dense and sparse spots are duplicated as necessary along the herd shape to preserve the local consistency of the distribution as the number of animals increases. Finally, the rightmost example proves to be the most complex one, due to the presence of strongly constrained formations such as lanes. This is indirectly encoded by the anisotropic ellipse distance and the density map, but is not strictly enforced by the system. This results in varying degrees of accuracy in the generated distribution. This is visible for 500 animals, where the formation of lanes is harder to recognize.

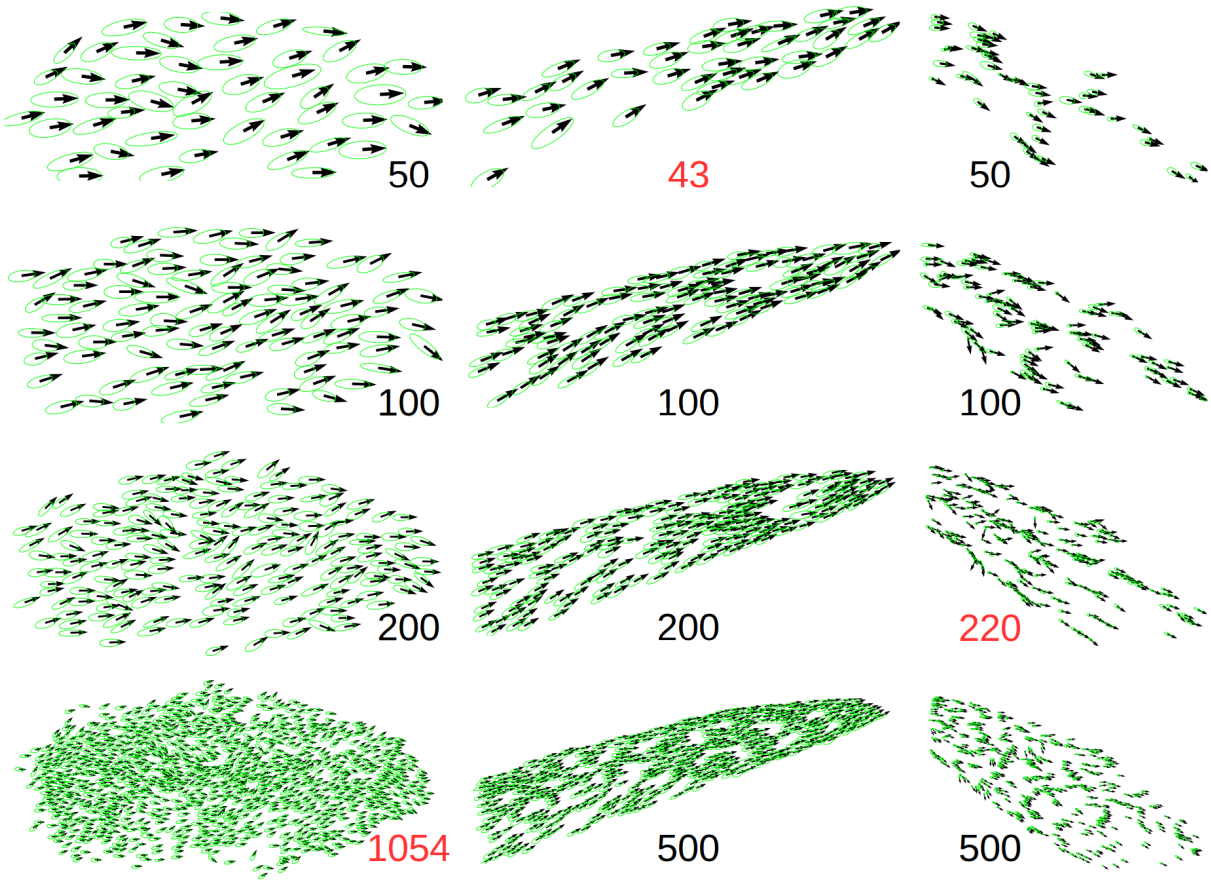


Fig. 4.8: Generated herds with different target herd sizes. The size in the input image is in red.

Animation results are depicted in Figures 4.2, 4.9 and 4.10. In particular, Figure 4.9 shows a comparison between the input image and the resulting animation of the animals close to the key-frame where this particular image is placed. Similarities in shape and disposition of the animals are visible, despite a different number of animals and minor distortions caused by the global path of the herd. Figure 4.10 demonstrates the interaction of a small herd with a tree. The steering agents used for individual movement are able to avoid the obstacle while maintaining the overall arrangement.

As these results show, our method can handle challenging animation scenarios with significant changes in the herd shapes due to user input or environmental constraints, while maintaining the global aspect of the herd distribution learned from the input herd photos.

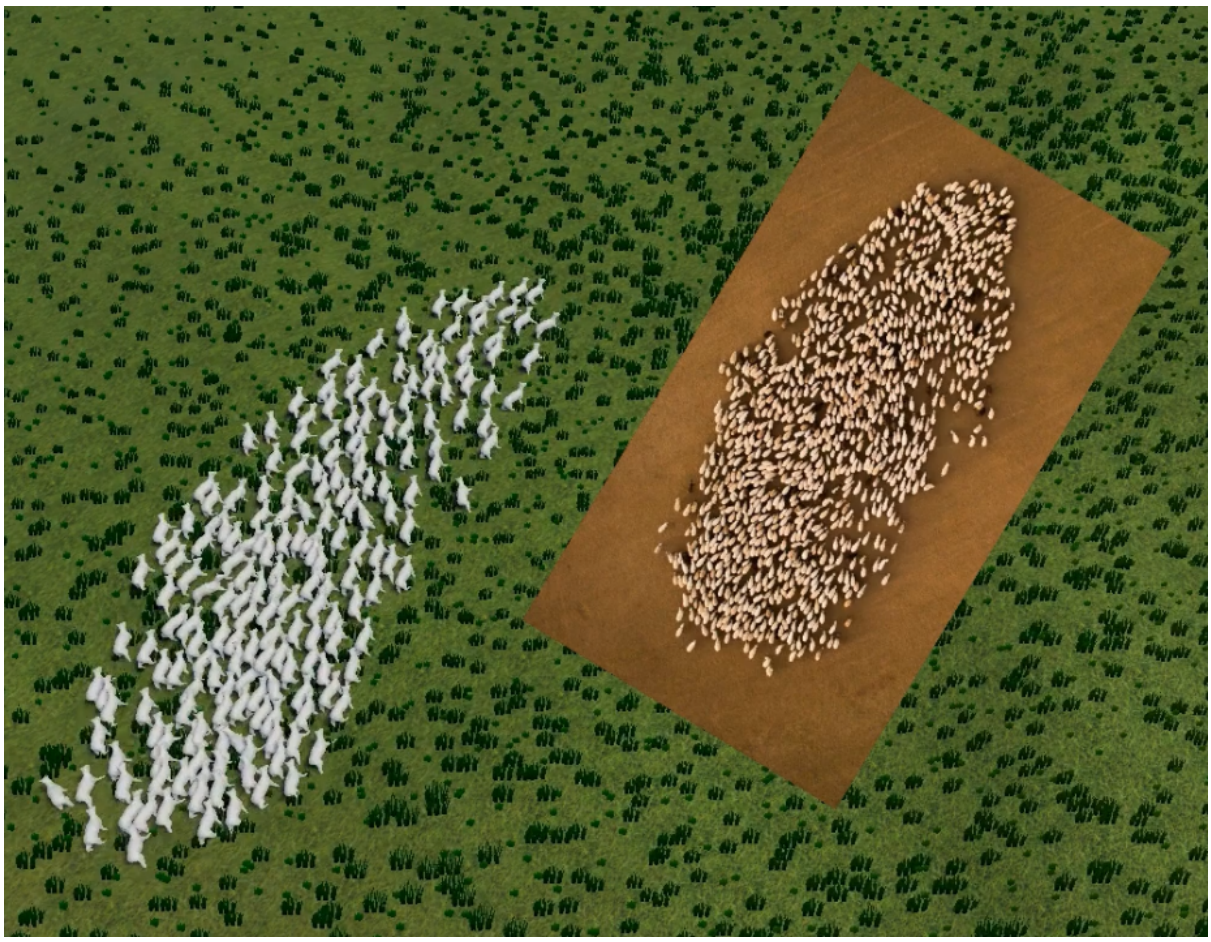


Fig. 4.9: Comparison of the result with the input image, with a far smaller number of animals.

Fig. 4.10: Frames from an animation showing interaction with an obstacle.

4.4.2 Limitations

Firstly, although orientation fields are properly reconstructed at herd key-frames, we failed to achieve an animation of individual animals that both matches this orientation field and results in natural motion, as can be seen in Figure 4.11. This comes from the fact that

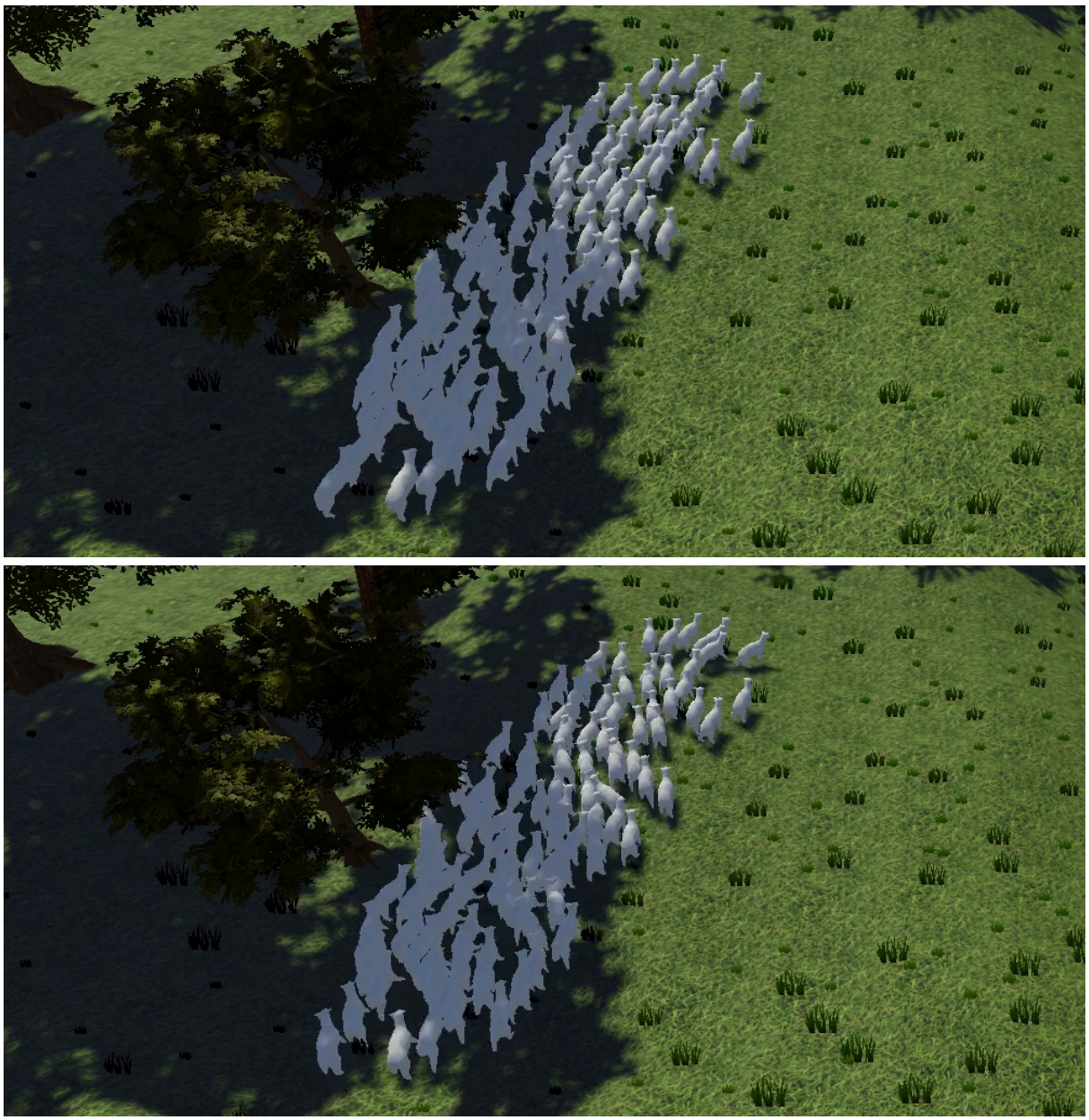


Fig. 4.11: Compared to computing orientations based on animal movement (top), interpolating orientations between key-herds (bottom) helps to reproduce variety in a herd, but can result in side-way movement.

linearly rotating guides between their target position and adding angular forces within the steering behavior produces strange behavior such as animals moving side-way or even back-way. Achieving animation that match orientation constraints is therefore left for future work. Secondly, although we offer a precise control of the statistical distributions of animals at key-frames, our method does not offer any guarantee on the plausibility of the distribution in-between. In particular, even when the same reference image is

used for two successive key-herds, the fact we use microscopic simulation to process local interactions in-between, may alter the intermediate distribution. A last limitation is the difficulty to reproduce constrained formations as the queue of animals in the rightmost example of Figure 4.8. This is due to the interpolation of the density map, which diffuses density where there was none in the input image, and to the distance used that does not discriminate front from side, apart for the inherent shape of the ellipses.

4.5 Towards herd animation from video

While the work presented in this chapter provides an intuitive tool designed for the authoring of herds of animals, its input is restricted to images by design. Compared to videos, this facilitates the extraction of data and increases the quantity of available data, but as a consequence contains much less information about the behavior of underlying animals. For example, the average and relative speeds, the type of motion (*e.g.,* kangaroos or rabbits hopping) as well as special events (*e.g.,* an animal falling down, getting back up and then running to get back to the herd) are all lost when only considering still images. We present in this section a work in progress intended for the learning, synthesis and authoring of animated herds, where information from short video sequences is used to control groups of animals.

4.5.1 Extracting meaningful data from video clips

We choose to focus on information from short video clips (of a few seconds) to avoid putting difficult constraints on a user looking for data. Just like for images, the first stage is to extract the relevant information from the input, which will later on be used for analysis and synthesis. Working with video data presents both pros and cons when compared to static images. On the positive side the successive frames provide additional information about the targets, which can be exploited for a higher accuracy. This includes mostly the movement of animals against a static background, which is easy to detect. However, video data also come at the cost of many constraints, such as blurred frames or a mobile camera that introduces a bias in velocity computations.

Figure 4.12 presents the different steps that we follow in order to successfully extract data from the input. Starting with the original video, we keep the same principles as for images and ask the user for a rough annotation of elements in the input. We provide a simple tool for this purpose (visible in Figure 4.12b) equipped with brushes for the different annotations and standard controls over the timeline of the clip (bottom of the image). The user is asked to mark a few sections of the background in red. In cases where the input is unstable, the user can interactively split the timeline to annotate different parts of the video independently. Optionally, the user can annotate some pixels as containing animals (green) or irrelevant data such as text displayed over the video (black). To extract movement information, we compute the optical flow (Figure 4.12c,

(a) Input video clip (b) User annotations (c) Optical flow

(d) Movement detection (e) Color detection (f) Animal detection

Fig. 4.12: Semi-automatic detection pipeline on video clips. Our method combines color and movement information from a few manual annotations to accurately detect animals in short clips.

with artificial colors representing orientation and intensity) over the entire example. In this case, we used PWC-Net [SYLK18] for the quality of the output over dynamic data with small elements, but our method is not bound to a particular optical flow computation method.

We use the optical flow in conjunction with the user annotation to compute the relative movement of pixels belonging to the background during the video. This allows us to estimate a 3×3 homography matrix H that represents the screen-space transformation of the ground pixels from one frame to the next. The matrix is optimized to satisfy

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.4}$$

for all points (x, y) mapped to (x', y') in consecutive frames. Inverting H provides us with a matrix that can be used to compensate the camera movement during the clip (illustrated in Figure 4.13), thus removing the initial bias and allowing the detection of moving objects over the static background. This stabilization with the help of an homography relies on the assumption that the terrain on which the animals move can locally be approximated to a plane.

We rely on two estimators to detect the location of animals, one based on their movement and the other one based on their color. The first one (4.12d) is computed from the difference between the measured optical flow and the theoretical optical flow representing the movement of the background. This movement map displays areas where

(a) Input clip

(b) Stabilized video

Fig. 4.13: Result of stabilization using the optical flow and inverted homography. Screenshots were taken after 0, 5 and 10 seconds of video, respectively.

objects are moving, but can produce false positives if the camera movement is not properly compensated or the lens of the camera produces distortions near the corners. We apply a temporal Gaussian blur and adaptive thresholding to minimize those issues. For the color-based estimator (4.12e), we use the same method as in Section 4.2.1: the image is binarized after training a SVM over the color of pixels annotated by the user. Combining the information from our two estimators provides an accurate detection of animals (see subfigure 4.12f) from a video, with a minimal amount of work from the user. While a false positive is visible at the top right of the image, it is a rare occurrence and can be easily corrected manually.

4.5.2 Avenues for animated synthesis methods

While we do not have completed our research regarding an authoring tool for fully dynamic herds, this direction seems promising so far. We will discuss here a few of the experiments that we have carried, or plan to carry out for this purpose. Two main approaches can be considered to extend our existing method from static to animated data.

The first option is to operate at the distribution level, and would aim at the synthesis of already animated distributions. In this formulation, the objective is to add a temporal component to curves encoding the distribution, making it possible to recreate the animals directly animated instead of relying on an independent animation step. This additional temporal component can take the form of a new dimension (turning the PCFs into 2D surfaces, on which computations would be expensive), a would require a more expressive metric, or a separate encoding mechanism (similar to the orientation and density maps used in Section 4.2.3).

An alternative approach considers this problem from an animation point of view. From the motion data extracted from the input video, we propose to generate a stochastic motion model for individual animals. This model could be used to individually predict plausible trajectories for each animal based on their neighbors and their direct environment. This can be seen as a more specific version of the steering agents used in this chapter, automatically generated to mimic behavior learned from the input data. On top of this microscopic simulation, global control over the shape and distribution of the herd could be applied with static synthesis methods. After moving the animals using the custom model, deviations from the target distribution would be computed and the positions of outliers be corrected (similar to the gradient descent used in [ÖG12]).

We plan to explore these research directions in the next few months.

4.6 Conclusion

We presented in this chapter an easy to use authoring tool for herd animations. It enables the user to key-frame a herd over a terrain by extracting the successive visual appearances from photos, while automatically taking the 3D environment constraints as well as the target number of animals into account. This is achieved by dissociating animation from placement and only considering static synthesis at specific locations. Once the target distributions are encoded at each key-frame, the user is allowed to manually update the orientation and density of animals in the herd thanks to the use of auxiliary fields representing these two variables during synthesis. At runtime, the animals in successive key-frames are then automatically paired and animated with an agent-based microscopic simulation method that handles collisions between animals and with the environment.

In addition to matching orientation and density fields, the animation could be further improved using known animal features such as their type of movement. For example, simulating movement ranging from smooth to cyclic (jumping of kangaroos or frogs) or to Brownian motion (flying insect swarms) would greatly increase the variety of species that can realistically be generated. An extension to 3D would also be beneficial when modeling flocks of birds or schools of fishes, since control can be difficult to achieve with traditional approaches.

Still, the velocity and behavior of animals cannot be learned from a photo. With this in mind, we discussed possible extensions to our method to support video clips as input data. Paired with our dedicated animal detection pipeline, this could allow the automatic synthesis of fully dynamic herds while retaining a simple and intuitive user experience.

In the next chapter, we take a step back and present a biology-inspired method to unify the instantiation of vegetation and animals on a terrain. At the scale of the entire environment instead of a local neighborhood, we explore the relationships between different species and propose a formulation allowing control, stability and plausibility without the need for an expensive and complex simulation.

Chapter 5

Authoring complete ecosystems

Contents

Beautiful landscapes result from the long and short-term co-existence of various forces and effects, such as weather, erosion, vegetation – and animals. While all these phenomena have a strong influence on landscape appearance, their effects are not always well-understood, and only some of them have been considered in Computer Graphics. In particular, a large body of previous work has addressed terrain generation and plant ecosystems, either in isolation or combined. Despite these efforts, humans can still easily distinguish fully automatically synthesized landscapes from real ones, and a significant amount of manual editing is often needed to improve realism.

Our first key observation is that animals have a critical visual impact on landscapes, as they compete for space and resources, shaping clearings and trails through the vegetation, which, in turn, affects erosion. Therefore, the standard pipeline where a terrain is generated, eroded, covered with plants, and where animals are only added on top while neglecting their interactions with the other elements, should be revisited.

License: Bernt Rostad, CC BY 2.0

Fig. 5.1: Narrow trail in a forest, used by deer and other wild animals.

The second key observation is that modeling a landscape as the result of the co-existence of a variety of life forms can be done consistently without a full simulation while enabling intuitive user control. This point is essential since launching a full, simultaneous simulation of terrain formation, vegetation, and wild-life would only bring indirect user control. Besides, the involved multiple time scales would make the simulation unmanageable.

We propose a solution to increase the quality of landscapes in Computer Graphics by generating full ecosystems with fauna and flora interacting with the terrain. Our approach unifies the treatment of plants and animals by considering them as different levels of a food chain, which is iteratively instantiated, from lower to upper levels. Moreover, we designed our method to allow intuitive authoring from the user,

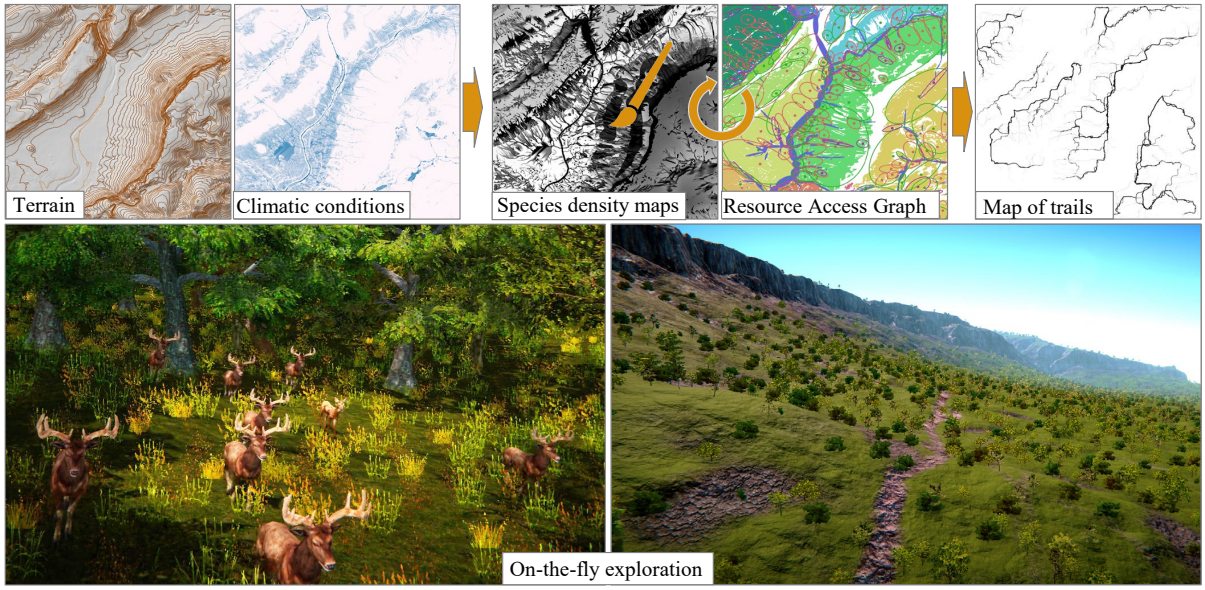


Fig. 5.2: From an input terrain, a set of climatic conditions, and proportions of competing species, our algorithm computes density maps per species by iterating on an embedded resource access graph, and finally extracts a map of animal trails (top). This enables the on-the-fly exploration of consistent ecosystems (bottom), showing vegetation, animals and the trails they generate. The user can edit maps at any stage using painting tools.

as well as the on-the-fly instantiation of the resulting ecosystem during interactive exploration, by positioning plants and animating animals along their daily paths between resources.

We achieved these results thanks to a trade-off between efficiency and exactness through a system providing several approximations and key choices: first, we allow authoring by taking as input the desired proportion of present species at each level of the food chain. We compute the actual number of specimens and their distribution over the terrain from this input, thanks to a new procedural competition algorithm ruled by the following hypothesis: the resulting ecosystem should not only be consistent in terms of resources, but also at a quasi-equilibrium state. The latter meaning that only the surplus produced by each species during one year can be used as food at the next food chain level. Second, the procedural competition method makes use of a novel hierarchical data-structure called the Resource Access Graph (RAG), which embeds resources for each species with their location and accessibility in the form of a layered graph embedded on the 2.5D landscape. This structure encodes both interactions with the previous level of the food chain and access paths between resources. Therefore, the completed RAG provides all necessary information to build a map of eroded trails over the terrain.

In a typical use-case, the user starts by specifying the terrain, and a set of species with optionally their desired proportions after competition. This input is used to compute the closest steady-state ecosystem, in the form of a set of editable density maps for each

species of plants, herbivores and then carnivores over the terrain. At each level of the food chain, the impact of the generated species is propagated back to the resources it consumes and the paths between them, enabling to account for grazing, foraging, and erosion along animal trails. The resulting ecosystem, composed of a set of density maps and trail maps, is finally instantiated during interactive exploration. In particular, we upsample the computed animal paths from the RAG while taking into account the terrain features and plant density maps, so that animated herds can be consistently displayed and animated. Figure 5.2 shows an example of interactive authoring and exploration of a populated landscape.

We claim the following scientific contributions:

- We introduce the RAG, a hierarchical, biology-driven graph of resources, which encodes interactions among food chain levels and enables us to model paths between resources.
- We introduce a procedural competition algorithm to build a fast approximation of a steady-state ecosystem at each level of the food chain.
- We propose a method for the consistent, on-the-fly instantiation of plants and animals during interactive landscape exploration.

We will start this chapter with a short case study of interactions and interdependencies similar to the case of ecosystems, but applied at a smaller scale and for a different context. This will be used to draw similarities and discuss the different challenges related to the case of ecosystems, but on a more constrained example. We will then provide an overview of the pipeline for general ecosystems, before detailing our data structure, competition algorithm and instantiation process.

5.1 Case study: effect and visualization of skiers

We focus in this section on the visual impact of skiers on snow-covered landscapes, as well as their interactions and effects on different types of snow.

5.1.1 Context

Snow-covered mountains, such as the one visible in Figure 5.3 (left), can present some of the most breathtaking landscapes. Their complexity stems from the impact of many different elements including the altitude of the mountain, the uneven melting of the snow depending on sun exposure, the effect of the wind on snow, or human activities. The goal of the work [CEG+18] from which this section originates is to simulate the main causes affecting the visual aspect of mountainous landscapes, to recreate their complexity while

Fig. 5.3: Snow-covered mountainous landscape (left) are shaped by many factors such as wind, altitude, sun exposure, and more. Human activities (right) also play an important role by leaving tracks, compacting snow and triggering avalanches.

allowing interactive user control over the result. To reach this goal, the above-mentioned phenomena are modeled as stochastic events computed on the GPU, and acting on a layered representation of the terrain describing both the underlying rock and multiple types of snow (*e.g.,* compacted, stable, unstable, powdery).

We will focus this section on one type of event that greatly affects the aspect of such landscapes, and presents similarities with the way animals interact with their environments in complete ecosystems. Indeed, skiers leave characteristic tracks in the snow, akin to how animals create trails to and from places of interest. They also have direct impacts on their environments, as they compact the snow in their path and can even trigger avalanches.

The size of our simulation grid (10m per cell) only allows a consideration of the general direction of the skiers. While this is sufficient when modeling the impact of skiing on snow state during the simulation, we need more precise paths to achieve realistic rendering. We thus opted to integrate a procedural method to generate plausible refined tracks on demand without increasing the computational costs during the interactive simulation.

5.1.2 Skiers in snow-covered landscapes

Global path search. Skiers can be either manually placed on the terrain by the user, or automatically placed by the system. In this case, each cell has a low probability of spawning a skier. This probability is influenced by the length and viability of a ski route. Therefore, we pre-compute a map registering the distance from each cell to its farthest down-slope end point (local minima or edge of the terrain). We use a simple cellular automaton that finds minima on the terrain and propagates the distance to them in subsequent steps to efficiently compute this map, directly on the GPU. This distance map is used both as a weighted mask when automatically spawning new skiers, and as a guide to discourage skiers from reaching dead-ends, as discussed next.

Fig. 5.4: Global paths (left) are computed in real-time and represent the rough trajectory of skiers going down the mountain. Refined tracks (right) are computed for rendering purposes, to detail the actual path taken by individual skiers.

Large scale paths are computed to approximate general direction and movement of the skiers, without the detail of the curves used to regulate their speed (Figure 5.4 (left)). For that, we take into account the slope of the mountain by defining an ideal slope angle s_t that skiers will be comfortable with and try to follow, and make sure that there is enough snow for them to ski on. In this work, skiers are modeled as independent agents, defined by their position and orientation, responsible for deciding their next move using a local search, based on the amount of snow present in neighboring cells, the corresponding relative slope s_n and a pre-computed weight w_d related to the distance to a terminus. In practice, skiers have a small lookahead window of a few tiles, and will steer in the best candidate direction defined by the center of a nearby cell. The probability for a skier aiming towards a given cell n is:

$$P(n) = \mathbf{1}_{snow>threshold}\, f(|s_n - s_t|)\, w_d, \tag{5.1}$$

where $\mathbf{1}$ is the indicator function, s_n is the slope between the current cell and cell n, and f is a function that assigns a weight, which can be changed to tune the behavior of skiers with regard to slope. To avoid sharp changes in direction, a smooth transition to the new steering direction is computed and applied to the orientation of the skier. The steering direction is re-evaluated at each animation step.

Refined tracks, visible in Figure 5.4 (right), are created for rendering purposes. They are approximated based on the observation that the precise movement of skiers is analogous to sine waves of varying amplitude and frequency, with sharp turns used to slow down and straight paths to gain speed. With this in mind, we model movement on each straight segment of a global ski trajectory using $\mathbf{p}(t) = a\sin(2\pi f t)$, where a is the amplitude and f is the frequency, dynamically updated with the terrain's varying slope.

Indeed, a skier moving straight down a mountain will go faster than one with a trajectory following the isoline. To account for this, we compute the effective slope s_e of the skier's trajectory as:

$$s_e = \arcsin \frac{\sin(s_n)}{2f\,l} \qquad l = \sqrt{\frac{1}{4f^2} + 4a^2}, \tag{5.2}$$

where l is the distance between sine curve extrema. This provides the local frequency value required for skiers to reach their comfortable target slope s_t:

$$f = \frac{\sqrt{\sin^2(s_n) - \sin^2(s_t)}}{4a \sin(s_t)} \tag{5.3}$$

Continuity with the previous refined position and orientation of the skier is ensured by choosing an appropriate starting phase value along the sine curve. At each animation step, the resulting movement detail is mapped on the fly onto the lower resolution trajectory computed using Eqn. (5.1). This is done using a local update to a ski-tracks texture layer covering the whole terrain. An alternative is to export this texture as a displacement map for off-line rendering.

Interaction between ski tracks and snow is two-way. Once a skier enters a cell it transforms a fixed amount of snow from unstable to stable, or from stable to compacted if no more unstable snow remains. If there is unstable snow still remaining then the probability of an avalanche is increased. Conversely, the impact of snow on rendered ski tracks is taken into consideration: as snow is deposited along the path or shifted by the wind, the tracks fade dynamically depending on the quantities involved.

5.1.3 Discussion

We demonstrated in this section the importance of interactions between individual elements and their environment. Apart from the direct mark left by skiers in the snow, their ability to compact snow under them and to trigger avalanches also results in drastic changes in the resulting landscape, that can give the user critical information on the environment. All these aspects have equivalents in the case of ecosystems. The ski tracks have an effect similar to that of animal trails, showing information on important places and recurring activities in the environment. They also be seen as similar to animal prints left in the snow, which instead give recent information about a single animal. Besides this, the main effects of skiers on the environment – compacting snow and triggering avalanches – can be seen as a parallel to animals consuming resources and being consumed by predators.

Fig. 5.5: Fresh bear tracks left in the snow.

This case study also provided interesting insights on how ski tracks and animal trails can be handled. Indeed, this example shows that a complete and precise simulation of the elements is not required to reach a polished result. In the case of skiers, we used a procedural model to recreate a detailed version of the tracks after the simulation, using

coarse information about orientation and the global path of skiers. This can be applied to the case of animals, where a simulation of animal movements over an extended period of time would be an intractable solution to generate trails.

We go back to the case of ecosystems in the next section, and provide an overview of our method designed to handle interactions between species and with their environment.

5.2 Populating a complex ecosystem: overview

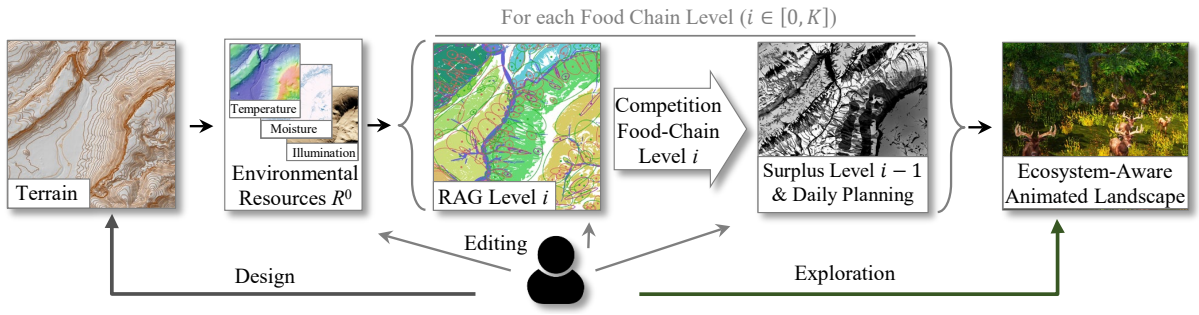


Fig. 5.6: System overview: After initializing environmental resources, we process each food chain level, starting from the bottom, and subsequently compute the corresponding level of the Resource Access Graph (RAG), the result of the competition algorithm for FCL^i, and the remaining surplus for FCL^{i-1}. Finally, we gather the resulting density maps (species and their surplus), yielding the ecosystem-aware-landscape, together with a map of eroded trails on the terrain, computed from the RAG and the density information. During the interactive exploration, the ecosystem is instantiated from its collection of maps. The user intuitively edits results by over-painting any of the maps, the edits being consistently propagated up the processing chain (the user being warned if they increase a species density beyond the available resources).

Both artists designing virtual worlds for movies or games and scientists studying the effect of climatic changes on ecosystems share an interest in generating and exploring consistent 3D landscapes with flora and fauna. Both, however, need control over the set of species they observe on the terrain, either to author a virtual world or to restore past ecosystems from data –such as animal bones and plant pollen found by paleontologists– that inform them about the nature and proportion of species.

In order to match these authoring needs, we do not simulate competition between species expressed as differential equations, since this standard approach would lead to a time-evolving ecosystem with little control over the resulting proportions of species. We instead build on user input to create a consistent landscape that matches the expressed authoring choices. This is done by iteratively considering a given level of the food chain, from bottom to top, to progressively generate the number of specimens for each species that will form a consistent ecosystem. The latter is supposed to be in a quasi-equilibrium

state, meaning that only the surplus produced by a given species is used as a resource that feeds the specimens of another species higher in the food chain.

Although the number of generated specimens can be limited, our solution ensures that the user-specified proportions between competing species at each food chain level are maintained. We use a novel data structure: a hierarchical directed graph called the **Resource Access Graph** (RAG), to encode the interactions between the different levels of the ecosystems, as well as their interactions with the terrain of interest. This structure enables us to output a modified terrain that embeds the trails eroded by animals' daily itineraries from a resource to the next.

The key concept behind our solution (see Figure 5.6) is the mutual interaction among species through food chains. For $i \in [0, K]$, a food chain level is denoted by $FCL^i = \{S^i_j\}_{j=0..N_i}$, and is a set of species S^i_j that feed from species in FCL^{i-1}. The FCL is updated in discrete iterations so that the previous state depends on the new one leading to a chain of $FCL^0 \to FCL^1 \to \cdots \to FCL^K$. In our implementation, we use three food-chain levels, namely plants, herbivores, and carnivores.

Let R^i_j denote the set of **resources** on the terrain used by a given species $S^i_j \in FCL^i$. These include areas where they can drink and areas where they can find either plants or animals of interest, among S^{i-1}. We denote by R^0 the natural resources for plants, including soil nutrients, sunlight, and moisture, computed over the terrain from the terrain soil map and climatic conditions.

5.2.1 Input and output

The input to our algorithm is a terrain given as a digital elevation model ($16\,\mathrm{km}^2$ with a resolution of 1 m in our examples), together with a map of soil nutrients, and the full set of species (including both plants and animals), denoted by $S = \{S^i_j\}^{i=0,\ldots,K}_{j=1,\ldots,N_i}$ the user defined on this terrain, with the desired proportions at each level i of the food chain.

The user may also directly provide maps of natural resources for plants such as moisture, temperature, and yearly light available over the terrain. In our implementation, we adapt the approach from [CGG+17] to consistently compute these maps from the terrain orientation, altitude, latitude, and some user-defined climatic information, including extreme temperatures at sea level and average sunlight and precipitations on a monthly basis.

In addition, the algorithm makes use of widely available information about living species. For instance, the plant database [TBDB] includes a set of needed resources (minerals, light, and water) as well as the minimal and maximal temperature the plant can withstand. For animal species, in addition to the needed resources (water, food in terms of species from the previous FCL), the database [ADWDB] also includes the average motion speed, the ability to climb slopes and to cross rivers; information that we use to compute their ability to travel over the terrain.

The output of our method is a set of maps over the terrain, representing the density of presence of any given species, and a map of eroded trails. Density maps for animals are uniform over a set of species-specific *confinement regions* that are delimited by frontiers such as rivers or cliffs that the animal cannot cross. We use uniform densities to represent the ability of an animal to move anywhere in such a region. In practice, species typically have non-uniform probabilities of presence over their confinement regions, depending on the location of the resources they need. Therefore, we also compute daily itineraries for animals within each confinement region, to model their typical routes. The resulting set of maps defines an ecosystem-aware landscape populated with animals, plants of various species, with clearly distinguishable trails.

5.2.2 The Resource Access Graph

The Resource Access Graph (RAG) is the main data-structure used during computations. This is a hierarchical, directed graph embedded in the terrain. At each level of hierarchy $i \in [0, K]$, the RAG's nodes are individual resources available for the set of species FCL^i, located on the terrain (represented by the centroid of the corresponding region if they have an extent in space, such as meadows of grass). Edges –labeled by indicators of the species– encode the ability of a given species to travel from one resource to another and they are valued by the average time it takes (we use directed edges if traveling times back and forth are different).

The RAG hierarchy represents the fact that species are resources for other species higher in the food chain, and are thus encoded within resource nodes (*e.g.,* a population of antelopes in a sub-region of the terrain, which is a food resource to wolves, is expressed as a resource node for carnivores in FCL^2). Therefore, nodes at each level of the RAG are built from the connected components of the RAG at the previous level (*e.g.,* the resource graph used by these antelopes, which live between the river and the cliff, and travel between different resources to feed in this region) as shown in Figure 5.8. Therefore, the RAG cannot be precomputed at once but needs to evolve progressively in order to reflect the interactions between a previously computed food chain level and the next, as detailed below.

5.2.3 Processing pipeline

Our algorithm (see Figure 5.6) first uses the input data (terrain maps and climatic conditions), discretized in a regular grid, to compute the set of natural resources R^0. We initialize the first layer of the RAG (*i.e.,* resources for plants) by considering each cell of the grid to be an individual node.

Then, we launch the iterative computation of the ecosystem, where we successively apply the same computational process at each level of the food chain (in our implementations: plants, then herbivores, then carnivores):

1. Building RAG^i: We create the i-th level of the RAG, which encodes the resources for species from FCL^i, their location over the terrain, and the ability of each $S^i_j \in FCL^i$ to travel between them;

2. Solving competition for FCL^i: We use a new procedural method to compute steady-state competition results for FCL^i that best matches the proportion of species specified by the user at this level of the food chain. We output maps with a uniform density of presence for each species in each of the confinement region in their accessibility map.

3. Computing surplus for FCL^{i-1}: The surplus is defined as the difference between production and death for a species during a given period (related to growth-rates for plants and birth/death rates for animals). It can be consumed by the species higher in the food chain with no impact on the ecosystem equilibrium. We take into account the use of the available resources on the terrain by species in FLC^i to compute density-maps of surplus for all species in FCL^{i-1}. These surpluses will be displayed in the form of younger animals or plants during interactive exploration (the species being supposed to be at equilibrium, older specimens are to die during the year).

Finally, the collection of density-maps is exported to form the ecosystem-aware-landscape, together with a trail map extracted from paths in the RAG, and used to model animal-driven local erosion.

Users can interactively explore the resulting landscape thanks to our on-the-fly instantiating method. They can also further edit the generated ecosystem and its local embedding within the terrain, by editing maps from any Food Chain Level. While increasing the quantity of a given species may not match realism due to the lack of resources (a warning is displayed), decreasing densities is always possible and will propagate up the processing chain for consistency. This can be used to model external events such as specimen destruction due to fire or diseases.

5.3 Resource Access Graph

We focus on medium-scaled terrains, *e.g.,* $4,000 \times 4,000$ Digital Elevation Model (DEM) grids with one-meter cells. Therefore, directly making queries onto a set of maps to check for types, quantities, and accessibility of resources, while considering the physical needs and traveling abilities of the considered species and their uses of rivers and other water sources, would not be practical. To simplify computations and allow interactive user feedback during editing tasks, we extract all the necessary information from maps that span the whole terrain and represent this information hierarchically using a graph structure. Each node of this graph represents a single grouped resource (*e.g.,* a meadow, a riverbank from where animals can drink, or a grazing herd of antelopes), and edges

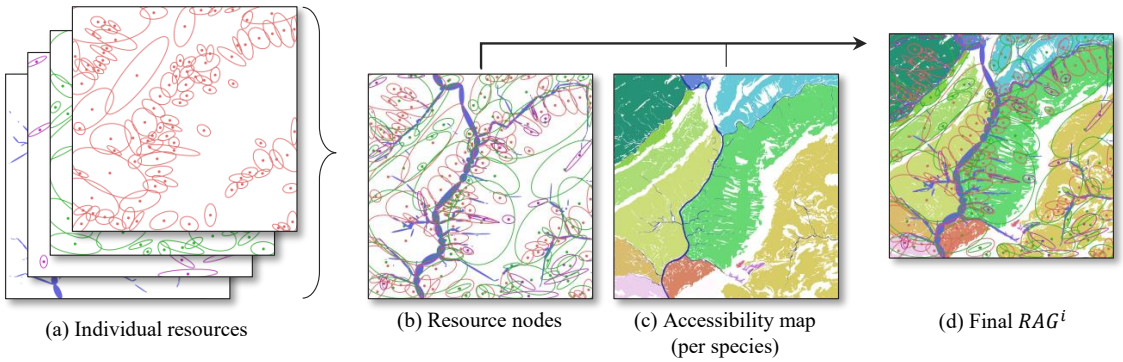


Fig. 5.7: Details of the computation of a RAG level over a terrain, from resource nodes (a,b) and accessibility maps (c) for each specie. In the final structure (d), all resource nodes lying in the same colored region are connected by an edge with the label of the considered species.

represent the existence of a path on the terrain, which allows a given species to travel from one resource to another. This graph is hierarchized into successive layers that correspond to the different levels of the food chain (see Figure 5.8).

5.3.1 Definitions

A Herd is defined as a group of animals, where the number of instantiated animals is extracted from zoological data. Depending on the species, we consider a single animal to be a herd of size one.

Spatialized resources on the terrain: In this work, we are progressively building and using areas where a given quantity of resources (*e.g.,* light, water, a given plant, or animal) is available on the terrain. Note that when the nutrient is another living species (plant, animal), only the **surplus** produced during one year is considered as an available resource. In contrast, we are using infinite quantities of resources for some other nutrients, such as water accessible from a riverbank. To provide an easily understandable visual representation, we depict spatialized resources on terrain maps as a set of ellipses whose center, size, and color correspond to the barycenter, area, and type of resource of the represented area, respectively (see Figure 5.7, left). We also keep track of the actual area through our hierarchical representation, and we use it to compute the accurate amount of resources during the competition (Section. 5.4).

Resource Access Graph (RAG): The RAG is a directed, hierarchical, and biology-driven graph encoding the localized resources on the terrain and the paths between them, for species in each of the food chain levels. The RAG's hierarchical structure allows species created at a given level of the food chain to serve as resources for species at the next food chain level. More precisely:

- At each level RAG^i, **RAG vertices** v are localized at the centroid of a spatialized

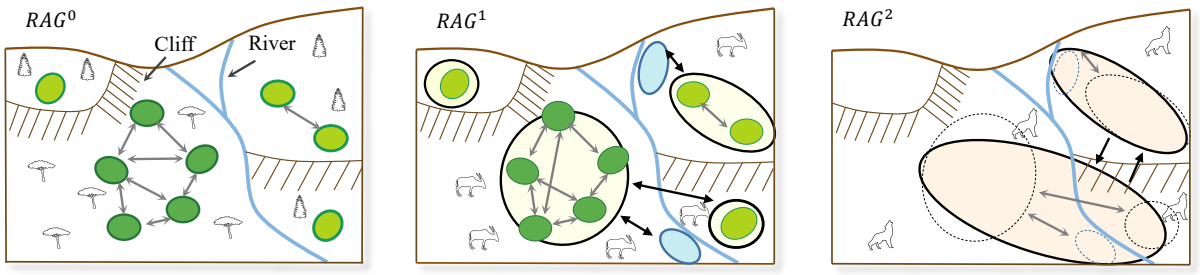


Fig. 5.8: Example of a three-levels Resource Access Graph. From left to right: Level 0 encodes mineral resources available for plant species. Level 1 encodes meadows where antelopes can graze and spots where they can drink. Antelopes can cross the river only at one ford, leading to two confinement regions. Level 2 encodes the RAG for a predator feeding on these two herds of antelopes. The predators can climb and go down cliffs, but with different traveling times up and down. Only parts of RAG^1 where antelopes herds were actually created are considered as resource nodes within RAG^2.

resource for species $s \in FCL^i$. As a vertex may represent several local resources available in the same area (see RAG initialization), we denote as $q(r, v)$, the quantity of the available resource r at v. This information is stored within the vertex data-structure.

- At each level RAG^i, **RAG edges** e are labeled by a species $s \in FCL^i$, and model the existence of an available traveling path between nodes (and thus between localized resources) for this specific species. The edge is valued, in each direction, by the average **Traveling time among resources**, proportional to the shortest path for an animal of the considered species between the centroids of the two resource regions on the terrain. This value roughly averages the travelling-time from/out of any point in the resource regions, making it usable even for large regions. Note that during instantiation, animal traveling speeds are based on the actual itineraries and zoological knowledge, not on these average values.

- **Hierarchical edges** are used to connect nodes at different hierarchy levels. More specifically, they connect a node of RAG^i to the nodes of RAG^{i-1} used to build it. They enable to find precise information about a specific resource, such as the exact areas and paths it is built on.

The RAG is initialized and then computed layer by layer, where RAG^i is built from RAG^{i-1} and from the solution of FCL^{i-1} *i.e.,* density maps for each species (see Figure 5.6). More precisely, the connected components of the graph at a given level of the food chain (modeling, for instance, the grazing area for a herd of antelopes) become the vertices of the next level of the RAG (surplus of antelopes as food for a carnivore), if the region ends up with a non-zero surplus density of antelopes (see Figure 5.8).

Computing RAG^i also makes use of accessibility maps (Section. 5.3.3) for species in FCL^i, modeling the ability of animals to travel from a resource to the next on the terrain.

We call **confinement region** an area of the terrain from which a given herd of animals cannot exit. Since a route may be infeasible due to dense vegetation, accessibility maps for animals are created after the computation of the vegetation layer, as detailed below.

5.3.2 Initialization with the vegetation layer

The resources needed by plants are the environmental resources R^0, which we compute from soil type and climate conditions in a regular grid as described in [CGG+17]. We convert each grid cell to a node of RAG^0, depicted as a circle centered in the cell, and assign the set of computed resources (light, moisture, nutrients) to the node. The edges of RAG^0 are created to connect each of these nodes with the four neighboring ones: Although individual plants cannot move, and will be computed based on local resources only, we use this feature so that contiguous areas where a given plant species grows (say, a grass meadow) be grouped within a single node at the next RAG level, where the connected components of RAG^0 with non-zero density will be used as resource nodes for herbivores.

5.3.3 Animal accessibility maps

Once RAG^0 has been used to compute competition between the plant species in FCL^0 (see Section. 5.4 for details), resulting in density maps for each of these species, we launch the computation of the accessibility maps for animals, based on their mobility capabilities.

While plants cannot move and only access local resources, animals travel, but may not be able to cross the whole terrain due to its topography (cliffs, rivers) or to areas with overly dense vegetation. To model the resulting confinement regions for a given species, we use a flood fill algorithm on the terrain after removing impracticable cells, including deep waters, dense forests, and steep slopes that exceed given, species-dependent thresholds.

The result of this step is a set of accessibility maps, segmented into confinement regions, in which the associated species is trapped (see Figure 5.7 (c), where we depict each confinement region with a uniform color). These regions will be used during the subsequent RAG computation steps to decide whether two resource nodes for a given species can be connected.

5.3.4 Computing the next level

Let us now consider the computation of RAG^{i+1} from RAG^i and FCL^i. The nodes of RAG^{i+1} represent all the localized resources available for species in FCL^{i+1}. Since we now deal with animals, we first initialize them with one node per region from where animals can drink (banks of a lake or river). We then add one node per available localized food source from FCL^i (either plants or other animals), as follows: Let us consider the

restriction of RAG^i to nodes where a given species s^i_j is present (as computed in the solution for FCL^i), and to the edges modeling this species's traveling abilities. Then, connected components in this sub-graph indicate regions where this specific s^i_j, now seen as a resource, is available. We create one node v in RAG^{i+1} to express this resource. The area associated with this node is directly given by the density map for s^i_j (the full connected component covered by a plant; or the full confinement region in the case of a herd of animals). We set the amount of available resource $q(r, v)$ to the local surplus produced during one year by s^i_j in this area (*e.g.,* the number of newborns from a herd of antelopes during one year, minus the number of dying animals), modeled as a percentage of the number of local specimens.

Once all nodes are generated, we create edges of RAG^{i+1}, labeled by each species s^{i+1}_j of animal in FCL^{i+1}, to model the ability for s^{i+1}_j to travel from one of the available spatialized resources to the next. This is done based on the accessibility map for the given species: if the two nodes (centers of ellipses in Figure 5.7) lie in the same confinement region of the map, an edge is created, and valued based on the traveling time for this species between the two connected nodes: we run a shortest path algorithm on the terrain grid to compute the path between vertices v_1 and v_2, while taking the mobility abilities of the species into account, in the form of the maximal slope the animal can climb comfortably (the other edges in the graph are removed). This results in curved paths taking the topography of the terrain into account. The value associated with the directed edge joining v_1 to v_2 with respect to species s^i_j is finally set to the associated travelling time, denoted by $e^i_j(v_1, v_2)$.

5.4 Competition algorithm

We introduce a procedural competition algorithm, to solve for species density maps at each successive level of the food chain (vegetation → herbivores → carnivores). Instead of modeling competition between species from different food-chain-levels (as done in prey-predator dynamic systems), we consider the full ecosystem as being in a state of quasi-equilibrium. Therefore, only the yearly surplus produced by a given species can be used as resource for another species. If some of this surplus remains unused, it will be displayed in the form of denser areas for plants or youngsters for animals during interactive exploration (Section. 5.5), but an equivalent number of specimens is supposed to die during the year since numbers are supposed not to evolve. In the following, the notation s^i_j for species in FCL^i is simplified to s wherever the information about the level of food chain is not needed.

5.4.1 Survival constraints

Resource consumption: In our model, some of the resources (*e.g.,* temperature and water sources) are modeled as unlimited and will not be *consumed* by species. For all

other resources r, our input contains a representative average consumption value per species s and per year, denoted by $n(s, r)$.

Survival thresholds: Each species s is associated with a set of resources *fitness ranges*, denoted by $\mathcal{F}(s, r)$, provided as input to our algorithm. The minimum value of this interval $\mathcal{F}_{min}(s, r)$ is the minimum quantity of resource r needed for the species s to survive over a unit of time (a month in our implementation). This value might be zero for a non vital resource, or even a negative value in case r represents the temperature. The maximum value $\mathcal{F}_{max}(s, r)$ is the maximum value of resource r that species s can stand in its environment (for instance, the maximum temperature tolerated by a plant) and can also be infinity. By definition, a species s fits in an environment, if and only if the available resource values belong to the corresponding $\mathcal{F}(s, r)$.

5.4.2 Solving for a Food Chain Level

The input of the competition algorithm for FCL^i are RAG^i, which provides the quantities of available resources and their location on the terrain, the confinement regions for each species (a single cell for a plant), as well as the set of adequacy ranges $\mathcal{F}(s, r)$. Given this data, we first compute adequacy functions, fit(s), modeling the ability for a species s to survive in each of its confinement regions. We then run a simple, procedural algorithm to infer the set of density maps for all $s \in FCL^i$ that best matches the user's specified proportion between species (if any), or instead maximizes the use of resources. Let us detail these steps:

By definition, a species fits in the environment if and only if at any time of the year, the minimum and maximum values of available resources are compatible with its corresponding survival intervals. The more the interval of available resources overlaps with the survival interval, the higher the adequacy of the species with the environment. Therefore, we define the adequacy function for a species $s \in FCL^i$, and for each resource r as the ratio of the available interval of the resource which belongs to the survival range of the species during an entire year composed of $n = 12$ time periods. The minimum of these intervals over all the resources needed by s then gives a global adequacy value for s, since the sparsest resource determines the ability for a species to actually survive. This leads to:

$$\text{fit}(s) = \min_{r \in R^i} \sum_{t=1}^{n} \frac{\min\left(\mathcal{F}_{max}(s, r), r^t_{max}\right) - \max\left(\mathcal{F}_{min}(s, r), r^t_{min}\right)}{n\ \left(r^t_{max} - r^t_{min}\right)}, \tag{5.4}$$

where r^t_{min} and r^t_{max} represent the minimal and maximal values of resource r during the period t. This adequacy function is computed for each species and confinement region of RAG^i and represents the likeliness of the species to survive and grow based on the locally available resources.

The method shown in Algorithm. 3 computes a solution to the computation of FLC^i that best matches the user-specified proportion between species at this level. Starting

from zero, this iterative algorithm tentatively increases the number of individuals in a species by a number set from the typical number of individuals in a herd, and only retains the solution that provides the closest proportions of species to the user-defined values. It then positions the newly created group in the best adapted confinement region on the terrain. This process is repeated until no new specimen may be created due to lack of resources. If the user did not provide any desired proportion for species in FCL^i, Algorithm. 4 is used instead to automatically infer plausible proportions based on the available resources in each confinement region of the terrain (we use the intersection of the species-specific confinement regions, in case the species in FCL^i have different travelling abilities).

The two versions of the algorithm have similar logics, but Algorithm. 3 makes use of the target proportions provided by the user, as an additional constraint enabling to guide population increase towards the most relevant species. In contrast, Algorithm. 4, used when no target proportions is given, makes use of a region-based strategy, ie. iteratively increases density of species per region, choosing species with closest adequacy. While these two deterministic algorithms are greedy and do not insure that globally optimal solutions are achieved, they guarantee completion, consistency and constraint satisfaction, since they only decrease the amount of surplus resources available in the environment.

Input: RAG^i, species s and their proportions
Output: densities of FCL^i
repeat
  Select s that would bring FCL^i closest to target proportions;
  Compute fit(s) in each confinement region C for s;
  Select C with highest fitness;
  Increase density of s in C;
  Decrease accordingly the available resources in C;
until *not enough resources to increase density*;

Algorithm 3: Competition algorithm with target proportions

Input: RAG^i, species s
Output: densities of FCL^i
repeat
  foreach *confinement region C (or their intersection)* **do**
    Compute fit(s) for each s;
    Select s with highest fitness;
    Increase density of s;
    Decrease accordingly the available resources in C;
  end
until *not enough resources to increase density*;

Algorithm 4: Competition algorithm with no target proportions

The results are a density map for plants - with a value per cell based on locally available resources, or a number of specimens - and thus a uniform density - per confinement region for animals. The remaining resources at the current level, in each confinement region (*i.e.,* the unused surplus of species of FCL^i) are exported as maps and later used to display extra specimens for FCL^i in the ecosystem-aware landscape.

5.5 Ecosystem-aware landscapes

5.5.1 Generating a map of trails

Our goal is to generate a set of eroded trails on the terrain consistent with their probability of being used by animals. We first evaluate the time each animal spends at the different RAG nodes in its confinement region. Then, we assign probabilities of fauna presence to RAG edges, and finally we refine these edges into trails.

Given an animal in the confinement region C, its probability of presence at a spatialized resource node v in the associated part of the RAG depends on the dependency its species has for each resource r, as well as on the available quantity $q(r, v)$ of this resource at v.

For simplicity, we assume that the animal consumes its vital resources $r \in R(s)$ uniformly over the relevant accessible RAG nodes; and the total time it spends in a specific node is proportional to its reference need $n(s, r)$ with respect to r, compared to its total need for all resources denoted as $n\left(s, R(s)\right)$. These assumptions lead to a model to compute the probability of presence of an animal of species s on RAG node v:

$$P_s(v) = \sum_{r \in R(s)} \left(\frac{q(r, v)}{q(r, C)} \cdot \frac{n(s, r)}{n(s, R(s))} \right), \tag{5.5}$$

where $q(r, C)$ is the quantity of resource r available in the region C.

Let us now consider the probability of the animal to travel from RAG node v to RAG node v' in C. We make use of the average travelling time information from v to v' for species s, denoted by $e_s(v, v')$ (see Section. 5.3.4), for providing a simple, approximate value for this probability. The higher the travelling time, the lower the conditional travelling probability:

$$P_s(v' \mid v) = \frac{e_s^{-1}(v, v') \cdot P_s(v')}{\sum_{v"} \left(e_s^{-1}(v, v") \cdot P_s(v") \right)}. \tag{5.6}$$

From these probability functions, we can now model the impact of each animal species s on local erosion (proportional to its number of specimen $\|s\|$ and their average mass

m_s) via the following weights computed for each *RAG* edge (considering all the trips in both directions).

$$w_{vv'} = \sum_s m_s \cdot ||s|| \Big(P_s(v) \cdot P_s(v' \mid v) + P_s(v') \cdot P_s(v \mid v') \Big). \tag{5.7}$$

We use these weights to estimate the tracks that have been developed by animals in the modeled ecosystem, assuming that the steady-state lasted for enough years to result in animal-driven local erosion, as follows: First, we weight each edge of the RAG with the expected total mass of animals that travel through the associated terrain segment. We then initiate the eroded trail as the available terrain segment with the highest weight and iteratively extend it by adding the highest weight neighboring edge at each end. Second, once all trails have been weighted and accumulated, we compute the final trail map by extracting the longest contiguous routes from portions of trails, starting from the most used ones, and smoothing them out with splines (Figure 5.9, left). We finally subtract the trails from the vegetation density maps for consistency, and render them as a rocky texture on the terrain (see Figure 5.9, right).

The main trails demonstrate the need of animals to pass through the same spots to access some of the resources (either because the resource is abundant, or because the environment forces passage through a specific location). From these main trails, animals will scatter to secondary resources. This will lead to recursive tree structures for the trails, which highlight the frequency of usage.

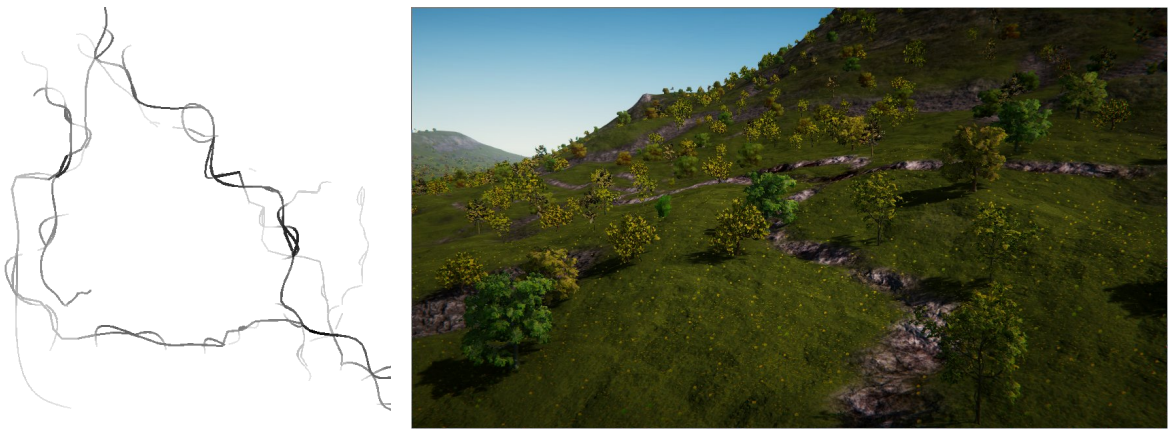


Fig. 5.9: Map of main animal trails for a specific region (left), where darker segments represent higher trail usage and consequently higher erosion. A rendered view of the trails on the terrain is also shown (right).

5.5.2 Daily itineraries and 3D instantiation

For computing daily itineraries of animals between resources, needed for 3D instantiation, we resort to the previously computed average travelling time $e_s(v, v')$ from node v to v'.

We build a semi-random daily planning for each herd of species s, (where the "herd" can be reduced to a single animal, depending on the species), by listing successive RAG nodes that can be visited in a single day, along with the associated duration of stay, as follows.

In real life, the typical consumption time of a resource r in hours per day may follow specific laws and our model would benefit from this specialized zoological knowledge. In our current implementation, we instead compute the average time $t(s, r)$ spent by an animal of species s at each node of resource r according to the needs of the species, and on the available quantity of resources at this node.

Since accessibility and adequate presence of resources to survive have already been enforced during the previous steps, our planning algorithm simply focuses on preventing too many animals to end up on the same resource at the same time, which is handled by reducing the capacity of resource nodes when occupied by a herd at a specific moment. We also facilitate the satisfaction of resource related constraints (such as accessing water once a day) by treating the mandatory resources that an animal must access during the day in priority. Algorithm. 5 summarizes the process for computing a daily planning, $Planning(h) = \{(v_1, t_1, t'_1)), \ldots, (v_n, t_n, t'_n))\}$, for each animal herd h of species s. $Planning(h)$ is a sorted list of the visited RAG nodes and corresponding time intervals.

Input: list of herds of species $s \in FCL^i$, RAG^i, $\forall r, t(s, r)$
foreach *herd h of species s* **do**
  $Planning(h) \leftarrow \emptyset$; $counter \leftarrow 0$; $j \leftarrow 1$;
  repeat
    if $\exists r$ *mandatory and not covered yet* **then**
      sample an available $v \in RAG^i$ containing r from $P_s(v)$;
    else
      sample any available $v \in RAG^i$ from $P_s(v)$;
    end
    $v_j \leftarrow v$; $t_j \leftarrow counter$; $t'_j \leftarrow counter + t_s(v)$;
    $Planning(h) \leftarrow (v_j, t_j, t'_j)$;
    decrease available space at v_j;
    $counter \leftarrow counter + t_s(v) + e_s(v_j, v_{j-1})$;
    $j \leftarrow j + 1$;
  until *$counter >$ length of an active day of species s*;
end
Output: daily planning of all herds of species s.

Algorithm 5: Daily planning algorithm for herds of species s.

The computed planning spots are then used as way-points along a daily itinerary for the herd. Herds are set to follow smooth paths on the terrain from one resource to the next. We take the terrain slope into account and use the A* path finding algorithm (over a quadtree) to compute their precise trajectory. The latter are then smoothed using spline curves. The effective speed of each herd along their trajectory is set from the zoological

characteristics of the species and the local slope.

The result can be visualized as 2D herd motion on a map that provides a quick lookup for the user to access areas of interest.

During the on-the-fly exploration, we instantiate individual animals depending on the camera position and viewing angle. Animals within herds are simulated using a boid model [Rey87] following the global path computed for the herd. This will prevent collisions and generate some relative motion within the herd. Note that this simple implementation, sufficient for 3D illustration in the context of this chapter, does not take species-specific herd shapes information (*e.g.,* the dynamic instantiation method presented in Chapter 4) into account and therefore leaves space for improvement.

5.6 Results and discussion

5.6.1 Interactive editing and exploration

Our system was implemented in Python and the interactive exploration part is done in Unity. The Unity engine allows interactivity at the cost of photorealism. We imported specific models of plants and animals to match the species that were required for the case studies. Generating the complete ecosystem with the eroded paths from the input takes on average 15 minutes, with around 5 minutes spent on vegetation and 5 - 10 minutes on the pre-computation of animal paths between resources. The density maps for animals are regenerated in a matter of seconds. Therefore, our method allows interactive editing through over-painting over any of the computed maps, with a re-generation time ranging between a few seconds and a few minutes, depending on the stage at which the user is making changes. Once maps, textures and daily itineraries are uploaded on Unity, the exploration is done at interactive rates.

5.6.2 Results

In order to validate the use of our system as an authoring and exploration tool, and to allow the visual validation of plausibility by experts, we focused on the modeling of two past ecosystems studied by a team of paleontologists at two different climate periods, in the valley of Tautavel. Period 1 corresponds to a cool and temperate climate around year -500,000. This climate is quite humid and allows the development of forests. In contrast, the second period refers to an earlier glacial period around year -550,000: a cold and dry climate harsher to vegetation. Please refer to Appendix A for tables giving the set of species for each of these ecosystems and the different parameters used in our experiments.

Figure 5.10 shows plant growth in the valley, and highlights their interaction with the river and cliffs (North West and South East) in the two different Periods. Figure 5.11

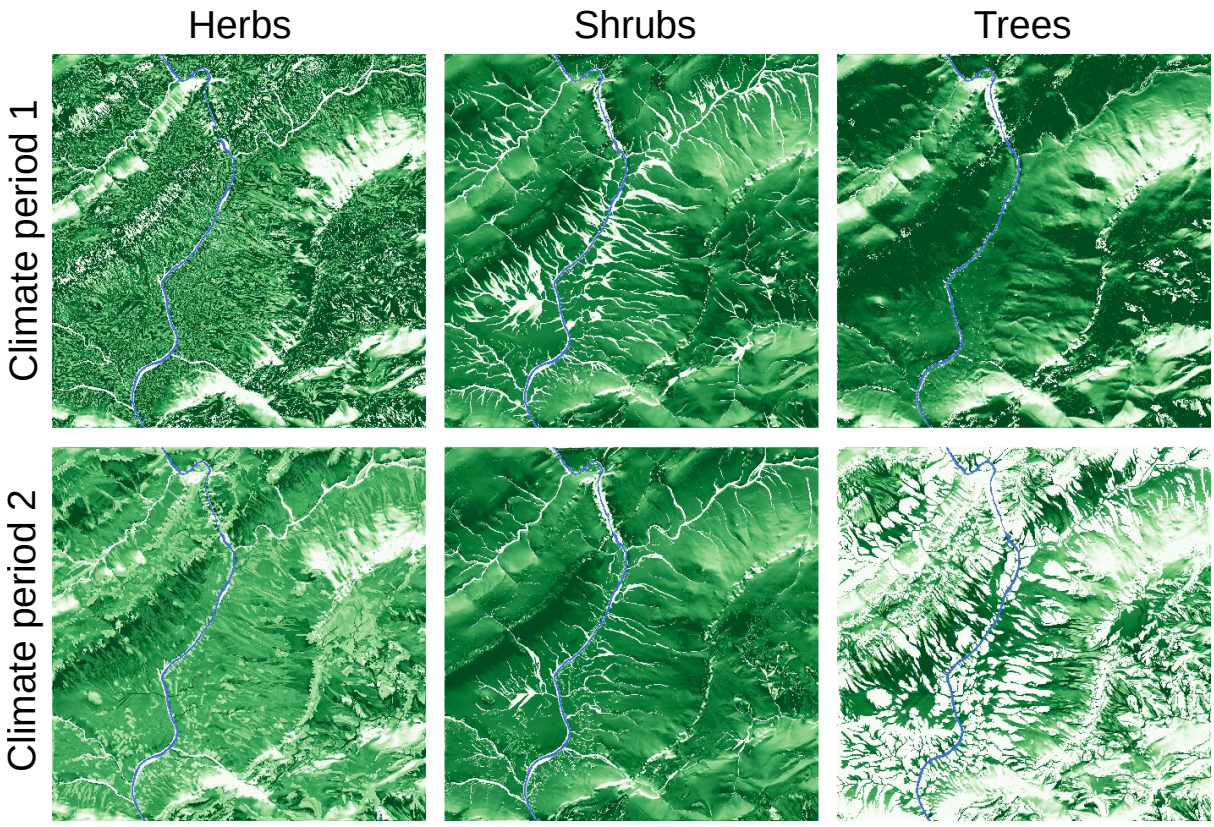


Fig. 5.10: Density maps computed for a variety of plant species grouped into herbs, shrubs, and trees. Darker colors mean higher densities. A river runs north to south in the valley (best seen in the top, right image), and is bordered on both sides by cliffs where vegetation has restricted access to water and more trouble growing. A forest developed north of the valley in Period 1, reducing the development of shrubs. In contrast we observe a reduced herbs density and much less trees in the colder climate (Period 2), except on the slope facing south.

shows how animals impact vegetation growth. More precisely, herbivores regularly consume a significant portion of the grass surplus in regions close to the river and away from steep cliffs. Note that herbs that have enough supplies of water to grow on cliffs are sheltered from herbivores, and manage to keep an overall higher surplus. This figure also shows trails where vegetation is fully removed, on both sides of the river (appearing in white in Figure 5.11, right). Finally, Figure 5.12 shows densities of herbivores for the two climate periods, while Figure 5.13 shows the exploration of the valley during Period 1.

5.6.3 Validation with expert users

Paleontologists work in multidisciplinary teams including experts in geology, plants biology and zoology. We worked with nine of these experts. Due to lack of effective computational tools for modeling, analyzing and visualizing past ecosystem, they are used to

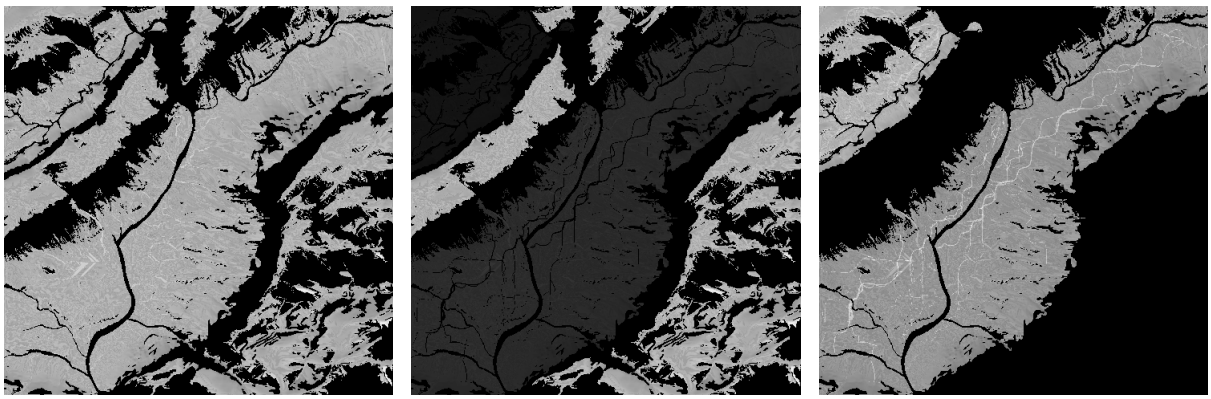


Fig. 5.11: Map of grass surplus without (left) and with (middle) the impact of animals (where dark means no surplus). The third image shows the difference between the two images, to ease comparison.

rely on visual comparison with current ecosystems in regions of the world with similar climates to validate their models. The latter express their assumptions about proportions and behaviors of species in past environments, based on multiple sources of findings on the terrain. To enable this visual comparison, they typically use maps, but have also tailored the development of a 3D virtual world, using standard software, enabling them to visually explore the past ecosystems they wished to validate. This previous experience with 3D gave them some grounds for comparison.

The nine experts in paleontology who used our system commented on the great benefits of enabling the interactive exploration of a 3D, animated virtual world, automatically built from their data, and which could be interactively edited. Their more detailed opinions on the project are summarized in the upper row of Figure 5.14. Users rated, in average, the completeness of the system as 3.9 out of 5, with the explicit mention that some species were missing (since we demonstrated our model on large animals only). They rated the user control as 4.1 out of 5, as they are already used to paint over density maps to manually create regions over the terrain. Finally, they rated the realism of the output of our system as 4 out of 5, mentioning that a good way to improve on this part would be to develop interfaces enabling them to add more information into the system and refine the input, as well as to account for extra species.

Additionally, we performed a second user study with ten end users experienced in working with virtual worlds, in the fields of computer graphics, animation, cinematography, video games, and museography. The participants were asked the same questions as the experts about completeness, user control, and visual realism, to allow comparison between the two studies. The results of this second user study are at the bottom of Figure 5.14 and they show the same general trend of an increased rating of our project as the user expertise increases (see Appendix B for details).

To provide more grounds for validation, the expert users also provided their own manually-created map of the most important passageways that were likely to be have

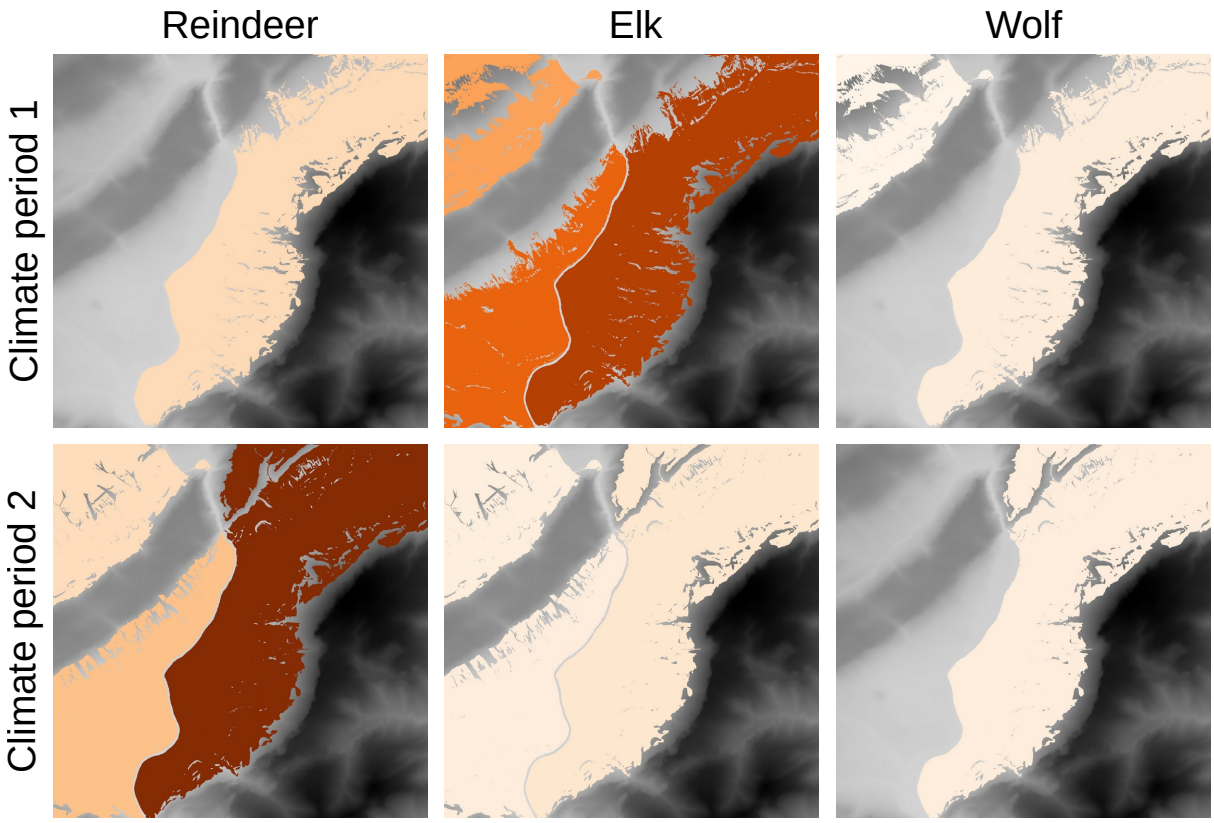


Fig. 5.12: Density maps (in white to brown as density increases) for three different animal species over the two studied Periods, based on different input proportions from paleontology knowledge. Note that the difference in proportions of reindeers (from 2% to 70% of the fauna) allowed them to colonize a much larger area in Period 2. The wolves on the other hand only settled on the side of the river where they could find their preferred preys in Period 2, given that their population was restricted to only around 1% of the fauna.

been used by species of large herbivores in the ecosystems they studied. We provide a comparison with our results on Figure 5.15.

Lastly, our vegetation density maps (Figure 5.10) were also validated by the experts through comparison with their own manually created maps. Each map was manually designed before the start of our project, from gathered data and expert knowledge, by three experts working together during one afternoon. For the sake of comparison, we recreated them with a similar rendering style based on the raw data generated by our method. We first grouped the species together by biome (*e.g.,* Mediterranean or Mountainous), and displayed for each biome the locations with the highest densities only (within the top five percentile) to get a clear overview (see Figure 5.16 left). We then copied the color code used by the experts on their map to ease visual comparison, outlining the dominant species (Figure 5.16 middle). We then discussed the differences between these maps with the experts. First, more river-related species appear in our version, located on smaller

Fig. 5.13: A herd of reindeer grazing in the valley.

branches of the river. These were omitted in the experts version, since they focused on the main branch. A second difference relates to the location of the Mediterranean biome (orange and striped orange/green) on the north-west plateau in their version but on the south side of this plateau in ours. This was attributed to an oversight on their end, ie. the fact that south slopes offer higher sun exposure and access to water sources, making it more sustainable for these species.

The validation of our results therefore emphasized not only the consistency of our method, but also its potential to save time and efforts. In the case of vegetation, our system provides more detailed density maps than the manual approach, providing information about each species's presence over the whole terrain, instead of only rough areas where given species are dominant. Our maps can also be generated and edited in a matter of minutes, which allows a far better assessment of hypotheses than manually created maps, requiring several hours of expert time.

5.6.4 Limitations

Although users emphasized that once configured with climate data, our system enables an easy exploration of flora and fauna hypotheses, our approach still includes a number of limitations.

First, the way we model the interactions between species is limited. Although we partly considered competition, cooperation between species [CBG08] was mostly neglected. In particular, while the fact that species bring resources to other species can

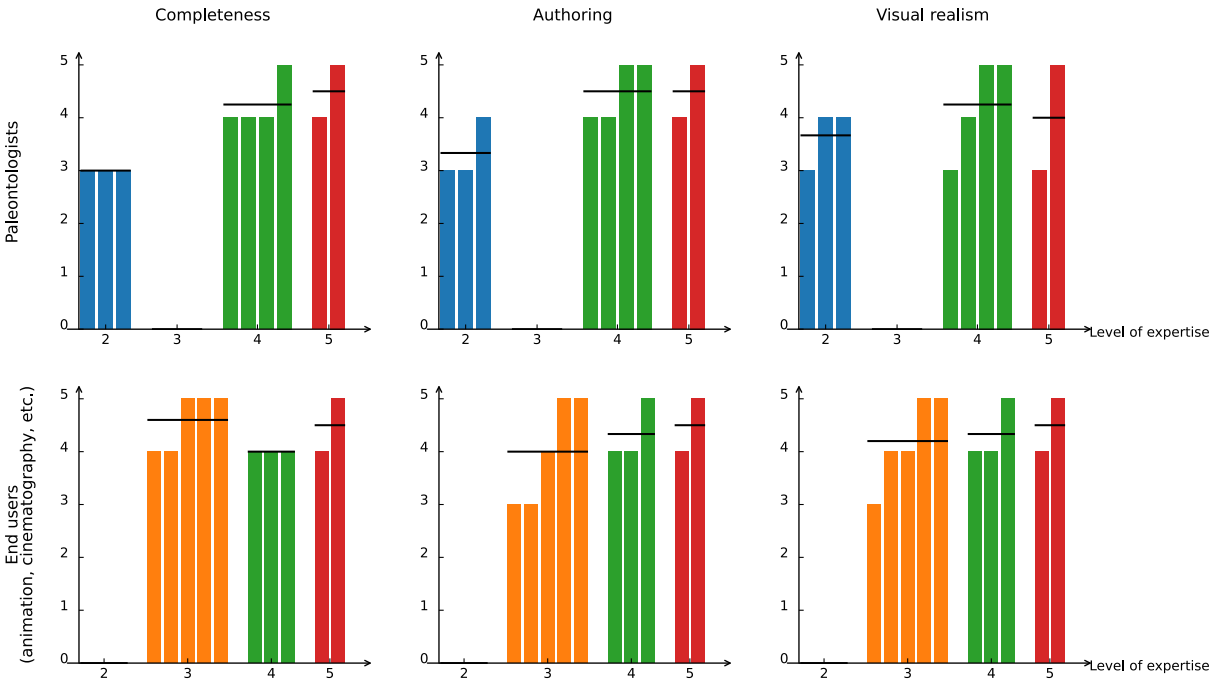


Fig. 5.14: Results of the user studies ran with nine experts in paleontology (top) and ten end users (bottom) who ranged their own expertise as respectively (2, 4, 5) and (3, 4, 5) out of 5.

be considered as cooperation, we neglected cooperation down the food chain, since we progressively build our solution from one food-chain-level to the next without considering any possible retro-action. Therefore, the fact that fauna can help fertilize soils for vegetation, or that some animals may serve as pollinators, is not modeled.

In addition, our model prevents us from modeling dynamic competition mechanisms between species. Indeed, we approximate the ecosystem as a series of procedural solutions, built level by level while trying to best match user-specified proportions and to maximize the use of available resources. In addition to bringing only a coarse approximation to a steady-state ecosystem, such a solution is by construction unable to model the dynamics of competition. Moreover, the realism of results heavily depends on the user-specified proportions between species. One of the main advantages of our model is its ability to consistently embed this coarse ecosystem onto a landscape which is automatically subdivided into relevant areas, to generate traveling itineraries for species between resources, and to model their impact on vegetation and erosion. We also allow authoring, where the user-defined proportions of species can be imaginary, extracted from real data, or could also be pre-computed at a different scale through simulation. Therefore, despite its limitations, we believe that our method can still be useful both for artists and for scientists wishing to visualize their results.

Lastly, although the set of species considered in our study is limited, larger and more complex food chains containing birds, smaller animal species and marine resources such

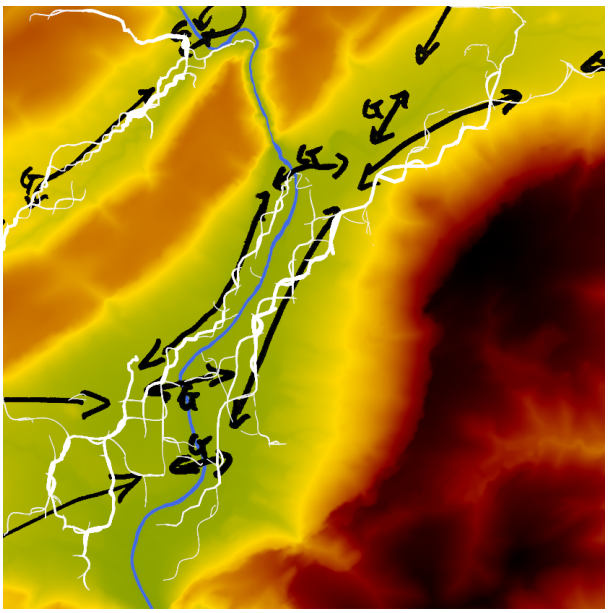


Fig. 5.15: Comparison of paths manually created by an expert (black), and the herbivores trails generated by our system (white). To ease comparison, we manually added exit locations to the top right and bottom left of our terrain, as well as three fords over the river, matching those indicated by the experts.

as fishes could be integrated in our model. However modeling marine food chains and ecosystems with similar methodology is left as an open and challenging research direction.

5.7 Conclusion and future work

We presented in this chapter a method for the consistent creation of a full ecosystem with flora and fauna over a terrain, enabling to model for the first time the impact of wild life on vegetation and erosion. Our method relies on a layered graph of available resources, positioned on the terrain, to progressively generate specimens of a steady-state ecosystem, built to match user-desired proportions. Starting with the terrain and other environment resources, we iteratively instantiate each level of the food chain, thanks to a procedural solution to the competition algorithm between species of the same level. The impact of instantiated specimens is then back propagated to the resources they use, and to the terrain in the form of eroded trails. Our solution provides both interactive authoring tools and an instantiation method enabling the user to interactively explore the resulting, animated landscape.

We expect both artists designing imaginary virtual worlds for films or games and scientists willing to visualize and explore past, present or future ecosystems, to share interest in our method. In this context, enabling users to specify proportions of species not only at a given steady-state, but also at specific points along a time-line, would

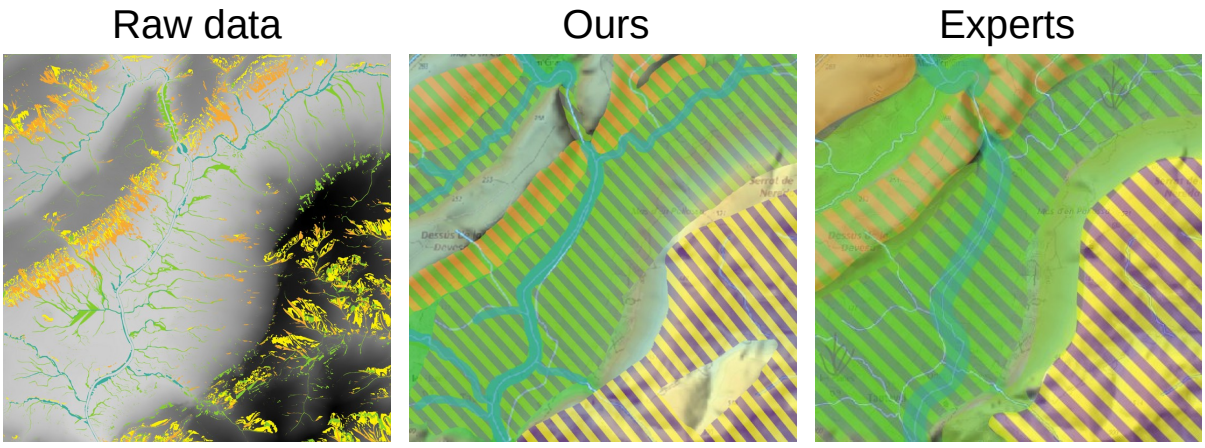


Fig. 5.16: Comparison of dominant vegetation maps generated by our system (left and middle) with one manually created by experts (right) for Period 1. Colors and stripes in blue, green, orange and yellow respectively represent Riparian, Ubiquist, Mediterranean and Mountainous biomes.

be a great extension for future work. The challenge is then to allow the exploration of intermediate landscapes, using a consistent ecosystem able to transition between two successive, predefined states. In addition to being useful for artistic story-telling, this could be a great tool in scientific applications such as paleontology, as a way to control and visualize time-evolving landscapes.

Another promising future research direction is the extension of our linear food chain model to a generalized food chain graph that would allow for a more accurate representation of the relationships between species.

Lastly, coupling our system with a standard, prey-predator simulation would bring several benefits: It would allow to predict consistent evolving ecosystems, with dynamic proportions of species, while enabling to stop at any point of in time and launch 3D exploration of the associated animated landscape. This could be achieved by considering long-term evolution as a succession of quasi-equilibrium states, and using predicted proportions from the dynamic simulation at a given time as an input to our terrain-embedded solution.

Chapter 6

Conclusion

In this thesis, we introduced novel methods for the authoring of ecosystems including both plants and animals, operating at different stages along the ecosystem creation process. We will summarize here our main contributions, and suggest promising directions of research in this domain.

6.1 Contributions

Our contributions are related to the instantiation of elements, applied to static and dynamic problems, as well as methods for the authoring of complete ecosystems. Control over the result remains an important priority throughout our work.

Static instantiation. We extended state of the art methods for the analysis and synthesis of distributions of elements, and reformulated them for the placement of objects, such as plant models, in virtual worlds. Given the variety of shapes and interactions that occur in nature, it is evident that the size of objects has a significant impact on the distribution of their neighbors. For example, small plants may need to avoid shade from taller trees, whereas parasitic mushrooms will grow very close or directly on top of other species. To make such interactions possible, we extended the PCF framework to distributions where different types of elements coexist, as well as to distributions of disks instead of points. To this end, we proposed a new 1D metric to evaluate the distance between two discs, which is able to encode information about the state of overlap between these disks based on a few critical configurations that we determined. To improve the robustness and usefulness of our method, we also enhanced PCFs to encode variance information from the input exemplars, and improved the analysis of distributions to support the extraction of data from distributions with arbitrary boundaries.

Dynamic instantiation. In the case of dynamic elements such as herds of animals, we developed a multi-scale approach where control over the appearance of the distribution is dissociated from control over the overall movement of the herd. We presented a new analysis and synthesis method based on the concept of PCFs but completely reformulated for terrestrial animals. To this end, we represented the animals as ovals instead of circles to better reflect their anisotropic nature while retaining information about their size. We also paid a particular attention to the distance between animals, their orientations and

local density, as well as the overall shape of the herd by introducing density and orientation fields that we reused during synthesis. We used photographs as input to allow an easy authoring process, and interpolated between exemplars to generate a smooth real-time animation. We also briefly presented a work in progress and directions towards herd animation from short video clips, with a mechanism to learn and synthesize relative speeds.

Complete ecosystems. Finally, we developed a general pipeline able to handle both plants and animals in a unified formulation. Our framework lies on the concept of an ecosystem in a quasi-equilibrium state, where animals and vegetation do not consume more resources than what is available, and as a result are able to keep a stable population at large time scales. In this context, our method tackles the problem of positioning plant species on the input terrain and computing daily cycles of animals, as well as modeling interaction with the terrain as eroding trails. To compute the available resources while keeping track of their accessibility to the different animal species, we introduced the Resource Access Graph, an abstract formulation of all resources embedded on the terrain. This graph is used by all living entities, and is updated gradually as requirements along the food chain are computed. This unified formulation allowed us to develop a common, efficient competition algorithm that we used to compute the requirements and presence of species at each level of the food chain.

6.2 Future work

The fields related to ecosystem authoring still offer plenty of promising research directions and challenges. We detail in this final section the main avenues for future research from the findings of this thesis.

Instantiation. We used in this thesis the PCF framework to handle analysis and synthesis of distributions, and extended them to handle simple shapes such as disks and ellipses. We used a custom metric in the case of disks, and additional information maps in the case of ellipses to guide computations towards their intended purpose. While this produced good results for our applications, we believe that PCFs and similar methods can still be developed much further. First, an extension to arbitrary shapes would help port the method to the general case, enabling its use in many more contexts. This generalization process would also need to be accompanied by the support of density variations and anisotropy in a more unified fashion to truly establish the method as the primary tool for distributions. We also believe that important performance improvements of the synthesis routine are possible, which could make PCFs a solution for interactive applications.

Group animation. The high level authoring method presented in this work was designed to provide the user with a way to intuitively control the aspect and formation of herds

of animals. Once this control was established, we relied on microscopic simulation to smoothly interpolate between configurations and handle collision avoidance and other interactions with the environment. This approach only covers basic interactions, and could be vastly improved by focusing on the response of animals and humans to environmental constraints. Combining the different stages of crowd animation methods can be one path to solve this problem. For example, merging local interactions of microscopic models, global control over the herd distribution, and global path planning algorithms could patch the holes in current crowd synthesis methods.

Ecosystem modeling. To the best of our knowledge, this thesis was the first to introduce full ecosystem modeling to Computer Graphics. However, due to the young nature of this question, it remains an open problem and our work should be regarded as only one step towards a more complete method for authoring full ecosystems. Our method could be expanded in many interesting directions, of which the most pressing one is perhaps the concept of food chain. While it is a good approximation for a limited number of species, it shows limitations as the types of species increase. Indeed, many species of animals have interrelations, which encourages a more open representation such as a graph of dependencies. Another barrier that prevents the development of more general models is our quasi-equilibrium assumption. We view this assumption as an approximation used to relax constraints and allow the computation of a suitable solution. In a more comprehensive system, the quasi-equilibrium assumption should be removed to allow the creation of fully dynamic ecosystems, if control over the result can be supplied by other means.

Landscapes. We outlined the importance of animals when depicting landscapes throughout this work. However, even without considering a direct representation of animals, they are often completely omitted when landscapes are studied in Computer Graphics. We are convinced that both direct and indirect (*e.g.,* trails, sounds, consumed food, remains) representations of animals are crucial to reach the level of quality that will be expected of virtual worlds in a few years. Impacts at a larger scale also have mostly been ignored, but could drastically affect the aspect of landscapes. For example, wolves regulating the biodiversity, and beavers completely reshaping their landscape and altering the paths of rivers both significantly affect their environment.

Control. While we paid a particular attention to user control and usability in our work, we presented mostly indirect approaches for the authoring of ecosystems. Indeed, example-based methods were used for instantiation problems, and we tackled ecosystems by allowing the user to paint information on intermediate information maps. There is, to the best of our knowledge, no method devised for a direct control over complete ecosystems, allowing artists to interactively shape the life over a terrain while retaining biological consistency. We believe that such a method would need a new control scheme due to the presence of both static and animated elements, and as such would be a great target for future research.

Appendices

Appendix A

Ecosystem parameters and notations

This appendix provides a detailed list of species parameters that are used in our ecosystem analysis, as well as a list of notations used in Chapter 5.

	Illumination	**Temperature**	**Moisture**	**Texture**	**Geological viability**
Cupressaceae	$(7, 9, 2)$	$(2, 9, 0)$	$(2, 4, 2)$	$(2, 8, 0)$	$(1, 0, 1, 1, 1)$
Pinus	$(7, 9, 5)$	$(6, 9, 0)$	$(2, 5, 3)$	$(1, 7, 0)$	$(1, 0, 1, 1, 1)$
Pistacia	$(7, 9, 2)$	$(7, 9, 0)$	$(3, 4, 2)$	$(3, 4, 0)$	$(1, 0, 1, 1, 1)$
Alnus	$(7, 9, 5)$	$(2, 8, 0)$	$(4, 8, 5)$	$(1, 4, 0)$	$(0, 0.8, 0, 0, 1)$
Apiaceae	$(3, 9, .2)$	$(1, 9, 0)$	$(2, 6, .2)$	$(1, 7, 0)$	$(1, 0, 1, 1, 1)$
Cichorium	$(7, 9, .2)$	$(6, 9, 0)$	$(4, 6, .2)$	$(3, 4, 0)$	$(1, 0, 1, 1, 1)$
Poaceae	$(5, 9, .2)$	$(1, 9, 0)$	$(1, 8, .2)$	$(1, 9, 0)$	$(1, 0, 1, 1, 1)$
Asteraceae	$(4, 9, .2)$	$(3, 9, 0)$	$(1, 7, .2)$	$(1, 8, 0)$	$(1, 0, 1, 1, 1)$
Quercus	$(7, 9, 5)$	$(5, 8, 0)$	$(3, 6, 5)$	$(2, 5, 0)$	$(1, 0, 1, 1, 1)$
Artemisia	$(7, 9, .2)$	$(2, 5, 0)$	$(2, 8, .2)$	$(2, 6, 0)$	$(1, 0, 1, 1, 1)$
Betula	$(7, 9, 3)$	$(3, 5, 0)$	$(4, 7, 3)$	$(1, 6, 0)$	$(1, 0, 1, 1, 1)$
Carpinus	$(6, 7, 5)$	$(6, 7, 0)$	$(4, 5, 4)$	$(3, 4, 0)$	$(1, 0, 1, 1, 1)$
Corylus	$(5, 6, 4)$	$(5, 6, 0)$	$(4, 5, 4)$	$(3, 4, 0)$	$(1, 0, 1, 1, 1)$
Plantago	$(5, 9, .2)$	$(1, 9, 0)$	$(2, 7, .2)$	$(2, 5, 0)$	$(1, 0, 1, 1, 1)$
Pinus mugo	$(7, 9, 3)$	$(2, 4, 0)$	$(3, 7, 3)$	$(2, 7, 0)$	$(1, 0, 1, 1, 1)$
Quercus ilex	$(7, 9, 4)$	$(7, 9, 0)$	$(4, 7, 4)$	$(2, 6, 0)$	$(1, 0, 1, 1, 1)$
Rubiaceae	$(3, 9, .2)$	$(2, 9, 0)$	$(2, 6, .2)$	$(1, 9, 0)$	$(1, 0, 1, 1, 1)$
Fabaceae	$(6, 9, .2)$	$(2, 9, 0)$	$(2, 6, .2)$	$(2, 8, 0)$	$(1, 0, 1, 1, 1)$

Table A.1: Plant parameters. Triplets denote $(min, max, consumption)$ and geological viability represents specific viability of a species per soil type (limestone, water, marl, fallen rocks, alluvium)

	Group size	**Mass**	**Presence in**	
			Period 1	**Period 2**
Bison Priscus	[10; 35]	[700; 1000]	1.16%	0.88%
Cervus Elaphus	[30; 45]	[70; 250]	41.28%	10.62%
Dama Roberti	[5; 14]	[30; 80]	34.30%	1.77%
Equus Mosbachensis	[6; 20]	[227; 900]	1.74%	0.88%
Hermitragus Bonali	[2; 23]	[36; 90]	2.91%	0.88%
Ovis Ammon	[2, 100]	[130; 160]	5.81%	7.08%
Rangifer Tarandus	[50; 150]	[100; 300]	2.33%	69.03%
Ursus Arctos	[80; 600]	1		1.77%
Ursus Spelaeus	[200; 500]	1	1.16%	
Lynx Spelaeus	[11; 15]	1	1.16%	0.88%
Canis Mosbachensis	[23; 80]	[5; 9]	1.74%	0.88%
Vulpes Vulpes	[3; 14]	1	1.16%	0.88%
Total			94.75%	95.55%

Table A.2: Animal parameters showing $[min; max]$ for each category. The target presence is also shown for both studied ecosystem.

Notation	Meaning
FCL	Food Chain Level
K	Number of $FCL - 1$
S^i	Set of species in FCL^i
s^i_j	Species j in FCL^i
N_i	Number of species in FCL^i
R^i_j	Set of resources used by species s^i_j
R^0	Set of natural resources used by plants
RAG	Resource Access Graph
v, e	RAG vertices and edges
$q(r, v)$	Quantity of resource r available at v
$e^i_j(v_1, v_2)$	Traveling time of species s^i_j along edge $e(v_1, v_2)$
$n(s, r)$	Need or average consumption of resource r by species s
$\mathcal{F}(s, r)$	Fitness ranges $[\mathcal{F}_{\min}(s, r); \mathcal{F}_{\max}(s, r)]$ for species s and resource r
$fit(s)$	Fitness of species s
t	Period of study
$[r^t_{\min}; r^t_{\max}]$	Minimal and maximal values of r during the period t
C	Confinement region
$q(r, C)$	Quantity of resource r available in the region C
$P_s(v)$	Probability of presence of an animal of species s on the RAG node v
$P_s(v' \mid v)$	Probability of species s to go to node v' if it is currently in node v
$\|\|s\|\|$	Number of specimen of species s
m_s	Average mass of species s
$w_{vv'}$	Weight of the RAG edge between nodes v and v'
$t(s, r)$	Average time spent by species s on a node r
$Planning(h)$	List of visited nodes and their timing for a specific herd h

Table A.3: Mathematical notations used in Chapter 5 .

Appendix B

Ecosystem user study

This appendix presents the detailed results of the user studies analyzed in Chapter 5. In total, they have been performed on nineteen users, including nine expert paleontologists and ten end users interested in the project from various domains and backgrounds.

User	Gender	Age	Domain	Career	Expertise	Completeness	Control	Realism
1	Male	18-25	VG, A	Industry	5	4	4	4
2	Male	18-25	CG	Academic	4	4	5	4
3	Female	18-25	CG	Academic	3	5	4	4
4	Male	18-25	CG	Academic	3	4	5	4
5	Female	26-39	CG	Academic	4	4	4	5
6	Male	26-39	A, C	Industry	3	5	3	3
7	N/A	40-65	M	Industry	5	5	5	5
8	Female	18-25	A	Industry	3	4	3	5
9	Male	18-25	CG	Academic	3	5	5	5
10	Female	18-25	CG, A, C	Industry	4	4	4	4

A=Animation, CG=Computer Graphics, C=Cinematography, M=Museography, VG=Video Games

Table B.1: User study results with ten end users with both academic and industry backgrounds, and covering a wide variety of domains.

Expert	Level of expertise	Completeness	User control	Realism
1	5	5	5	5
2	4	4	5	5
3	4	5	4	5
4	2	3	4	4
5	5	4	4	3
6	2	3	3	4
7	2	3	3	3
8	4	4	4	4
9	4	4	5	3

Table B.2: User study results with nine expert paleontologists, with different levels of expertise.

Bibliography

[AD05] Monssef Alsweis and Oliver Deussen. Modeling and visualization of symmetric and asymmetric plant competition. *Eurographics workshop on Natural Phenomena*, page 7, 2005. 15

[AD06] Monssef Alsweis and Oliver Deussen. Wang-tiles for the simulation and visualization of plant competition. In *Advances in Computer Graphics*, volume 4035, pages 1–11. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. Series Title: Lecture Notes in Computer Science. 16

[ADWDB] Animal-Diversity-Web. https://animaldiversity.org/, DB. Accessed: 2020-01-22. 87

[Aur87] Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. 43

[BAS09] Bedrich Benes, Nathan Andrysco, and Ondrej St'ava. Interactive modeling of virtual ecosystems. page 8, 2009. 14

[BBJA02] O. Burchan Bayazit, Jyh-Ming Lien, and N.M. Amato. Roadmap-based flocking for complex environments. pages 104–113. IEEE Comput. Soc, 2002. 23

[BBT+06] Pascal Barla, Simon Breslav, Joëlle Thollot, François Sillion, and Lee Markosian. Stroke pattern analysis and synthesis. *Computer Graphics Forum*, 25(3):663–671, 2006. 12

[BE03] B. Benes and E.D. Espinosa. Modeling virtual ecosystems with the proactive guidance of agents. In *Proceedings 11th IEEE International Workshop on Program Comprehension*, pages 126–131, May 2003. ISSN: 1087-4844. 26, 27, 28

[BMJ+11] Bedrich Benes, Michel Abdul Massih, Peter Jarvis, Dadniel G. Aliaga, and Carlos A. Vanegas. Urban ecosystem design. In *I3D*, pages 167–174, 2011. 26

[BSK+15] Gwyneth Bradbury, Kartic Subr, Charlampos Koniaris, Kenny Mitchess, and Tim Wayrich. Guided ecological simulation for artistic editing of plant distributions in natural scenes. *Journal of Computer Graphics Techniques*, 4(4), 2015. 15

[BSK16] Adam Barnett, Hubert P. H. Shum, and Taku Komura. Coordinated crowd simulation with topological scene analysis. *Computer Graphics Forum*, 35(6):120–132, September 2016. 23

[BvdPPH11] Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, December 2011. 70

[BWWM10] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6):166:1–166:10, December 2010. 8

[CBG08] Franck Courchamp, Ludek Berec, and Joanna Gascoigne. *Allee effects in ecology and conservation*. Oxford University Press, 2008. 103

[CEG+18] Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani. Interactive generation of time-evolving, snow-covered landscapes with avalanches. *Computer Graphics Forum*, 37(2):497–509, May 2018. 82

[CGG+17] Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. Graph.*, 36(4):134, 2017. 15, 26, 28, 87, 92

[CGW+13] Jiating Chen, Xiaoyin Ge, Li-Yi Wei, Bin Wang, Yusu Wang, Huamin Wang, Yun Fei, Kang-Lai Qian, Jun-Hai Yong, and Wenping Wang. Bilateral blue noise sampling. *ACM Trans. Graph.*, 32(6):216:1–216:11, November 2013. 8

[Ch'11] Eugene Ch'ng. Realistic placement of plants for virtual environments. *IEEE Computer Graphics and Applications*, 31(4):66–77, July 2011. Conference Name: IEEE Computer Graphics and Applications. 15

[Ch'13] Eugene Ch'ng. Model resolution in complex systems simulation: Agent preferences, behavior, dynamics and n-tiered networks. *SIMULATION*, 89(5):635–659, May 2013. 26

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, January 1986. 8

[DAPB08] Funda Durupinar, Jan Allbeck, Nuria Pelechano, and Norman Badler. Creating crowd variation with the ocean personality model. page 6, 2008. 19

[DGA04] Brett Desbenoit, Eric Galin, and Samir Akkouche. Simulating and modeling lichen growth. *Computer Graphics Forum*, 23(3):341–350, September 2004. 14

[DHL+98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomir Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *Proc. of Sigg.*, SIGGRAPH '98, pages 275–286. ACM, 1998. 14, 15

[DMCN+17] T. B. Dutra, R. Marques, J.B. Cavalcante-Neto, C. A. Vidal, and J. Pettré. Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum*, 36(2):337–348, May 2017. 20

[DMLG02] J.-M. Dischler, K. Maritaud, B. Lévy, and D. Ghazanfarpour. Texture particles. *Computer Graphics Forum*, 21(3):401–410, 2002. 8

[DSZ17] Oliver Deussen, Marc Spicker, and Qian Zheng. Weighted linde-buzo-gray stippling. *ACM Trans. Graph.*, 36(6):233:1–233:12, November 2017. 8

[ENMGC19] Pierre Ecormier-Nocca, Pooran Memari, James Gain, and Marie-Paule Cani. Accurate synthesis of multi-class disk distributions. In *Computer Graphics Forum*, volume 38, pages 157–168. Wiley Online Library, 2019.

[ENPMC19] Pierre Ecormier-Nocca, Julien Pettré, Pooran Memari, and Marie-Paule Cani. Image-based authoring of herd animations. *Computer Animation and Virtual Worlds*, 30(3-4):e1903, 2019.

[EVC+15] Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. Worldbrush: interactive example-based synthesis of procedural virtual worlds. *ACM Transactions on Graphics*, 34(4):106:1–106:11, July 2015. 9, 10, 16, 17, 30, 34, 39, 50, 55

[FRDC06] Laurent Favreau, Lionel Reveret, Christine Depraz, and Marie-Paule Cani. Animal gaits from video. *Graphical Models*, 68(2), 2006. 18

[GD13] Qin Gu and Zhigang Deng. Generating freestyle group formations in agent-based crowd simulations. *IEEE Computer Graphics and Applications*, 33(1):20–31, January 2013. 24

[GGG+16] Eric Guérin, Eric Galin, François Grosbellet, Adrien Peytavie, and Jean-David Génevaux. Efficient modeling of entangled details for natural scenes. *Computer Graphics Forum*, 2016. 16, 26

[GLCC17] James Gain, Harry Long, Guillaume Cordonnier, and Marie-Paule Cani. Ecobrush: Interactive control of visually consistent large-scale ecosystems. *Eurographics*, 36(2), 2017. 9, 10, 15, 16, 17, 30, 43, 47, 50, 53

[GPEO12] T Geijtenbeek, N Pronost, A Egges, and M H Overmars. Interactive character animation using simulated physics. *Computer Graphics Forum*, page 23, 2012. 19

[GPG+16] Francois Grosbellet, Adrien Peytavie, Éric Guérin, Eric Galin, Stéphane Mérillou, and Bedrich Benes. Environmental objects for authoring procedural scenes. In *Computer Graphics Forum*, volume 35, pages 296–308. Wiley Online Library, 2016. 26

[GSP+07] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 3D collage: expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering - NPAR '07*, page 7, San Diego, California, 2007. ACM Press. 11

[GvdPvdS13] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6):1–11, November 2013. 19

[HBDP17] Torsten Hädrich, Bedrich Benes, Oliver Deussen, and Sören Pirk. Interactive modeling and authoring of climbing plants. *Computer Graphics Forum*, 36(2):49–61, May 2017. 14

[HKS17] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, 36(4):1–13, 2017. 19

[HLT+09] Thomas Hurtut, Pierre-Edouard Landes, Joëlle Thollot, Yann Gousseau, Remy Drouillhet, and Jean-François Coeurjolly. Appearance-guided synthesis of element arrangements by example. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*, pages 51–60. ACM, 2009. 9, 10, 12, 30

[HM95] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, May 1995. arXiv: cond-mat/9805244. 19

[HPZS02] Jim Hanan, Przemyslaw Prusinkiewicz, Myron Zalucki, and David Skirvin. Simulation of insect movement with respect to plant architecture and morphogenesis. *Computers and Electronics in Agriculture*, 35(2-3):255–269, August 2002. 27

[HSK14] Joseph Henry, Hubert P. H. Shum, and Taku Komura. Interactive formation control in complex environments. *IEEE Transactions on Visualization and Computer Graphics*, 20(2):211–222, February 2014. 25

[Hug03] Roger L. Hughes. The flow of human crowds. *Annual Review of Fluid Mechanics*, 35(1):169–182, January 2003. 21

[HWYZ20] Chen-Yuan Hsu, Li-Yi Wei, Lihua You, and Jian Jun Zhang. Autocomplete element fields. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Honolulu HI USA, April 2020. ACM. 11

[IMIM08] Takashi Ijiri, Radomír Mêch, Takeo Igarashi, and Gavin Miller. An example-based procedural system for element arrangement. *Computer Graphics Forum*, 27(2):429–436, 2008. 8, 12

[JCC+15] Kevin Jordao, Panayiotis Charalambous, Marc Christie, Julien Pettré, and Marie-Paule Cani. Crowd art: density and flow based crowd motion design. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 167–176. ACM, 2015. 22

[JCP+10] Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. Morphable crowds. page 1. ACM Press, 2010. 21

[JPCC14] Kevin Jordao, Julien Pettré, Marc Christie, and Marie-Paule Cani. Crowd sculpting: A space-time sculpting method for populating virtual environments. In *Computer Graphics Forum*, volume 33, pages 351–360. Wiley Online Library, 2014. 22

[KBG+13] Mubbasir Kapadia, Alejandro Beacco, Francisco Garcia, Vivek Reddy, Nuria Pelechano, and Norman I. Badler. Multi-domain real-time planning in dynamic environments. page 115. ACM Press, 2013. 23

[KHHL12] Manmyung Kim, Youngseok Hwang, Kyunglyul Hyun, and Jehee Lee. Tiling motion patches. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 117–126, Goslar Germany, Germany, 2012. Eurographics Association. 22

[KO04] A. Kamphuis and M.H. Overmars. Motion planning for coherent groups of entities. pages 3815–3822 Vol.4. IEEE, 2004. 23

[KO12] I. Karamouzas and M. Overmars. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406, March 2012. 24

[Koo10] S. A. L. M. Kooijman. *Dynamic Energy Budget Theory for Metabolic Organisation*. Cambridge University Press, 2010. Google-Books-ID: R8OCVR9rOhUC. 27

[LCF05] Yu-Chi Lai, Stephen Chenney, and ShaoHua Fan. Group motion graphs. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 281–290, New York, NY, USA, 2005. ACM. 21

[LCHL07] Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–118. Eurographics Association, 2007. 21

[LCMP19] Axel López, François Chaumette, Eric Marchand, and Julien Pettré. Character navigation in dynamic environments based on optical flow. *Computer Graphics Forum*, 38(2):181–192, 2019. 20

[LD06] Ares Lagae and Philip Dutré. Poisson sphere distributions. In *Vision, Modeling, and Visualization*, pages 373–379, 2006. 10, 34

[LGH13] Pierre-Edouard Landes, Bruno Galerne, and Thomas Hurtut. A shape-aware model for discrete texture synthesis. In *Computer Graphics Forum*, volume 32, pages 67–76. Wiley Online Library, 2013. 10, 11, 12

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, March 1968. 13

[LNW+10] Hongwei Li, Diego Nehab, Li-Yi Wei, Pedro V. Sander, and Chi-Wing Fu. Fast capacity constrained voronoi tessellation. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 13:1–13:1, New York, NY, USA, 2010. ACM. 8

[Lor10] Michel Loreau. *From populations to ecosystems: Theoretical foundations for a new ecological synthesis (MPB-46)*, volume 50. Princeton University Press, 2010. 27

[LP02] Brendan Lane and Przemyslaw Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. *Graphics Interface*, page 13, 2002. 15

[LPLL19] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics*, 38(4):1–13, July 2019. 18, 19

[LSM+19] Thomas Leimkühler, Gurprit Singh, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. Deep point correlation design. *ACM Transactions on Graphics (TOG)*, 38(6):226, 2019. 10

[LWSF10] Hongwei Li, Li-Yi Wei, Pedro V. Sander, and Chi-Wing Fu. Anisotropic blue noise sampling. *ACM Trans. Graph.*, 29(6):167:1–167:12, December 2010. 8

[MAAI17] Domingo Martín, Germán Arroyo, Rodríguez Alejandro, and Tobias Isenberg. A survey of digital stippling. *Comput. Graph.*, 67(C):24–44, October 2017. 8

[MALI10] Domingo Martín, Germán Arroyo, M. Victoria Luzón, and Tobias Isenberg. Example-based stippling using a scale-dependent grayscale process. In *Proceedings of the 8th International Symposium on Non-Photorealistic*

Animation and Rendering, NPAR '10, pages 51–61, New York, NY, USA, 2010. ACM. 8

[MHS+19] Milosz Makowski, Torsten Hädrich, Jan Scheffczyk, Dominic L. Michels, Sören Pirk, and Wojtek Palubicki. Synthetic silviculture: Multi-scale modeling of plant ecosystems. *ACM Trans. Graph.*, 38(4):131:1–131:14, 2019. 15

[MLM+11] Gonçalo M. Marques, António Lorena, João Magalhães, Tânia Sousa, S. A. L. M. Kooijman, and Tiago Domingos. Life engine - creating artificial life for scientific and entertainment purposes. In George Kampis, István Karsai, and Eörs Szathmáry, editors, *Advances in Artificial Life. Darwin Meets von Neumann*, volume 5778, pages 278–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science. 27

[MNR+17] Darryl I MacKenzie, James D Nichols, J Andrew Royle, Kenneth H Pollock, Larissa Bailey, and James E Hines. *Occupancy estimation and modeling: inferring patterns and dynamics of species occurrence*. Elsevier, 2017. 27

[MP96] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410. ACM, 1996. 14

[MWH+06] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *Acm Transactions On Graphics (Tog)*, volume 25, pages 614–623. ACM, 2006. 13

[MWT11] Chongyang Ma, Li-Yi Wei, and Xin Tong. Discrete element textures. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, pages 62:1–62:10, New York, NY, USA, 2011. ACM. 10, 11, 12, 51, 53

[NENMC20] Baptiste Nicolet, Pierre Ecormier-Nocca, Pooran Memari, and Marie-Paule Cani. Pair correlation functions with free-form boundaries for distribution inpainting and decomposition. *Eurographics 2020 short paper proceedings*, page 4, 2020.

[ÖG12] Cengiz Öztireli and Markus Gross. Analysis and synthesis of point distributions based on pair correlation. *ACM Transactions on Graphics (TOG)*, 31(6):170, 2012. 9, 30, 32, 34, 35, 38, 39, 51, 53, 78

[OPOD10] Jan Ondřej, Julien Pettré, Anne-Hélène Olivier, and Stéphane Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics*, 29(4):1, July 2010. 20

[PAB07] Nuria Pelechano, Jan Allbeck, and Norman Badler. Controlling individual agents in high-density crowd simulation. *ACM SIGGRAPH Symposium on Computer Animation*, page 12, 2007. 19

[PGMZ12] Tao Pei, Jianhuan Gao, Ting Ma, and Chenghu Zhou. Multi-scale decomposition of point process data. *GeoInformatica*, 16(4):625–652, 2012. 39

[PGT08] Julien Pettre, Helena Grillon, and Daniel Thalmann. Crowds of Moving Objects: Navigation Planning and Simulation. *ACM SIGGRAPH Computer Graphics*, page 6, 2008. 23

[PH95] Przemyslaw Prusinkiewicz and Mark Hammel. The artificial life of plants. *Artificial life for graphics, animation, and virtual reality*, page 39, 1995. 13, 27

[PLH+90] Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S. Hanan, F. David Fracchia, and Deborah Fowler. The algorithmic beauty of plants. *Color Research & Application*, 18(2), 1990. 13

[PM01] Yoav Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308. ACM, 2001. 13

[PMKL01] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, pages 289–300. ACM Press, 2001. 14

[PPD07] Sébastien Paris, Julien Pettré, and Stéphane Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674, September 2007. 20

[QCHC17] Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. Wasserstein blue noise sampling. *ACM Trans. Graph.*, 36(5), October 2017. 10

[Rey87] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, August 1987. 19, 70, 99

[Rey99] Craig W Reynolds. Steering behaviors for autonomous characters. page 21, 1999. 19, 21, 23

[RFDC05] Lionel Reveret, Laurent Favreau, Christine Depraz, and Marie-Paule Cani. Morphable model of quadrupeds skeletons for animating 3d animals. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '05*, page 135. ACM Press, 2005. 18

[RÖG17] Riccardo Roveri, A Cengiz Öztireli, and Markus Gross. General point sampling with adaptive density and correlations. In *Computer Graphics Forum*, volume 36, pages 107–117. Wiley Online Library, 2017. 10, 51, 55

[RÖM+15] Riccardo Roveri, A. Cengiz Öztireli, Sebastian Martin, Barbara Solenthaler, and Markus Gross. Example based repetitive structure synthesis. *Computer Graphics Forum*, 34(5):39–52, 2015. 10, 12, 51, 57

[RPH+20] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M. Kitani, Dariu M. Gavrila, and Kai O. Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, July 2020. arXiv: 1905.06113. 23

[RRS13] Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. Interactive by-example design of artistic packing layouts. *ACM Transactions on Graphics*, 32(6):1–7, November 2013. 11

[SHL+17] Stephen R Shifley, Hong S He, Heike Lischke, Wen J Wang, Wenchi Jin, Eric J Gustafson, Jonathan R Thompson, Frank R Thompson, William D Dijak, and Jian Yang. The past and future of modeling forest dynamics: from growth and yield curves to forest landscape models. *Landscape ecology*, 32(7):1307–1325, 2017. 27

[Sim94] Karl Sims. Evolving virtual creatures. pages 15–22. ACM Press, 1994. 18

[SJ13] Ben Spencer and Mark W. Jones. Progressive photon relaxation. *ACM Transactions on Graphics*, 32(1):1–11, January 2013. 8

[SKSY08] Hubert P. H. Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. Interaction patches for multi-character animation. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 114:1–114:8, New York, NY, USA, 2008. ACM. 22

[SP19] Ci Song and Tao Pei. Decomposition of repulsive clusters in complex point processes with heterogeneous components. *ISPRS International Journal of Geo-Information*, 8(8):326, 2019. 39

[SPK+14] Ondrej Stava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomir Měch, Oliver Deussen, and Bedrich Benes. Inverse procedural modelling of trees: Inverse procedural modeling of trees. *Computer Graphics Forum*, 33(6):118–131, September 2014. 14

[SRH+09] L. Skrba, L. Reveret, F. Hétroy, M-P. Cani, and Carol O'Sullivan. Animating quadrupeds: Methods and applications. *Computer Graphics Forum*, 28(6), 2009. 18

[SYLK18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 76

[TBDB] Tela-Botanica. `https://www.tela-botanica.org/`, DB. Accessed: 2020-01-22. 87

[TCP06] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics*, 25(3):1160, July 2006. 21

[TLH19] Peihan Tu, Dani Lischinski, and Hui Huang. Point pattern synthesis via irregular convolution. *Computer Graphics Forum*, 38(5):109–122, August 2019. 10, 12, 55

[TYK+09] Shigeo Takahashi, Kenichi Yoshida, Taesoo Kwon, Kang Hoon Lee, Jehee Lee, and Sung Yong Shin. Spectral-based group formation control. *Computer Graphics Forum*, 28(2):639–648, April 2009. 25

[vdBGLM11] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In Bruno Siciliano, Oussama Khatib, Frans Groen, Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, *Robotics Research*, volume 70, pages 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 20

[vTCG12] Wouter G. van Toll, Atlas F. Cook, and Roland Geraerts. Real-time density-based crowd simulation: Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23(1):59–69, February 2012. 23

[vTP19] Wouter van Toll and Julien Pettre. Connecting global and local agent navigation via topology. In *Motion, Interaction and Games*, pages 1–10, Newcastle upon Tyne United Kingdom, October 2019. ACM. 23

[WBCG09] Jamie Wither, Frédéric Boudon, Marie-Paule Cani, and Christophe Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. In *Computer Graphics Forum*, volume 28, pages 541–550. Wiley Online Library, 2009. 14

[Wei10] Li-Yi Wei. Multi-class blue noise sampling. In *ACM Transactions on Graphics (TOG)*, volume 29, page 79. ACM, 2010. 10

[WZDJ14] Xinjie Wang, Linling Zhou, Zhigang Deng, and Xiaogang Jin. Flock morphing animation. *Computer Animation and Virtual Worlds*, 25(3-4):351–360, May 2014. 24

[XJY+08] Jiayi Xu, Xiaogang Jin, Yizhou Yu, Tian Shen, and Mingdong Zhou. Shape-constrained flock animation. *Computer Animation and Virtual Worlds*, 19(3-4):319–330, 2008. 24

[XK07] Jie Xu and Craig S. Kaplan. Calligraphic packing. In *Proceedings of Graphics Interface 2007 on - GI '07*, page 43, Montreal, Canada, 2007. ACM Press. ISSN: 07135424. 11

[XWL+08] Xuemiao Xu, Liang Wan, Xiaopei Liu, Tien-Tsin Wong, Liansheng Wang, and Chi-Sing Leung. Animating animal motion from still. *ACM Trans. Graph. (Siggraph Asia issue)*, 27(5), 2008. 18

[XWY+15] Mingliang Xu, Yunpeng Wu, Yangdong Ye, Illes Farkas, Hao Jiang, and Zhigang Deng. Collective crowd formation transform with mutual information-based runtime feedback. *Computer Graphics Forum*, 34(1):60–73, February 2015. 24

[YGJ+14] Dong-Ming Yan, Jianwei Guo, Xiaohong Jia, Xiaopeng Zhang, and Peter Wonka. Blue-noise remeshing with farthest point optimization. page 10, 2014. 8

[YLL+09] Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. *Computer Graphics Forum*, 28(5):1445–1454, July 2009. 8

[YLvdP07] KangKang Yin, Kevin Loken, and Michiel van de Panne. SIMBICON: Simple Biped Locomotion Control. *ACM Transactions on Graphics*, page 10, 2007. 18

[YW13] Dong-Ming Yan and Peter Wonka. Gap processing for adaptive maximal poisson-disk sampling. *arXiv:1211.3297 [cs]*, August 2013. arXiv: 1211.3297. 8

[ZHWW12] Yahan Zhou, Haibin Huang, Li-Yi Wei, and Rui Wang. Point sampling with general noise spectrum. *ACM Trans. Graph.*, 31(4):76:1–76:11, July 2012. 9, 12

[ZSKS18] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics*, 37(4):1–11, August 2018. 19

[ZZC+14] Liping Zheng, Jianming Zhao, Yajun Cheng, Haibo Chen, Xiaoping Liu, and Wenping Wang. Geometry-constrained crowd formation animation. *Computers & Graphics*, 38:268–276, February 2014. 24

ECOLE DOCTORALE

Titre : Création d'écosystèmes cohérents et animés : Apprentissage efficace à partir de données partielles

Mots clés : informatique graphique, modélisation, animation, simulation de foules, création, apprentissage

Résumé : Grâce aux récentes améliorations de puissance de calcul, les mondes virtuels sont maintenant plus vastes et complexes que jamais. Alors que ce type de contenu se généralise dans de nombreux médias, les utilisateurs attendent une expérience de plus en plus réaliste. En conséquence, de nombreuses recherches ont été effectuées sur la modélisation et la génération de terrains et de végétation, et parfois leurs interactions. Néanmoins, les animaux ont reçu moins d'attention et sont souvent étudiés en isolation. Avec le manque d'outils d'édition intuitive, ces problèmes font de la modélisation d'écosystèmes une tâche difficile pour les artistes, qui se retrouvent limités dans leur liberté créative, ou forcés d'ignorer le réalisme biologique.
Dans cette thèse, nous présentons des nouvelles méthodes adaptées au design et à l'édition d'écosystèmes virtuels, permettant la liberté créative sans pour autant renoncer à la plausibilité biologique. Notre approche a pour objectif de fournir des outils basés sur des données concrètes pour permettre une édition efficace des écosystèmes, tout en ne nécessitant qu'un nombre peu élevé de données. En incorporant les connaissances existantes sur la biologie à nos modèles, nous sommes capables de garantir à la fois la cohérence et la qualité des résultats.
Nous présentons des méthodes dédiées à l'instantiation précise et intuitive d'éléments statiques et animés. Comme les éléments statiques peuvent présenter des interactions complexes, nous proposons une méthode précise, basée sur l'exemple et adaptée aux recouvrements. Nous appliquons un concept similaire à l'édition de troupeaux, en utilisant des photographies ou vidéos comme entrée d'un algorithme de synthèse par l'exemple. À une échelle plus large, nous utilisons des données biologiques pour formuler un processus unifié gérant l'instantiation globale et les interactions de long terme entre la végétation et les animaux. En plus de garantir la cohérence biologique, ce modèle offre un contrôle sur le résultat via l'édition des informations à toute étape. Nos méthodes fournissent contrôle et réalisme durant le processus de création d'écosystèmes, en prenant en compte les éléments statiques et dynamiques, ainsi que leurs interactions à plusieurs échelles. Nous validons nos résultats à l'aide d'études utilisateur, ainsi que des comparaisons avec des données réelles ou d'experts.

Title : Authoring consistent, animated ecosystems: Efficient learning from partial data

Keywords : computer graphics, modelling, animation, crowd simulation, authoring, learning

Abstract : With recent increases in computing power, virtual worlds are now larger and more complex than ever. As such content becomes widespread in many different media, the expectation of realism has also dramatically increased for the end user. As a result, a large body of work has been accomplished on the modeling and generation of terrains and vegetation, sometimes also considering their interactions. However, animals have received far less attention and are often considered in isolation. Along with a lack of authoring tools, this makes the modeling of ecosystems difficult for artists, who are either limited in their creative freedom or forced to break biological realism.
In this thesis, we present new methods suited to the authoring of ecosystems, allowing creative freedom without discarding biological realism. We provide data-centered tools for an efficient authoring, while keeping a low data requirement. By taking advantage of existing biology knowledge, we are able to guarantee both the consistency and quality of the results.
We present dedicated methods for precise and intuitive instantiation of static and animated elements. Since static elements, such as vegetation, can exhibit complex interactions, we propose an accurate example-based method to synthesize complex and potentially overlapping arrangements. We apply a similar concept to the authoring of herds of animals, by using photographs or videos as input for example-based synthesis. At a larger scale, we use biological data to formulate a unified pipeline handling the global instantiation and long-term interactions of vegetation and animals. While this model enforces biological consistency, we also provide control by allowing manual editing of the data at any stage of the process.
Our methods provide both user control and realism over the entire ecosystem creation pipeline, covering static and dynamic elements, as well as interactions between themselves and their environment. We also cover different scales, from individual placement and movement of elements to management of the entire ecosystem. We validate our results with user studies and comparisons with both real and expert data.

Institut Polytechnique de Paris
91120 Palaiseau, France