# Dynamic sketches : Hierarchical modeling of complex and time-evolving scenes

Pauline Olivier

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAX019

Thèse de doctorat

# Dynamic Sketches: Hierarchical modeling of complex and time-evolving scenes

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

École doctorale n°626
Ecole doctorale de l'Institut Polytechique de Paris (EDIPP)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le 08 mars 2022, par

**PAULINE OLIVIER**

Composition du Jury :

| | |
|---|---|
| Tobias Isenberg<br>Directeur de recherche,<br>INRIA Saclay (LRI, UMR 8623) | Président |
| Joëlle Thollot<br>Professeure des universités,<br>Grenoble INP (LJK, UMR 5216) | Rapporteure |
| Daniel Sýkora<br>Professor,<br>Czech Technical University in Prague | Rapporteur |
| Frédéric Cordier<br>Maître de conférences,<br>Université Haute-Alsace (LMIA, EA 7499) | Examinateur |
| Marie-Paule Cani<br>Professeure des universités,<br>École Polytechnique (LIX, UMR 7161) | Directrice de thèse |
| Pooran Memari<br>Chargée de recherche CNRS,<br>École Polytechnique (LIX, UMR 7161) | Co-directrice de thèse |
| Renaud Chabrier<br>Docteur,<br>École Polytechnique (LIX, UMR 7161) | Invité |

# Dynamic Sketches: Hierarchical modeling of complex and time-evolving scenes


Pauline Olivier
Ecole Polytechnique

A thesis submitted for the degree of
*Doctor of Philosophy*

January 2022

# Acknowledgements

When I started this thesis three and a half years ago, I could never have imagined it would have turned into such a fulfilling journey. It has allowed me to grow not only scientifically but also personally. However, the achievement of this thesis would not have been possible without a series of people to whom I am grateful for their discussions, guidance, and, to a greater extent, support.

First of all, I am sincerely grateful to my advisors. Thank you Marie-Paule for your constant support and all that I have learned from you, it helps me grow as a researcher and person. Thank you Pooran for our inspiring discussions and your availability.

Next, I would like to thank all the jury members for accepting the invitation. In particular, I address some special thanks to Joëlle Thollot and Daniel Sýkora for their very kind reports and interesting questions during the defense. In addition, I am grateful to Tobias Isenberg for his detailed reading and comments that allowed me to improve this manuscript. I also thank Frédéric Cordier for his questions during the defense.

I would like to address some special thanks to Renaud Chabrier for our precious discussions which made me discover the world of art and sketching. I would like to thank Jean-Luc Coll for taking the time to explain to me some biological phenomena and for our interesting discussions. In addition, I thank Julien Husson, Philippe Chavrier and Matthieu Piel who participated in my pre-study. I also address my thanks to the architecture agency SCAU (Paris 5ème), in particular, François Gillard and Eric de Thoisy, for permitting this collaboration. In addition, I am grateful to all the architects who took the time to explain to me their practices and their needs and for having tested my prototype. Furthermore, I thank all the users who participated in my online study.

I address some big thanks to Damien for his precious help throughout this thesis, both on a technical and general level. I also address some special thanks to Magali for our breaks that helped me escape my work and to Mathieu for your support and precious advice. In addition, I thank all the people that I had the chance to spend time with during these couple of years both in and out the lab: Pierre, Thomas, Maud, Eduardo, Ariel, Jiayi, Christophe, Vicky, David-Henri...etc.; Gome, Isa, Georges, Thomas, Audrey, Pierre, Roman, Nicolas,..etc.

I express my thanks to Corentin, Thibault, Sylvain, Laure, and Carine for the good time we had between our board game sessions and excursions.

Finally, I am grateful to you, Mum, for being an inspiration to do a thesis and for always supporting me. To conclude, I deeply thank you, David, for being by my side, for helping me improve my English, and for bringing so much joy into my life.

# Abstract

Visual representations are essential to explore and communicate an idea or a phenomenon. As digital software for 3D modeling and animation are still complex and specialized, they usually do not favor creativity. In particular, they offer no easy way to quickly draft a series of alternative options. Thus, up to now, sketching on a physical medium remains the only simple and general means to create such representations. Recently, sketch-based modeling techniques were intensively studied to create 3D models but only few techniques use sketching as input to create and immerse the users into a 3D environment, guide the motion of shapes or explore hypotheses.

In this thesis, we focused on the real-time modeling of complex and time-evolving scenes using only sketching as input. More precisely, the long-term vision would be to provide users with an augmented pen enabling them to interactively create a 3D scene composed of shapes that can be put into motion or deformed while enabling refinement both on the creation and motion without any editing pipeline.

Through a collaboration with architects, we first introduce *Nested Explorative Maps*, a new type of 3D sketch for the easy creation and exploration of ideas applied to the preliminary design of man-made shapes. Our model enables coarse-to-fine sketching of nested structures to progressively shape a 3D building from the floor plan to interior design while keeping the original strokes and allowing interactive navigation through the alternative design that the sketch visually suggests.

We then tackle the synthesis of anisotropic distributions from a sketch as a means for the general creation of content both in 2D and 3D. From a simple multi-resolution analysis of the shape distributions, we propose an efficient method to synthesize the input distribution into an extended 2D domain but also a 3D embedding of this extended distribution in addition to an illusion of depth to enable users to immediately explore a 3D environment inspired by their sketch.

Finally, we collaborated with biologists to explore animated 3D sketches, where motions and deformations of organic shapes can be expressed and refined through the use of a simple depiction vocabulary inspired from standard representations in their field, and key-frame snippets.

# Résumé

Les représentations visuelles sont essentielles pour explorer et communiquer une idée ou un phénomène. Alors que les logiciels numériques pour la modélisation 3D et l'animation restent complexes et spécialisés, ils ne favorisent généralement pas la créativité. En particulier, ils ne permettent pas l'ébauche rapide d'options alternatives. Ainsi, le croquis sur un support physique reste, jusqu'à maintenant, la manière la plus simple et générale pour créer de telles représentations. Récemment, les techniques de modélisation par esquisse ont été intensément étudiées pour la création de modèles 3D mais seulement peu d'entre elles se servent du croquis comme entrée pour créer et immerger les utilisateurs dans un environnement 3D, guider le mouvement ou encore explorer des hypothèses.

Cette thèse se concentre sur la modélisation temps-réel de scènes complexes et évoluant dans le temps à partir d'entrées croquis. Plus précisément, la vision à long terme serait de fournir aux utilisateurs une sorte de crayon 'augmenté' leur permettant de créer interactivement une scène 3D composée de formes qui peuvent être mises en mouvement ou déformées tout en permettant le raffinement à la fois sur la création et le mouvement, et, sans se voir imposer d'ordre spécifique dans ce processus de création.

Grâce à une collaboration avec des architectes, nous avons tout d'abord mis en place *Nested Explorative Maps*, un nouveau type de croquis 3D dédié à la création rapide et l'exploration d'idées pour le design préliminaire de formes architecturales. Notre modèle permet d'esquisser des structures imbriquées, du grossier aux détails, afin de donner forme à un bâtiment en 3D, du plan de sol aux détails d'intérieurs et de façade tout en gardant les traits de l'utilisateur et permettant une navigation interactive à travers les designs alternatifs suggérés visuellement par le croquis.

Nous avons ensuite abordé la synthèse de distributions anisotropes à partir d'une esquisse comme un outil général de création de contenu à la fois en 2D et en 3D. À partir d'une analyse multi-résolution sur les distributions de formes présentes dans un croquis, nous proposons une méthode efficace pour la synthèse de ces distributions dans un domaine 2D étendu. Une intégration 3D de cette nouvelle distribution a également développée, complétée par une illusion de profondeur afin de permettre aux utilisateurs une immersion immédiate dans un environnement 3D qui s'inspire de leur croquis.

Enfin, nous avons collaboré avec des biologistes afin d'explorer les croquis 3D animés, dans lesquels les mouvements et déformations de formes organiques peuvent être exprimés et raffinés à travers l'utilisation d'un vocabulaire schématique inspiré des représentations standards de leur domaine et d'encarts d'image clés.

# Contents

# List of Tables

# List of Figures

# 1 Introduction

## 1.1 Motivation

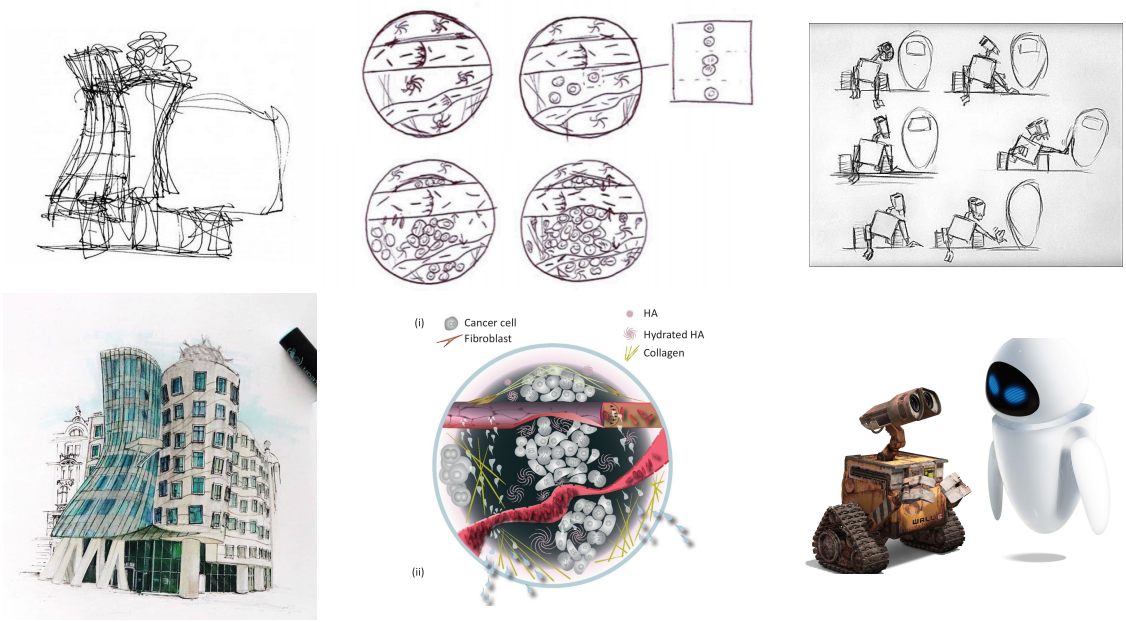



**Figure 1.1:** From the preliminary sketches to the final design of: (left) the Dancing House in Prague @Frank Gehry and @Vlado Milunić; (middle) tumor progression in the human body; Wall-E @Disney

From illustrations to schematic drawings, visual representations are fundamental in everyday life to explain an idea or phenomenon. However, it is almost impossible to directly create appropriate illustrations right away without both strong artistic skills and a clear vision of the object of interest. So it usually starts with a simple sketch. Using a pencil on a piece of paper, a marker on a board, anyone can try to convey their vision through simplified shapes, visual indications, and, trial and error. This process is illustrated for different applications in Figure 1.1: the top pictures represent annotated sketches, as first instances of the final design or phenomenon depicted below.

Regardless of the field of activity, sketching remains a natural and essential step in any creative or communication process. A sketch is a rough, easily grasped visual representation of a mental vision. While a drawing is meant to be finished work and usually requires artistic skill, a sketch retains the freedom to be modified, refined, improved, discussed, and can be considered "alive" in a certain way. It can be used by anyone as a tool for reflection, but also by others as a means of quick communication and inspire new ideas. For instance, in the creation process, it is usual to adopt visual cues to express additional information, as shown in the pictures at the top of Figure 1.1: over-sketches are used to represent areas of uncertainty, curves to express volumes. To explain a phenomenon in a simple way, it is also common to use arrows or special symbols such as those used for storyboarding, or in the specific field of study.

The strokes that compose a sketch can therefore have different purposes. In particular, these strokes can be used to:

- model 2D shapes, or even suggest a surface or a volume in 3D space,
- indicate information through annotations to:

  connect or link elements,

  illustrate different hypotheses through uncertainty or over-sketching,

  drive animation.

Up to now, the use of paper and pen remains the most convenient sketching medium. However, it has some major drawbacks. Even if everyone can find a piece of paper quickly enough, using it to represent objects in 3D or to put elements in motion to explain a phenomenon will require a lot of sketching. Moreover, it is difficult to anticipate the space needed to represent a mental image, which is left to refinements. If the support is too small, it will be necessary to make a collage of different pieces of paper, or even start again on a larger surface. If it is too large, the artist may tend to distort or disproportion parts of the project and run out of room anyway. Distortion or disproportionality can also occur when isolating particular areas of a project to specify details using zooming inserts. The latter, as the depiction of different hypotheses through uncertainty or over-sketching, can highly impinge the sketch's clarity and consequently makes it lose its essential easy to grasp nature.

Recently, some industrial digital sketching software has tried to overcome the main drawbacks of the paper and pen medium. By using a 3D environment, artists can directly create 3D objects of specific categories from a sketch and give them simple rigid movements. In an available space that seems infinite, they can navigate and isolate parts of a project without fear of disproportions. Still, the artist's control is limited to the tools of the software and these are usually very specialized

and highly restrict the sensibility and creativity. In addition, the original strokes that convey necessary but implicit information are usually replaced by single vector curves. The sketch then becomes much more polished, is difficult to refine, and thus loses its essential nature of rough visual representation.

Thus, when it comes to complex sketches, the choice of physical or digital medium favors one nature over another. The duality of the sketch, which is a rough and easily grasped visual representation of a mental vision, is then hardly accessible.

Over the last two decades, sketch-based modeling techniques have been intensively studied for creating 3D models from an input sketch to bridge the gap between physical and digital media. These methods range from general interactive systems that limit the number of priors on the modeled shapes to dedicated tools that are strongly priors-based and thus target only specific applications. However, they are generally focused on the creation of an isolated object. While refinements on the creation and simple animation can be made possible on the newly created shape, they usually prevent refinements on both the creation and the animation. Another approach is to create a design sketch that lives in 3D, although, the current techniques do not provide any animation mechanism or visualization of alternative options.

Finally, to our knowledge, proposing a digital tool that immerses users in a 3D environment inspired by their sketch and in which they can guide the movement and explore hypotheses on both creation and animation with the help of visual indications has never been considered.

In these times when our digital world is put forward, could we not design a novel digital tool to support creation? How can we increase the ability of humans to explain in a few gestures, animate, refine and clarify their mental visions while keeping the simplicity of sketching on paper?

## 1.2 Objective

The objective of my thesis is to explore the use of sketch-based techniques to propose a general methodology for the fast creation and progressive refinement of complex and time-evolving scenes. Time-evolving can refer to user-induced modifications/refinements or the intrinsic dynamic nature of shapes as actors of an animated phenomenon. The long-term vision would be to provide users with a kind of "augmented" pen, allowing them to sketch and use gesture metaphors to quickly define a coarse scene, to express movements and deformations on its elements without having to complete the modeling, to add hypotheses or prior knowledge through constraints to be maintained, and to progressively arrive at refining certain aspects of the modeling or movement without being imposed a specific order in the creation process.

The digital prototype must be interactive throughout the process allowing the user to manipulate it and virtually test different hypotheses.

## 1.3 Methodology

Even though sketching is used independently of the field of activity, not every specialist will look for the same tools in their ideal digital sketching system. Through collaborations with specialists, we focus a few specific case studies to immerse ourselves in different applications and identify specific needs and constraints. These studies serve as both inspiration and validation, the main objective being to extract a common and global methodology, at least applicable to all case studies.

In particular, these studies have allowed us to identify issues that have not yet been addressed, such as

- the need for nested sketches,
- the creation of anisotropic distributions from a sketch
- the ability to extract motion and deformation from a sketch, especially to navigate between options or to convey an animated phenomenon.

In this thesis, we focused on three case studies: sketch-based creation of man-made shapes with an application to architecture, sketch-based synthesis of anisotropic distributions of shapes in 2D and 3D space, and sketch-based creation and animation of organic shapes for a cell biology application.

Although an architectural project is composed of static elements, the sketch must be able to evolve through deformations and refinements to follow the creative process. Refinements should also be made possible when computing shape distributions from a sample sketch via returns to the sketching interface to edit the input before performing a new synthesis. On the other hand, for the cell biology application, the sketch may need some refinements and adjustments, but it must first and foremost represent an animated phenomenon composed of dynamic shapes, which must be set in motion and evolve in their environment. These case studies are thus good examples to explore the duality of the 3D shapes' time-evolving nature.

## 1.4 Contributions

This thesis presents novel approaches to extend sketch-based modeling techniques for exploring options on the creation, distribution, and animation of complex, possibly nested shapes.

**Expressive modeling** Through a collaboration with architects, we center Chapter 3 on an interactive sketching tool applied to architectural design. We propose a new type of 3D sketching based on a nested structure for a coarse-to-fine free-form design. Our creative process allows users to design coarse 3D models and gradually improve and refine the interior and exterior while preserving their original strokes and without any editing pipeline. This 3D sketch is combined with an exploration method based on local attraction to dense stroke regions to allow for the interactive navigation through alternative designs that the nested sketch visually suggests. We validated our model through a user study conducted in the SCAU agency, which enabled us to highlight the potential of our tool for conceptual design in architecture.

**Sketch-based distribution synthesis** We focus Chapter 4 on an interactive and flexible tool for real-time synthesis of anisotropic distributions of shapes in both 2D and 3D space, using a 2D sketch as an exemplar. The latter can represent a combination of bounded (isotropic or not) and/or fiber-like shapes and can be edited by the user at any time to start a new synthesis. We build our solution on a novel data structure, computed by a simple multi-resolution analysis of the distribution of the shapes along the main anisotropy directions in the sketch and the deviations from them. Based on this support structure, we propose two synthesis methods, depending on the desired output domain. In the case of volumetric distributions, we let users explore the infinite 3D environment inspired by their sketch. We validated our model with an online user study aimed at a large audience. In addition to equaling the perceptual performances of the best synthesis methods for 2D anisotropic distributions of bounded elements, we extend the distributions to the case of fiber-like shapes and extend sketch-based modeling to 3D anisotropic environments.

**Expressive animation** In Chapter 5, we jointly explore the creation of shapes and their animation based on their dynamics and we focus our work on the exploration of animated 3D sketches for an application in cell biology. For this project, we collaborated with the biologist Jean-Luc Coll to characterize animation with two types of tools: a simple vocabulary inspired by standard representations in biology and key-frame snippets. Using sketching as input to define and refine the animation, we introduce a new type of 3D sketch based on an evolving nested structure that interactively adapts to the user's sketch input. We illustrate the potential of our system using two narrative scenarios proposed by our collaborators, which allows us to highlight the potential of our tool for both exploration and communication of a dynamic phenomenon.

### Publications, Talks & Software

*International publications*

- **2019 - Nested Explorative Maps: A new 3D canvas for conceptual design in architecture.** Pauline Olivier, Renaud Chabrier, Damien Rohmer, Eric De Thoisy, and Marie-Paule Cani. *Computer and Graphics (Shape Modeling International 2019 Technical paper)*

*Under preparation for submission*

- **2022 - Synthesizing Anisotropic Distributions of Shapes from a Sketch.** Pauline Olivier, Pooran Memari and Marie-Paule Cani.
- **2022 - Narrative sketches - Application to cell biology phenomena.** Pauline Olivier, Renaud Chabrier, Pooran Memari, Jean-Luc Coll and Marie-Paule Cani.

*French workshops*

- **2018 - Interactive 3D canvases for a coarse-to-fine sketching - Application to conceptual design in architecture.** Pauline Olivier and Marie-Paule Cani. Short Paper and talk at the Computer Graphics French Days 2018 (Journées Françaises d'Informatique Graphique), Poitiers (France), November 2018
- **2021 - Perceptual distribution of anisotropic 2D strokes** Pauline Olivier, Pooran Memari and Marie-Paule Cani. Talk at the French Workshop on Geometric Modeling (Journées du Groupe de travail en Modélisation Géométrique), online, March 2021

*Software*

- **2019 - NEM: Nested Explorative Maps**. Software written in WebGL and serving as the prototype to illustrate the method

  Available at : https://www.lix.polytechnique.fr/geovic/software.html

- **2021 - Online user study written in HTML/CSS/Javascript/PHP**

  https://www.lix.polytechnique.fr/Labo/Pauline.Olivier/UserStudy/Texture/

# 2
# Related work

As presented earlier, this thesis focuses on sketch-based techniques for the easy creation and progressive refinement of complex and time-evolving scenes. To create dynamic scenes, we want users to have access to intuitive sketch-based tools for creating and animating shapes. In particular, the creation process should be made possible for a wide variety of shapes by interpreting the input strokes or synthesizing a provided sample. In contrast, animation could be applied globally or locally for a specified set of elements. In addition, we are looking for tools that also offer the ability to edit both a shape and its animation.

In what follows, we present an overview of existing methods and advances that might be suitable for our purpose and discuss their limitations. This chapter is structured as follows: Section 2.1 focuses on sketch-based modeling techniques; Section 2.2 centers on the synthesis of object distributions; and Section 2.3 on current sketch-based animation methods. Note that we will find the same three topics and in the same order, in the following chapters presenting the contributions of this thesis.

## Contents

## 2.1 Sketching to represent 3D shapes

Until recently, paper was the medium of choice for all creative design. As shown in Figure 2.1, artists or even engineers sketched their projects using annotations and standard representations of surfaces, curves, and areas of uncertainty. To obtain the closest representation of the desired 3D model, it was common practice to draw a project from different points of view and, if possible, to supplement these drawings with physical scale models to adjust dimensions or check plausibility. Although the paper medium has some drawbacks (see Introduction 1) and some unforeseen problems may arise when creating the final model in real life, this medium is a natural and direct way to create and explore ideas.

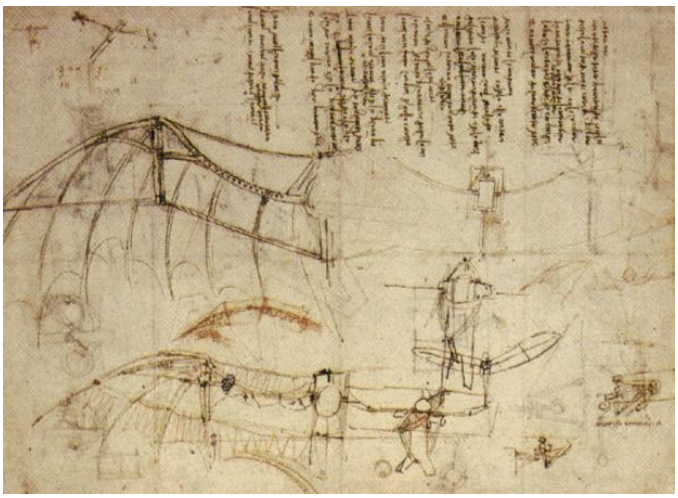


**Figure 2.1:** Leonardo DaVinci's design for a flying machine.

The spread of digital design tools, whether for industry or the general public, has thus overturned the former approach of paper sketches as it offers the possibility to design directly in a virtual 3D environment. In addition, the creative process is now accessible to a wider audience, as artistic skills, such as the ability to draw in perspective, are no longer required to design in 3D. Nevertheless, artistic freedom is further discouraged, given the complexity of the provided interface (see Figure 2.2), the smoothing of the user's strokes into splines, or the need to follow a specific sequence of steps (often non-intuitive) to create an innovative model. Digital tools are indeed not as easy to handle as the paper medium because they generally require a significant amount of time to master all the proposed functionalities. As a result, the user's attention may, unfortunately, be focused more on the technical issues related to the software's functionality and interface than on the creative process. While there have been some efforts to bring back the paper into digital sketching: initially in the form of smartpens and traditional paper (Anoto pen [Saa]) or more recently in the form of e-ink tablets with pen inputs (reMarkable [reM17], Boox Max Lumi2 [Inc21]), these technologies still need to be improved before they can be used by a wider

audience.

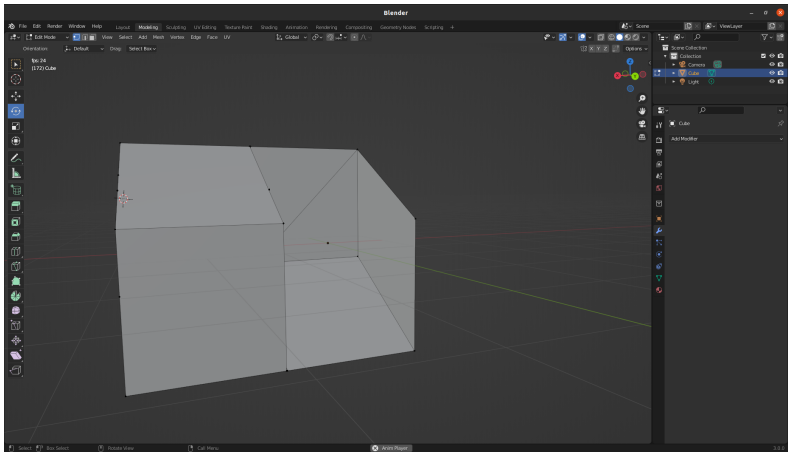


**Figure 2.2:** Modeling an object on the complex interface of Blender [Roo95].

In contrast, a series of recent advances in Computer Graphics research aims at providing more intuitive tools for creative and digital design. That is usually done through direct manipulation or sketching input. Closer to the modeling approach of industrial software, systems that rely on the first category typically provide a default 3D model that can be shaped by the user using 2D gestures [DLCB11] or sculpting [MWCS13, SCCS13, PXW18] metaphors. However, these techniques generally require the development of a wide variety of tools to handle shape deformation at different scales and for different purposes. The reader may refer to the survey by Cani and Angelidis [CA06] for an overview of early direct manipulation techniques. In comparison, we focus on sketch-based systems that do not face the same challenges since the input is a user-supplied 2D sketch. Indeed, while it is easier for a designer and novice user to use sketches as a means of expression, they can be more difficult to interpret given the diversity of artistic styles, which has led to a significant amount of research.

Sketch-based modeling focuses on creating an augmented representation (2.5D or 3D) from 2D sketches. However, the interpretation of a provided sketch can be ambiguous as its content can be highly dependent on the artist's style, not to mention the lack of depth information inherent in 2D data. Therefore, these systems relied primarily on prior knowledge to guide the creation of the appropriate model. This knowledge may, for example, result from general perceptual studies of how a shape or design is perceived, from domain-specific constraints, or even from a specific 3D surface choice. Moreover, additional user intervention may be required to complete the latter. The reader can refer to the survey by Olsen et al. [OSSJ09] and Cordier et al. [CSGC16] for a more general overview.

#### General characteristics

Below is a summary of the different options that can be used to set up a sketch-based modeling system in terms of input, creative process, and the desired output:

*Input sketch*

- **Source:** offline (drawn beforehand) or online (drawn on an interactive interface)
- **Dimension:** single 2D view, multiple 2D views or 3D sketch (e.g., a set of 3D strokes drawn in an immersive system)
- **Richness:** from rough/draft aspect to clean drawing

*Input/Output process*

The creative process can be carried out either in one pass (creation of the input and then generation of the output) or by going back and forth between the sketch in an interactive interface and completing the current output.

*Output objective*

These techniques interpret the user's sketch to derive a 3D geometric model (Section 2.1.1) or represent strokes as part of a 3D sketch (Section 2.1.2). In what follows, a 3D model refers to a mesh that is a closed object with surfaces, even if only some edges are drawn.

### 2.1.1 Inferring a 3D model from one or several sketches

This part focuses on sketch-based 3D modeling, i.e., the creation of geometric models from one or more sketches. These methods range from general systems, which limit the number of priors on the shape to be modeled, to specialized systems, which are strongly knowledge-based and thus restrict the creation to a specific set of objects.

#### From general systems

As mentioned earlier, it is challenging to propose a general system for creating the desired 3D model from any input sketch. A general approach is to search for the closest object, i.e., to use a database of 3D objects to replace the user's sketch with the best-fitting model. This approach has been widely explored for isolated object retrieval [FMK*03, YSSK10, SXY*11, ERB*12], extended to 3D scene creation [SI07, XCF*13] and even to object design [LF08, GLX*16]. However, even using deep learning models [WKL15, YLL16, GJS18] during the feature extraction phase, this modeling process remains too restrictive for our goal of promoting the user's creativity. Indeed, these techniques first require a very rich database containing sufficiently numerous and varied elements. Then, a robust matching function must be defined to determine the closest

models from a user's sketch that can be drawn by a novice, be incomplete or even contain noise. Furthermore, two sketches representing the same global object, for instance, in two different styles, will yield the same 3D model. As shown in Figure 2.3 left, only the coarse features of a sketch are usually taken into account, which can be all the more frustrating to a user who has drawn a complete sketch with details.

**Figure 2.3:** Data-driven approaches: (left) examples of 3D shapes retrieval [WKL15]; (right) sketch-based modeling with [LPL*18].

More recently, and in contrast to the last approach, a few methods have taken advantage of improvements in deep learning techniques to create a new 3D shape from 2D sketches. While Lun et al. [LGK*17], Delanoy et al. [DAI*18], and Su et al. [SDY*18] train their network on specific categories of objects, Li et al. [LPL*18] propose a method to create generic free-form 3D surfaces from a single-view or multiple-view sketch (see Figure 2.3 right). From a provided 2D sketch, their system creates an input map consisting of the sketch, a foreground/background binary mask, and optional user annotations, such as depth or curvature information, to infer the 3D surface using two sub-networks. A DFNet sub-network generates a flow field from an input map, and these two are fed into a GeomNet sub-network to determine the depth and normal maps of the surface, as well as a confidence value specifying the most ambiguous area of the sketch. Their approach allows for coarse-to-fine design as the user can start modeling a coarser sketch and gradually add new annotations to the drawing to refine the generated surface. However, this system seems limited to shapes that can be represented by an elevation model in a flat silhouette. Different sketches must be used and combined when the desired shape does not meet these requirements, such as the fish with a side fin for the Figure 2.3, right.

As an alternative approach, while immersive systems have been widely studied for 3D sketching (see Section 2.1.2), Surface Brush [RRS19] tackles free-form modeling from 3D sketches. The authors rely on the characteristics of 3D brush drawing to generate a 3D manifold

free-form surface that matches the user's input. From a 3D sketch built incrementally by the user (see Figure 2.4 (a)), their method infers the closest 3D surface using a discrete constrained optimization framework, to first cluster closed-by stroke regions into manifold partial surfaces (Figure 2.4 (b)) and then fill the gaps between them, while preserving the manifold property (Figure 2.4 (c)). The main strength of their system is the ability to generate man-made and organic shapes, however, their surface reconstruction is not real-time or interactive. Indeed, the user must create a complete input before starting the surface reconstruction. Furthermore, while the 3D brush removes any ambiguity about depth, their system requires specific hardware with which users must be familiar, especially to follow their fence-painting metaphor.

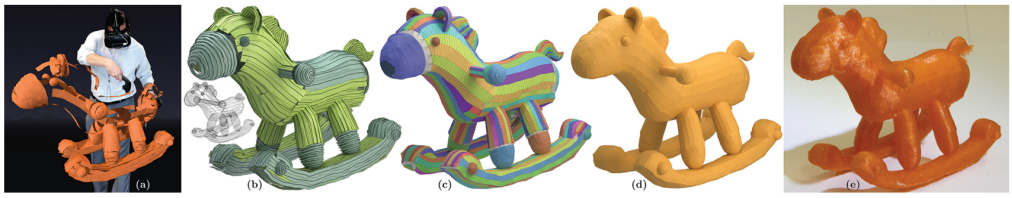



**Figure 2.4:** Surface Brush [RRS19] modeling process, from the user's 3D strokes drawn with a VR brush (a) to the final output (d) and fabricated model (e).

In the spirit of the paper & pencil approach, modeling 3D shapes using 2D sketches remains the simplest solution. Nevertheless, due to the ambiguity of 2D projection, it is not possible to directly convert a 2D sketch into the most appropriate 3D model without additional user intervention or prior knowledge. However, instead of providing a complete sketch, one solution is to let the user gradually shape the desired model, through iterative sketches on an interactive interface. In particular, using a specific mapping between a 2D sketch and a 3D model allows for the generation of various shapes without needing additional assumptions. In what follows, we focus on two different inputs: sketching "gestures" and 2D silhouettes.

*Sketching "gestures"*

Inspired by studies on drawing perception and visual understanding, Zeleznik et al. introduce SKETCH [ZHH96] an interactive system for rapid design and editing of 3D scenes based on sketching gestures. This approach can be seen as an alternative between detailed paper sketches and digital software primitives selection as the authors define correspondences between a predefined set of strokes, also called gesture, and a 3D model (see Figure 2.5 left). Following certain assumptions about object placement, their method replaces each sketch gesture by its associated 3D model relative to the stroke environment, which allows for the creation of complex structures. In addition to the modeling process, some editing tools, based on additional strokes or direct manipulation gestures, can be used to manipulate shapes or apply specific constraints (see Figure 2.5 right). Although this system emphasizes the importance of sketch-based digital solutions for the rapid

prototyping of 3D models from sketch inputs, it can inhibit creativity. Indeed, the user is limited to the recorded gestures, and thus, modeling is restricted to the associate 3D models.

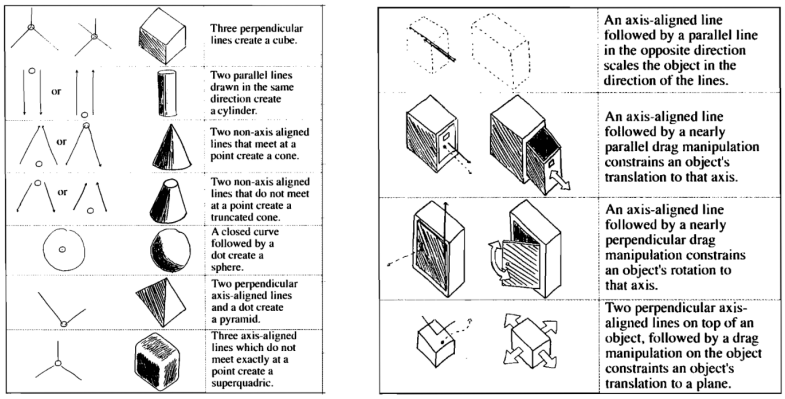



**Figure 2.5:** SKETCH [ZHH96]: (left) Examples of sketch gestures and their associated models; (right) Examples of sketch gestures to manipulate shapes and apply constraints.

As shown in Figure 2.5, each sketching gesture in SKETCH [ZHH96] is composed of two to three inputs, which can be strokes or points. While this gesture solution is intuitive for creating angular shapes such as a cube or a pyramid, it is not as natural in the case of a sphere where a simple circular outline might suffice. In particular, some perceptual studies [Kof55, AF69] have established that, without prior knowledge, the user will mentally infer the simplest 3D shape from a 2D silhouette. In this state of mind, any 3D model with spherical topology, also called an organic shape, could be represented by a 2D contour, interpreted as the silhouette of the shape.

*Sketch 2D silhouettes*

In particular, the Teddy system [IMT99] is a pioneer in the interactive modeling of free-form organic shapes by inflating 2D sketched silhouettes. On its interactive interface, the user successively draws closed free-form 2D contours. From each contour, the system first retrieves the underlying 2D shape before computing a triangulation and the medial axis. Then, it lifts the axis vertices according to their distance from the contour to deduce the corresponding 3D shape. As illustrated in Figure 2.6, this modeling process can be completed by free-form drawing on a surface as well as some sketch-based editing operations, such as extruding, cutting, deleting, smoothing, and distorting a model.

Following Teddy's approach of inflating 2D silhouettes, some authors choose to model the underlying 3D shapes using implicit surfaces instead of polygonal meshes. Such a surface can be defined by the set of points $p \in \mathbb{R}^3$, such that: $f(p) = c$, with $f : \mathbb{R}^3 \to \mathbb{R}$, a field function and $c \in \mathbb{R}$, an isovalue. Therefore, each surface can be described by these two parameters. While this specific surface representation facilitates Constructive Solid Geometry (CSG) and blending

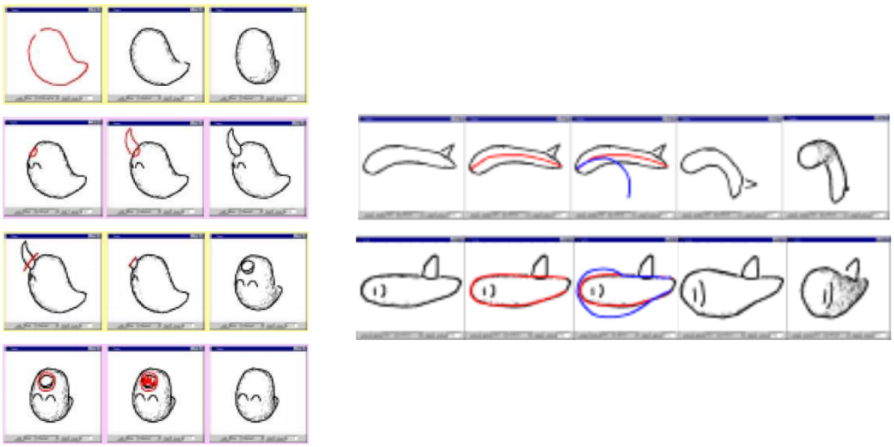


**Figure 2.6:** The Teddy system [IMT99]: examples of different operations (modeling, extrusion, cut, erasing and distorting).

operations, the main challenge lies in determining the appropriate parameters to create a model that matches the user's sketch. Existing methods can be classified into two categories depending on whether they inflate their surface from the contour of the underlying 2D shape or from the medial axis. For the first category, Karpenko et al. [KHR02] base their modeling on variational implicit surfaces [TO99]. However, their model depends heavily on some constraints that increase the computation time in the case of complex structures. In contrast, ShapeShop [SWSJ07] relies on Hierarchical Implicit Models [WGG99] to facilitate the creation of complex shapes. From a 2D variational implicit curve fitted from the user's 2D contour, this system infers a 3D surface by determining a 2D scalar field bounded around the curve and sweeping this field along the depth axis. As illustrated in Figure 2.7, the use of a hierarchical structure allows the authors to provide editing modes such as cutting holes or removing some volume. Their system also offers two additional modeling processes by extrusion or revolution symmetry, as well as the ability for the user to adjust depth parameters using sliders.

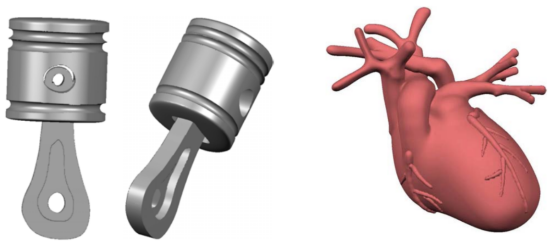


**Figure 2.7:** Examples modelled with ShapeShop [SWSJ07].

Although ShapeShop provides realistic results, these contour-based approaches require the placement of additional points in depth that serve as constraints, in addition to the use of an optimization function to restrict the surface to fit the input contour. To counter this problem,

Alexe et al. [AGB04] and Tai et al. [TZF04] define their implicit surfaces from the skeleton of the shape, extracted using Teddy's [IMT99] method. In the system of Alexe et al. [AGB04], implicit spheres (or blobs) are positioned at the location of each vertex of the skeleton and the surface is reconstructed by blending the contributions of the blobs. However, due to the blending of individual spheres, their resulting surface is not guaranteed to be smooth. To counter this issue, Tai et al. [AGB04] compute the field contribution of each segment of the skeleton (and not the points, as before) and rely on a convolution model on the skeleton to define their surface.

In addition to following this modeling approach, Bernhardt et al. innovate the input data in Matisse [BPCB08] by relying on a painting metaphor, as opposed to drawing only the outline of the 2D shape. In particular, this choice avoids any prior smoothing operation on the user's input data. With this metaphor, the user paints a region on a texture image, on which the medial axis of the underlying 2D shape is extracted by iterative erosion [Hal89] and a distance image is computed using a Weighted Distance Transform. From these two structures, their method derives a convolution skeleton defined as a graph of branching poly-lines. Their modeling process relies on the convolution surface model of Tai et al. [TZF04] to infer the surface from the skeleton while adjusting the convolution weights to the resolution of the input image. As illustrated in Figure 2.8, users can create complex models by progressively painting regions with brushes of various widths, providing them with direct visual feedback on the appearance of the created surface.

**Figure 2.8:** Modeling process with Matisse [BPCB08]: (left) bridging between two shapes using the painting metaphor; (right) original artwork from Matisse taken as inspiration to create the complex 3D model on the right.

Conventional implicit models such as those described above have some drawbacks, such as the inability to create sharp edges or the loss of small details when merged with a large shape. To address these issues, Zanni et al. present SCALIS [ZBQC13], a scale-invariant implicit model that warps the shape skeleton before convolving it to create a scale-invariant field. In addition, their approach permits radius control by adding an extra point or segment skeleton on the skeleton extremities or maximum radius vertices to adjust the desired field value. As shown in Figure 2.9, their method allows for the creation of sharp features as well as the preservation of small details,

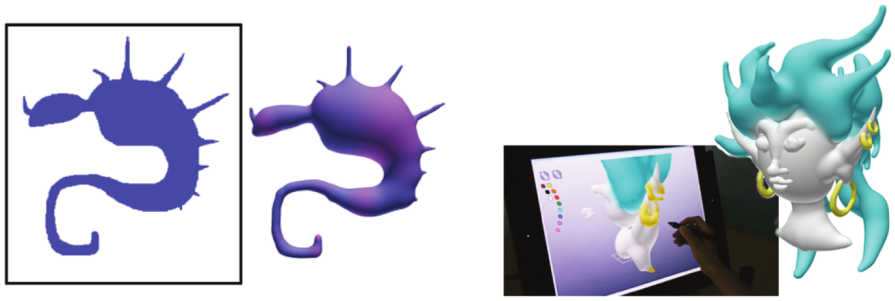


**Figure 2.9:** Results modelled with Scalis [ZBQC13]: (left) possibility to model sharp features; (right) a complex model composed of elements at various scales, using Scalis [ZBQC13].

even when blended with larger structures.

As an alternative approach to promote the creation of sharp features, Nealen et al. propose Fiber Mesh [NISA07], a system in which the user's strokes define control curves, serving as 2D silhouettes to create 3D shapes but also as handles to deform the latter. To integrate this last functionality, the authors define the desired 3D shape by a polygonal mesh computed by triangulating the underlying 2D shape of the contour before deducing the depth by functional optimization. The user can deform a model by adding new control curves, changing the type of existing ones (from smooth to sharp and vice versa) but also more locally by drag-and-drop gesture on a curve. For this last process, the authors rely on two least-square minimization frameworks to deform the curve while preserving the details and updating the surface in real-time. Following Teddy's approach, their system also offers additional sketching tools such as cut, extrusion, and tunnel operations. With their concept of control curves, their system provides an interesting tool that allows both the creation and edition of shapes from sketches and direct manipulations. Since this model defines the control curves as positional constraints only, the deformation range of these curves may not be sufficient to model a higher level of curvature deformation as in more constrained but offline methods such as [LPL*17].

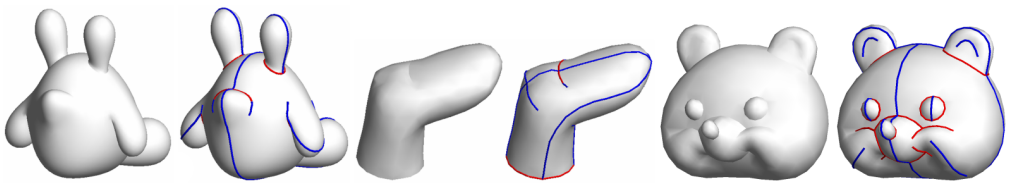


**Figure 2.10:** Modeling results in FiberMesh [NISA07], the user draws strokes that serves as control curves (blue = smooth curve, red = sharp curve).

While these systems can model a wide variety of shapes using classical, deep learning, or even virtual reality techniques, none of them provide the perfect tool that favors the user's creativity while allowing for the easy creation of all the shapes. Indeed, for some specific applications,

additional knowledge, either through assumptions, or user intervention is essential to guide the modeling towards the desired result.

**Toward more constrained methods**

Depending on the chosen constraints, the methods in this category can range from a specific type of surface for the 3D model, such as parametric surfaces [NGDA*16, HKYM17, HGY17] to the creation of very specific shapes, like self-occluding objects [KH06, CS07].

In what follows, we first give a brief overview of some domain-specific methods from applications that we have not explored in this thesis. Then, we focus in more detail on sketch-based 3D modeling methods applied specifically to architecture and biology.

*Brief overview domain-based approaches*

From the perspective of creating more diverse 3D scenes from sketches, we present an overview of application-specific techniques using sketch inputs. For instance, existing methods have approached terrain modeling by drawing a single silhouette as a profile curve [WI04] or feature curves, either to edit an existing terrain model [BMV*11] or to generate a new model from a 2D sketch [HGA*10, GDG*17]. This terrain can be filled with trees interactively drawn by the user in a single sketch [OOI06], in a coarse-to-fine manner [WBCG09] or in an immersive environment [ZLC*21, YH21]. To populate this 3D world, the user can model 3D characters, either progressively on an interactive interface [GIZ09] or from a static sketch, which can be side-view to model a wide variety of animals using symmetry assumptions [EBC*15, DSC*20], or from a more random view but completed by a 3D skeleton [BCV*15]. This character can be dressed from fashion design sketches using developable surfaces [RSW*07, JHR*15, FBR*17, FRH*21] or interactively and directly on the 3D model for layered structures [DPS15]. Finally, hair modeling has been studied by inferring a helix model from a 2D sketch [WBC07], or by using deep learning approaches [XNC*19, SZF*21]

We will now focus in more detail on our two application domains, namely architecture and biology.

*Sketch-based 3D modeling applied to architecture*

As in any specific application, an architectural design follows certain rules and constraints inherent to the field. For instance, most architectural designs can be characterized by man-made shapes, and some by only organic shapes or a mix of both. In addition, they are usually composed of one main item containing a facade with 2D or 3D features and an interior with inner floors and other elements. Therefore, sketch-based modeling methods addressing general, man-made shapes

are not suitable, as they do not support the modeling of specific features inherent to architectural models. Although mostly used for garments, methods dedicated to developable surfaces have been illustrated with architectural examples [RSW*07], but, as shown in Figure 2.11, they limit the modeling to developable facades and roofs.

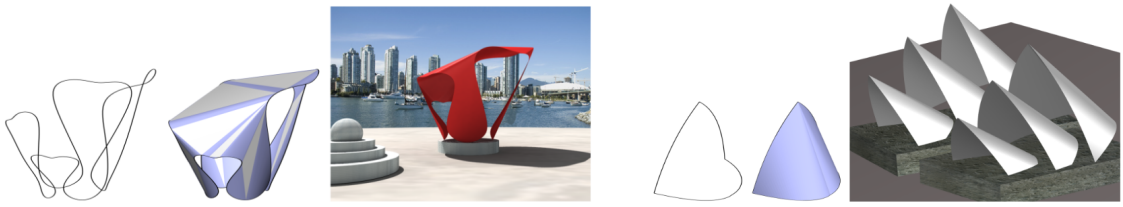


**Figure 2.11:** Architectural designs modelled with developable surfaces [RSW*07]: (left) gazebo; (right) Opera House.

Only few methods have been specifically proposed for interactive sketch-based modeling of architectural models: Facetons [SCSI15], Sketching Reality [CKX*08], Interactive Sketching of Urban Procedural Models [NGDA*16] and BuildingSketch [LZC21].

Facetons [SCSI15] is dedicated to the interactive design of building exteriors in an immersive environment. Equipped with a head-mounted display and a six-degrees-of-freedom input device, the user designs an architectural model by positioning and editing oriented 3D points, or "facetons". Each point represents a face primitive, such as a plane or a cylinder, from which the 3D model is derived by delineating the primitive relative to its intersections with others. As illustrated in Figure 2.12, their creation process is limited to the assembly of plane and cylinder primitives by positioning their "facetons".

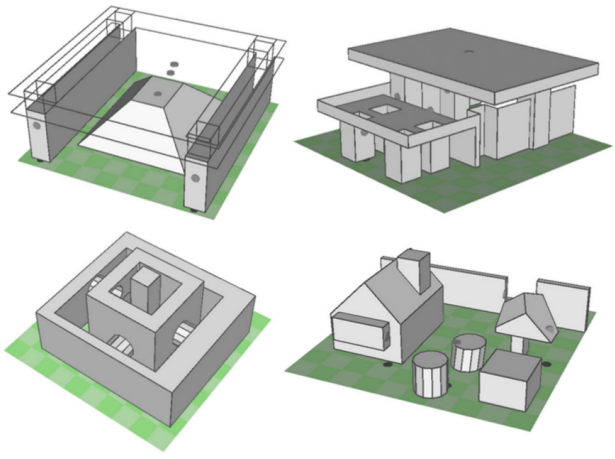


**Figure 2.12:** 3D architectural models designed in VR with Facetons [SCSI15].

In contrast, Sketching Reality [CKX*08] provides an interactive interface on which the user's strokes are interpreted into 2D geometric primitives (dot, circle, straight line, or Bézier curve). After drawing a stroke, the user specifies its type (primitive, detailed geometry, texture) to help a maximum likelihood algorithm determine the closest 3D model from a database. In addition, an iterative refinement mechanism allows for coarse-to-fine design, as illustrated in Figure 2.13.

**Figure 2.13:** Example of a rotunda created with Sketching Reality [CKX*08].

Finally, Urban Procedural Models [NGDA*16] and BuildingSketch [LZC21] combine a sketch-based interface with procedural modeling of buildings. In the first system, the authors train a set of convolutional neural networks to recognize procedural parameters of 3D models belonging to the same category, such as mass buildings or roofs. As illustrated in Figure 2.14, the user progressively shapes an architectural building through iterative sketches and the determination by the CNNs of the most plausible 3D models. This approach allows for coarse-to-fine model refinement while letting the user choose the final model from a small set of selected models proposed by the system.

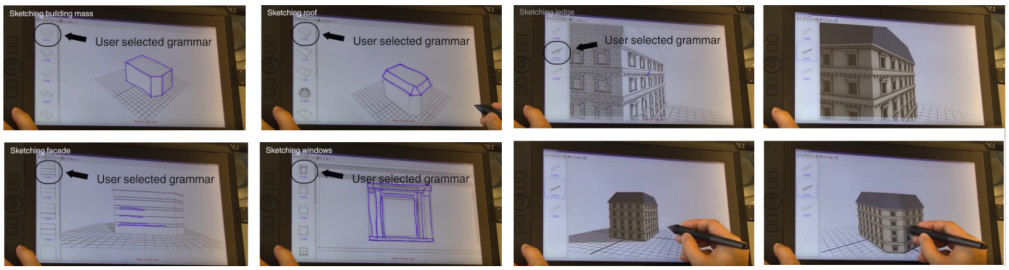



**Figure 2.14:** Modeling process with Urban Procedural Models [NGDA*16].

In contrast, BuildingSketch [LZC21] relies on an immersive system that interprets a set of 3D strokes drawn by the user into the closest procedural models, as depicted in Figure 2.15. The authors rely on a recurrent neural network to determine the category of the 3D model from the sketched strokes, and in the case of a free-form shape, a PointNet helps to decompose the stroke into regular parts. Finally, the system automatically generates the 3D model by deducing the height and tilt of the 3D model from the 3D sketch. In addition, their model allows for direct manipulation of the generated models to move them in space. While very effective for quickly authoring a set of nice-looking buildings for a virtual city, these two last systems limit the user's

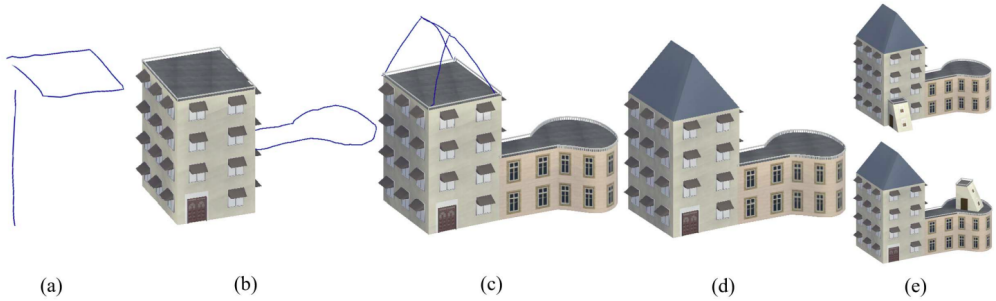



**Figure 2.15:** Modeling process with Building Sketch [LZC21].

creation to the pre-recorded procedural models.

None of these methods promote the user's creativity as they restrict the design to only the exterior of a building and the modeling to already pre-defined 3D models. Indeed, none of them authorizes the design of free-form shapes, nor does it allow the joint design and exploration of the interior and exterior of buildings. In contrast, in Chapter 3 we focus on an intuitive system assisting architects during the early stages of design where the building shape is still under exploration and users can sketch visual information to guide further refinement.

*Sketch-based 3D modeling applied to biology*

Biological shapes are generally organic. Moreover, as in the case of architecture, these structures are usually nested, i.e., a human body contains vessels, which themselves enclose blood cells, etc. However, creating a biological model is not as restrictive as other domain-based approaches. Indeed, this field is still under intensive study. The main challenge is to provide simplified shape representations that are more understandable than real, reconstructed biological data. That leaves a lot of freedom in the style and level of detail of the representation of these shapes. Therefore, biologists are looking for intuitive and creative tools to clarify, visualize, or even communicate their representation of some shapes but also explore some hypotheses. We refer the reader to [VGH*05, Ise15, LVPI18] for more details on visualization techniques applied to the illustrative rendering of biological shapes. In contrast, the survey by Preim and Saalfeld [PS18] presents an overview of virtual human anatomy education systems. In what follows, we focus on existing sketch-based methods applied to modeling anatomical structures, especially at the cellular scale.

While character modeling and animation lead to a great deal of research on muscle modeling [LGK*10], Abdrashitov et al. [ABL*21] propose the first interactive sketching tool for creating and exploring musculoskeletal structures. Based on the skeletal and skin representation of a 3D model, the user can sketch profile curves whose extremities lie on the same bone to create

a muscle around the latter. The system then deduces a muscle using a diffusion process and, as shown in Figure 2.16, the user can edit the latter by changing the diffusion rate. Their approach provides an interesting tool for real-time creation and exploration of other anatomical structures, especially by generating the muscles' tetrahedral meshes only at the end of the modeling. The main limitation of their current system is to restrict the definition of a muscle to a single curve.

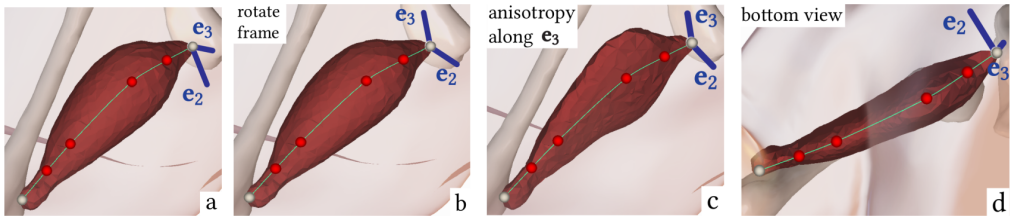



**Figure 2.16:** Sketch-based modeling of muscles [ABL*21]: after the modeling, the user can edit the shape by changing the rate of diffusion along certain directions.

More relevant to our target application, two methods tackle the sketch-based modeling of vascular systems as a tool to support anatomy teaching. Pihuit et al. [PCP10] rely on the sketching conventions used in anatomical drawings to model vascular systems from a single sketch, as illustrated in Figure 2.17. Their method follows Matisse [BPCB08] skeleton extraction process to retrieve from a provided sketch both a surface and a contour skeletons. From these two skeletons and their intersection, their system identifies the properties of each vessel, such as its orientation, curvature, and connections to others, based on the sketching conventions. From this analysis, it constructs a 3D skeleton from which an implicit surface is inferred by convolution. This system can be an interesting tool for teaching anatomy while preserving the sketching conventions. However, using a single sketch as input is too restricted for our objective of iterative refinements, especially if the user needs to provide, from the start, a sketched representation of the entire 3D scene.

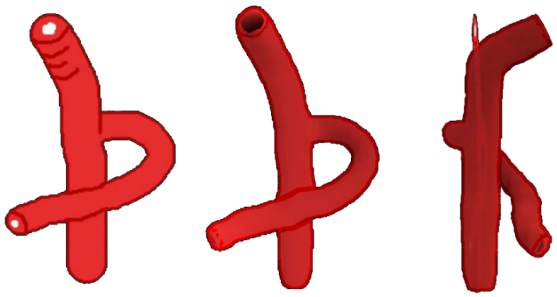


**Figure 2.17:** Sketch-based modeling of self-occluding vascular system on [PCP10].

Closer to our modeling objective, Saalfeld et al. [SSPOJ16] propose an interactive and semi-immersive system to create vascular systems but also edit them to promote the explanation of vascular pathologies. Using a six-degree-of-freedom stylus, the user can directly define a vessel in 3D by sketching its centerline. Their model then samples and smoothes each stroke by a Gaussian kernel to ease the creation of a continuous implicit surface generated by the blending of Metaballs of constant radius. As illustrated in Figure 2.18, this system can be used to represent vascular pathologies by updating the weight of Metaballs (left) and activating a blood flow inside a vessel (right) through a local drag-drop gesture. While their method tackles challenges similar to those covered in our Chapter 5, the authors' modeling approach smooths out the user's strokes and imposes constant width centerlines thus, resulting in similar width for the implicit surfaces. In addition, their animation process requires significant user intervention by dragging the stylus to each branch to allow blood flow throughout the vascular structure.

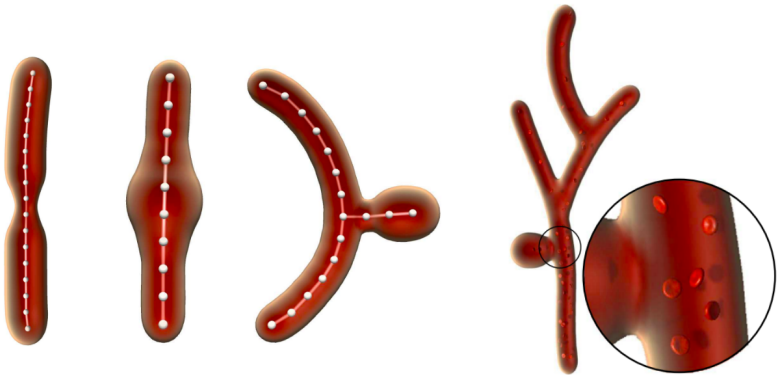


**Figure 2.18:** Sketch-based modeling on [SSPOJ16]: (left) vessel editing to represent some pathologies; (right) blood flow visualization.

### 2.1.2 Creating a 3D sketch from user's strokes

In the same mindset as FiberMesh [NISA07] which allows the user to place strokes on the 3D model, some methods do not seek to replace the input strokes and even encourage their creation in 3D through the use of support volumes or surfaces; the other end of the spectrum being to promote 3D sketch creation without any support model. As illustrated in Figure 2.19, a sketch often conveys a richer representation of an object of interest than its corresponding 3D model because it carries more information and, in particular, the artist's style. This is why sketches are still massively used, even in 2D, in the early design stages.

For this reason, many methods tackle the problem of creating a design sketch that lives in 3D instead of constructing a 3D geometric model. Indeed, instead of targeting the creation of such 3D geometric models that may only roughly fit the user's strokes, 3D sketching systems focus on guiding the user through the progressive sketching of the desired 3D model. The main challenge

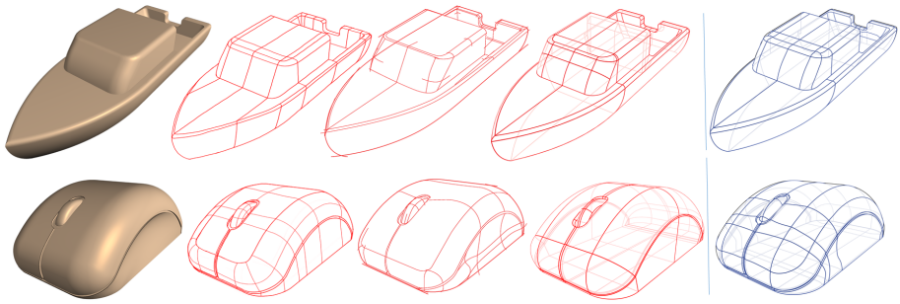


**Figure 2.19:** The comparison between artists' generated networks: (left to right) the input model, the generated 2D design drawings and 3D network from artists (in red), and FlowRep's [GSV*17] algorithmic result (in blue).

is to favor the creativity of users while easing the 3D positioning of strokes. In what follows, we present various alternative input techniques to address this challenge by direct creation, constrained strokes, strokes' projection onto support structures, and finally sketching in a predefined context.

### Direct creation

The first approach consists in letting the user create a 3D sketch directly, whether in 2D or immersive systems.

*2D inputs & prior knowledge*

Closer to sketch-based 3D modeling techniques, some methods rely on prior knowledge to interpret 2D sketches into 3D. These techniques aim either at generating an illusion of 3D by rendering or image blending techniques or at creating a network of 3D curves.

To generate an illusion of 3D from a 2D sketch, a first approach is to compute the normal field of the underlying shape of the input sketch, either by estimating the curvature lines [SBSS12, IBB15] or the underlying 3D model [SKv*14]. However, these methods do not allow 3D navigation since the sketch remains 2D. On the other hand, Bourguignon et al. [BCD01] propose a solution to render a single sketch through different viewpoints by interpreting the user's 2D strokes as silhouette contour. Their method deduces the silhouette's differential geometric properties to generate a local surface around it. Then, it adapts the rendering intensity of the stroke according to the current viewpoint to highlight the confidence of this surface. As shown in Figure 2.20, the user can also edit an existing silhouette by sketching a new stroke from a different viewpoint. Although this system provides a simple solution to infer depth, the strokes are all planar. In addition, extreme refinement of a shape by sketching silhouettes from multiple viewpoints can impair the readability of the sketch.

**Figure 2.20:** Illusion of 3D in [BCD01]: artistic illustration seen from three viewpoints.

To avoid this issue, SketchSoup [ADN*17] encourages exploratory ideation in conceptual design. Using a set of raw ideation sketches drawn on an interactive interface and the correspondences between them, their system embeds this set into a 2D interpolation space using both image warping and blending. As illustrated in Figure 2.21, the user can navigate into this continuous design space to explore solutions. Moreover, additional sketches can be input into this space to refine the exploration. SketchSoup is an appealing tool for the interactive exploration of alternative options. However, the user must draw each sketch individually, which can be both an advantage and a limitation.

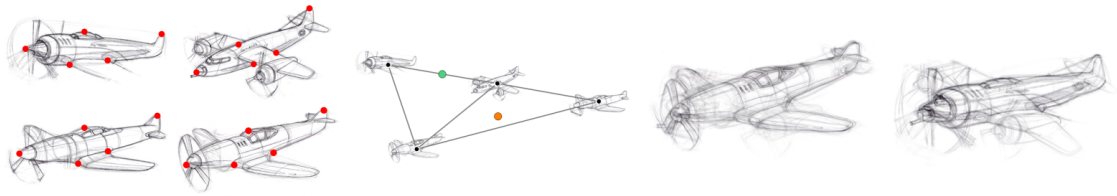


**Figure 2.21:** Exploratory ideation in SketchSoup [ADN*17]: (left) input sketches and their correspondences; (middle) 2D interpolation space; (right) interpolations at the orange and green dot.

Another approach is to create a 3D curve network from a provided sketch. Cordier et al. [CSMS13] propose a solution to lift a set of mirror-symmetric curves in 3D by exploiting the topology of the curve and, in particular, by recovering symmetry parameters from the connectivity of the curves. As an extension, the system True2Form [XCS*14] uses the regularity properties of spline-based design sketches as a hard constraint to lift control points from input strokes while preserving sketch fidelity, using an optimization framework. However, using piecewise cubic Bézier splines, this system only takes clean drawings as output. More recently, Gryaditskaya et al. [GHL*20] introduce a system for lifting rough 2D design drawings containing scaffolds and surface curves by taking advantage of the geometric cues provided by the construction lines. Although this method extends the previous ones to rough sketches, all of these methods take as input complete 2D sketches drawn by artists.

While being able to turn around an existing 2D sketch is very useful in the early stages of design, the current methods do not allow to use this new representation as a context in which additional details can be drawn directly. Thanks to its interactive interface, the system of Bourguignon et al. [BCD01] offer an interesting solution to allow a sketchy representation of a 3D world from strokes. However, they limit the strokes to planar curves drawn on a drawing plane facing the camera and whose depth position can be adjusted on the user interface.

*3D inputs & accuracy*

With recent advances in immersive systems, the most intuitive solution for creating a 3D sketch is to let users directly sketch 3D strokes. Since the pioneering 3-Draw [SRS91] and Holosketch [Dee95], virtual reality systems have sought to make the user experience as familiar as possible, for instance, by allowing 3D painting in a cave environment [KFM*01] or using hand motions to create 3D shapes [SPS01]. However, compared to the ease of drawing on a 2D paper or interface, direct 3D sketching can lead to difficulties in accurately positioning strokes in space and relative to each other [AKA*17, BMSA19]. Thus, to solve these issues, recent methods have tried to reduce the user's mental load by providing tools to guide the creation of a stroke or by applying some stroke neatening [YAS*21, YDSG21].

In particular, this creation guidance can be provided either by precisely defining a stroke in 3D space or relying on support structures onto which the strokes will be projected.

**Constrained stroke**

The precise definition of the position of a stroke in 3D space can be done globally, for example, by drawing the desired curve from different points of view or locally by iterative guidance of the current local tangent.

*Multi-stroke sketching*

Early methods have tackled the creation of non-planar 3D curves from 2D sketches through multi-stroke sketching. Indeed, defining a 3D curve by providing its projection from at least two viewpoints allows for removing the ambiguity of depth. Two families of approaches have been used to follow this concept: single-view or multi-view sketching. While the former avoids any camera rotation between the two sketches, it limits the user to drawing all curves from the same viewpoint. For instance, Cohen et al. [CMZ*99] propose a single-view sketching system in which a 3D curve is created and edited by matching a curve sketched from the current viewpoint with its shadow drawn on a provided ground plane. As another option, Karpenko et al. [KHR04] rely on the epipolar projection system to precisely define a 3D curve by its projection along two viewpoints. In their system, the user keeps sketching 2D strokes along different viewpoints

to refine the inferred 3D curve. As an alternative to these two approaches, Bae et al. present in ILoveSketch [BBS08] a rich set of different 3D sketching modes, including a single-view sketching mode, in which the user sketches a pair of symmetric curves with respect to a given center plane and a multi-view mode relying on the two-views epipolar system. In all of these approaches, the user must have a good understanding of 3D space and the spatiality of the curve in that space. Therefore, such systems are most often limited to expert users or require additional training to be used by novice users.

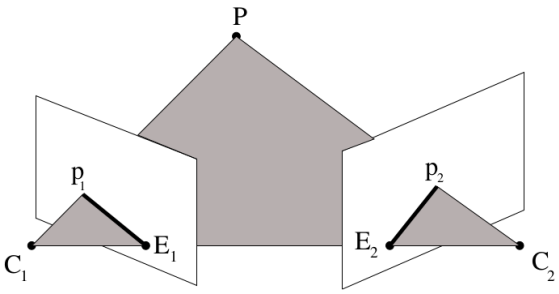



**Figure 2.22:** Overview of the multi-view epipolar projection system to precisely define a point P in 3D space.

*Tap Drawing*

The tap drawing technique consists in using one hand as a guide to making an accurate sketch with the other hand. Originally introduced for the creation of large-scale structures using a physical tap, Balakrishnan et al. [BFKB99] first extend it to a digital version and then Grossman et al. [GBK*02, GBS03] adapt it to 3D and immersive systems. More recently, the system Drawing on Air [KZL07] is built on this metaphor and in an immersive system to provide users with two complementary modes of 3D sketching: a one-handed drag manipulation, and a two-handed tape drawing, as well as a tangent preserving method for switching between them. This technique allows for a precise definition of a non-planar 3D curve, but it requires either alternating between guiding and sketching the stroke or a good synchronization between the two hands.

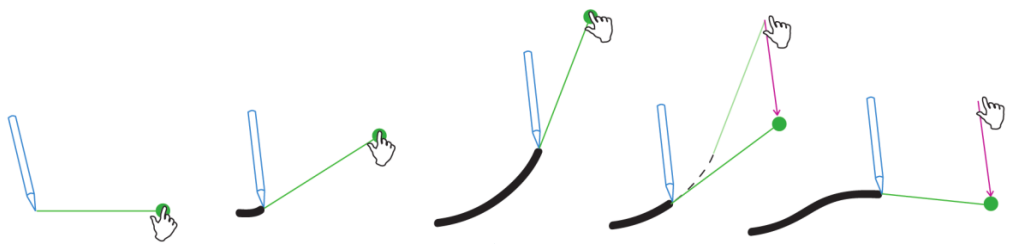


**Figure 2.23:** The bi-manual tap drawing technique: the drawing direction is determined by the position of the non-dominant hand and the current stroke endpoint. Drawing a curve stroke requires to synchronize the motion of both hands.

Creating a 3D curve using multi-stroke sketching or tap drawing can accurately define non-planar 3D curves. However, it may require specific expertise compared to the processes of direct creation and drawing on support structures.

**Drawing on support structures**

The alternative approach to guide the creation of stokes in 3D space is to project them onto an existing surface (canvas) or a more global structure (scaffold). Thus, both the choice of structure and its position in space characterize the general shape of the stroke that will be projected onto it. Indeed, a planar surface cannot accommodate a non-planar curve such as a helix.

*Planar canvas*

Aiming at the creation of a 3D scene from 2D strokes, Cohen et al. introduce in Harold [CHZ00] a basic environment composed of one flat ground and a sky on which the user can interact by sketching. In particular, a stroke can serve as a silhouette to edit the ground but also to create a 2D billboard orthogonal to the latter and on which strokes can be projected. To preserve the relationship between strokes when the viewpoint changes, their system also allows for the creation of bridge billboards to connect two existing ones. Their use of billboards as the support structures is interesting because it remains simple and easy to comprehend. However, since such structure is always facing the camera, it does not permit users to create a complex 3D sketch in which sketching from different points of view will give real depth to an object. For instance, inspired by architectural drawings, the system Mental Canvas [DXS*07] is a pioneer in the concept of using semi-transparent 2D planes, also called canvas, as support primitives to position strokes in space. As illustrated in Figure 2.24, the user interactively places a predefined canvas in 3D space before drawing strokes on them. The main innovation of their system is the ability to project a stroke from one canvas to another canvas to facilitate the creation and visualization of 3D sketches.

However, none of these systems favors the creative workflow as the user keeps switching between interacting with the scene and changing modes on the menu. Targeting a user-friendly interface, ILoveSketch [BBS08] and EverybodyLovesSketch [BBS09] rely on sketching to create and switch modes. Indeed, the authors implement a small set of pen gestures to help the user stay focused on the design while changing modes. To enhance the range of support surfaces, the authors propose two solutions for creating them: a tick-based approach that allows the user to create a plane by positioning points on an existing curve and taking the plane that contains them; or a sketch-based approach to defining an extrusion surface by sketching a profile and a direction. However, in both of these concepts, surroundings surfaces or curves are required to enable the creation of a new one. Therefore, the user may need to follow a certain creative workflow or

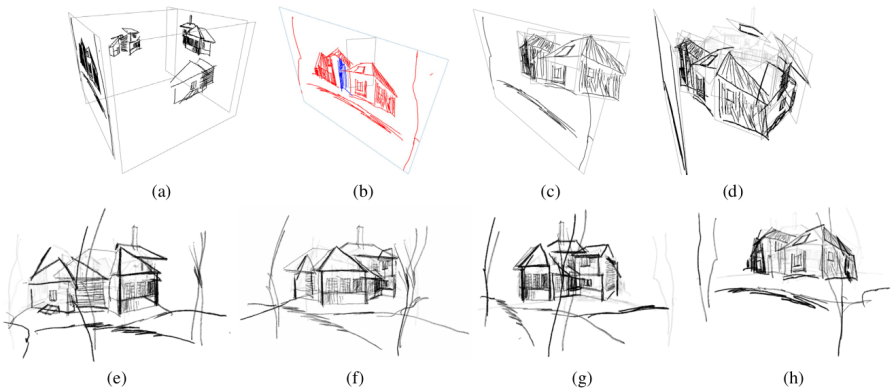



**Figure 2.24:** The creation of a suburban house in Mental Canvas [DXS*07]: a) the designer sketches on four different canvas positioned in 3D space; b) and c) highlight the impact of the strokes' projection in 3D; d) the house can be viewed from novel viewpoints; e)—h) landscape elements are added to the design.

identify an adequate series of surface construction to obtain the desired 3D curve.

*More complex structure*

To support the design of non-planar 3D curves, Schmidt et al. [SKSK09] rely on a scaffold structure to guide the creation of strokes in a perspective 3D space. During the design process, the user alternates between creating, editing the scaffold structure, and sketching on it. The system provides visual guidelines to help the user define a regular scaffold. In addition, it relies on an inference strategy to determine the spatiality of a curve in 3D space from its anchor point on the scaffold. While scaffold structures can provide an easy decomposition of the 3D space, the appropriate scaffold structure might be difficult for novice users to define without knowledge of analytic drawing.

To counter these issues, Kim et al. [KALB18] make use of hand motions to generate rough 3D shapes on which strokes can be projected. The authors rely on a spray of polymer particles metaphor to generate a 3D shape out of a hand motion. The advantage of this feature is that the user has great control over the position and orientation of these shapes without having to rely on existing surfaces. In addition, visual displays are used to highlight the hand position relative to the already created scaffold and the user can remove entirely or partially some air-scaffold structure to reduce the overflow of information. The main drawback of this approach is that it only supports the creation of planar 3D curves.

All of these approaches provide valuable tools to ease the creation of 3D curves while promoting the user's creativity. However, as these structures are generally 2D surfaces, projecting strokes on them will not allow the creation of non-planar curves. In addition, most of these systems (except Mental Canvas) presume that the user has already settled on the design as they smooth the

input strokes to favor visual accuracy over the user's creativity. In addition, some recent immersive systems are based on the same paradigms as the one presented above: VRSketchIn [DGK*20] combines mid-air 3D sketching with a six-degree-of-freedom tracked tablet that serves as a support surface for more precise sketching; Hand Painter [JZF*21] allows the user to use the non-dominant hand as a canvas, onto which the other hand can project strokes. Finally, for some applications, it may be essential to rely on context to ease the creation of the desired design.

**Drawing in a pre-defined context**

In this part, we focus on context-based systems that use external sources as a guide for the creation. In particular, this guide can be 2D inputs such as images but also a 3D mesh, from which we can extract canvas. As context-based systems are intensively studied and especially in immersive environments, we provide a brief overview of the main approaches or contexts before presenting in more detail the few methods that specifically tackle design sketches in architecture (in addition to Mental Canvas).

*Brief overview*

Existing methods have explored various contexts and sketching paradigms. For instance, the system NapkinSketch [XSS08] restricts the design space to a physical napkin on which the user will anchor support surfaces to create a 3D sketch in an augmented reality system. Another approach consists in using a 2D input as a reference image. LiftOff [JK16] is a virtual reality system that lets users import a 2D sketch as the background image to help them lift their strokes in 3D. Quite similar to 3D modeling, the system Sweep Canvas [LLZ*17] uses an RGB-D reference image inside which the user can create free-form extrusion surfaces which are rendered only by a sketch outline. In contrast, Model Guided 3D Sketching [XFZ*19] uses a 3D model as a reference and also as a pre-defined exterior scaffold on which the user can project strokes.

In contrast to previous approaches, the OverCoat system [SSGS11] extends the 2D painting metaphor to 3D by allowing the user to choose a 3D texture effect and apply it interactively to a 3D model. In the same mindset, the authors of the recent Mid-Air Curves system [AS21] look at the texturing of virtual objects using a virtual environment and how to improve the 3D projection of the strokes.

On the other hand, Sketching With Hands [KB16] relies on a bi-manual interaction in which a virtual model of the user's non-dominant hand is used as the context of creation but also as a guide for creating planar surfaces on which strokes can be projected. Finally, an alternative to VRSketchIn [DGK*20]+ but in augmented reality and for the context of physical objects,

SymbiosisSketch [AHKG*18] provides a mix of 3D drawing in mid-air and surface interaction on a tablet. However, this combination is hardware-heavy, and switching between drawing modes is not intuitive for users.

*Design sketches in architecture*

The pioneering work of Mental Canvas [DXS*07] inspires the development of a digital sketching system dedicated to architectural design. For instance, its canvas concept is first extended to allow the integration of images, in addition to user strokes, in the context of a cultural heritage application [CMH*10, RLT*17]. Another extension, called Insitu [PKM*11], focuses on conceptual design in a target 3D environment, which can be natural or man-made. As illustrated in Figure 2.25, a site representation is created by merging several types of data (elevation map, photographs, aerial map, site map), while a user interaction similar to Mental Canvas [DXS*07] allows conceptual design directly in-situ.

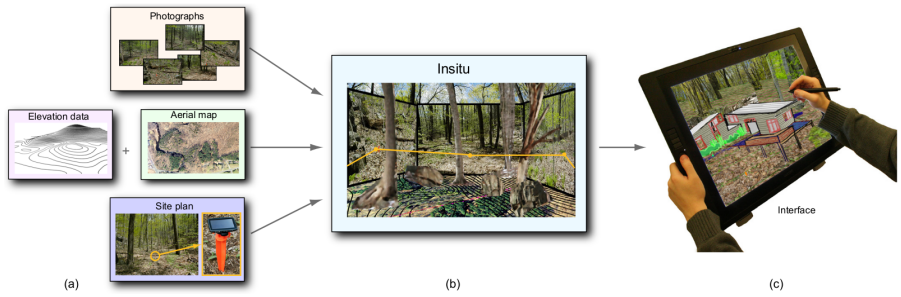



**Figure 2.25:** Insitu's overview [PKM*11].

In contrast, Smart Canvas [ZLDM16] allows the user to draw directly on a reference 2D image or sketch from which vanishing lines are extracted. The user's 2D strokes are dynamically interpreted and sorted into co-planar groups from which 3D polygonal surfaces are extracted by optimization and based on adjacency relationships. The user can accept, modify or create new relationships at any time. The model is rendered in a non-photorealistic way to make it look like a drawing, but the user's original strokes are not retained.

Being able to design an architectural model in context is a real advantage to foster the creative process. However, the presented systems rely on planar support structures that can limit the design of free-form 3D strokes. In addition, they only address the design of the exterior of an architectural model.

**Figure 2.26:** Concept developed by a practicing architect on Smart Canvas [ZLDM16].

### Discussion

Through this detailed section on related work, we have presented an overview of existing methods using classical, deep learning, and immersive techniques for both sketch-based 3D modeling and 3D sketching.

**Original strokes:** In our desire to encourage the user's creativity and relative to the identified characteristics of the sketch, we can notice that most approaches use clean drawings or apply a pre-processing spline conversion to the user's hand-drawn strokes. While we agree that hand-drawn strokes can be noisy due to the chosen input device, such smoothing steps can also polish the user's strokes and erase some information that is all the more important during the early stages of design. In particular, none of the existing methods exploits over-sketching or lighter strokes area as means of alternative options.

**Nested structures:** Moreover, while sketch-based modeling techniques have been intensively studied, none has yet address the interactive creation of nested structures without any constraint on the order of creation. In particular, for some applications, being able to represent such structures is essential to the creation process.

Now that we have explored the existing tools for creating shapes, we want users to be able to populate its 3D scene from a small sample of elements either in 2D or 3D.

## 2.2 Synthesizing a distribution of objects

From tree barks, brick walls, or schools of fish, to fields of collagen fibers and arrangements of cells inside a vessel, the world is filled with diverse distributions of shapes or textures at various scales, as depicted in Figure 2.27. However, reproducing such distributions manually or even virtually is a time-consuming technical task that requires good artistic skills. Moreover, any change on the output size or some part of the content can result in starting the process all over again.

**Figure 2.27:** Examples of common distributions of texture and elements. From left to right: a tree bark; a brick wall with windows; school of fish; vessel network and red blood cells.

A good alternative to this tedious approach is to focus the creation on a small sample, representative of the desired distribution, and then rely on computational algorithms to generate an output, visually similar to the provided example. In particular, this visual similarity can be characterized first, by some general visual constraints on the resulted output, such as avoiding artifacts or unnatural salient repetitions—even in the case of synthesis in an extended output domain—and second, by matching some proximity criteria with the input, determined using statistical approaches or perceptual validation. This process, also called example-based synthesis, consists of two main parts: an analysis of the input, in which the main features of the input are extracted; and a synthesis in an output domain, based on the result of the analysis. Example-based synthesis methods can be classified into two broad categories, depending on the type of input which is either an image (Section 2.2.1) or an arrangement of discrete elements (Section 2.2.2).

### 2.2.1 Texture synthesis

Texture-based synthesis methods take as input an image composed of one or more patterns or structures and aim at generating a new texture image that matches the provided sample. The reader can refer to the survey of Wei et al. [WLKT09] for more details on early methods addressing this type of synthesis. Existing methods can be classified according to the desired degree of similarity between input and output, i.e., whether the goal is to match some neighboring properties or a statistical model.

### Local similarity

*Neighborhood search process*

A first approach is to decompose the input into entities such as pixels [EL99, WL00] or patches [EF01, LLX*01, KSE*03], and rearrange them on an output image to generate a new texture, considered perceptually similar to the input one. These techniques generally assume that the input texture can be viewed as a realization of a local stationary random process, also defined as a Markov random field process. Thus, a new texture is generated iteratively by successive neighborhood search mechanisms on the input image. More particularly, the synthesis process starts by randomly picking a seed (a pixel or a patch in these cases) from the input image to position it in the middle of the output image. Then, the neighborhood of this seed is analyzed on the input image to determine the possible entities that can be copied around it, on the output image. The final output is thus generated progressively by selecting an entity, positioning it on the output image, and searching for its neighborhood in the input image to find the next most plausible candidate. The main challenge of this process lies in determining an adequate neighborhood size that maintains the input perceptual properties while avoiding too regular patterns. In addition, the improvement of patch entities has made the partitioning of the input texture more complex and introduced the need for smooth stitching between these patches. However, it highly improved the computational cost during the neighborhood process while ensuring the preservation of the input structures, as illustrated in the first three results in Figure 2.28.

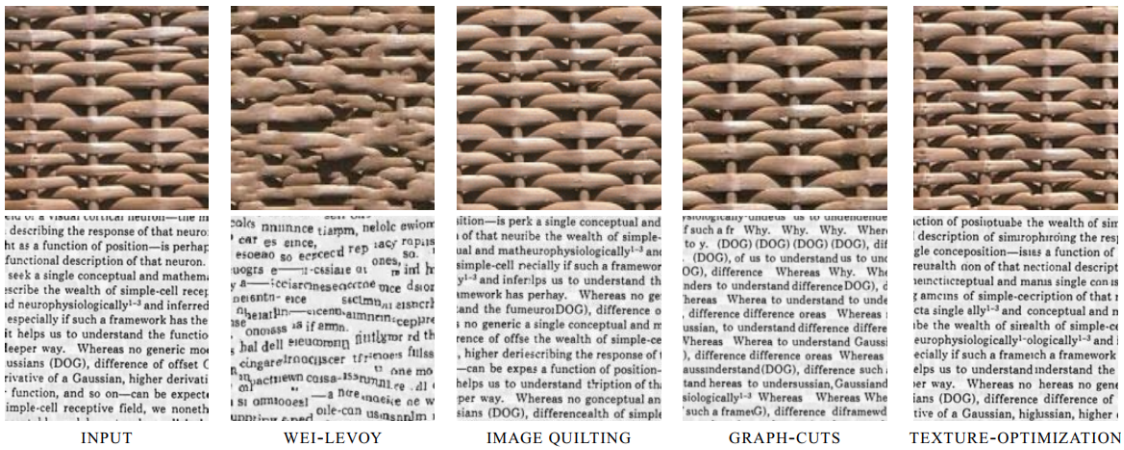



**Figure 2.28:** Comparisons between texture synthesis techniques. From left to right: input model; pixel-based synthesis [WL00]; patch-based synthesis using image quilting [EF01]; patch-based approach using a graph cut technique [KSE*03]; texture optimization [KEBK05].

*Local optimization*

In contrast to this region growing mechanism, some methods approach texture synthesis as a local optimization problem. Early methods [KEBK05, WSI07] introduce this process as a hybridization of the pixel and patch approach, defining a similarity measure based on the Euclidean distance between the color of two pixels and minimizing these values over a set of neighboring pixels (or patch) during the optimization. As shown in Figure 2.28, Kwatra et al. [KEBK05] provide an output texture perceptually comparable with the patch-based methods ones. However, the patch-based and optimization results can highlight some undesirable repetitions, in addition to not preserving the multi-scale texture details, since the similarity optimization and neighborhood search are applied at a patch scale. To counter these issues, Kaspar et al. [KNL*15] extend the similarity measure of Kwatra et al. [KEBK05] to handle large-scale structures, avoid repetitions and preserve regular and quasi-regular patterns (see Figure 2.30 top left). While Kaspar et al. still base their model on optimizing overlapping patches by calculating the similarity between a source and a synthesized pixel, they compute the Euclidean distance on both the color and a guidance channel. In particular, this channel provides additional information by storing the distance to the nearest feature. In addition, their method supplements this distance with a global histogram matching constraint. Finally, the authors implement a fully automatic initialization strategy based on a self-similarity analysis of the exemplar.

As described in the last approach, applying local similarity on pixel colors alone is not sufficient to preserve all the input information while avoiding repetition. In particular, the texture optimization methods rely on a global analysis of the source image. In this sense, an alternative direction, also widely tackled in the related work is to compute a more global characterization of the input sample, by a signature model, in the sense that all the texture images matching this model would be considered visually similar to the input.

**Statistical characterization**

*Histogram matching*

Based on perceptual studies, Julesz [Jul62] presents the concept that two textures with the same statistical characterization can appear visually similar to a human. His model, based on a pixel-based approach, represents the signature of a texture by its N-th order joint histogram of pixels. Following this work, studies on early vision and texture perception [BA88, MP90] try to discriminate textures by more global approaches, such as applying a series of convolutions with linear filters [FS89]. In particular, Heeger and Bergen [HB95] propose a two-step method to transform a noise image, into a texture considered similar to the reference one by alternately matching the histograms of the two images as well as of their multi-scale image pyramidal

representations. This process is then generalized and improved by Portilla and Simoncelli [PS00] by matching statistical constraints on both the pyramid and the image, instead of histograms. However, this structure is not sufficient to capture the features of natural textures.

*Deep learning approaches*

More recently, Gatys et al. [GEB15] improve the classical methods by using deep learning techniques. Unlike previous approaches, the authors take advantage of the feature space provided by the VGG-19 network, a convolutional neural network that has been trained for object recognition. During analysis, their method describes the input image by a set of Gram matrices containing the activations of an image at each layer of the network. Then, during synthesis, the system successively updates the pixels of a white noise image to match the activations of the input texture. This method is then improved by Sendik et al. [SCO17] to deal with structured textures, by introducing structural energy to identify regularities in a texture.

The other network commonly used in texture synthesis methods, as well as neural style transfer, is the Generative Adversarial Network (GAN) [GPAM*14]. Unlike a simple convolution network, such as the VGG-19 used in the above methods, which is trained beforehand and attempts to reduce the loss between a reference image and a progressively updated white noise one, the GAN consists of a generative network whose purpose is to synthesize an output, in addition to a discriminative network that determines whether or not this generated output is real. Since this network is unsupervised, its robustness in generating various outputs depends first on the diversity of the training textures, but also on the convergence of the network, in the sense of identifying diverse features without overfitting. For instance, Zhou et al. [ZZB*18] use a GAN to synthesize non-stationary textures in an extended domain. To achieve this, they take a database composed of non-stationary textures and train their GAN on $k$ x $k$ patch samples of the same image, allowing them to tune both their generative and discriminative networks by comparing the resulted output with the ground truth using a pre-trained VGG-19 network, as illustrated in Figure 2.29. Although one important limitation of their network is the need to train a dedicated generator on any new input image before it can be synthesized.

Figure 2.30 presents an overview of the best existing methods. In particular, it highlights the potential of the optimization framework of Kaspar et al. [KNL*15] for structured textures, but also the limitations of their patch-based optimization for anisotropic and non-stationary structures. Furthermore, it emphasizes the importance of using a complex neural network such as a GAN to synthesize various textures.

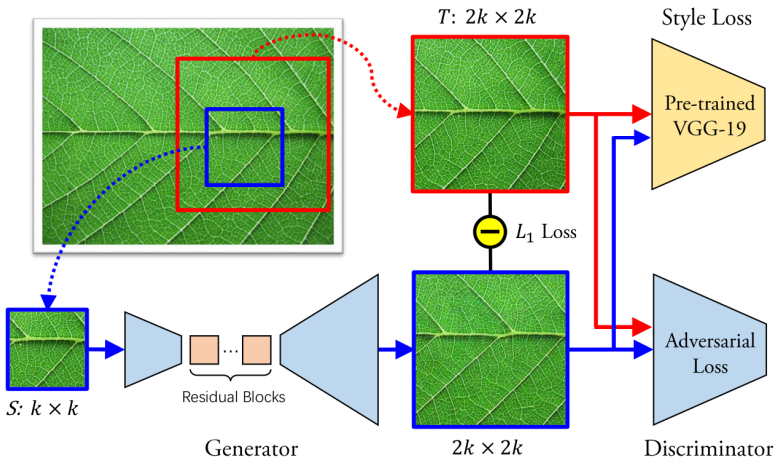



**Figure 2.29:** Overview of Zhou et al.[ZZB*18] method. The generator learns to expand a *kxk* texture blocks into 2*kx*2*k* ones using a combination of adversarial loss, $L_1$ loss and style loss.

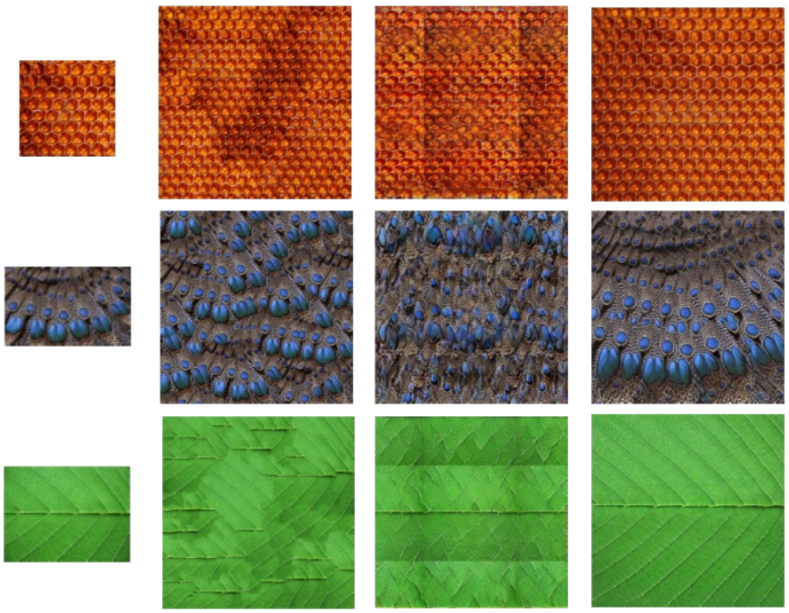


**Figure 2.30:** Comparison of the main approaches: (left) input; Optimization-based texture synthesis [KNL*15]; deep learning with structural energy [SCO17]; GAN approach [ZZB*18].

Although these methods show promising results in capturing, at least partially, local and global correlations in an input exemplar, they are limited to image-based input, and they do not extend to discrete element distributions. However, when it comes to the synthesis of the latter, the shapes presented in the input will necessarily be replicated (like the first approach in texture synthesis [EF01, KSE*03, KEBK05]) but following some model or characterization as introduced in the second approach.

### 2.2.2 Discrete shape distributions

Discrete element or example-based synthesis methods take as input a set of discrete shapes positioned in space and aim at generating a new set of these elements that can be considered visually similar to the input one. Because point distributions are composed only of identical, simple, and small shapes, we have separated the approaches that address these distributions from those that deal with more general shape arrangements. Although some post-processing renderings can be necessary to position the shapes at the location of the generated points, the synthesis methods do not present the same challenges.

#### Point patterns

Since point distributions are of significant importance in various fields of Computer Graphics such as rendering, sampling for anti-aliasing, or physical simulations, the synthesis of such distributions has been widely studied in recent years. In particular, one common and validated assumption is that any point distribution is built on a random structure in addition to relying on some rules. This has therefore motivated the use of statistical measures to discriminate an input. Moreover, since these distributions can generally be defined by the spatial correlations between point samples, synthesis methods mainly rely on stochastic point process techniques. The reader may refer to the course of Öztireli and Singh [OS18] for a detailed explanation of point processes and stochastic techniques and their various uses.

Point-based synthesis can thus be characterized by two main steps: an analysis of the input point distribution to identify a stochastic model defining the provided distribution; a synthesis of this distribution based on a random point process model and Monte-Carlo techniques but adjusted to the defined model. In what follows, we will briefly present the main approaches and recent improvements in point distributions synthesis.

*Sampling and frequency measurements*

Initially intended to solve anti-aliasing problems [Coo86], blue noise distributions have been intensively studied because of their recognizable signature in both the frequency and spatial domain. In the frequency domain, such distributions are characterized by a lack of low-frequency energy and an absence of structural bias. In the spatial domain, the point samples are randomly and uniformly distributed in space while guaranteeing a minimal pair-wise distance. Although alternative approaches have been explored, dart-throwing remains a reference to creating such distributions. An output is progressively created by iteratively generating a random position in space and checking if a new point sample can be positioned there while respecting the distance constraint. This approach can be replaced or complemented by a Lloyd relaxation. While many works addressed the isotropic property of such distributions, Li et al. [LWSF10] introduce the

possibility of dealing with anisotropic structures using a warping or sphere surface analysis (see Figure 2.31). The reader may refer to the survey of Yan et al. [YGW*15] for a general overview.

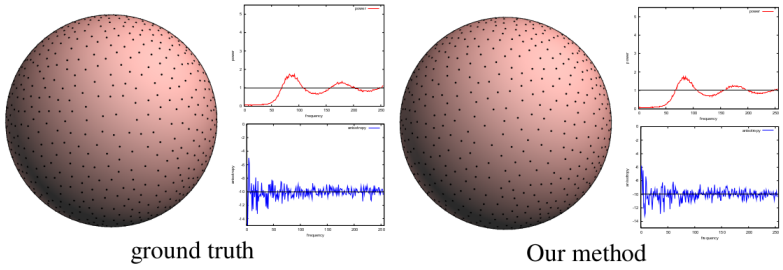



**Figure 2.31:** Sphere surface sampling and spectral analysis via spherical harmonics [LWSF10].

Although blue noise distributions synthesis methods can provide statistical accuracy, as well as handling isotropic and anisotropic distributions, these techniques highly depend on the blue noise property of the input distribution.

*Pair Correlations functions*

As an alternative approach, Öztireli and Gross [OG12] extend the dart-throwing approach to more general stationary distributions. In particular, they introduce a new distribution model defined by a pair correlation function. This measure encodes the perceptual features of the distributions into a normalized continuous function, in addition to providing a visual and interpretive signature, as illustrated in Figure 2.32. In particular, this measure characterizes a distribution by its point density relative to the distance of one point from all others. During synthesis, a generalized dart-throwing algorithm creates a new output whose PCF is closed to that of the target. Gradient descent is then applied to the generated point samples to minimize this difference. Roveri et al. [ROG17] first extend this approach to more general distributions such as local stationary processes and spatially varying correlations. Then, Ecormier et al. [ENMGC19] introduce the first generalization of PCF to disks that can have controllable overlap.

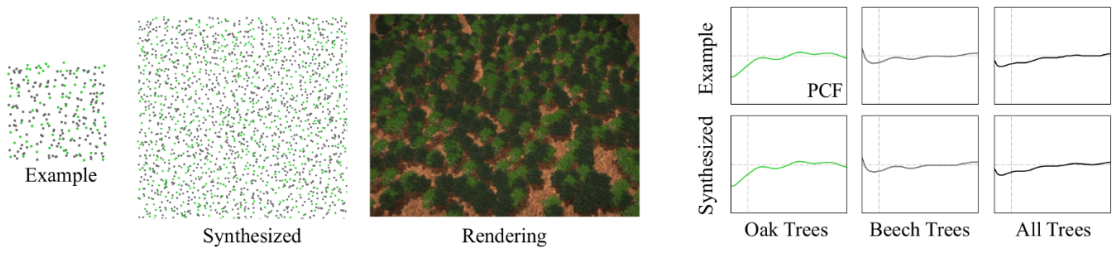



**Figure 2.32:** Example-based distribution synthesized with the PCF approach [OG12].

Although pair-correlations functions provide a new statistical model to characterize the point distribution, this model is only based on pair-wise distance, so it is restricted to isotropic

distributions. Another very important limitation of PCF-based techniques is that they cannot capture structures of the distribution (see Figure 2.33), which is a central property we target for the synthesis of anisotropic distribution.

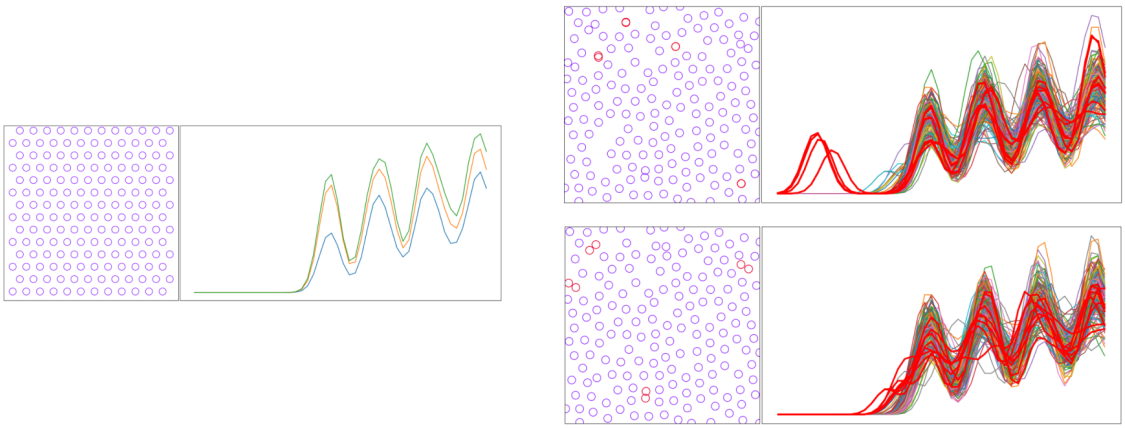


**Figure 2.33:** Failure case of PCF-based approaches on a hexagonal grid taken from [ENMGC19]: (left) exemplar input; (right) standard disk synthesis (top) and synthesis with outlier removal (bottom). Disks with the most outlying PCFs are colored in red at each stage.

*Deep learning approaches*

While deep learning techniques have been widely studied for texture-based synthesis, only one method tackles point-based synthesis. Inspired by texture-based deep learning techniques [GEB15, SCO17], Tu et al. [TLH19] apply an irregular convolution layer on a set of input and default output points to generate feature maps on a regular grid, which can be fed to a trained VGG-19 network. Then, their method converts the output back to the corresponding set of 2D points before updating the convolution layer weights in a coarse-to-fine manner and applying an additional optimization constraint on the output. Then, their system iterates this process while applying end-to-end optimization. As shown in Figure 2.34, this method provides an interesting optimization approach while taking advantage of the benefits of a neural network. However, their current pipeline prevents the synthesis from being real-time, in addition to not preserving visually perceptible shapes in the input.

In addition to standard point-based synthesis, Leimkühler et al. [LSM*19] focused on designing point patterns by proposing a deep architecture to learn a distribution on the fly from a user-specified loss. During training, the parameters of the network filters are updated to fit the desired distribution by minimizing the loss over a set of random points. During deployment, the desired distribution can be generated from a new set of random points.

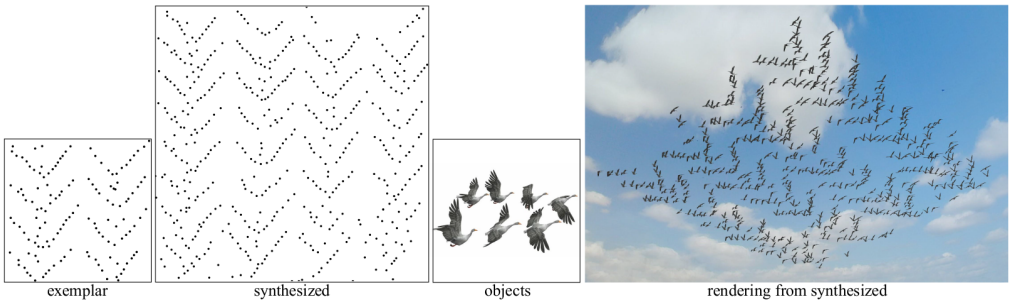



**Figure 2.34:** Point pattern synthesis via Irregular Convolutions [TLH19]: (left) input and synthesized data; (right) post-processing object placement.

### Discrete element distributions

This part focuses on element distributions, i.e., vector textures formed by arrangements of discrete elements. Unlike point distributions, these distributions are less constrained to stochastic approaches, however, the discrete elements composing the input distribution can be of various shapes. In addition to avoiding inter-penetrations, methods in this category must treat each shape individually and as part of the underlying distribution.

The methods belonging to this category consist of two steps: an analysis of spatial correlations between shapes; and a synthesis process that preserves input shape distributions in addition to validating the general visual constraints (no self-intersection or undesired repetitions).

*Centroid-based approach for shape distributions*

The first approach consists in generating distributions of discrete elements by simplifying the shapes into their centroids. The analysis is then computed on the point distributions and the synthesis is decomposed into two parts: generating the centroids and then retrieving the associated shapes. The pioneering work of Barla et al. [BBT*06] focuses on the synthesis of stroke patterns, as illustrated in Figure 2.35 left. Their method begins by grouping input strokes into elements using a user-specified pattern size before determining the connectivity of the distribution through Delaunay triangulation on the centroid of the elements. Next, their system synthesizes the point distributions using Lloyd's method [Llo82], and then the shapes are recovered using a combination of partial neighborhood comparison and perceptual studies. Following a similar approach, Ijiri et al. [IMIM08] base their synthesis on successive point placement using local growth and followed by a relaxation of the currently generated point distribution (see Figure 2.35 middle). However, these two methods are respectively limited to quasi-uniform distributions or a search limited to only immediate (1-ring) neighborhoods.

To handle more general shape distributions such as those in Figure 2.35 right, Hurtut et al. [HLT*09] rely on stochastic point process approaches. During the analysis, they use perceptual

studies to create a histogram grouping together the most similar shapes based on appearance constraints. Then, the authors make use of this structure to characterize the input distribution by a probability density function, part of a Gibbs point process. Their method generates a new arrangement using Monte-Carlo chains adapted to their model. As an extension to the synthesis and editing of virtual worlds, Emilien et al. introduce WorldBrush [EVC*15], an interactive system on which the user first defines a scene sample composed of predefined elements and whose distributions are encoded in a pipette. Then, the authors rely on painting systems operators to let the user brush, paint, and coherently edit virtual worlds. They take advantage of statistical and procedural models to encode both the elements' procedural parameters and their inter-relationships with and between categories of scene elements. In contrast to Hurtut et al. [HLT*09], in WorldBrush, the object types and categories need to be pre-set.

Simplifying the input shapes into points does not allow for the analysis of the correlation, orientation, or spatial placement of shapes. Therefore, it does not provide a model carrying all the input information. For instance, in the case of elongated shapes, inter-penetrations cannot be avoided. In addition, none of these methods can analyze and synthesize structured input.

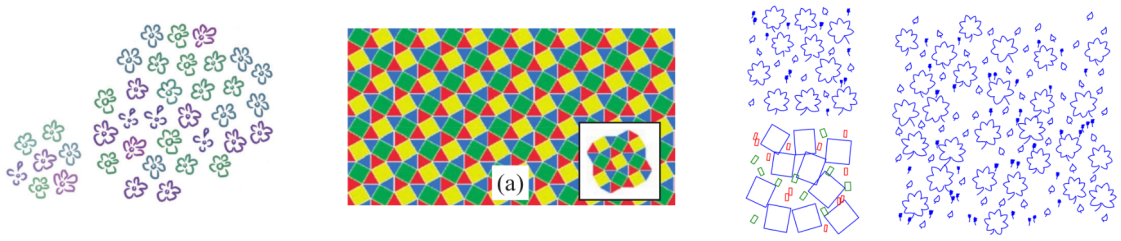



**Figure 2.35:** Overview of the centroid-based approaches: (left) Barla et al. [BBT*06]; (middle) Ijiri et al. [IMIM08]; (right) Hurtut et al. [HLT*09].

*Multiple points synthesis*

Rather than using a single centroid point, Ma et al. [MWT11] represent each input shape by a set of sample points. Using a new neighborhood metric and an energy optimization process, they manage to insert individual shapes in a pre-defined output domain, as depicted in Figure 2.36 left. Their approach is later extended to dynamic textures [MWLT13], stroke auto-completion [XCW14] and adapted to other texture workflows [KIZD12, DSJ19]. While the use of multiple point samples enables to broaden distribution synthesis to arbitrary shapes, these methods tackle bounded elements only—as opposed to unbounded, fiber-like shapes—and require some post-treatment to avoid inter-penetrations at the synthesis stage, preventing their real-time use.

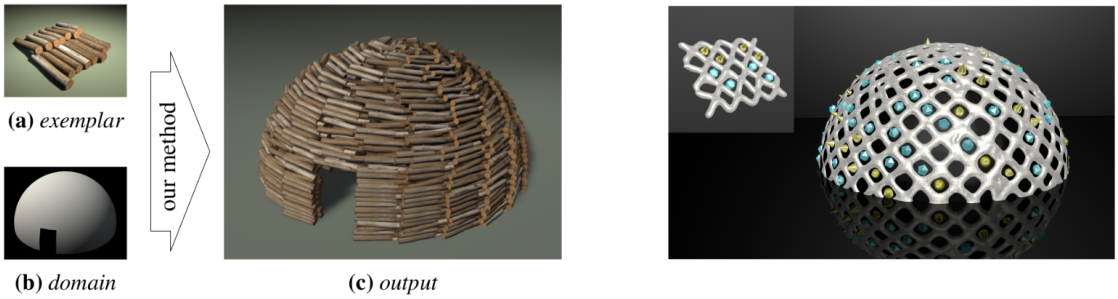



**Figure 2.36:** Multiple-points based approaches: (left) Discrete Element Textures [MWT11]: (left) from a small input and a user-specified domain, their method synthesizes the following output; (right) the synthesis of mixtures of discrete elements (gems) with continuous structures in [ROM*15].

In contrast, Roveri et al. [ROM*15] introduce the first example-based distribution synthesis method applicable to both bounded and unbounded shapes (see Figure 2.36 right). The authors decompose each shape—no matter its dimension—into point samples that are encoded into a functional representation. They define a similarity measure in the associated functional space to quantify the similarity between the input and output. The synthesis is achieved through neighborhood matching and energy optimization. In addition to not guaranteeing real-time synthesis, the main limitation of this method is the requirement of repetitive enough patterns to avoid bad local minima and thus distortion in the synthesized structures.

*Shape-aware distribution*

Rather than sampling the elements in the input exemplar, Landes et al. [LGH13] propose to simplify shapes into proxy geometries, as shown in Figure 2.37. They introduced a spatial relationship measurement that takes into account the inter-space between pairs of elements and their relative orientations. Extending stochastic models for point distributions [OG12, ZHWW12], their synthesis method successfully maintains both the distributions of distances and relative orientations of elements. Moreover, their method can generate 3D distributions from 3D input. Although it handles anisotropic distributions, their model does not meet our goals, since it does not offer real-time performances and is limited to distributions of bounded objects.

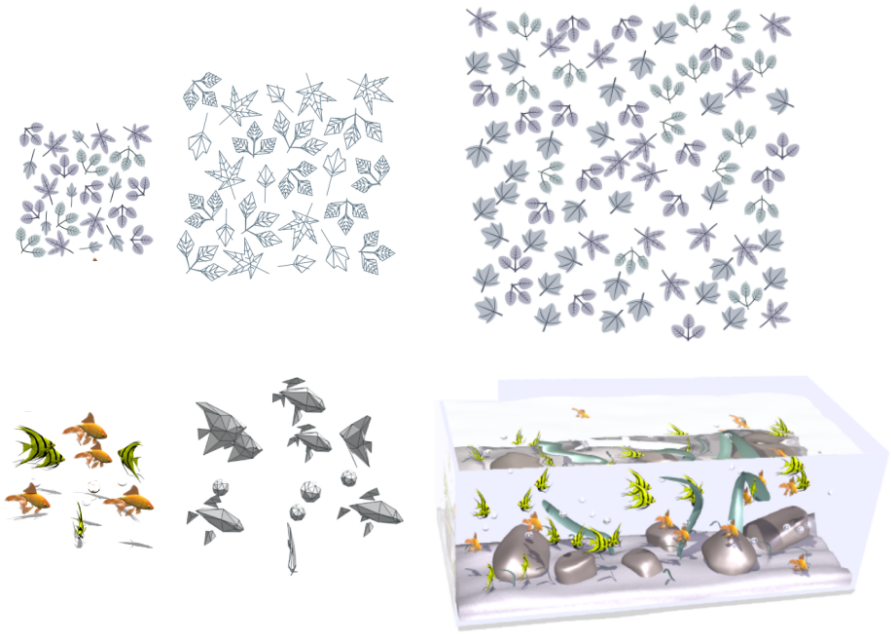


**Figure 2.37:** Overview of Landes et al. [LGH13] discrete model synthesis: (left) input; (middle) proxy geometry: 2D polylines or 3D meshes; (right) synthesized distributions.

### Discussion

Example-based synthesis has been widely studied for texture and discrete elements distribution. However, none of the existing methods handles the real-time anisotropic distribution of bounded and unbounded shapes from a sketch or the immersion of this distribution in a 3D environment.

While the best existing methods for texture-based synthesis rely on deep learning techniques, it will be difficult to train a neural network to interpret the shape distributions present in a sketch. Furthermore, they would not be easily extensible to the synthesis of 3D distribution.

Although point distributions synthesis methods offer statistical accuracy, this may come at the cost of losing real-time performance or visual structures present in the input. Although the few deep learning techniques show interesting results, they do not preserve the shapes that would be recognized by a human. Moreover, a single model [ROM*15] succeeds in determining an appropriate model for the distribution but requires repetitive enough patterns.

Finally, although current discrete element synthesis techniques provide appealing results on bounded and unbounded as well as for the 3D distribution of elements, only one addresses the anisotropy of the distribution, but only for bounded shapes.

## 2.3 Sketching to animate

Traditionally, an animation was entirely hand-drawn by artists, frame by frame. Based on the well-known principles of animation [JT81, Las87] and as shown in Figure 2.38, this creative process was either linear (*straight ahead*) or through keyframes and in-betweens (*pose to pose*). While the *straight ahead* approach can be considered the most creative and spontaneous, a certain lack of coherence can occur between the beginning and end of the animation. On the other hand, the *pose to pose* approach imposes to define all the important steps at first, which allows for better control of the animation as a whole.


**Figure 2.38:** The fourth principle of animation [JT81]: straight ahead and pose to pose, illustrations by ©Alan Becker.

Regardless of the chosen approach, hand-drawn animation requires a high level of artistic skill to generate the desired drawings but also to convey the rhythm of the animation from a set of still images and a timing chart. Moreover, drawing each image individually can be very tedious, especially since small changes, either in the visual characteristics of an element or in its motion will require redrawing some parts or even all of the drawings.

Recently, computer animation methods have attempted to facilitate this entirely hand-drawn animation process by providing tools to animate either an individual object or a set of elements. These techniques can be classified according to the level of creativity and freedom they offer. For example, parametric models such as procedural or physical-based systems [NMK*06, Bri15] have been intensively studied in the context of natural phenomena. However, they are highly dependent on parameters, which may also be difficult for the developer to estimate or for the user to adjust. Therefore, this indirect and not intuitive control is not favoring the creative process. An alternative, which is also standard in industrial animation software, is to let the user interact with an environment through direct manipulation. While it is easy to drag and drop an object in space, precise manipulation of a shape, which can also be complex, is more challenging. In particular, existing methods rely mainly on the manipulation of anchors positioned on a model. These can be located manually by the user [BKLP16, DSC*20] but also predefined by some support structures

such as a bounding cage or more classically the model rig. While the first case offers greater freedom of movement, extreme dragging may result in unnatural poses as the consistency of the geometry will not be preserved. On the other hand, the manipulation of predefined anchors can be considered too rigid since control is limited to the degrees of freedom of the handles. In addition, regardless of the chosen approach, it may take several iterations, if not impossible, to create a specific expressive pose. Unlike the pose, timing may also be difficult to estimate from user input, and relying on a fixed timeline [PWKK20] may be insufficient for an expressive animation. Therefore, a user may prefer to rely on sketching (or even combine it with direct manipulation) to ease the animation process. Indeed, as with sketch-based modeling, it may be more intuitive for the user to use sketch inputs to express animations but more computationally complex to represent the desired motions and deformations.

In what follows, we focus on existing methods for sketch-based animation of an individual object or a set of elements. Specifically, these approaches can be classified according to their use of sketching, which can serve to represent keyframes (Section 2.3.1), design the trajectory path of an individual object (Section 2.3.2) or characterize motion guidelines for a set of elements (Section 2.3.3).

*Brief overview of 3D character representation*

Before entering the heart of this section, we present a brief overview of the classical skeleton-based animation pipeline and, in particular, the various terms related to it. Since the geometry of some complex 3D models may contain a large number of vertices, a common usage in the classical animation pipeline is to rig this geometry by a 3D skeleton (or **rig**), i.e., a hierarchical set of **bones** connected by **joints**. In contrast, the geometry of the 3D model is called the **skin**. The main interest of this structure lies in the direct mapping (also called **skinning**) between the skin and the rig, allowing to deform a model from the rig, which has the effect of preserving the core structure of the model and avoiding to compute transformations at the scale of the detailed geometry. Indeed, each bone of a rig is defined by a set of transformations that specify its position in space. The use of skinning allows to compute the position of the vertices of the skin by blending the transformations of the bones that surround them. Finally, during an animation, the joint transformations are updated according to the process of **forward kinematics** or **inverse kinematics**. While forward kinematics computes the transformation by following the skeleton hierarchy, inverse kinematics, which is also the most frequently used, propagates the transformation in reverse order. To illustrate this concept, when moving the fingertip of a human avatar, forward kinematics first determines the transformation of the upper joints in the hierarchy, such as those connecting the shoulder to the arm, the arm to the elbow, etc., before reaching the target joint, whereas inverse kinematics first updates the bone selected to reach the target and calculates from it the appropriate intermediate transformations up to the hierarchy.

### 2.3.1 Sketching keyframes

As introduced previously, the *pose to pose* approach first consists in defining *keyframes* (the main steps of the animation) and then progressively smoothing the animation by the use of in-betweens. Related work aimed at keyframe animation mainly focuses on approaches that either infer a transition (or in-betweens) from predefined *keyframes* or represent key poses in a coarser, easier representation.

#### Inferring in-betweens from predefined keyframes

Closer to the traditional approach, this part focuses on determining in-betweens from already defined keyframes. As a shape can undergo deformations between two keyframes, a common process to smoothly move from one shape to another is morphing. This technique has been widely studied for images, 2D shapes, and even 3D meshes. In addition, the morphing process can generally be decomposed into two main steps: 1) finding correspondences between two successive keyframes, 2) determining a mapping function to create transition states (i.e., the desired in-betweens).

In what follows, we classify the approaches according to the type of data or structure used to create an animation from a set of keyframes.

*Image morphing*

Introduced by Beier and Neely [BN92], image morphing is still the subject of active research. Indeed, while the creation of transition states from defined correspondences is generally based on an interpolation algorithm, the automatic determination of such correspondences from images is a challenge. For instance, the first methods (see the survey by Wolberg [Wol98]) rely heavily on user intervention which does not scale to entire animations.

As a solution, Shechtman et al. [SRAIS10] introduce the concept of regenerative morphing. Inspired by texture-based synthesis methods, the authors bypass the correspondence step by generating their intermediate shapes as a combination of patches already present in the source images. They define their morphing as an optimization framework whose objective is to maximize each generated image, its visual similarity with the source images, and its temporal coherence with the direct neighbors while allowing some variation. Using a hierarchical Gaussian pyramid on an initial set of images, their model optimizes the intermediate images in a coarse-to-fine manner. To preserve visual structures during the transition, Darabi et al. [DSB*12] has extended the previous similarity distance to take into account local gradients in addition to patch color values. Furthermore, Browning et al. [BBRF14] has adapted this approach to obtain a resulting

motion guided by a predefined simulated flow (see Figure 2.39).

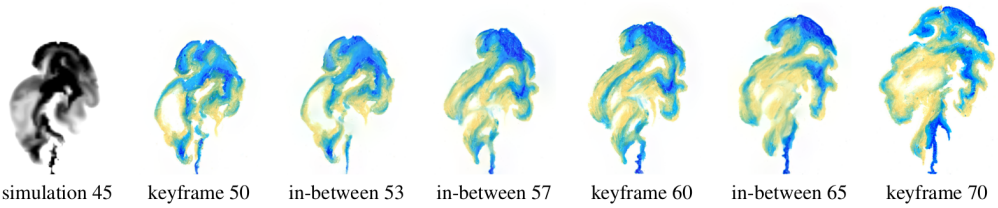



**Figure 2.39:** Overview of [BBRF14]: (left) an underlying simulation is used to guide the artwork; (right) Frames 50, 60 and 70 are hand-drawn by an artist, while the remaining in-betweens were automatically generated by their system.

Other alternatives have been explored, such as estimating these correspondences by finding a linear path that maps a pixel of one image to the other [MHM*09] or using a 2D vector field over a domain halfway to define the desired map by convolving two halfway mappings [LLN*14]. Zhu et al. [ZLWH16] successfully determine region correspondences by optimizing a network flow graph but they rely on a full animation sequence as input, which is generally not the case in image morphing. Finally, Li et al. [LZLS21] let the user provide a sketch to guide the animation between two keyframes. They base their model on a neural network to estimate the correspondences between the animation and the sketch on a cross-domain.

These methods only characterize the input images at the pixel level. However, this representation is not adapted to depict 2D shapes precisely. Indeed, taking directly the shape itself as input allows for a richer description of its topology or geometry. In what follows, we mainly separate the approaches that take into account the boundaries of the contour from those that focus on its interior. Moreover, as shape manipulation was intensively studied, we only describe some significant approaches.

*2D Shape interpolation*

A first approach consists in interpolating the input shapes by their contour. This representation is very useful to have information about the topology of a shape and thus avoid the generation of invalid in-betweens (self-folding, etc.) and can even allow the morphing of topologically different shapes. For example, Kort [Kor02] rely on heuristics and rules to first analyze the components of the shape sketched by a user but more importantly to determine whether a stroke in a drawing can match another one in a second drawing. In contrast, Whited et al. [WNS*10] target the specific case of tight keyframes and rely on a stroke graph to represent a shape. Their system finds the correspondences between two shapes by simultaneously traversing the graph of each shape and checking for similarities. As an extension of this stroke graph to animation, Dalstein et al. introduce Vector Animation Complex [DRvdP15], a data structure to replace the sequential

keyframe process with a topological one that promotes morphing with time-varying topologies, such as the example shown in Figure 2.40.

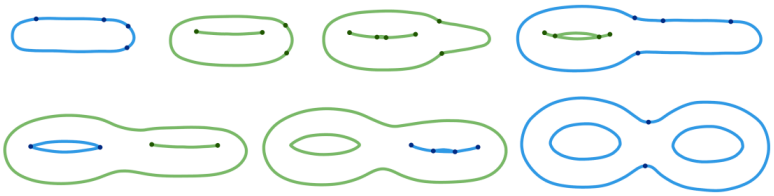


**Figure 2.40:** Example of morphing from a torus to a double torus in [DRvdP15].

The other current approach is to characterize a shape by its interior instead of its contour. While time-varying topology morphs are no longer valid, this definition allows for the preservation of the geometric structure of the shape.

For instance, Alexa et al. [ACOL00] propose a shape interpolation model that is as-rigid-as-possible, i.e., morphing one shape into another in the least distorting way possible. As the focus is on rigid transformations only, the authors assume the correspondences between the contour of two shapes to be already predefined by a bijective map. Their method defines the interior of each shape by computing its Delaunay triangulation followed by some optimization to preserve the isomorphism of the map and thus find the correspondences of the shapes at the vertices of the triangle level. To obtain their as-rigid-as possible shape interpolation, the authors first determine the optimal least-distorting deformation between two matching triangles to obtain a local ideal model. They then define an affine mapping between the two shapes as the default global model. Finally, their method characterizes the interpolation as the minimization of a quadratic error function computed from the difference between the local ideal of each triangle and the default global models. By expressing this function as a matrix and discretizing it in time, the authors define a closed-form vertex path. As shown in Figure 2.41, they can apply their technique to various inputs. Their as-rigid-as possible concept is then extended to shape manipulation [IMH05], image registration [SDC09] and used as the basis for a rigidity-preserving layered deformation model [DSC*20].

**Figure 2.41:** Application of ARAP interpolation [ACOL00] through the morph of the photograph of an element and into the photograph of a giraffe.

The alternative representation to preserve the internal structure of a shape is to extract its skeleton and apply the morphing to this simplified structure before recovering an in-between. Initiated for 2D polygons [SR95], it is then intensively used for the morphing of implicit surfaces [BBB*97]. In particular, Galin et al. [GLA00] extend this concept to the metamorphosis of the Blobtree. The latter structure is defined as an implicit surface composed of skeletal elements or referred to as blobs. This surface is represented by a scalar field generated by the sum of the contributions of each blob. Moreover, this scalar field can be defined by the convolution of a field function with a distance to a skeleton. Therefore, the authors characterize each blob $b$ by its skeleton, its distance function, and its field function. To morph an initial model $(A)$ into a final $(B)$ one, the authors first let the user define a coarse correspondence between the elements of each shape and a refinement step, split some components into sub-entities to obtain a bijective graph. Then, their method defines a time-varying Blobtree by the evolution of generic components $(g_i)$ associated with each matched pair $(g_i^A, g_i^B)$. Therefore, each skeleton, distance function, and field function of a generic component is defined from those of their associated pair. The time-varying skeleton, field function, and distance functions are then computed by the Minkowski sums or an adaptation of it of the initial and final parameters of the associated components. Figure 2.42 illustrates this morphing process.

**Figure 2.42:** BlobTree metamorphosis based on Minkowski sums [GLA00].

While contour-based approaches allow for topological changes during metamorphosis, using interior or skeleton-based techniques provide greater stability of surfaces and volumes. For instance, Zhu et al. [ZPBK17] extend the variational interpolation technique to handle topology changes by adding a CoMesh optimization framework. However, their method rely heavily on the user to specify cuts and correspondences between the keyframes which could both bring control to the artist but also be a drawback. In addition, although CoMesh optimization significantly improve the variational interpolation approach by refining the topological connectivity of the initial meshes, this operation is not real-time.

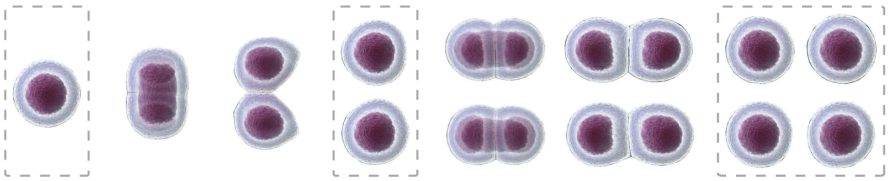


**Figure 2.43:** Overview of [ZPBK17]: interpolation of inbetweens (unboxed) from sparse key drawing shapes (boxed) across arbitrary topology changes and extreme deformations.

*3D mesh morphing*

As some of the methods presented above deal with the 3D case, we let the user refer to the survey by Alexa [Ale02] for an overview of the early methods addressing mesh morphing. In particular, for the 3D case, finding the proper correspondences between two 3D models is all the more challenging and constitutes a whole field of research, for instance, using functional maps [OCB*16].

More recently, deep neural networks have been used in the context of generating in-betweens from a set of keyframes by retrieving learned data [ZvdP18]. Furthermore, Harvey et al. [HYNP20] propose a neural-based technique to generate high-quality motion from only a few keyframes used as animation constraints. In particular, their model makes use of robust transition generators that adapt the in-betweens to variations in keyframes, as illustrated in Figure 2.44.

**Figure 2.44:** Overview of Harvey et al. [HYNP20]: From a few keyframes (in blue), the transitions (in brown) are automatically generated. For clarity, only one in four generated frames is shown.

The methods presented above take as input a set of keyframes and generate the missing in-betweens. Although several morphing techniques were explored, they are usually specific to a certain class of inputs or certain types of transformations. Indeed, some properties such as rigidity preservation and shape elasticity may be incompatible. Following this key-framing mindset, the next part focuses on intuitive tools that allow the user to directly sketch the 3D pose of a complex 3D model. In particular, in a combined framework, the user could design the 2D or 3D poses and then refer to, for instance, the methods presented above to create the underlying animation.

### Abstract representation of keyframes

For a complex and detailed object such as a 2D or 3D character, hand-drawing one keyframe after the other can be very laborious. For this reason, some attention was given to intuitive and abstract representations of key poses based on sketches especially, for 3D models. However, as with sketch-based modeling (see Section 2.1), it can be challenging to infer the desired 3D pose from 2D sketches. Although here the 3D model is already defined and provided, the goal is to first identify the user's desired pose from a sketched 2D abstract representation and then adapt the model accordingly.

The first approach consists in interacting directly with the model's geometry. For instance, Kho and Garland [KG05] propose an approach to deform a 3D model from two curves sketched in the screen plane (see Figure 2.45). The authors describe the first curve as the reference curve and use it to highlight the local region of interest (ROI) determined by a graph-cut partitioning. The second or target curve represents the desired deformation profile of this region. From an arc-length parameterization of the two curves, they define two 1D local frames on which they can express the projection of the vertices of the ROI and deduce their image on the target frame. Their method characterizes the deformation by the rotation angle between each vertex and its image. Then, they apply some post-processing on this resulting local deformation relative to the rest of the 3D model through optimization and adaptive refinement allowing for the generation of a smooth triangulation while preserving the fidelity to the target curve. The authors propose a sketch-based morphing by linearly interpolating the length and rotation angles on the reference and target curves to obtain intermediary curves. While other methods tackle mesh deformation by sketching curves or silhouettes [NSACO05, ZNA07], their model is more suitable for mesh editing as it can be more complicated to infer smooth animation between the original and deformed models.

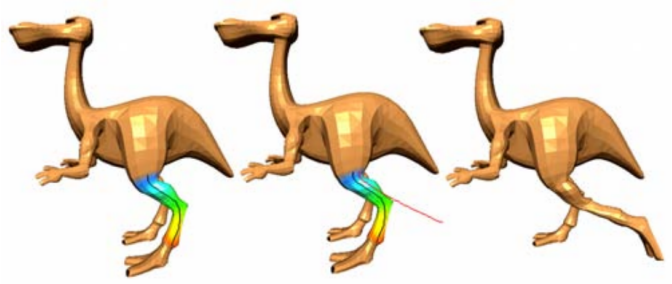


**Figure 2.45:** Sketch-based mesh deformation in Kho and Garland [KG05]: (left) the user draws a reference curve that is projected on the 3D model; (middle) then a target curve; (right) the leg is deformed accordingly.

Although this method allows for local deformations to be sketched directly onto a 3D model, there are no restrictions on the amount of deformation, which can lead to unrealistic poses. To counter this problem, other methods rely on the skeleton (or rig) of the model as an intermediary between the 2D sketches and the 3D mesh. These techniques can be classified into two main approaches, those that take a complete sketch as input and those that allow the user to sketch strokes directly on the 3D model (as was done previously).

*Sketch representation*

Several methods use sketched 2D stick figures as coarse representations for character poses. Inspired by the early stages of hand-drawn animation, this structure is simple and quick to draw. In addition, providing a correspondence between a 2D stick figure and its 3D representation can ease the posing of various models composed of the same rigging. However, as with any 2D data and especially with this representation, different 3D skeletons can be represented by the same 2D stick figures. Therefore, as with sketch-based modeling techniques, these methods usually refer to prior knowledge to determine the most plausible 3D pose. In addition, the following methods use only a human 3D model and thus rely on prior knowledge related to the human anatomy, such as the natural degree of freedom of joints or body balance. For example, Davis et al. [DAC*03] determine up to two potential depths for each bone relative to their 2D foreshortening, and used joint angles priors to eliminate invalid poses. Their system ranks the valid poses using an optimization framework constrained by a set of preferences over human priors before displaying the most plausible one to the user and offering some alternative choices in a thumbnail. The user can add further annotations to refine the desired pose. In contrast, Mao et al. [MQW05] target more natural poses by providing visual guidance to the user during the sketching phase and allowing over-sketching as a depth indication. Their method also automatically recognizes the correspondences of each sketched bone and thus their degree of freedom. Finally, as a tool to quickly generate motion sequences from rough and incomplete sketches, Choi et al. [CYI*12] generate a database of 2D figure poses supplemented by trajectory hints from original motion clips. Their system identifies key human body features from the input sketches to determine the closest sequence of motion in terms of trajectory correspondence.

While 2D stick figures may be encouraged for their ease in quick prototyping animation, they are very limited. First, as Figure 2.46 points out, they are inherently ambiguous and can generate multiple geometrically valid and plausible solutions. Second, it can be tedious, if not impossible, for a user to represent a complex skeleton or pose with this representation. Indeed, since it represents a high simplification of the final model, it can be confusing for the user to keep in mind the finer details of the real geometry. Finally, even if the resulting 3D pose matches the 2D sketched input, there is no guarantee that it is a valid 3D skeleton pose for a 3D model

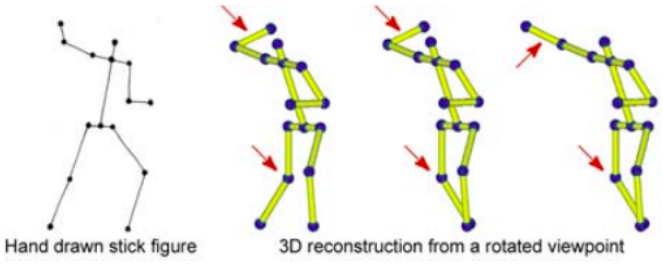



**Figure 2.46:** Overview of the 3D pose ambiguity from a single 2D stick figure [DAC*03].

containing details not accounted for in the 3D skeleton representation.

To address these issues, Bessmeltsev et al. present Gesture3D [BVS16], a 3D pose system that takes as input a gesture drawing. As illustrated in Figure 2.47 b) and e), this drawing represents the outline of the projection of a 3D character from a specific viewpoint.

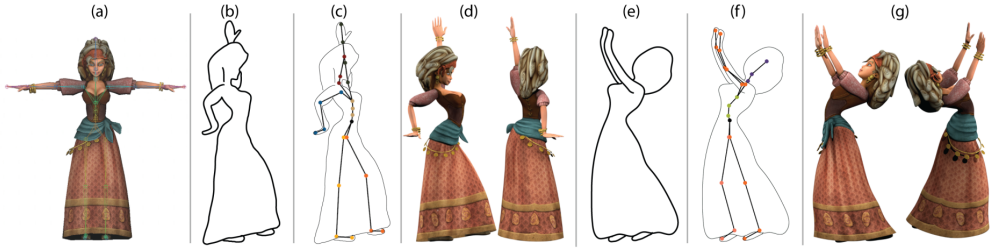



**Figure 2.47:** Bessmeltsev et al.'s Gesture3D [BVS16]: gesture drawings (b,e) of an input character model (a); the estimated 2D skeleton projections (c,f) and the new poses automatically computed from the drawings (d,g).

Based on the characteristics of the gesture drawing, a provided 3D model, and a gesture drawing, the authors determine the desired 3D pose in two steps. First, their system identifies the projection of the 3D skeleton pose corresponding to the contour drawing by computing the extended radius of the 3D skeleton joints. To do that, their method first considers each joint's environment in the 3D model to find their potential locations in the contour drawing by computing the probability of their projection being present within the drawing contour using the proportion of intersecting contours. Then, this first approximation is refined by optimizing an energy function composed of this likelihood, bone connectivity constraints, pose preferences, and constrained by a global consistency term over the entire skeleton. The authors rely on a discrete solver to minimize this function. Finally, a second optimization constrained the joint to their allowed degrees of freedom using a random walk and an ICP variant. As a second step, their system recovers the depths of each joint by relying on the characteristics of the drawing (simplicity, imprecision, regularity) to define an objective function constrained by the regularity and order of the joints. Following the kinetic approach, their method describes 3D positions in terms of twist coordinates and relied on the Taylor expansion to linearize the resulting expressions and define an optimization

framework as a sequence of constrained quadratic optimizations. Since their model depends heavily on the characteristics of the gesture drawing, it constrains the user to follow this design convention. In addition, their greedy discrete solver prevents real-time computation of their 3D pose system.

While 2D stick figures may be considered an oversimplification of the underlying model, gesture drawings may be too technical for novice users. Moreover, both approaches take a complete sketch as input and are therefore strongly dependent on the quality of this input. Indeed, in both cases, the representation of depth requires the shortening of bones, the quantity of which can be a bit difficult to estimate without skills or a good representation of proportions. To counter these problems, the following methods allow the user to sketch strokes directly on a provided 3D model.

*Stroke representation*

Offering the ability to sketch directly on the 3D model allows for the user to focus solely on the desired pose and not worry about the shape proportions or high drawing conventions. In addition, it is much easier to determine the correspondences between the 2D input and the 3D model or skeleton.

In the spirit of a local 2D stick figure representation, Wei and Chai [WC11] let the user locally deform a model by sketching a curve on the limbs or torso of the 3D model to specify the desired local pose. The authors define the deformation as a maximum a posteriori framework that estimates the most likely pose as an energy minimization combining likelihood energy measuring the similarity between the input and the parameters of the generated pose, and prior energy based on a prior distribution of poses to validate the degree of naturalness. In particular, they infer their prior distribution and generated pose from their data-driven model based on a mixture of analyzed factors and an Expectation-Maximization algorithm. Although their model preserves the naturalness of the generated pose, relying on a database restricts the range of available poses to the prerecorded ones.

Extending this representation to a non-data-driven model, Hahn et al. [HMC*15] base their approach on the concept of sketch abstraction, i.e., a set of rigged curves that form an illustrative 2D representation of the 3D model from a specific viewpoint (see Figure 2.48). These abstraction curves can either be provided beforehand or directly sketched by the user.

**Figure 2.48:** Sketch abstraction highlighted in red in [HMC*15].

In particular, their method transforms an arbitrary curve into a rig by computing its projection onto the 3D model and by calculating for each point of the curve its barycentric coordinates relative to the vertices of the intersection face. The authors characterize the curve both by these local coordinates and by the projection matrix of the camera. This description allows them to define the sketch abstraction as a deformation tool. Indeed, such an abstraction curve is by definition mapped to the 3D model geometry but this geometry is already skinned to a 3D skeleton characterized by a set of rig parameters, thus creating a mapping between the 2D sketch abstraction and the rig parameters. Given a deformation curve sketch on the 3D model composed of sketch abstractions, the authors follow the ICP approach to formulate the deformation using a matching energy defined as the sum of the weighted distance between the sketch abstraction and all the points of the deformation curve, with the weight representing the potential correspondences between a point and a sketch abstraction curve. This energy function is minimized by alternating between the calculation of weights and updating the rig parameters to deform the surface. To counter the under-constrained property of this objective function, the authors add three regularization terms to limit the amount of deformation, favoring deformation in the view-plane and favoring local deformation over global deformation. Through the use of sketch abstraction, this system is generic, flexible, and can accommodate any rigging parameters; however, the deformation is also severely limited by rigging limitations. In addition, the encoding is quite cumbersome, so increasing the accuracy of the deformation by adding multiple rigged curves can adversely affect real-time performance.

The two methods presented above define a local 3D pose process by drawing strokes locally on certain parts of the 3D model. However, they limit the movement to the degrees of freedom of the selected bones. Moreover, if the user wants to define a global pose, it can be tedious to draw several curves without the guarantee of obtaining the desired result. Therefore, to improve the abstraction of keyframing for more expressive poses, the two methods presented below rely on the line of actions for a global pose and secondary lines for more detailed and local poses.

Focused on motion expressiveness, Öztireli et al. [OBP*13] introduce a differential blending algorithm, extending the traditional rigid transformations to extreme deformations such as twists or

extreme bendings. The deformation process begins with the user selecting the relevant bones using the sketch as a selection tool and sketching a target curve, representing the desired deformation. This process can be repeated after each deformation. As shown in Figure 2.49, the authors provide two sketch interactions: the line of action, for a more global deformation, and the detailed bone shape line, for a more detailed deformation.

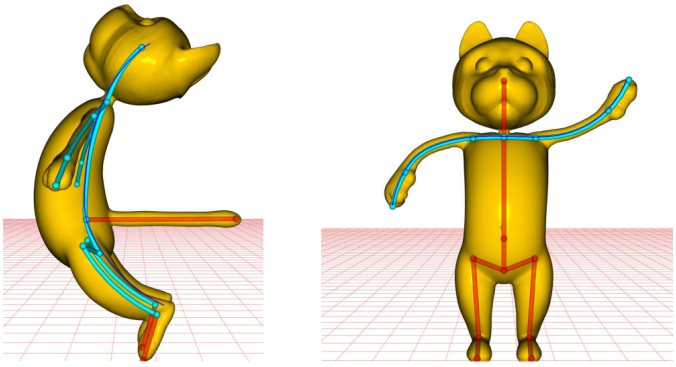


**Figure 2.49:** Differential blending skinning [OBP*13] allows for both line of action sketching (left) and detailed bone shape sketching (right).

The authors rely on arc-length cross parameterization [KG05] to compute the deformation between an input curve and the 2D projection of the relevant bones in the view plane. Since they aim for expressive and extreme deformations, they used curved bones (see the bending of the bones in Figure 2.49 left) that represent a set of continuous transformations. Their differential blending method relies on decomposing an extreme deformation into a set of smaller transformations, which are then blended with the shortest path method and compounded to get the final weighted average. In particular, the use of curved bones allows them to obtain a discretization of the bone space into smaller entities capable of storing local transformations. These transformations are updated relative to the user's sketch and used to determine the deformation of the mesh. Their method stores the differential transformations at each node along the path from the root to the current one. Their system computes the transformation of each vertex of the mesh by the linear interpolation of the transformation samples around it. The main limitation of their method is that the user must specify for each deformation the bones involved.

Targeting the pose of 3D articulated characters from a single line of action, Guay et al. [GCR13] propose a sketch-based interface on which the user can sketch a line of action (LOA), and the system automatically aligns the 3D model to fit this line, as shown in Figure 2.50.

In addition, the user can add secondary lines to refine the generated pose from other viewpoints. The authors define an LOA as a C- or S-shaped curve guiding the pose both in terms of position and tangents. Their method sets the deformation region by a bodyline, i.e., the maximum connected linear chain in the kinematic tree of a character's skeleton. Following their formal description

**Figure 2.50:** Line of Action [GCR13]: expressive character poses created in a few seconds each, by sketching intuitive lines of action.

of the line of action, their system retrieves the desired pose by solving an optimization problem. The strength of this system is that it allows to pose a shape using only one single stroke; however, LOAs are only limited to C and S-shaped. To simulate the motion of the line of action, Guay et al. [GRGC15a] present a physics-based line interpolation that guides transitions between two key poses (defined by the lines of action) while preserving bone constraints. To do so, the authors consider the 2D line of action as a piece-wise rigid chain with elastic behavior, which transitions from one key pose to another by forward simulation.

In all the presented approaches, the deformation/posing is performed relative to the view plane. While this eases depth inference, it does not allow for non-planar or more complex deformations. Furthermore, while these tools offer interesting solutions to facilitate keyframe creation, the inherent problem with this concept is that the focus is only on specific spatial features for a given time. Because of this decorrelation, the transition between two keyframes may lack rhythm, which is an important characteristic of expressive animation. A solution that has already been used by some methods could be to locally or globally define a timing curve or to use a gesture before transitioning from one state to another.

### 2.3.2 Sketching the trajectory path of an isolated object

Compared to the previous concept, the direct sketch of a motion path can provide information about spatial location and timing; therefore, it does not raise the same issues. Inferring a 3D motion from a 2D sketch, also called motion synthesis, consists of three steps: specifying the constraints (user input + external such as contact), generating the motion, and applying some post-processing steps, e.g., to smooth the animation or enforce some constraints. In particular, in the following methods, the complete motion is specified by the user at once and the algorithm infers the motion from all data.

*Data-driven model*

The first approach consists in using preregistered data and defining the motion synthesis problem by a search graph or gesture recognition.

To provide a flexible tool to generate various human locomotion from a user sketch input, many methods rely on motion graphs. The main idea is to record a set of human locomotion clips (see Figure 2.51 top) belonging to the same model in a graph, compute potential transitions between these clips, and generate a motion sequence that matches the user's intended path. Although existing methods change the type of graph, the database, or used different similarity functions, the motion synthesis is primarily defined as a graph walk. For example, Kovar et al. [KGP02] base their method on the directed graph structure. Their system computes the transitions from point clouds of the character's position between two images using a weighted sum of squared distances and a branch-and-bound search. Their search approach is very effective for constrained motion such as the sketched trajectory, however, using the directed graph can lead to a large and unnecessary amount of search in the case of a rich dataset.

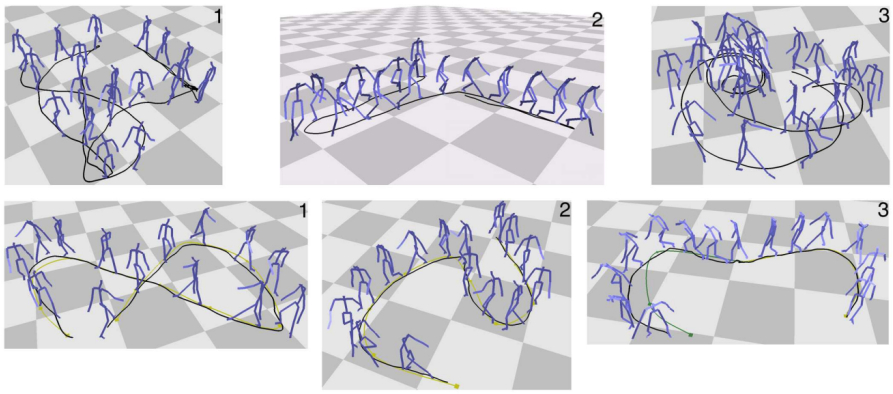



**Figure 2.51:** Top: original motion capture, Bottom: motion synthesis by motion graphs [KGP02].

In contrast, Lee et al. [LCR*02] rely on a two-layer structure: a low layer that models the motion data as a Markov process to detect plausible transitions between motion segments; a high layer that groups the prerecorded data into similarity and constructs a forest of clusters to encode the choices available to the avatar among the same cluster. Arikan and Forsyth [AF02] create a graph hierarchy by grouping images from the same motion sequence together, setting a cost value for the graph edge representing the dissimilarity of two images, and applying a random search of a graph hierarchy to satisfy user constraints. Finally, Safonova and Hodgins [SH07] define motion synthesis as an interpolation of two time-scaled paths through the graph, using discrete optimization to compute the transition between the poses in the database.

The application of motion graphs in these methods is limited to human locomotion and prerecorded clips. In particular, it does not provide much freedom to the user in terms of motion

creation. More specially, while the motion search process can be accelerated by graph walks, this process does not allow for additional constraints such as obstacles in an environment.

As an enrichment of movement but still relying on prerecorded motions, Thorne et al. present Motion Doodles [TBvdP04], a system in which the user can design a movement by sketching a sequence of gestures, composed of arcs, lines, and loops. In the same spirit as SKETCH [ZHH96] but applied to motion, the authors associate a set of 2D and 3D sketched trajectories to specific motions. Their method first segments and analyzes each input stroke to recognize the underlying motion and the stroke parameters. Then, it applies a keyframe-based parametrized motion synthesis to create the animation by decomposing the determined motion into keyframes extracted from a database. These keyframes are then interpolated using Catmull-Rom interpolants and an inverse kinematics solver to preserve contact with the environment. Due to depth ambiguity, their model is mainly restricted to planar motions, in addition to offering only the prerecorded set. In contrast, Min et al. [MCC09] represent a set of provided motion examples by a generative model characterized by two deformable parameters, encoding variations related to the geometry and timing of the motion. The authors thus defined the motion synthesis problem as a maximum a posteriori framework estimating the most plausible parameters from the user input.

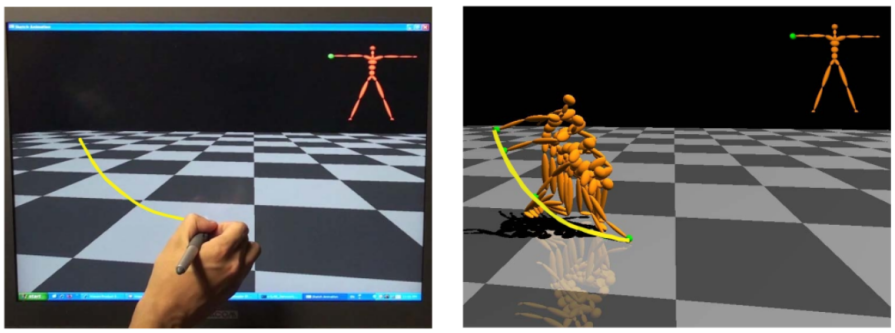


**Figure 2.52:** Overview of [MCC09]: the pen-based sketching interface (left) and interactive motion generation with deformable motion models (right): (left) pen-based sketching interface; (right) motion filtering and foot.

However, the use of prerecorded motion limits the creative process because the motions are restricted to those present in a database, and also the animation cannot be expressive or modified. In particular, recent works have addressed the correlation of space and time encoded in the user's path. In addition, these approaches all propose a coarse-to-fine process by letting the user sketch a global path that defines an initial animation that can be refined by additional paths.

*Sketching space-time curves*

Extending the keyframe abstraction concept to dynamics, Guay et al. [GRGC15b] introduce a space-time sketch abstraction, encoding a complete coordinated motion in a single sketch curve called space-time curve (STC). In particular, this curve encodes both trajectory and velocity, as illustrated by Figure 2.53 right. The user can sketch other curves in the environment to refine the motion.

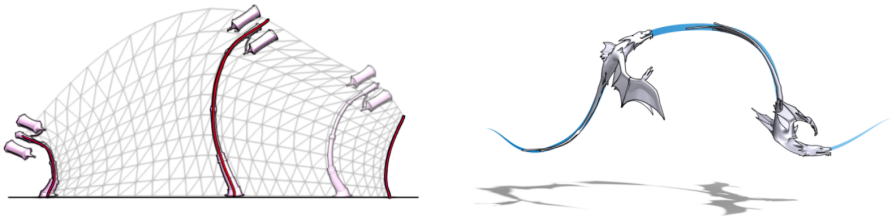


**Figure 2.53:** Space-time curve [GRGC15b]: (left) representation of the space-time surface (DLOA) defined by the sketched strokes (in red) depicting key poses; (right) result of the space-time sketching abstraction, enabling to sketch shapes and paths from a single curve (in blue).

The authors rely on certain assumptions about the desired animation behavior, such as the correspondences in space and time of the shape and the provided keyframe, to define the concept of dynamic line of action (DLOA). Such a DLOA is a 2D parameterized surface where time is seen as a spatial parameter, and the constant time slices are static lines of action (see Figure 2.53 left). Unlike the LOA of Guay et al. [GCR13], the complete DLOA, i.e., the keyframe sequence and the timing, is extracted from the input "space-time curve" drawn in screen space. In addition, the authors extend the linear interpolation to preserve the local length and C1-smooth motion. From the user input, their method uses projective constraints to compute the DLOA that drives the motion. In particular, they achieve that by linear parameterization, followed by splitting the parameter into a spatial and timing variable, from a warping. This linear parameterization allows for squash and stretch effects since the drawing speed is not constant. In addition, this solution also allows for bouncing and rolling effects by recognizing singular points or loops and respectively adding a correction term to handle take-off and landing and an additional constraint on the DLOA. With the STC concept, the authors introduce a means to correlate space and time under the same single curve. However, their method is limited to simple shapes (without limbs).

As an extension of the previous approach to more complex characters such as articulated figures, Choi et al. present SketchiMo [CiRL*16], a system dedicated to editing a predefined 3D articulated motion using sketch inputs and direct manipulation. During the editing process, the user alternates between defining a sketch target and editing the motion using the three sketch spaces, as shown in Figure 2.54.

**Figure 2.54:** SketchiMo [CiRL*16] variety of visualizations: (top left) joint path in the world space; (top middle) relationship between a joint and its parent; (top right) between two coordinated body parts; (bottom) temporal coherence of the motion.

Four types of sketch targets are available: body line, joint path, selected joints connection, or end effector connection. The authors rely on the body line definition of Guay et al. [GCR13] but let the user select the desired body line. Then, their system infers the joint paths from the extremities of this bodyline, also serving as a visual guide for the joint trajectory. Their method uses the remaining sketch targets to connect the movement of two skeletal joints. With the provided sketch spaces, the user can modify the global motion of the body line (global space), the local motion of an end joint (local space), or expand time to facilitate the fine editing of both these motions. In addition, the authors provide a brush to re-time some sections of a path and a noise removal tool for the motion data. The multi-resolution property of their system makes it efficient to act on different levels of motion. The authors rely on the minimization of a constrained energy function to update the current motion. Although their interface allows users to easily toggle between different levels of detail, which provides great freedom in the creation process, their system is solely focused on editing motion and not on generating new content from the user's sketch.

As an alternative to traditional trajectories, Ciccone et al. [CGNS17] focus on designing motion cycles. The user animates an element (full 3D model, rig controller, bone, etc.) by sketching multiple loops of the same motion around the relevant entity. As illustrated in Figure 2.55, multiple motion cycles can be sketched for different elements, allowing for more complex animation.

**Figure 2.55:** Motion cycles [CGNS17]: the user draws several loops (left), a looping motion cycle is extracted from the noisy input (middle); the user can combine different motions to create a complex animation (right).

The authors define a motion cycle as a set of sketched nearly cyclic repetitions. The user sketches as many sets of cycles as desired, which reduces inaccuracies and potential noise in the input data. From this data, their method identifies repeating patterns in three steps. First, it relies on an analysis of the spatial and velocity variations of each pair of points to estimate the cycle period. Then, it computes the correspondences between cycles to find the similarity of the extracted cycles and thus, defining an average cycle. Finally, it fits a Bézier curve to each component of this cycle, i.e., one for translation, rotation, and scaling. Their method inserts this resulting cycle into their interactive interface, allowing for the edit of a cycle in both time and space and the current viewport using direct manipulation. As each Bézier spline maps time to position, orientation, or scale, their system computes the bijective function giving the time of each curve to a specific value. Thus, such functions provide correspondences between time and space, orientation and scale. On their interface, the user can update the positions of the points in the cycle by direct manipulation and the cycle's orientation, and scale by interacting with their oriented and elongated arrows and propagating the transformations throughout the cycle. In addition, the user can sketch a projective line to highlight the presence of contact constraints. Finally, their method allows users to update the cycle timing by moving points closer or farther apart, resulting in a smooth time-warping expressed as the minimization of an energy function. Although the authors provide some editing tools, their system is highly dependent on the quality of the input, and an inaccurate spline model can be found in the case of too much heterogeneity between the user's cycles.

In summary, we see these contributions as a real improvement, from the early data-driven methods to the generation and editing of space-time curves from single curves and minimum user intervention. Another feature of the last three methods is the coarse-to-fine design process, allowing the user to progressively refine the animation, while already visualizing the motion and deformation. This coarse-to-fine design is also a feature that we propose in Chapter 5.

### 2.3.3 Sketching motion guidelines for a set of elements

In this part, we discuss the few methods aiming at the animation of a set of elements from sketch inputs. We classify these systems according to the type of objects manipulated by this animation process.

*Inferring a flow field from sketches*

As illustrated in Figure 2.56, Zhu et al. [ZIH*11] propose interactive illustration and animation of fluid systems progressively enriched by sketch inputs. The authors rely on a fluid simulation system in the background to intensify the illustration. They represent the fluid system by

a hydraulic graph built incrementally from the user's sketches and encoding the topological relationship between regions and pipes. A linear system based on hydraulic rules computes the flow in the circuit. Finally, a multi-layered solver driven by the flows in this network infers the flow patterns within the local fluid regions.

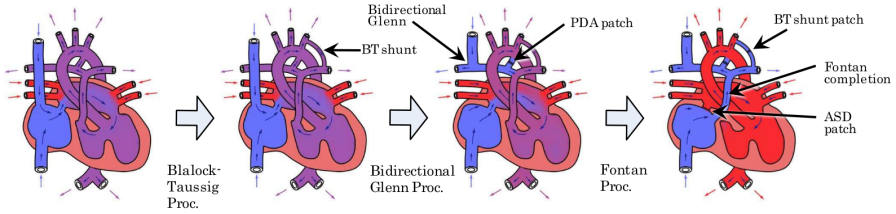



**Figure 2.56:** An example of a dynamic illustration representing the surgical repair procedure of an heart defect through progressive user sketching and editing in [ZIH*11].

Creating such a dynamic illustration of fluid systems starts with the creation of contour regions and pipes on a 2.5-dimensional canvas. The creation of an exterior structure implies the automatic creation of an inflow or outflow and its propagation throughout the structure. The user can modify the parameters of any flow using a menu. Among the other features, pipes and regions can overlap thanks to the 2.5D canvas and the depth order can be changed locally at any time. The user can sketch inner obstacles, additional force, or even add a flow source inside a region to change the flow pattern, any pipe can be blocked or unblocked at any time. Direct manipulation interaction can serve to erase, move, rotate or deform the contour of elements. To provide only valid results, automatic detection of an invalid subgraph stops the simulation in it. In addition, this system can be used to illustrate or communicate cardiac defects or surgical procedures (see Figure 2.56).

In contrast, the Energy Brushes system [XKG*16] targets passive and secondary animations.

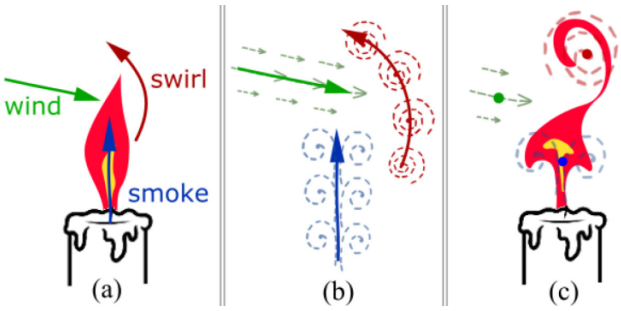



**Figure 2.57:** Overview of Energy Brush [XKG*16]: a) the colored arrows represents different types of energy brush (wind, swirl, smoke); b) the energy brush and underlying particles flow; c) the influence of energy brushes on a given shape.

The user sketches a stroke that can represent the outline of a shape, a color-filled region, or a texture. Then, he can define flow particles around this shape to generate a local energy pattern represented by a velocity field within a given radius. The authors proposed three types of velocity

fields: wind, swirl, and smoke. Finally, an energy brush is a stroke that represents the coarse direction of energy and forces. As shown in Figure 2.57, the authors correlate each energy brush to a specific type of flow particle and each brush continuously emits those particles according to the sketched trajectory. The user can change the emission interval, speed, strength, and brush size. In addition, the user can set anchors, stretching effects, or rigidity constraints.

These two methods tackle the simulation of fluid whose flow is guided by a sketched environment. Given the large number of particles interacting in these systems, constrained physical simulation remains the simplest and less computationally expensive to handle such systems.

*Objects*

In a similar approach as the previous one, Kazi et al. [KCG*14] propose Draco, an interactive system dedicated to animated illustrations. The authors rely on the concept of kinetic texture, defined by a combination of data samples (distribution of objects drawn by the user) and a set of motion properties (global and granular). Global and granular motions can be added to any kinetic texture and their parameters can also be adjusted at any time. The global motion affects the main trajectory while the granular one is applied individually to each object. As shown in Figure 2.58, the authors offer two types of kinetic textures, one emitting and one oscillating.

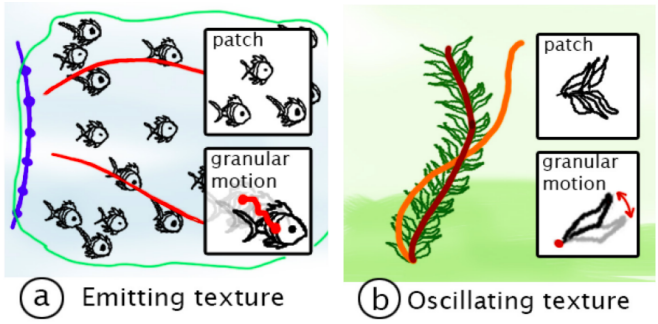



**Figure 2.58:** Kinetic textures in Draco [KCG*14]: (a) Emitting texture, defined by a source patch, emitter (dark blue), global motion paths (red) and granular motion, outline (green blue). (b) Oscillating texture, defined by source patch, brush skeleton (brown), oscillating skeleton (orange), and granular motion.

For both types, the user begins by creating a trajectory composed of a set of sketched elements. The authors characterize the emitting texture by a sketched emitter line (in blue in a)) from which these elements emanate and the motion of the texture represents both the local and global trajectory and guides the motion field. The user can also control the dynamics of an emitting texture (control the velocity, frequency, cohesion) and specify two additional features to delimit the propagation of the distribution in space: an outline, from which objects disappear; a mask that hides all the objects crossing. In contrast, an oscillating texture is defined by a reference skeleton (in

brown in the Figure 2.58) along which the patch is uniformly distributed, and a target oscillating skeleton defines the range of spatial oscillation. A motion profile can be displayed to sketch the desired scale of the velocity of an element. This system has been extended in Kitty [KCGF14] to incorporate interactivity and functional relationship between entities by direct manipulation.

Focused on motion instantiation, the Hierarchical Motion Brushes system [MNB*14] leverages a painting metaphor to allow users to brush animated content directly onto a 3D scene. In addition, this content can be stored in a higher-level brush to support coarse-to-fine animation and promote the creation and reuse of animated content at different levels of detail. The authors define a motion brush as an elementary digital scene composed of geometries that move in time. The atomic content of this type of brush is created offline using traditional animation tools to define an animation sequence of 3D content, further stylization of this sample can be achieved using the Overcoat 3D painting system [SSGS11]. During the interactive session, the user selects a motion brush, sets the brush parameters (size, spacing, and opacity), and draws a stroke embedded in 3D space, similarly to the Overcoat system [SSGS11]. The 3D content distribution, also called motion stamp, is instantiated in the local frame of the stroke relative to the brush spacing value. As shown in Figure 2.59, the user can also combine multiple brushes and choose which one to use on which part of the strokes to draw.


**Figure 2.59:** An example of creation in Hierarchical motion brushes [MNB*14]: (a) the user paints a set of motion brushes; b) he creates another scene containing two hands (b); by combining these two brushes, the user can paint only one stroke to create a complex fire effect that starts from one hand and reaches the other over time (c).

In addition, the authors offer some editing tools such as the ability to repaint existing stamps to update the parameters, coordinate the parameters of a brush or stamp to a predefined parameter, or let the user draw additional field strokes from which a scalar or vector field is determined and mapped to the chosen stamp parameter. In addition to these spatial tools, a curve editor allows users to adjust the timing of the animation, as well as organize the order of the contents of multiple brushes. Finally, the main component of this system is the hierarchical structure that allows to instantiate complex, multi-resolution animated 3D content. Providing an intuitive authoring tool that encodes a hierarchical set of animated 3D content addresses a significant challenge in the coarse-to-fine design of animated distributions of elements. However, in their current system, the atomic level of the animated 3D content is defined offline and there is no data processing relative

to potential structures of the sample distribution. To account for the latter problem, the same authors [MGC*16] introduced hierarchical spatio-temporal clustering of a set of simulated points to recognize the similarity of the distribution before replacing these points with the content of a motion brush.

Finally, Gu and Deng present Formation Sketching [GD11], a sketch-based crowd grouping and formation system on which the user sketches the target shape of the crowd (see Figure 2.60). Their method characterizes a provided distribution of agents (or a crowd) by computing for each agent its formation coordinates relative to the center of the group and also its global coordinates relative to world space. When a user defines the target shape, their method transforms the region of interest into a formation template by sampling the contour boundaries and using a Flood-Fill extension in the form of a horizontal scan-line traverse to fill that region with even-space template points. The main idea is to find correspondences between these two structures so that each agent is assigned a suitable target position. The authors rely on KD-tree structures, one for each coordinate type and a neighbor search process between the template points and the agent coordinates to find the exact profile and distribution of the target formation. A few relaxation steps are performed to optimize the general distribution. To provide a smooth and natural transition between the starting and target point of each agent, the authors create a two-level hierarchy group control scheme to divide the whole group dynamics into internal and inter-group dynamics. As a result, their method defines the final velocity of an agent by a weighted sum of local formation velocity, local collision avoidance, and local navigation velocity (inter-group part).

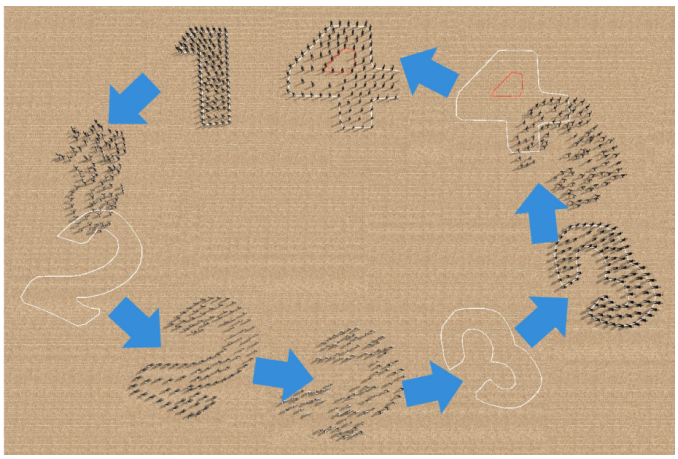


**Figure 2.60:** Crowd shaping and transitions [GD11] for a single group of 100 agents heading to four navigational way-points (center of sketching).

In summary, these solutions for animating a set of elements have introduced interesting sketch-based tools for animating 3D content. However, each of them has its limitations. Draco [KCG*14] limits the animation to 2D content and predefined movements. In contrast, Hierarchy Brushes [MNB*14] innovates the creative concept with a hierarchy motion brush structure that allows for the spatial combination of different brushes and more importantly the ability to encode a complex animated 3D scene into a higher-level brush and thus create nested structures of animated content. However, the authors take a predefined atomic animation, and no distribution analysis is performed to preserve the underlying structure. Finally, the last method deals with this creative animation process from a shaping perspective that can also be used in a more general case to provide variety in the motion of elements while preserving the group characteristics. Furthermore, while accurate simulation techniques could provide interesting results, it would be difficult to interactively adapt the simulation to match the user's sketch.

### Discussion

Both keyframing and motion trajectory approaches have their advantages and limitations. While keyframing provides a high level of control over the spatial representation of an element and allows for a fine-grained description of the detailed state of pose or deformation at a specific time, it is generally decorrelated to the timing process. In contrast, motion trajectories facilitate temporal control, but this may come at the cost of a lack of detailed spatial information at certain key points in the trajectory. The choice of one approach over the other will depend heavily on the context of use. For example, in an environment with obstacles, it may be easier to favor trajectory-based motion because collisions can be detected and treated as constraints on the motion. More recently, some methods introduce space-time curves as a solution to correlate both space and time in a single curve. However, they limit shape deformation to squashing and stretching or more local control that may require multiple iterations by the user. In addition, researchers have not yet explored painting metaphors to define a coarse keyframe that can also be combined with local or global trajectories as a quick solution to exploit both approaches.

Among the features of the presented systems we are interested in, the Fluid Illustration system [ZIH*11] allows one to represent an animated phenomenon and explore alternative options. However, only one option is displayed at once which results in overriding the previous one so there is no possibility to revert to previous designs. In addition, Hierarchy Brushes [MNB*14] introduces the concept of nested motion brushes, allowing for a coarse-to-fine design process as well as the creation and reuse of animated content at different levels of detail. This feature is highly important in our dynamic sketch concept as we want users to be able to create small samples of data (animated or not) and instantiate them elsewhere. Unlike their method, we aim to allow the user to create the desired animated 3D anatomical content through a sketch rather than relying on an offline process.

## 2.4 Conclusion

In this detailed section on related work, we have presented existing approaches to sketch-based modeling, example-based synthesis, and sketch-based animation. However as described in the respective Discussion part, the current methods are not sufficient to achieve our goal of fast creation and progressive refinement of complex, time-evolving scenes.

In particular, as uniquely exploited by Milliez et al. [MNB*14], the property of nesting structures greatly enriches the creative process while remaining user-friendly. In addition to this property, using over-sketching as a visual cue has barely been exploited and has never been used to explore alternative options. Furthermore, in the context of architecture, there is a real need for digital and exploratory tools to design coarse 3D structures and progressively refine the exterior and interior while using the sketch as a visual guide (Chapter 3). In addition to this creative design, the synthesis of anisotropic distribution from the user's sketch in both 2D and 3D has never been explored (Chapter 4). Finally, for our biological application but also towards a more general objective, we propose in our last Chapter 5 an in-going prototype of a sketch-based system allowing modeling, distribution synthesis, and also animation and this without any specific editing pipeline.

# 3

# Sketching evolving environments: the example of architecture

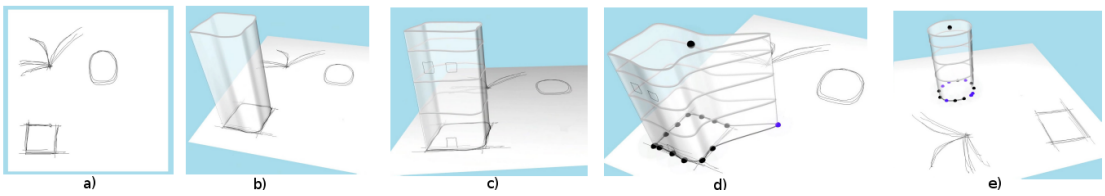



**Figure 3.1:** Overview of our tool: the user can draw strokes to represent uncertainty or add visual details to a surface (a) and (c), also create 3D surfaces from sketching (b), at any design stage, the exploration of hypotheses can be performed by deforming the basis of any surface using a drag-and-drop gesture (d-e).

In this chapter, we use the example of architecture to explore how 3D sketching could be used to draft, explore, and progressively improve new designs. Architects typically use paper and pen to go through the design phase, where they proceed in successive iterations from an initial idea to a finished model. This process is rarely linear. Thus, phases of uncertainty, i.e, areas of indeterminacy, can easily be represented on paper by areas of lighter strokes or by an over-sketch (several strokes in the same place). While these areas provide crucial information during the creative process, a paper sketch does not offer a solution to favor an option over another without affecting the readability of the sketch. On the other hand, even though digital software offers a solution for directly visualizing 3D models, it imposes on users an overly rigid editing pipeline, making it impossible to represent and explore alternative designs.

To favor exploratory design in 3D, this chapter presents *Nested Explorative Maps* (NEM), a new system dedicated to interactive design in architecture. Our model allows coarse-to-fine sketching of nested architectural structures to progressively shape a building in 3D from the floor plan to interior design, through a series of nested maps able to spread in 3D. Each map allows for the visual representation of uncertainty as well as the interactive exploration of the alternative and tentative options as shown in Figure 3.1. We validate our model through a user study conducted with professional architects to highlight the potential of the NEM system for conceptual design in architecture.

## Résumé en français

Dans ce chapitre, nous prenons l'exemple de l'architecture pour explorer comment un croquis en 3D pourrait être utilisé pour dessiner, explorer et améliorer progressivement de nouveaux designs. Les architectes utilisent généralement le papier et le crayon pour parcourir la phase de création, où ils passent par des itérations successives afin de progresser d'une idée initiale au modèle final. En particulier, ce processus est rarement linéaire, et des phases d'incertitude, c'est-à-dire des zones d'indétermination, peuvent facilement être représentées sur un papier par des régions de traits plus légers ou par une sur-esquisse (plusieurs traits au même endroit). Si ces zones apportent des informations cruciales au cours du processus créatif, un croquis sur papier ne permet pas de privilégier une option plutôt qu'une autre sans en affecter la lisibilité du croquis. D'autre part, même si les logiciels numériques offrent une solution pour visualiser directement des modèles 3D, ils imposent aux utilisateurs une suite d'étapes trop rigide, rendant impossible la représentation et l'exploration de designs alternatives.

Pour favoriser le design exploratoire en 3D, ce chapitre présente *Cartes Imbriquées et Exploratoires* (NEM en anglais), un nouveau système dédié à la conception interactive en architecture. Notre modèle permet d'esquisser des structures architecturales imbriquées, du grossier aux détails, afin de façonner progressivement un bâtiment en 3D, du plan de sol au design intérieur, grâce à une série de cartes imbriquées et capables de se déformer en 3D. Chaque carte permet la représentation visuelle de l'incertitude ainsi que l'exploration interactive des options alternatives et provisoires, comme le montre la Figure 3.1. Nous avons validé notre modèle par une étude utilisateur menée auprès d'architectes professionnels, afin de mettre en évidence le potentiel du système NEM pour le design conceptuel en architecture.

### Contents

## 3.1 Motivations

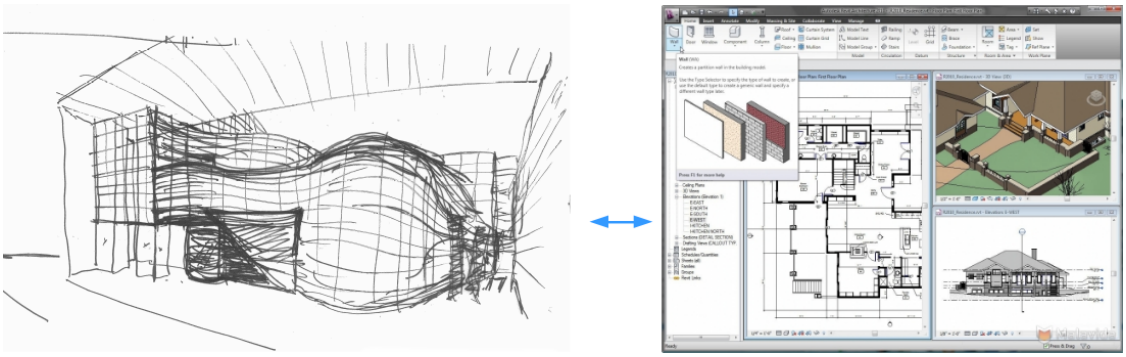


**Figure 3.2:** Comparison between: (left) a paper sketch from the SCAU agency and (right) the Autodesk's software Revit [Aut02] using the BIM tool.

At the end of the 20th century, the arrival of digital tools revolutionized the architectural profession. Traditionally, the architects were conceiving a new project in two steps: a sketching phase, during which they were drafting and exploring ideas using a paper and a pen; a construction step, in which one or several physical and scaled models were built not only to obtain a 3D representation of the resulted outcome but also, determine how plausible an idea was. Now, the digital software is providing a solution to combine both of these steps by letting architects directly create 3D models in a digital version of the context environment, and inside which the final design will be built. Moreover, the digital world has taken so much room in the architecture profession that nowadays, the customers are requesting architects with a digital model of their proposal.

Such digital software can appear at first sight as a real improvement over the classical medium since professionals are not restricted anymore to the suggestions of surfaces and volumes, and they can even directly examine the 3D representation of an idea in the construction context. However, conceiving a project on the currently existing software is far from being as intuitive as drafting ideas on a paper. Indeed, creating each model may take several hours since many mandatory details need to be specified, from the precise dimensions of each element to the choice of the construction material. This cumbersome modeling stage does not encourage trials or creativity (see Figure 3.2 right), as some parts of the design may still be rough place-holders, with several possible alternative versions that are still envisioned but cannot be visually represented. Moreover, it can also lead professionals to reuse elements from a former project (by a simple copy/paste operation) rather than designing adapted ones from scratch.

In particular, the BIM tool ("Building Information Modeling" [Aut02]) has recently imposed itself in the profession. This tool has the benefit of allowing different stakeholders to work on the same project model and provides a better study of the building performance and construction planning. However, the software using BIM still imposes on architects a non-logical hierarchy in

the design phase, such as adding on a current model, an element with already fixed characteristics (material, dimensions).

Therefore, in practice, architects still alternate between manual drawings and digital software. Using the paper & pen medium allows for a more global and natural vision at the design stage, in addition to providing the opportunity to represent areas of uncertainty, through lighter strokes or over-sketching (Figure 3.2 left). However, it comes with the usual flaws of any 2D physical medium: each sketch represents buildings under a single static view; progressively adding details must be compromised with sketch readability; jointly modeling the exterior and the interior of a building can only be done to a certain extent. These flaws can be solved by representing the sketched design in 3D, however, as explained previously, the current software is too rigid to do that. Thus, architects currently need to re-start their design from scratch on 3D software, sometimes enabling the late detection of strong inconsistencies that need to be solved along the way.

In this chapter, we tackle the problem of proposing new 3D modeling paradigms, dedicated to satisfying the architects' needs at the conceptual stage of design. In particular, this work involved a collaboration with a professional architecture agency, the SCAU[1] agency in Paris.

Throughout this collaboration, we conducted two user studies: a pre-study to identify the major needs; and a final user study to present our prototype and validate our new concepts. For instance, the architects expressed a genuine need for an easy-to-use tool, enabling free-form drawing in 3D, i.e., the ability to change the viewpoint, while sketching rough strokes to represent uncertainty as easily as with paper and pen. In addition, they also pointed out the importance of coarse-to-fine modeling such as being able to quickly draft the general view of the outside of a building while allowing a progressive refinement by adding the relevant details both on the inside and outside and, in any appropriate order. The last and most challenging request was to authorize the interactive exploration of the different options indicated within the drawing for each element of the building. Note that this brings a new challenge to sketch-based modeling systems, namely turning the user drawing into an interactive exploration tool rather than merely extracting a single 3D model from it.

Based on the architects' needs, we introduce a new type of 3D sketch, called *Nested Explorative Maps* (NEM). At any time, the user can draw and progressively refine a 3D sketch built as a hierarchy of nested and deformable 3D canvases – such as the ground surface of a building, its facade, or inner floors. These canvases are associated with a texture image, called a map, which conveys the original user's strokes, in addition to the information of local uncertainty or certainty, respectively expressed through lighter strokes or over-sketching. Thanks to these maps, the user

[1] http://www.scau.com/fr/home

can draw strokes to define and refine preferred positions for each of the 3D canvases before exploring alternative architectural options through a drag-and-drop gesture to move and deform them to the most relevant design. At any stage, the user can either edit a map, create a new 3D canvas, or select and move any existing element to a new tentative position on the underlying map while the consistency of nested details is automatically maintained.

In addition to the concept of NEM itself, our main technical contributions include:

- a recursive solution to create a 3D sketch by drawing on semi-transparent 3D canvases, themselves extruded from the user's strokes – with the associated visualization and editing tools;
- a method, based on the local attraction to dense stroke regions, allowing for the interactive navigation through the alternative designs that the nested sketch visually suggests. This navigation may not only drive displacement but also free-form deformations of the nested 3D canvases carrying the user's strokes.

Figure 3.3 illustrates an example of a conceptual design draft, efficiently created by a professional architect.

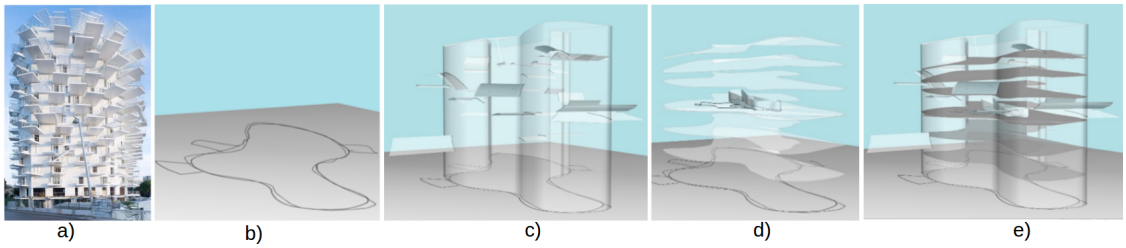



**Figure 3.3:** Our Nested Explorative Maps' system can be used to quickly draft and explore architectural models. (a) Inspiring photograph: Sou Fujimoto's White Tree building; (b) Ground map showing two alternative designs; (c-e) Nested canvases, enabling the progressive refinement of the outer and inner parts of the building in less than 10 minutes.

## 3.2 Pre-user study

We started this project with a week of pre-study at the SCAU agency to analyze their workflow, identify their current needs and the constraints to add to the ideal design tool. In particular, we were interested in understanding how architects used both paper and digital media and, more importantly, at which step of their creative process. This study involved 9 architects, 7 men, and 2 women, from 6 months to 30 years of experience.

From the feedback of a mere intern to an expert architect, using the paper medium was, for all of them, essential to start the conception of a new design. It was also considered the best medium to think, express, and communicate ideas. Providing a direct tool for the creation that does not require any translation ("*the hand is the extension of the brain*", "*the direct translation of the thought*"), enables professionals to "*prioritize decisions*", "*keep the memory of the different iterations of the thought*" as well as highlighting uncertainty areas which "*let room for interpretation for both the drawer and other people*". In particular, in their profession, a sketch completed by some explicating lines is referred to as a "*figure*".

However, they are also aware of the limitations of a paper sketch, such as how "*difficult*" it can be to sketch a 3D model on a 2D medium as well as defining "*precise dimensions and scale*". While defining such data is possible on the current digital software, architects are not satisfied by the rigidity of its editing pipeline ("*non-intuitive hierarchy of steps*", "*need to follow an imposed pipeline to reach the expected result, which may require several trials or a complex succession of steps*") and the reversal of the steps compared to the natural creative process ("*reflection towards the details and not to the basic elements*", "*freeze a lot of elements that will be definitive and not modifiable*").

From this feedback, we focus our project on an intuitive tool favoring creativity in digital design, through augmented sketches, able to express and explore uncertainty.

To achieve this goal, we identify four main criteria to include in our model:

- **(C1) Immediate usability**, as easy to use as paper and pen,
- **(C2) Coarse-to-fine design**, enabling the creation of coarse 3D models (e.g., not asking for precise dimensions or details at first) and then progressively improving and refining them both the outside and inside, without imposing any specific editing pipeline,
- **(C3) Free-form strokes and uncertainty representation**, allowing the interactive sketching while keeping the user's original strokes instead of over-simplifying them so that strokes can be used to indicate uncertainty, non-geometric details, or any other information (e.g., through arrows or hatching),
- **(C4) Exploration of the alternative options**, providing a way to try the alternative designs depicted within a given sketch.

In the above criteria, we do not specify 3D navigation or import of external data because they are usually already present in digital solutions and do not bring major scientific challenges.

Based on the architects' insight, we evaluate the capabilities of paper and pen design, and of the digital software used in the agency (such as *Revit* [Aut02] and *SketchUp* [Tri00]) regarding these needs. Among our findings, we notice that sketching on paper enables professionals to suggest 3D surfaces from a few rough strokes while using other ones for decoration or to refine the design. This succession of strokes progressively builds the mental image of the model, allowing for a coarse-to-fine design. In contrast, 3D modeling software usually requires users to assemble volumetric primitives to directly materialize a building with the appropriate dimensions including thickness. Among them, *Revit* requires several months of training but imposes a non-logical hierarchy of steps which does not allow coarse-to-fine modeling. In contrast, even though *SketchUp* necessitates only several days of training, it is more adapted to prototyping. However, it still does not handle the representation of uncertainty nor the exploration of options.

We then position the state-of-the-art solutions in Computer Graphics (see Section 2.1 for more details) against these criteria. To the best of our knowledge, only four systems are specific to the interactive sketch-based modeling of architectural models: Sketching Reality [CKX*08], Facetons [SCSI15], Interactive Sketching of Urban Procedural Models [NGDA*16] (referred to as Sketching Procedural in the table below) and Building Sketch [LZC21]. Even though these methods provide immediate usability (**C1**) and a coarse-to-fine design (**C2**), the user's strokes are always replaced by some primitives either inferred from a database or using a convolutional neural network or even a six degree of freedom point. Closer to this work are the methods focused on the creation of a design sketch that lives in 3D and applied to architecture: Mental Canvas [DXS*07],

Insitu [PKM*11], Smart Canvas [ZLDM16] and CHERish [RLT*17]. Following the idea of drawing strokes on semi-transparent 2D planes (or canvases), these methods are, for all of them, validating (**C1**) and also almost a coarse-to-fine design (**C2**) as well as free-form drawing, and even to the extent of representing the uncertainty (**C3**). However, they limit the design to the creation without allowing any exploration.

To sum up, while easy-to-use 3D sketching tools have been introduced in recent work, allowing for both coarse-to-fine design and the representation of uncertainty through stroke depiction on 3D canvases (criteria **C1** to **C3**), none of the existing solutions meets all the expressed criteria, and, more specifically, none addresses (**C4**).

Table 3.1 presents a summary of the current architecture design media and the state-of-the-art methods in Computer Graphics.

| | C1 | C2 | C3 | C4 |
|---|---|---|---|---|
| Paper & pen | ✓ | P | ✓ | X |
| Revit [Aut02] | X | X | P | X |
| SketchUp [Tri00] | P | ✓ | P | X |
| Sketching Reality [CKX*08] | ✓ | ✓ | X | X |
| Facetons [SCSI15] | ✓ | ✓ | X | X |
| Sketching Procedural [NGDA*16] | ✓ | ✓ | X | X |
| Building Sketch [LZC21] | ✓ | ✓ | X | X |
| Mental Canvas [DXS*07] | ✓ | ✓ | ✓ | X |
| Insitu [PKM*11] | ✓ | ✓ | ✓ | X |
| SmartCanvas [ZLDM16] | ✓ | P | P | X |
| CHERish [RLT*17] | ✓ | ✓ | ✓ | X |

**Table 3.1:** Evaluation of the existing digital tools—including two software used by professionals and recent research solutions—in respect to four of the criteria expressed by architects. "X" means not handled, "P" means partly handled. Our work tackles the introduction of a tool matching all the criteria and in particular, C4, which was never considered so far.

## 3.3 Overview

### 3.3.1 Our solution: Nested Explorative Maps

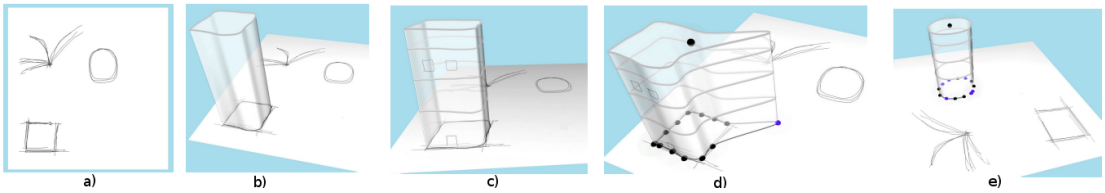



**Figure 3.4:** Overview of our tool: the user can draw strokes to represent uncertainty or add visual details to a surface (a) and (c), he can also create 3D surfaces from sketching (b), at any design stage, the exploration of hypotheses can be performed by deforming the basis of any surface using a drag-and-drop gesture (d-e).

We present a new solution for 3D sketch creation in architecture, focused on achieving all the four criteria extracted from the pre-study. To achieve that, we first rely on an interactive drawing on a digital tablet, also serving as a screen—to keep the interaction as close as possible to the paper and pen medium—thus avoiding any steep learning curve **(C1)**. Then, we based our solution on a coarse-to-fine, progressive design **(C2)**, that directly uses the free-form strokes as elements of the 3D sketch, as well as visual feedback of uncertainty **(C3)**. Finally, our system is the first to offer the possibility to navigate interactively through the alternative designs suggested by the sketch **(C4)**.

This proposed solution is built on the new concept of *Nested Explorative Maps* (NEM). In addition to the concept itself, our main technical contributions include:

- a nested structure for a coarse-to-fine, free form design:

  modeling from a coarse outside to details both on the outside and inside,

  while keeping the original strokes,

  without any specific editing pipeline,

- a method, based on local attraction to dense stroke regions, allowing for the interactive navigation through the alternative designs that the nested sketch visually suggests.

Figure 3.4 depicts an overview of our framework, showing the progressive creation and exploration.

### 3.3.2 Terminology

Before describing our approach in detail, let us define three technical terms that will be used throughout this chapter:

- **3D canvas**: Surface on which the user can draw a sketch. The term is used by analogy with the traditional artist's canvas, but in our case, a canvas can be any free-form surface in 3D space.
- **Footprint**: Vector curve defined from a user's stroke and serving as the basis for the extrusion of a new 3D canvas. Footprints are dynamic elements that can be interactively deformed to allow the exploration of alternative options.
- **Map**: Layer defined on top of a canvas containing both user's strokes and a texture image to depict both the uncertainty and the stroke density to guide subsequent footprint deformations.

Figure 3.4 illustrates these elements: a) shows the user's over-sketching strokes stored on a map over a flat canvas; b) displays a footprint used to extrude a new 3D canvas; this new canvas is itself associated with a map storing sub-sketches in c); d) and e) show the result of the deformation of a footprint carrying the associated canvas and map towards another optional design.

### 3.3.3 Presentation of our interface

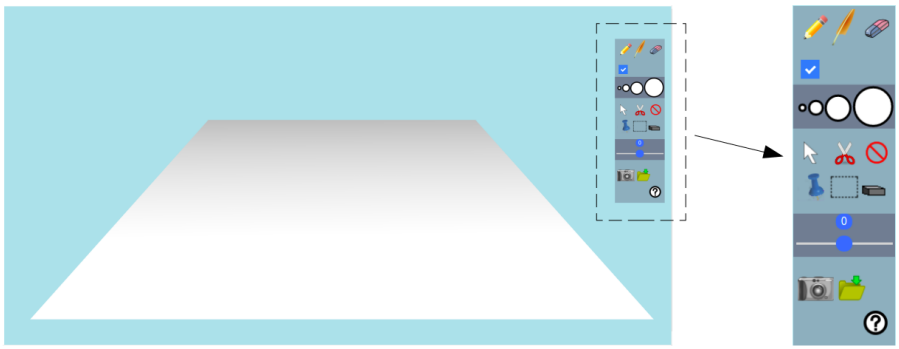


**Figure 3.5:** Overview of our interface.

Figure 3.5 presents our interface composed of two main parts: on the left, the default 3D canvas, that is, the ground surface (a simple plane), on which the user will start the design; on the right, our toolbox. As we render the scene using a perspective camera, the user can at any time control the camera rotation or zoom parameters to navigate in the 3D sketch.

Elements composing this toolbox can be classified into four categories: sketching modes, direct manipulation, display tools, and load/save buttons. In addition, a help pop-up can be

displayed from the question mark button to remind the user of the specificity of each button.

Unlike a physical paper on which users can use the same pencil to draw strokes carrying different goals, we decided to separate each function into different modes, represented by a specific button. We offer users four different sketching modes along with an eraser and a width picker. As described in more detail in the next section, these modes are the following: the pencil for free-form drawing; the feather for footprint strokes from which surfaces are extruded along the default direction if the checkbox is activated; the scissor to sketch a cutting curve to cut one or several 3D canvases at once; the brick for the specific case of footprint strokes that define inner floors.

Our system is also complemented by interactive tools ranging from direct manipulation to display modes, as well as an undo button to return to the last action. For instance, to allow for the precise design of details, the dotted rectangle button lets users select an area of their 3D sketch to zoom in. However, this may not be sufficient for designing inner floors or on canvases hidden by others. To solve this problem and improve the readability of the sketch, we provide a slider (in blue in the toolbox) that allows users to switch between viewing the entire structure of a building and only a part of its interior or exterior, layer by layer.

Finally, the main feature of our system is the exploration of alternative options suggested by the sketch. Users can achieve that by activating the navigation mode from the cursor button and then applying a drag-and-drop gesture on any 3D canvas to refine its shape. In addition, the user can secure parts of the surface by using the pin button to block their displacements or deformations.

## 3.4 Nested structure for a coarse-to-fine, free form design

Our goal is to provide a tool enabling users to fully draw a building, from a tentative map on the ground surface to facade details and interior design.

### 3.4.1 Coarse-to-fine, free-form design

The creation of a NEM follows a recursive approach, starting with the ground surface as the first canvas, as depicted in Figure 3.5. During an interactive session, the user keeps alternating between sketching strokes to edit the maps on the displayed canvases, drawing footprints to create new child canvases, and editing canvases (see Figure 3.4). All these operations are based on free-hand drawing and can be done from any arbitrary viewpoint with any number of structures displayed. This enables the creation and refinement of any part of the NEM, at any time, which is essential in

the design phase of a project.

In this project, we interpret the user's strokes as:

- free drawing and representation of different hypotheses through uncertainty or over-sketching;
- basis of new child 3D canvases, the latter being either,

  3D surfaces constructed by extrusion of the stroke along a chosen direction,

  inner floors created by expanding the stroke along a parent surface;
- cutting gestures to cut 3D surfaces, similarly to scissors cutting.

For all the following sketching modes, the user stroke, i.e., the polyline captured on the screen space, is projected along the camera viewing direction to the closest displayed 3D canvas using the ray-tracing approach. In addition, in the case of a complex 3D sketch, the user can have access to hidden canvases by setting the display on or off for each of the existing canvas in the sketch.

**Free drawing and uncertainty representation** To be as close as possible to the paper/pen medium while enabling drawing in 3D, the user's original strokes are stored on the closest canvas' map and more particularly in local coordinates relative to this canvas. They are displayed without any simplification or smoothing steps to preserve the depiction style of the architect, as well as their usual expression of uncertainty (such as over-sketching a curve or drawing it with dashed lines).

As illustrated in Figure 3.6, the strokes can be drawn on the ground surface (left) or be projected on any 3D canvas (middle and right).

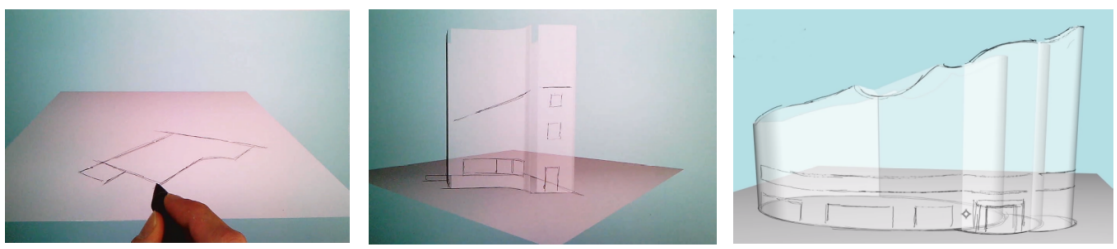


**Figure 3.6:** Examples of map sketching mode: on the ground surface (left); on 3D canvases (middle and right).

**Footprint curves and 3D canvas** In contrast to the previous mode, the user draws a single continuous stroke interpreted as the footprint of a new 3D canvas. This footprint can be an open or closed curve from which the child canvas is extruded along, either the vertical direction, the horizontal one, or the averaged normal directions of the support canvas. While verticals and horizontals fit standard architectural designs, the averaged direction allows free-forms 3D design. Each new canvas is extruded by a fixed default height which can be adjusted from a cutting stroke (as described below). In addition, the extruded surface is associated with a map defined from the canvas's texture coordinates.

We chose to represent the footprint curves as cubic interpolating Catmull-Rom splines to help generate smooth canvas surfaces. In addition, we provide a better level of smoothing for quickly drawn strokes by setting the control points as a combination of the drawing speed of the user and the distance between the original stroke points. From the user's perspective, sketching fast allows to model smooth curves while sketching slower enables the precise design of sharper curves.

Having a closed footprint and thus a generalized cylinder as the resulting canvas is the condition to add inner floors. To ease the creation of such footprints, we snap the first and end vertices together if their distance is less than a threshold.

*The special case of footprint curves and inner floors*

To ease the creation of inner floors or separating surfaces, we introduce a specific case for a footprint as the contour of a new 3D canvas. As illustrated in Figure 3.7, such a footprint is drawn on the surface of an already existing closed 3D canvas (generalized cylinder) to create the child canvas at the level of the contour curve. This is done as follows. We first project the user's stroke and expanded it along the support surface to model a closed 3D curve. Then, to limit the total number of polygons, we generate the interior surface from a coarse triangulation of a sub-sampled set of points from the footprint and a few central points on its medial axis.

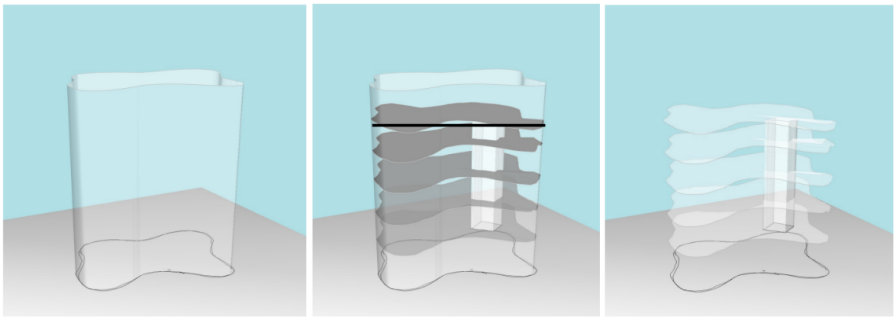


**Figure 3.7:** 3D canvas and floors creation: (left) a 3D canvas; (middle) the creation of a new inner floor; (right) the inside structure composed of a set of inner floors and an additional 3D canvas.

**Cutting stroke** As explained earlier during the footprint part, each 3D canvas is extruded by the same default height. We made this choice to provide users with direct visual feedback without requiring another manipulation. However, we let the user interactively cut canvases by sketching a free-form cutting line over one or several displayed surfaces, as illustrated in Figure 3.8. Similarly to scissors cutting a paper, the cutting line is expected to span the entire cross-section of the canvases it should cut. Indeed, we apply the cutting to all the displayed canvases from which this property holds. Therefore, the user can cut several canvases, embedded within one another or not, in a single cutting gesture. After an automatic selection of the target canvases, we cut the associated meshes so that the border of their projection on the screen matches the cutting stroke. Then, we update the associated maps to take into account the cut without modifying the remaining part.

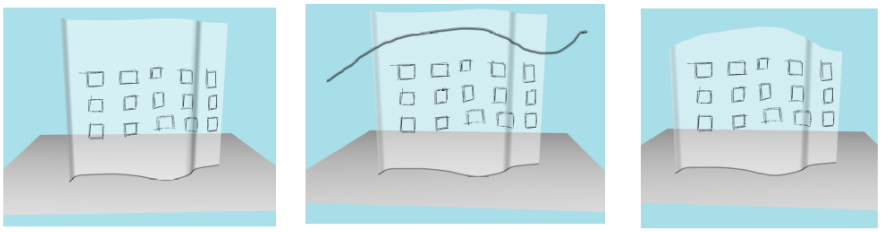


**Figure 3.8:** An example of cutting a canvas with a free-form stroke.

Since the user's strokes are stored in local coordinates relative to their parent canvas, an alternative post-cutting approach could have been to update the newly cut canvas' texture coordinates and thus, propagate this update to the associate map. Therefore, the user's strokes, such as the windows in Figure 3.8, would have been down-scaled but also shifted to the bottom of the canvas to maintain the local coordinates in the updated local frame. However, since we create each 3D canvas with the same height, we thought it was less distracting for users to simply cut the piece of surface they considered extra instead of seeing the structure changed after each cut.

Now that we have presented all our sketching tools, we will focus on the importance of having a nested structure for the visualization and the preservation of the design consistency throughout the edits.

### 3.4.2 Nested structure

As illustrated in Figures 3.3 and 3.7, we specifically account for the fact that buildings are composed of nested elements in a global way:

- inner floors are nested inside a closed 3D canvas,
- balconies on facade details are nested on the facade surface.

Although facade details are not geometrically nested within the rough surface facade, their footprints are. This property of nesting, in a broad sense, enables us to represent a 3D sketch by a hierarchical structure.

**Concept of NEM** An important feature of our system is that most of the geometric elements of a building can be designed from a series of footprints sketched on free-form canvases and extrusion operations. For instance, a footprint drawn on the ground surface will lead to the creation of external walls, facade details can be extruded from strokes on the facade, and internal walls will result from footprints sketched on different inner floors.

Therefore, the user is recursively sketching and creating a hierarchy of nested 3D canvases and associate maps, each new canvas being created from a sketched footprint. As previously explained, we defined as nested child canvases, both the geometrically nested ones such as inner floors and the ones used as facade details such as balconies or protruding windows. In addition, and as described in Section 3.5.2, all footprints can be interactively deformed based on the map lying on their support canvas, and, thus, any child canvas may dynamically evolve, allowing the user to visually explore alternative designs. To express their dynamic nature enabling exploration, we call this new type of 3D sketch: *Nested Explorative Maps* (NEM).

**NEM representation** In practice, a NEM can be represented by a hierarchical structure between 3D canvases, their associated maps, and the footprints drawn on them, as depicted in Figure 3.9. Starting with a ground surface (the first 3D canvas) and its map, the user can enrich the map by sketching free-form strokes or drawing some footprints curves to define new 3D canvases. In particular, a footprint can be seen as the parent element of new child canvas+map, created from the extrusion of the curve along a specific direction. Thus, all the operations applied to the ground surface can be done on any newly created canvas+map with the extra feature to create inner floors inside any closed 3D canvas.

Using this hierarchy benefits both visualization and editing tools as the user can display or hide any part of the hierarchy to sketch on any exterior or interior canvases from the same

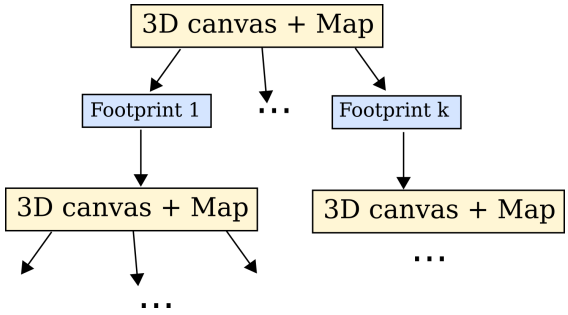



**Figure 3.9:** The hierarchical representation used for *Nested Explorative Maps* (NEM): Each 3D canvas and its associated map can be the parent of an arbitrary number of footprint curves, each serving as the basis for a new pair of 3D canvas and map.

viewpoint as depicted in Figure 3.7 middle and right. Moreover, this hierarchical structure helps us maintain design consistency throughout the edits, thanks to the attachment of sketched strokes, nested footprints, and maps onto the surface of the parent canvas. In particular, this provides great freedom during the creative process, as any creation or edition steps, including the deformations of various options suggested in the design, can be done in any order.

That said, we will now focus on the core feature of our system, which is the exploration of alternative options depicted by the sketch.

## 3.5 Interactive exploration of alternative options

A key point of our method is to let users sketch strokes without any simplification or smoothing steps, enabling them to represent uncertainty and alternative options, thanks to over-sketching. Based on the local density of ink/graphite, these strokes are used to model a continuous field stored within their map's texture image. In particular, during the user's interactive manipulation, this field is used to guide and attract the footprints towards high-density regions. Through this process, the user can interactively deform and explore the different options depicted on a map using a simple drag-and-drop gesture on a canvas's footprint. Indeed this operation can be applied at any level of the NEM hierarchy. Figure 3.10 presents an overview of our representation and exploration of uncertainty.

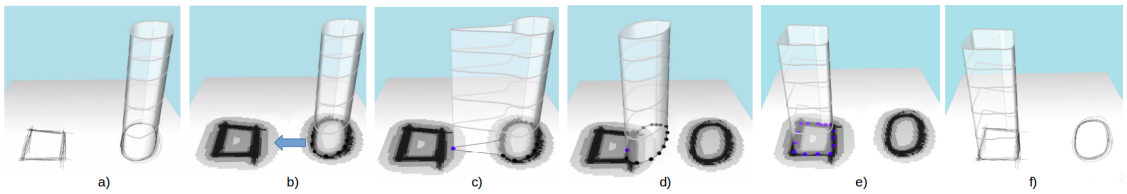



**Figure 3.10:** An example of representation and exploration of uncertainty: an initial cylinder shape with inner floors is interactively dragged by the user and automatically adapts its shape to a squared one, as sketched on the ground map.

### 3.5.1 Confidence field from a set of strokes

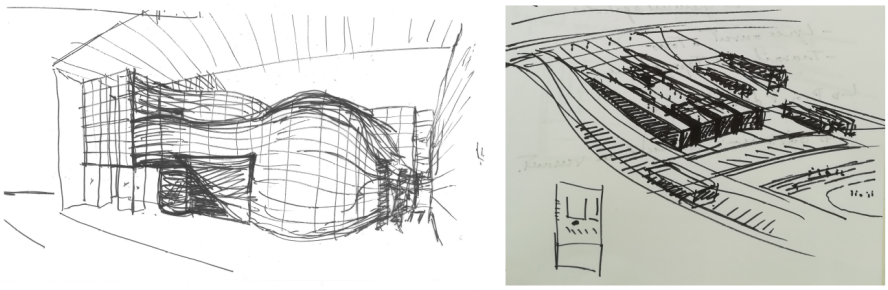


**Figure 3.11:** Examples of uncertainty representation in architectural sketches. @SCAU.

As illustrated by the architectural sketches in Figure 3.11, uncertainty is traditionally represented using several strokes of various densities and thicknesses. Through observations, we have estimated that locally, a high density of strokes expresses strong confidence of presence, in the sense that an element should be placed here, while sparser strokes express uncertainty.

To express this variation of density, we introduce a correspondence between dense strokes regions on a map and high field values on a 2D texture image. The latter, also called *confidence field* $F$, is added to the user's strokes to form the map on the top layer of a canvas. Footprint curves will then be interactively attracted and deformed towards high field values, corresponding to dense strokes regions on the map. The confidence field $F$ corresponds to a 2D grid of scalar values, stored as a texture layer and associated with a canvas. We used the natural $(u,v)$ parameterization given by the extrusion operation used to create the canvas: $u$ varies along the footprint direction, and $v$ varies towards the extrusion direction.

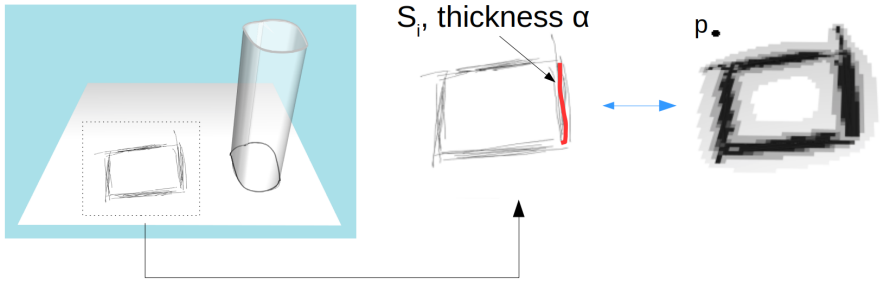



**Figure 3.12:** From the user's strokes to the confidence field: the contribution of a new stroke $S_i$ of thickness $\alpha$ is locally increasing the field values.

As depicted in Figure 3.12, at each grid point $p$, $F(p)$ is computed as the cumulative 2D field generated by the convolution of a kernel $\kappa$ along each of the user's strokes $S_i$ drawn on the current map:

$$F_i(p) = \int_{S_i} \alpha \, \kappa(p,s) \, \mathrm{d}s \tag{3.1}$$

where $\alpha$ is the thickness of stroke $S_i$ and the kernel $\kappa$ is defined as:

$$\kappa(p,s) = \frac{1}{d(p,s)^3} \tag{3.2}$$

$d(p,s)$ being the distance between the grid point p and a point $s$ of $S_i$.

Generating this field as a convolution along a curve allows us to create a smooth field and blend the contribution of each new stroke to the latter. In addition, the integral along a curve of this specific kernel has a closed-form solution ([CH01]) thus, in practice, F is computed incrementally, the contribution of each new stroke on a map is added to its confidence field.

In addition, since both the user's strokes and confidence fields are stored using their canvas local texture coordinates, any canvas can freely be deformed while maintaining the consistency of its map.

### 3.5.2 Plastic deformations of footprints and canvases

As illustrated in Figure 3.10, the user can trigger at any time the exploration of alternative options by simply using a drag-and-drop gesture on any existing footprint. The latter will then be interactively deformed and thanks to our nested structure, this deformation will automatically be propagated to the associated child canvas as well as the corresponding sub-tree of the hierarchy.

We aim at modeling a hybrid approach for deformable footprint curves: while being interactively guided and deformed by the user's interaction gesture, the curve should be attracted toward close-by high confidence regions, expressed by high values of $F$. Footprints should also evolve differently depending on the type of deformations applied to them: they should tend to preserve their features such as length and curvature under small deformations, however, they should be able to accommodate more drastic changes of shape to adapt to larger deformations while remaining at its new equilibrium position when the user releases the footprint.

To model such plastic deformations, we used a set of mass particles connected by plastic springs, i.e., springs able to absorb all deformations that exceed a specified threshold by changing their rest length if needed. More precisely, we set the particles with associated mass, position, and speed parameters on the footprint control points (as depicted by the spheres in Figure 3.14).

Plastic springs are set to connect neighboring particles, as well as second neighbors, i.e., particles sharing a common direct neighbor, to ensure better shape preservation when no strong force is applied.

In addition to the usual elastic and friction forces, particles are subject to an attraction potential given as:

$$P_{attraction}(p) = \exp\left(-(F(p)/\sigma)^2\right) \tag{3.3}$$

Note that the potential smoothly decreases towards 0 in the vicinity of the user strokes.

Similarly to deformable contour approaches [KWT88], we consider the associated force that derives from this potential as $F_{attraction}(p) = -\nabla P_{attraction}(p)$.

Lastly, the user interaction is modeled as a hard positional constraint using a drag-and-drop gesture applied to a particle.

To sum up, in our proposed model, when small deformations are applied to a footprint curve, the described forces lead to a standard elastic behavior, that is, a smooth deformation of the curve, which attempts to preserve its shape while being attracted to local high confidence values. In contrast, under larger deformations, the footprint can undergo extreme spring-length elongation or contraction thus its springs will switch to their plastic behavior by absorbing the deformations. This is done through a change of their rest length while the associated elastic force remains at zero.

Thanks to this mechanism, the rest shape of the footprint curve can evolve. In addition, subsequent deformations will just act the same way on the new curve, enabling users to go on exploring options until they are satisfied.

Note that the deformation associated with these plastic springs is efficiently computed in the 2D parametric space. After each animation step, the corresponding 3D positions are displayed on the canvas.

Finally, to enrich the exploration of options and avoid unwanted deformations of some part of the footprint, we introduce two extra specific behaviors: a glue mode (the pin button in the toolbox) to fix the position of some particles and then allow some precise local deformations, such as to explore the possibility of a building extension, as illustrated by Figure 3.13; a global rigid translation of the entire curve when the displacement of the particles exceeds a certain threshold.

Note that this global displacement can take place before the user releases the footprint curve and thus before the action of the plastic springs, as depicted in Figures 3.10 and 3.14.

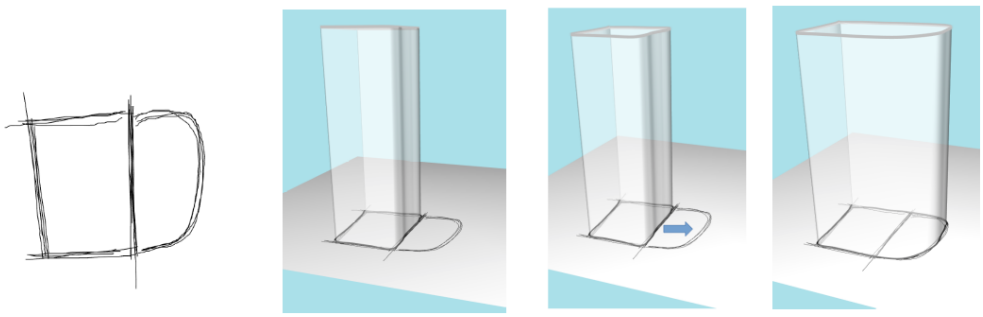


**Figure 3.13:** Uncertainty representation for the footprint of a canvas and exploration through interactive local deformation, which gets locally attracted to a new part of the map.

To detail the displayed Figures, Figure 3.10 presents an example of a deformation applied to a nested structure. First, the user sketches two geometric shapes (a square and a circle) on the ground surface and over-sketches them both. Then, the user draws a footprint to model the circular tower, inside which inner floors are added. Through a drag-and-drop gesture on some particles of the footprint, the entire NEM structure is progressively updated to a new equilibrium position to turn into a square-shaped building. As illustrated in Figure 3.14, such plastic deformations can also be applied to child elements on a 3D canvas where a protruding window is moved and deformed from a circular basis shape to a rectangular one. Finally, Figure 3.15 presents other examples of uncertainty representation and the exploration of alternative options.

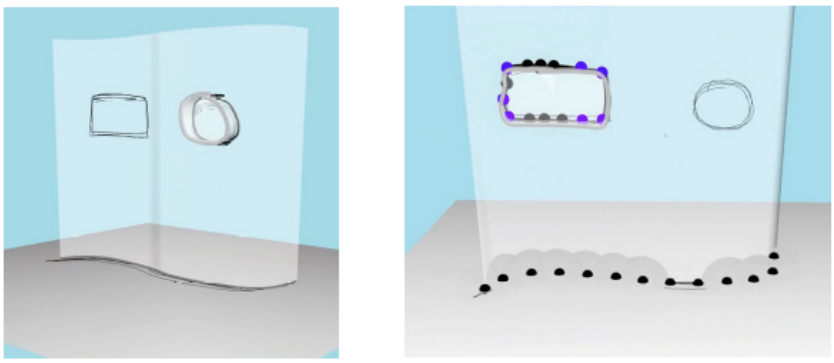


**Figure 3.14:** An example of representation and exploration of uncertainty: an initial cylinder shape with inner floors is interactively dragged by the user and automatically adapts its shape to a squared one, as sketched on the ground map.

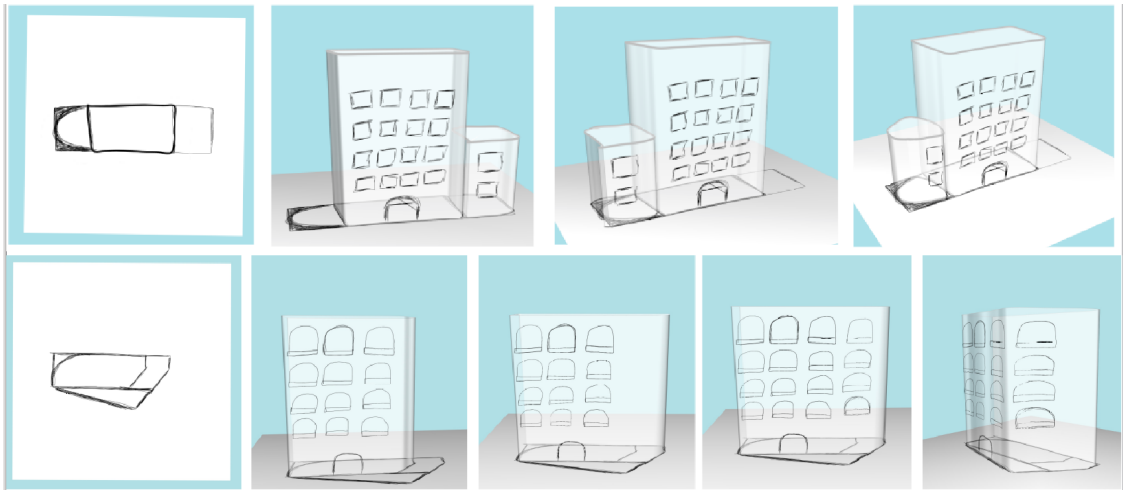


**Figure 3.15:** Two examples of uncertainty representation and the exploration of alternative options.

## 3.6 Results & Validation

For our system, we developed a prototype in WebGL Javascript using the `THREE.js` and `Orbit Controls` libraries. Using this programming language enables us to propose an interactive system running on a computer and any touchscreen device. In particular, some of the presented examples were performed using a mouse on an Intel(R) Core(TM) i7-7920HQ processor at 3.10 GHz while the user study was done on a WACOM MobileStudio Pro tablet.

Our software, also serving as a prototype to illustrate our method as well as for the user study, is available at: https://www.lix.polytechnique.fr/geovic/software.html

In addition to the previous illustrations, we provide below some screenshots (Figure 3.16) of the accompanying video to highlight the potential of NEM for easy prototyping without any specific editing pipeline.

Starting from the last step of Figure 3.4 (also taken from the video) during which the user had turned a nested squared shape building into a nested circular tower (a), now he or she is first trying to explore the remaining option depicted by the three branches on the ground surface. From a drag-and-drop gesture, the footprint and thus the underlying canvas is first deformed (b) before following a rigid global translation (c). After landing on the desired option, the plastic springs undergo a classical elastic behavior while being attracted to the high density of strokes. The user can then apply other drag-and-drop gestures to lead the shape transformation towards the desired result (d-e). To have access to a hidden canvas, the user can set the display of the closest canvases to off (f-g). It is now possible to draw a footprint (i) and create a new child canvas on the second floor (j). As this new canvas is defined with the default height, the cutting mode is used to define the cut from a sketched stroke (k-l). Finally, after all these steps, the resulting NEM is presented in (m-n).

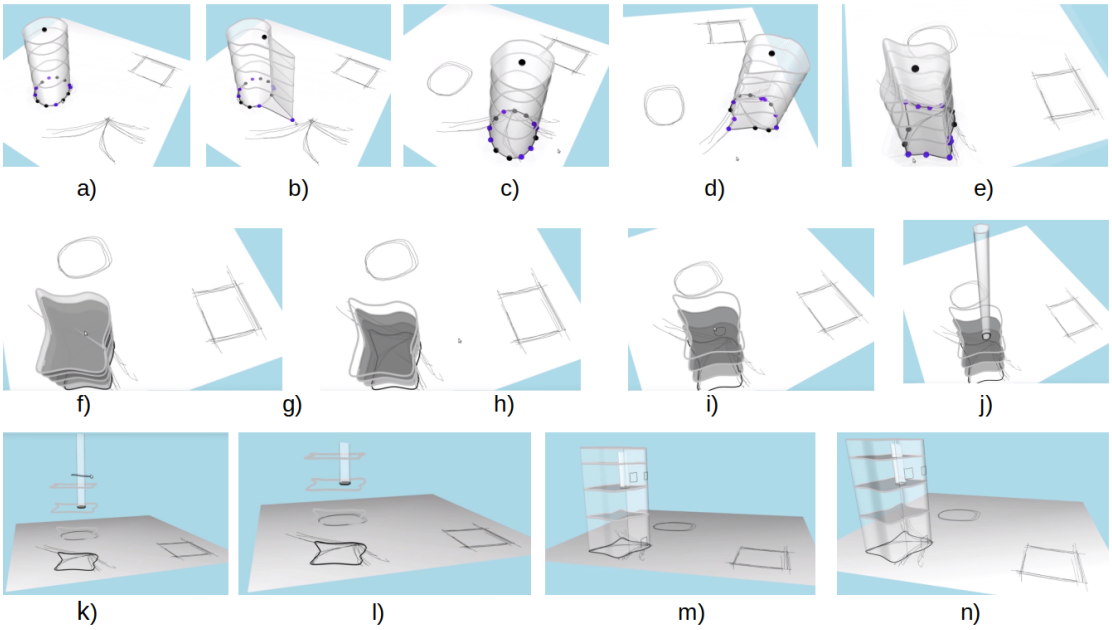



**Figure 3.16:** Screenshots from an extract of the accompanying video depicting all the proposed features: (a-e) exploration of the alternative options depicted on the ground surface; (f-g) display on/off on 3D canvases to have access to hidden layers; (i-j) footprint drawing, followed by the creation of a new 3D canvas+map; (k-l) a cutting stroke and the resulting canvas cut; (m-n) the resulting NEM.

### 3.6.1 User study

*Experiment setup*

We validate the NEM system through a user study conducted with professional architects of the SCAU agency in Paris. Our goal was to validate the four criteria established during the pre-study (Section 3.2), in addition to two additional hypotheses comparing NEM with the current digital software. We added these hypotheses to validate the convenience of our tool regarding the existing digital solutions used in the profession. Below is a quick reminder of these criteria, as well as a presentation of our two additional hypotheses:

- **C1**: Immediate usability,
- **C2**: Coarse-to-fine design + no specific editing pipeline,
- **C3**: Free-form shapes and representation of uncertainty,
- **C4**: Exploration of alternative options,
- **H1**: Creating a 3D sketch on NEM is faster than using any existing professional software,
- **H2**: Our tool is best adapted to the creative design phase of a project than the industrial software our users are familiar with.

Each session of this study was composed of two main parts: a creative part on a WACOM MobileStudio Pro tablet for users to experiment with our prototype while keeping the interaction as close as possible to the paper medium; a survey section for them to express free comments as well as providing some opinions regarding our criteria and hypotheses.

Our creative phase consists of three stages:

1. an explanation of the concept followed by an interactive demonstration (5 minutes);
2. a learning phase whose objective was to reproduce a basic model presented during the demonstration (5 minutes);
3. an exercise phase aiming at creating an imaginary model (5 to 10 minutes).

For the final step, we let the architects choose whether to create their model from scratch or to be inspired by examples of creative architectural projects containing free-form buildings that we thought would be difficult to model using standard industrial software. These visual references are available in Appendix A. During this experiment, the system was regularly saved as well as some screenshots taken.

Following the creative phase, we asked each user to fill out a paper sheet (also provided in Appendix A) to give scores from 1 (strongly disagree) to 5 (strongly agree) to validate or refute our different criteria and hypotheses. In addition, for our hypotheses, we request the user to precise the most frequently used software to compare our prototype with it.

*Results of the user study*

This study was conducted by 17 users, including 13 males and 4 females, from 23 to 57 years old. This group included 2 students in architecture and 15 professional architects, with experience varying from 6 months to 40 years. 16 of them were right-handed, and one was left-handed. Figure 3.17 presents a summary of the users' profile. In addition, each session (without the explanation part) lasted about 15 minutes.

Figures 3.18 and 3.19 depict some results created by the architects during the user study. In particular, Figure 3.18 illustrates the progressive construction of a 3D sketch.

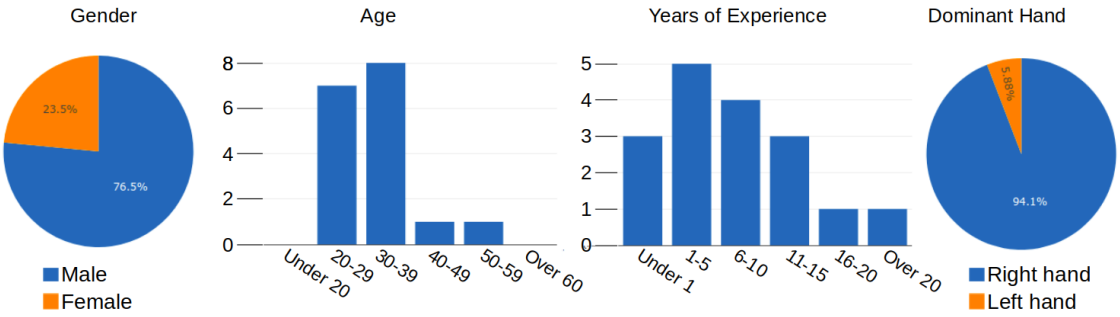



**Figure 3.17:** Summary of the users' profile for our user study.

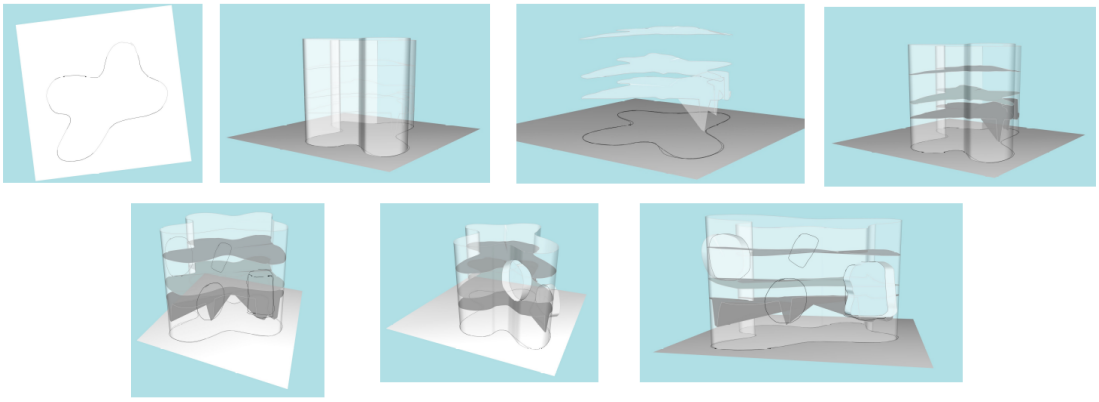


**Figure 3.18:** The progressive construction of a 3D sketch by an architect, during the user study.

From the architects' feedback, our tool was mainly perceived as an original and impressive "augmented paper" able to replace the paper and pen medium, as well as any digital software during the creative design phase of a project. They particularly appreciated the direct relationship between sketching on a tablet (also serving as a display screen) and the resulting 3D model, which provides immediate visual feedback, in addition to the general freedom to quickly model even complicated shapes in 3D. Moreover, our system was found to be playful and very convenient to draft ideas, explore and communicate them.

To validate or refute our criteria, we compute the mean and standard deviations of the answers obtained for each question. Figure 3.20 presents the global result of this survey with, in blue the mean and in black the standard deviation of the answers obtained. On average, the results for the four criteria were pretty good which permitted us to validate all of them.

We intentionally separated our hypotheses as the users' answers depended on their preferred software. In particular, more variations can be found regarding the habits of the users with respect to their preferred software and especially to determine if NEM was better for creation.

As specified above, the architects highly appreciated the possibility of directly drawing and modeling in our system using a touchscreen device. In addition, they were able to achieve their

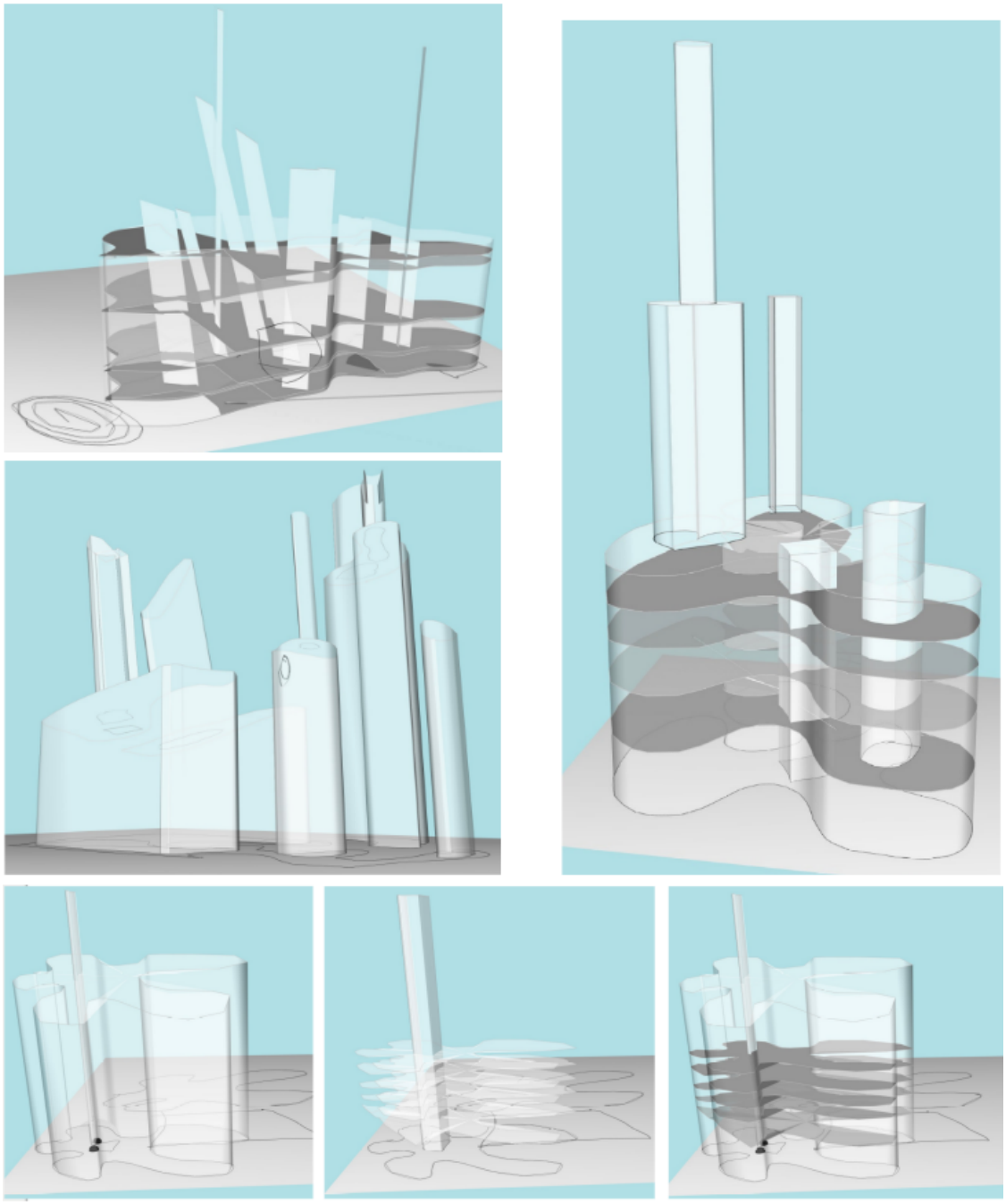


**Figure 3.19:** A few creative designs conceived during the user study's exercise phase.

first designs in less than 15 minutes while using all the operations allowed by our system. Our criteria (**C1**) was thus validated.

Users validated that a coarse-to-fine design (**C2**), combined with a progressive and simultaneous design of the exterior and interior parts of a building, was possible. They especially highly appreciated the possibility to visualize, at the same time, both the exterior and interior

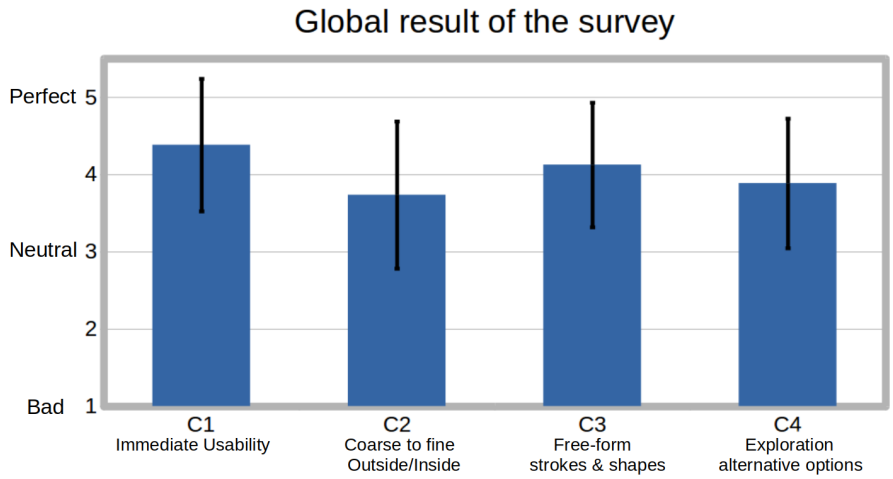



**Figure 3.20:** The global result of the survey relatively to our criteria.

parts, thanks to our semi-transparent canvases. However, two participants noticed a lack of precise dimensions or scale.

Most of the users appreciated the preservation of the original strokes and the possibility to represent uncertainty (**C3**). Accustomed to their digital software that replaces a stroke with a simple vector shape, the architects found it very interesting to be able to keep their rough strokes, which also allow them to create free-form shapes as illustrated in Figure 3.19. In addition, they mentioned that keeping the rough aspect of a paper sketch was very important to communicate, sketch ideas and approximate a shape with several strokes.

Finally, the ability to explore multiple options was considered a very positive addition, bringing more freedom to the modeling process. Furthermore, one architect even pointed out that the combination of the elasticity and plasticity behavior was well adapted to their needs.

For the next part, we separated the users' answers relatively to the name of the software they precised in the paper survey and applied the same process to display the results, in blue the mean of the answers and in black the standard variation. The number next to a software's name specifies the number of users that compare it with our prototype. Figure 3.21 depicts the results for our hypothesis **H1** on how creating a 3D sketch is faster than using any existing professional software. Figure 3.22 presents the answers for **H2** to determine whether NEM is more adapted for the creation than the current software.

Even though our prototype system proposed much less functionality than the professional software, the participants were able to achieve their first designs in a couple of minutes and even when using all operations allowed by our system, which validated our hypothesis (**H1**). In

addition, several participants qualified some of the functionalities as intuitive, such as the cutting operation and the creation of a floor inside a building.

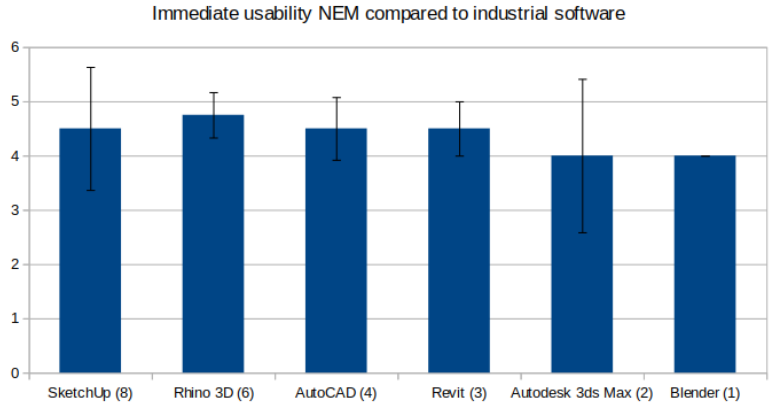



**Figure 3.21:** The comparison of immediate usability between our prototype and the users' preferred software.

Determining if NEM was better for the creation as the existing software had been more open to discussion, as the results were lower on average in addition to having greater variations. We could note that these deviations were largely influenced by the habits of the participants relative to their preferred software. From Figure 3.22, one could remark that the score from the participants using the software Rhino 3D was associated with the lowest value, and such users considered their usual software as equally creative as the NEM approach. In comparison, the immediate usability of NEM was homogeneously acknowledged, independently of their habits, as shown in Figure 3.21. During this user study, we also noticed that professional architects were not only using standard BIM tools such as Revit but also more generic 3D creation tools from 3DSMax to Blender.

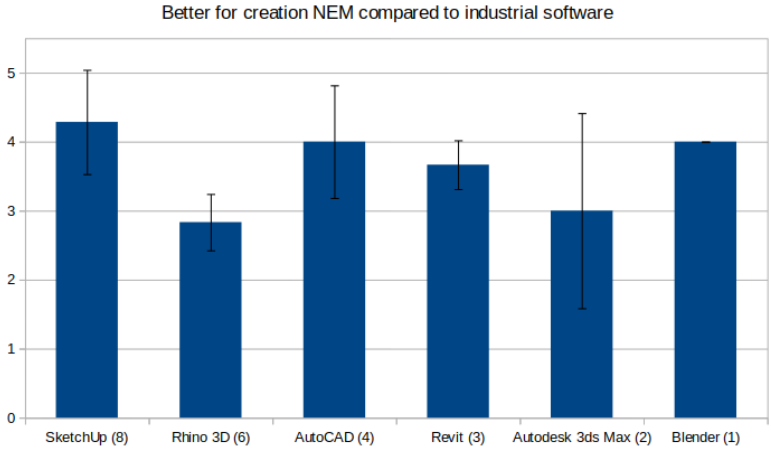



**Figure 3.22:** Comparison as a creative tool with the preferred software chosen by each participant. The indication below each bin points out the number of users who compared NEM to this specific software.

To conclude this user study, we remarked that the participants' experience and position in the agency influenced their perception of the tool. While we had too few participants to develop a valid quantitative study, we still observed the following behavior: experienced architects, who had been subject to a longer training with pen and paper, were more enthusiastic about the general idea of the tool, and especially the idea of being able to explore uncertainty. Moreover, they strongly encouraged further development of the tool towards creativity, for instance, allowing the creation, exploration, and progressive refinement of arbitrary, free-form shapes. In contrast, while architecture students and beginning professional architects particularly appreciated the effectiveness of the tool compared to their usual software to generate draft buildings, they less frequently mentioned the importance of being able to represent and explore uncertainty.

### 3.6.2 Discussion and limitations

While our system has been generally well perceived by users, we are aware that our current implementation could be optimized and that several functionalities were missing. For instance, several users complained about the few seconds of latency during the exploration of uncertain options, probably due to our CPU implementation of confidence maps. The models we chose to compute these maps and our deformation of surfaces also impacted the expressiveness of our system. In fact, in our method, all the strokes sketched on the same 3D canvas are part of the same confidence field. In particular, the user could prefer to use each stroke as an independent, mutually exclusive option, even in the case of two intersecting strokes. Similarly, while hatching strokes are common representation in sketched design to express specific information such as orientation or curvature, we did not implement any detection mechanism to handle them. Finally, although the architects had stressed the importance of designing a model in the context environment by importing external data such as height fields, this was not implemented in our prototype. Indeed, while mapping a texture on a surface is quite common in Computer Graphics, it would require another interactive tool to help map such data onto a 3D canvas, especially at the appropriate orientation and scale.

In order to improve our system while keeping the main features, some extensions could be considered. First, allowing the extrusion of a 3D canvas along a free-form path drawn by another stroke, like in SweepCanvas [LLZ*17], would highly extend the potential of our system to more complex architectural pieces, such as the Sydney Opera House. Second, extending our cutting operator to create holes inside a canvas could improve the creative process and even be a first step towards the possibility of moving doors or windows while shaping a building. Third, enabling the user to dynamically adapt the sampling of the deformed footprint, or change the type of deformation, could allow a wider variety of shapes.

Lastly, while our prototype can be very useful during the creative design stage to directly explore ideas by deforming 3D models, the lack of scale can be problematic for the next design steps. In addition, there is currently no possibility to export the resulting 3D sketch into any existing software while keeping all the information. When discussing with architects the adoption of our tool by professionals and on a larger scale, the easiest solution would be to update our current implementation to insert it as an extension of existing prototyping software such as Rhino 3D. Thus, standard quick prototyping methods could be leveraged with confidence fields stored as texture maps on top of surfaces, allowing to navigate through uncertain options.

## 3.7 Conclusion

We proposed a new type of 3D sketches dedicated to the design phase of architectural projects, which is hardly covered so far by standard industrial tools. Based on a pre-study to identify the needs of architects, we were able to introduce the concept of *Nested Explorative Maps*, allowing the recursive creation of a 3D sketch from the user's strokes while maintaining the sketch readability and offering an exploration of alternative options visually suggested on the sketch.

Our tool is the first, to our knowledge, to allow users to interactively explore the different design options represented in a 3D sketch. We do this by assigning a plastic behavior to our 3D canvases and allowing the user to move their footprints on a map where the density of the strokes triggers an attraction. As validated in our user study, being able to represent uncertainty and navigate between different options is a real improvement in the early stages of conceptual design.

Another salient contribution of this work was to provide a tool enabling for sketching not only the outside geometry of a shape but also internal structures and details. Although mandatory for architectural design, such nested structures could serve as inspiration for sketching systems in other domains—such as in biology, as discussed later in this thesis.

**Future work:** While our system has provided important functionalities to help architects prototype ideas during the early stages of design, it does not allow an architect to draw a sample of an element such as a window, or even a series of them, to generate it elsewhere or in an extended domain. Indeed, architectural models are often composed of repetitive elements such as windows on the facade or even similar interior layouts, which can be tedious to sketch one after the other. While implementing a simple copy/paste operation will be easy to add, analyzing and synthesizing a sketched distribution in real-time will require addressing new challenges such as extracting individual elements from of a sketch, finding an efficient representation of the underlying distribution as well as providing a real-time synthesis method based on this distribution.

In the next chapter, we focus on a tool dedicated to the creation of repetitive content. Specifically, we discuss the synthesis of arrangements of shapes from a sketched input. In particular, this sketch can be composed of bounded or unbounded shapes. We have specialized our approach on anisotropic distributions of elements, i.e., shapes aligned along a specific direction. From this input, we propose two synthesis methods, the first one in a 2D extended domain (Section 4.4) which can be used, for instance, to propagate windows on a building facade; the second one in a 3D domain (Section 4.5), to directly immerse the user in a 3D environment inspired by the provided sketch.

# 4

# Creation of repetitive contents: Anisotropic distributions of shapes from a sketch

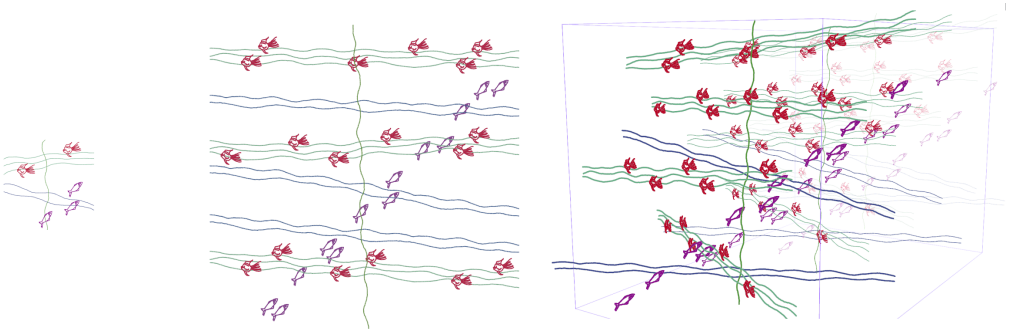



**Figure 4.1:** Based on a few perceptual and depiction hypotheses, our method extends an input sketch (a) into 2D (b) or 3D (c) distributions. Both bounded shapes and unbounded curves are seamlessly handled.

Compared to the previous and following chapters, this work tackles the creation of general yet self-similar content with an intuitive and flexible tool for real-time synthesis of anisotropic distributions. We take as input a sketch interactively drawn by the user and composed of bounded and unbounded shapes. Our fine-to-coarse method successively reduces the input strokes into linear structures to build our new data structure, called *Support Structure Hierarchy*. Based on this hierarchy, we propose a coarse-to-fine synthesis method in an extended 2D domain. In addition, we also introduce a 3D embedding of this extended distribution, enriched with some simple perceptual adjustments, to provide an illusion of depth and allow users to immediately explore a 3D environment inspired by their sketch. The user may tune the default settings of the synthesis environment and also return to the sketching interface to edit the input and perform a new synthesis. As shown by a user study, the generated distributions equal the perceptual performances of the best synthesis methods for 2D anisotropic distributions while expanding them to the case of unbounded shapes and extending sketch-based modeling to 3D anisotropic environments.

## Résumé en français

Contrairement aux chapitres précédent et suivant, ce travail porte sur la création de contenus plus généraux, mais auto-similaires, à travers un outil intuitif et adaptable, dédié à la synthèse, en temps réel, de distributions anisotropes. À partir d'un croquis dessiné par un utilisateur sur notre interface et composé de formes à la fois bornées et non bornées, notre méthode d'analyse suit une approche des détails au grossier, en réduisant successivement les traits du dessin d'entrée en structures linéaires, afin de construire notre nouvelle structure de données, appelée *Hiérarchie de Structures de Support* (Support Structure Hierarchy). En se basant sur cette dernière, nous proposons tout d'abord, une méthode de synthèse du grossier aux détails, dans un domaine 2D étendu. En complément, nous présentons également une intégration 3D de cette distribution élargie, enrichie de quelques paramètres perceptifs simples, afin d'offrir une illusion de profondeur et de permettre aux utilisateurs d'explorer immédiatement un environnement 3D inspiré de leur croquis. De plus, l'utilisateur peut ajuster les paramètres par défaut de l'environnement de synthèse, revenir à l'interface de dessin pour modifier l'entrée et effectuer une nouvelle synthèse. Comme démontré par notre étude utilisateur, les distributions générées par notre système égalent les performances perceptives des meilleures méthodes de synthèse pour les distributions anisotropes 2D, tout en les élargissant au cas de formes non bornées, et en étendant la modélisation par esquisse aux environnements 3D anisotropes.

### Contents

## 4.1 Motivation

Structured distributions have been used intensively in 2D for decoration purposes, from mosaics and wallpapers to the distribution of windows and architectural decorations on building facades. Such anisotropic distributions of shapes are also ubiquitous in natural environments, in 2D as well as in 3D, and at multiple scales: from fibers and cellular organisms at microscopic scales to seaweeds, schools of fishes, and alignments of trees at a larger scale. The perceived structure emerges from the anisotropy of these shape distributions. In particular, the specific ranges and variances of perceived orientations, both in terms of salient shapes and alignments, convey their unique visual appearance, as illustrated in Figure 4.2.

**Figure 4.2:** Examples of 2D and 3D distributions of anisotropic shapes, serving as inspiration for our work. From left to right: illustration of windows composing a building facade; rods; seaweeds; collagen fibers.

This work tackles the interactive, sketch-based design of such 2D and 3D anisotropic distributions. The idea is to extract the underlying structure of an input 2D sketch and extend it to a larger 2D or 3D domain, leading to an illusion of spatial extent in a perceptually consistent and non-repetitive way.

While example-based synthesis has been extensively studied in recent years (see Chapter 2.2.2), existing methods have mostly focused on point distributions. They have achieved statistical accuracy for noise models or through continuous representations such as pair correlation functions or probability density functions. In addition, shape connectivity has also been targeted via neighborhood metrics and energy optimization. However, the few methods that tackle anisotropic distributions of shapes rely on multiple point samples or proxy geometries. To the best of our knowledge, and as our comparative study in Section 4.6.2 will confirm, none of them has been able to efficiently capture the underlying structure of the input distribution, namely the inter-dependencies and variations in space and orientation within a distribution of bounded and unbounded shapes. Furthermore, sketch-based synthesis of 3D anisotropic distributions, such as those in Figure 4.3, has never been attempted.

In this chapter, we present a real-time analysis and synthesis method for structured anisotropic distributions of shapes. The real-time performance of our method allows us to apply it to a mere 2D sketch, interactively drawn by a user, which may represent any collection of simple bounded shapes as well as elongated (fiber-like) unbounded strokes. Our synthesis framework and visual interface allow users to seamlessly explore a larger spatial extent around their sketch and navigate within the resulting distribution. To emphasize the interactivity of our tool, we not only let users tune the default parameters of the synthesis environment, but we also let them return to the sketching interface to edit the input before performing a new synthesis. This makes it an interesting sketching tool for both creation and communication, for instance, in the domain of biology, as illustrated by our result in Figure 4.3.

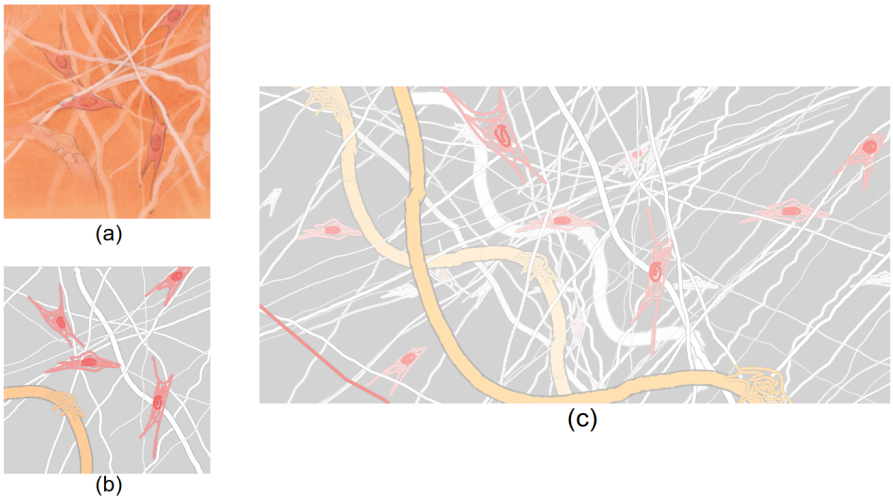



**Figure 4.3:** a) Biological illustration depicting individual cells that navigate in a 3D distribution of fibers; b) Input sketch inspired from a); c) Snapshot of the 3D virtual world resulting from our 3D synthesis method.

Real-time analysis and synthesis of distributions require an efficient representation, encoding both local and global correlations between elements. Our insight is to introduce a compact encoding for anisotropic distributions, called the *Support Structure Hierarchy*, where individual supporting structures are lead directions of alignment or line skeletons computed from the user strokes at various scales. This representation leads to a particularly simple and efficient multi-scale analysis of the distribution of orientations in the input sketch. It also allows for efficient domain extensions in both 2D and 3D spaces, leading to the interactive immersion tool that we used for illustration.

While both must preserve the input anisotropic distribution when the viewpoint does not change, our two synthesis methods face different challenges, namely avoiding unwanted overlaps when extending the user's sketch to a larger 2D domain; and inferring depth when a 3D distribution is generated. Therefore, our contributions are threefold:

- a fine-to-coarse analysis method that hierarchically clusters the user strokes into a Support Structure Hierarchy based on their proximity in position, orientation and alignment;
- a coarse-to-fine planar synthesis method that extends the pattern around the input sample based on the extracted hierarchy and a set of perceptual hypotheses validated by a user study;
- a second coarse-to-fine synthesis method that extends the pattern around the input sample both in 2D and 3D to create a 3D distribution of shapes, perceptually similar to the input sketch and inside which users can navigate.

We also present a user study conducted to validate our design choices and the perceptual consistency of our results.

## 4.2 Overview

### 4.2.1 Hypotheses on Depiction & Perception

Extending a sketched input requires making some assumptions about the user's depiction and perception of the resulting distribution. Our key hypothesis common to most sketch-based modeling systems is that users see their input as a *general view* of the 2D or 3D distribution they want to create. Therefore, we expect the input to include all the necessary information in a perceptually representative way. This leads us to three design hypotheses:

**(H1) Groupings and Alignments are meaningful** All alignments and groupings are intentional, especially they cannot be considered to be due to a specific viewpoint during the depiction of a 3D scene;

**(H2) Repetitiveness is explicit** All the shapes the user wants to see replicated in the output, are repeated in the input;

**(H3) Non-overlapping shapes should remain disjoint** Shapes that do not overlap in the input should not overlap in the output.

These three hypotheses are used as guidelines for our method at the design stage and then validated by a user study aimed at a broad public (see Section 4.6.2).

### 4.2.2 Creation and pre-processing of the Input Sketch

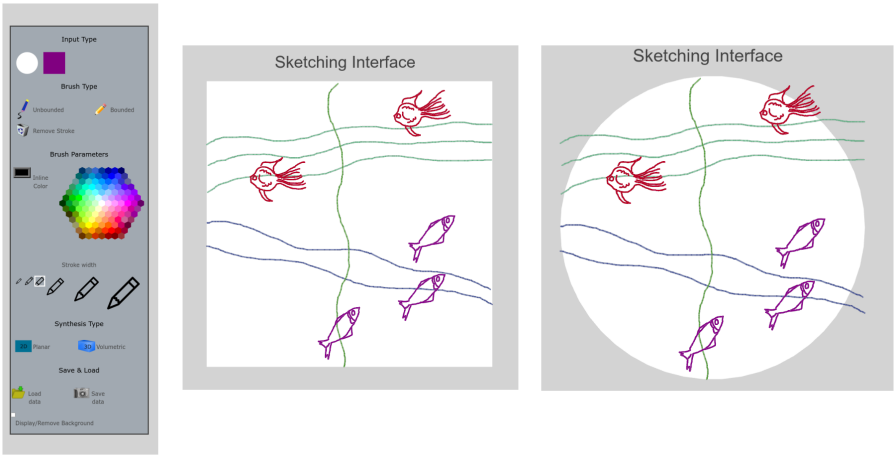



**Figure 4.4:** The sketching interface (in a)) includes, from top to bottom: the choice between a square or a circle input domain (in b) and c)); two pencils for drawing unbounded vs. bounded shapes and an eraser to remove an already drawn stroke; a color picker; a picker for stroke thickness; two buttons for launching 2D and 3D synthesis; and load and save buttons.

During a sketching session, the user interactively draws strokes on a 2D Input Space ($IS$) as illustrated in Figure 4.4. The input domain can be either a square domain of side $l_i$ or a circle domain of radius $r_i$. While squared input domains are common for texture or distribution synthesis, a circle input space gives the illusion of drawing and exploring distributions through the lenses of a microscope, which is interesting for our application to biological illustrations.

We provide two different pens to denote bounded and unbounded strokes. The first ones are limited to the dimensions of $IS$ and will be repeated as they are. The second ones are interpreted as extending beyond the input domain, either in both directions if both extremities reach the border of the $IS$, or in a single direction. In the case of 3D distribution synthesis, the output space is considered as the depiction of a cube—or a cylinder—of volumetric material. Therefore, we also allow unbounded strokes with both extremities inside $IS$: these strokes are then considered as strongly oblique, and their extension will mostly take place along the depth axis of the output domain.

**Strokes storage:** To avoid restricting $IS$ to a specific size, we analyze the input data within a normalized space ($NS$) being either a unit square or a unit circle. The input points $(x,y) \in IS$ are then transferred to $NS$, using: $X = \frac{x}{l_i}$ and $Y = \frac{y}{l_i}$, for a square domain, or polar coordinates for a circle domain.

We represent each sketched stroke by a set of data containing the coordinates of its points both in *IS* and *NS*, its color and thickness parameters, its type (bounded or not), and a principal direction computed from the *Principal Component Analysis* (PCA) of the coordinates of its points in *IS*.

The color is the only parameter left from the analysis, as we directly consider it at the final rendering stage. In particular, a shape can be composed of strokes of different colors.

### 4.2.3 Processing pipeline

During our fine-to-coarse analysis, we iteratively construct the *Support Structure Hierarchy* by successively clustering the input strokes by proximity, both in position and orientation. In reverse, the synthesis stage processes the Support Structure in a coarse-to-fine manner: the structures at the top of the hierarchy are first extended to the user-selected 2D or 3D larger domain, and the Support Hierarchy is then traversed top-down to the individual strokes, to generate the extended distribution of elements.

**Extraction of the *Support Structure Hierarchy*:**

Based on our perceptual hypotheses, our analysis consists in progressively extracting a fine-to-coarse hierarchy of support structures (the Support Structure Hierarchy) from the input strokes according to alignments and multi-scale repetitions in the input (see Figure 4.5). We first cluster the bounded strokes into *shapes* composed of one to several strokes and consider each unbounded stoke as an individual shape (**Level 0**). We then simplified the bounded shapes either into a *central point* or a *support segment* depending on their degree of anisotropy (Figure 4.5c). The central points and support segments are clustered according to both orientation and position to find alignments and then grouped into *fibers* (Figure 4.5d), forming the **Level 1** of the Support Structure Hierarchy. Other fibers are directly extracted from the unbounded strokes (Figure 4.5c'). To capture large-scale repetitions, fibers of similar orientation are clustered into *fiber medians* (**Level 2**, Figure 4.5f), which are finally grouped into *lead directions* (**Level 3**, Figure 4.5g).

During this process of clustering and hierarchical simplification, we progressively partition the input domain *IS* into a hierarchy of *ribbons* that express the variability of position of each substructure around its parent structure. We use this partitioning to allow for an appropriate degree of variability while avoiding unwanted overlaps in the synthesis stage. See Section 4.3 for details.

**Synthesis stage:**

Unlike most existing approaches, our method to synthesize distributions consists in directly replicating the local and global correlations between the input shapes encoded by our Support Structure Hierarchy. To avoid exact repetitions, this is done by instantiating each structure from top to bottom of the hierarchy while perturbing their positions within adequate *allowed areas*. We compute these areas to prevent unwanted overlaps between strokes belonging to the same lead

direction and at a low cost since no further overlap detection will be required. We defined our output domain by an enlargement of the input space by a scale factor $k$ ($k > 1$). For the 3D case, we complement this 2D extended space by a depth extrusion of parameter $h$. Although they rely on the same analysis, the 2D and 3D distribution syntheses pose different challenges, leading to distinct solutions.

*2D Distribution Synthesis* requires avoiding any unwanted overlap between the generated elements, otherwise, $H3$ (a.k.a, no additional overlap of shapes in the output) would not be met. Starting at the highest level of the hierarchy, we consider repetitions to clone structures and then compute the areas within which the child structures can be extended or generated without creating any unwanted overlaps. In particular, we introduce a coarse-to-fine adjustment method that fits the generated elements within a predefined displacement area at the price of slightly bending their supporting structures to avoid unwanted overlaps when extending the domain. This process is iterated down the hierarchy until the generation of the stroke distributions.

*3D Distribution Synthesis* addresses the challenge of inferring depth from 2D data. Starting with a simplified planar synthesis (with no avoidance of overlaps when elements should not be at the same depth), we rely on perceptual studies to embed this newly created 2D distribution into a cube or cylinder of fixed depth. During exploration, the resulting volume of 3D matter is replicated forward and backward to create an unlimited 3D distribution of strokes. The rendering is adapted to the distance to the camera to achieve intelligible 3D visualization with a good perception of depth, as illustrated in Figure 4.1, right.

### 4.2.4 Interactivity and refinements

During a session, the user starts by drawing the input on the sketching interface. To ease the creation of a more complex distribution, this interface has a menu that allows users to load data such as a background picture or an existing input. The user can also save the current state of a sketch at any time. When the input is complete, the user can perform one of two syntheses. On the synthesis interface, the user can display a menu to tune some parameters to change the default environment. For the planar synthesis, the main modifications concern the size of the output domain by updating the scale factor $k$ or the amount of curvature by changing the $\alpha$ parameter (see Section 4.4). For the 3D distribution, the user can modify the scale factor and also the 3D immersion type, the depth of a layer, the number of layers, and the shift range parameter between layers. The user can deactivate or activate the navigation and adjust its speed. For both syntheses, it is also possible to load the data from a previously saved synthesis and save the current one.

The main feature of our tool is to offer users two syntheses as well as the ability to return to the sketching interface at any time to modify the current input or perform a new synthesis.

## 4.3 Fine-to-Coarse Analysis

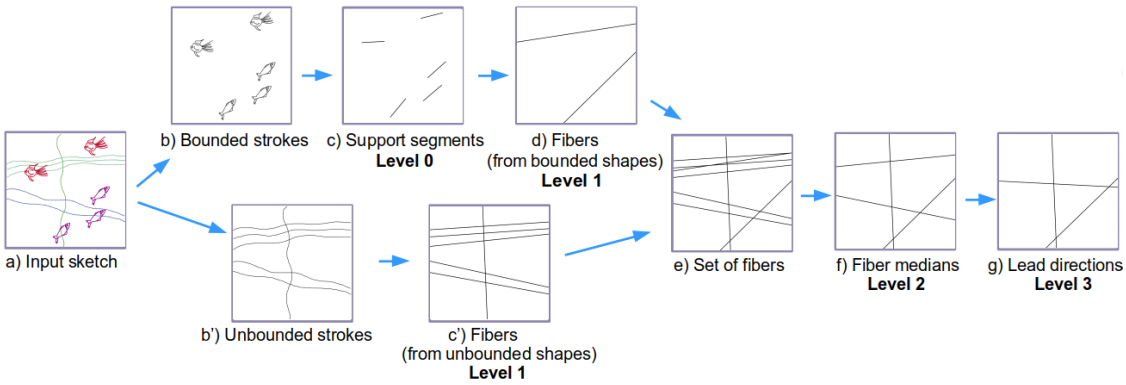



**Figure 4.5:** Processing pipeline for the fine-to-coarse analysis of a sketch into a Support Structure Hierarchy.

Our goal is to simplify the input data into one or several coarser structures carrying groups of strokes that are the closest in direction, position, and alignment. To achieve this objective, we introduce a new representation, called Support Structure Hierarchy, on which strokes can be wrapped, as illustrated in Figure 4.5. In addition to encoding the main features of the input compactly, this structure enables us to avoid high computational costs.

### Level 0: from Strokes to Shapes

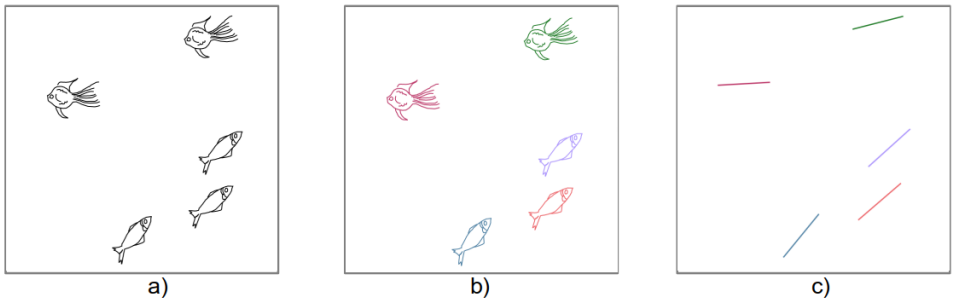



**Figure 4.6:** From Bounded Strokes to Shapes: a) input bounded strokes; b) clustering into shapes (random color per cluster); c) support segments **(Level 0)**.

In the first step and as shown in Figure 4.5 a) to b) and b'), the bounded and unbounded strokes are analyzed separately to extract supporting lines, which we process in a combined manner.

As users are authorized to draw shapes composed of several strokes, we start by clustering the bounded strokes by their position to retrieve the user's intended shapes. For this specific clustering, we explored two data representations: a centroid (point) approach and a bounding box (box) approach.

The first approach is to represent each stroke by its centroid. For the latter, we first compute the local directions between the adjacent points of a stroke to determine its curvature and whether the stroke should be considered opened or closed. For an opened stroke, we define the centroid as the mid-point of the extremity points, otherwise, it is the barycenter of the point coordinates. We then apply the *Mean Shift* algorithm to cluster the strokes with the *Euclidean distance* and a bandwidth computed from the mean of the pair-wise distances.

The other approach focuses on a coarser representation using oriented bounding boxes computed from each stroke's main direction and point coordinates. We describe each bounding box by its corners and two axes. Our distance metric between two bounding boxes is defined as follows. For each box, we project its corners along both its axis and the other box's axis to look at its minimum and maximum values on each axis. If these ranges of values intersect on all four axes, then, the boxes are considered to overlap and we set their distance to 0. Otherwise, we compute for each box the projection of its corners on the other box segments and take the minimum distance between a corner and its projection. Finally, if necessary, we compute the minimal distance between the corners of one box and the ones from the other box. We then apply a basic clustering algorithm using this distance metric and a threshold determined from the pair-wise bounding boxes distances.

While both of these approaches can provide the same output for most examples, some specific cases require to favor one over the other (see Section 4.6.4 for more details). For instance, for the detailed example above (see Figure 4.6), a bounding box representation can ensure that individual stokes representing details at the extremities of a fish are clustered with the corresponding fish main body's stroke.

From the output of the clustering algorithm, we reduced bounded strokes either into a single *central point* or into a single *support segment* depending on the degree of anisotropy of the shape. As illustrated in Figure 4.6, elongated shapes such as fish are reduced into support segments. The resulting set of support segments and central points is a first simplification of the input efficiently encoding the main directions and the approximate positions of the bounded elements. In addition, the unbounded strokes (if any) are considered individual unbounded shapes for further processing. This allows for separate processes for the fishes and the wavy curves in Figure 4.1.

### Level 1: from Shapes to Fibers

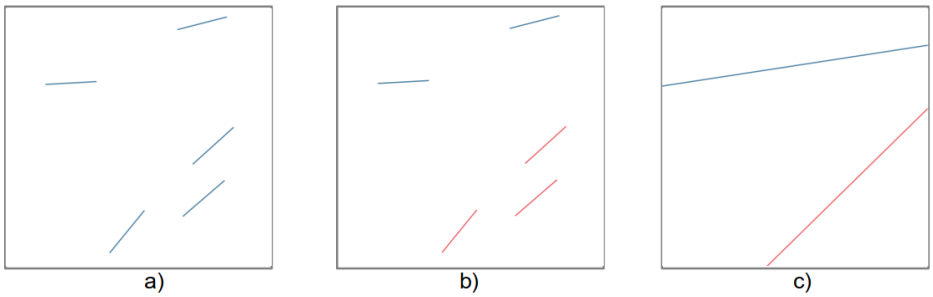



**Figure 4.7:** From support segments to fibers: a) orientation-based clustering; b) position-based clustering; d) representative fibers.

For a relevant distribution synthesis, we need to extract and preserve the underlying structure and correlations between shapes. To this end, we approximate each unbounded shape by the line that best matches its principal direction and position. This support line enriched with an thickness estimated to contain the entire shape defines a representative *fiber*.

For bounded shapes, finding such fibers requires an extra analysis to capture the anisotropic information, such as alignments (see Figure 4.7). As central points are not highlighting a dominant direction, we separate fibers coming from central points from the support segments ones.

*Central points into Fibers*

To determine fibers from central points, we first cluster the points by position using the *Euclidean distance* to group the closest ones. We then apply the *Principal Component Analysis* on each resulting cluster of points to detect alignments and obtain a set of support lines. We define the representative fibers by adding a thickness to each of these lines to ensure that all the underlying bounded strokes are included in this new structure.

*Support Segments into Fibers*

For support segments, we consider that the anisotropy information is already encoded in the main direction of a segment. Thus, we begin our reduction by first grouping the segments by orientation to determine which segments belong to the same anisotropic distribution. We do this by representing each segment by a point $(cos(\theta), sin(\theta))$ where $\theta$ is the angle between its main direction and the horizontal axis. To take into account the circular aspect of the angular data, we also consider the opposite point on the unit circle defined as $(-cos(\theta), -sin(\theta))$. The orientation distance between two support segments is thus defined as the minimum distance between a representative point on one support segment and both on the other. We then apply the

*Mean Shift* algorithm to cluster the support segments based on this distance. From each cluster, we determine a dominant orientation defined as the average of the orientations of its elements.

To reduce the position of a segment to a one-dimension value, we first simplify it into its midpoints (since the segments have already been clustered by orientation). Then, we use the linear representation $ax+by+c=0$ to compute the $c$ values of all lines parallel to this dominant orientation and passing through the midpoints of the segments. Finally, we apply the *Mean Shift* algorithm on each set of $c$ values to determine support lines, and, as usual, we associate a thickness to these lines to create fibers containing all its associated shapes.

### Level 2: from Fibers to Fiber Medians

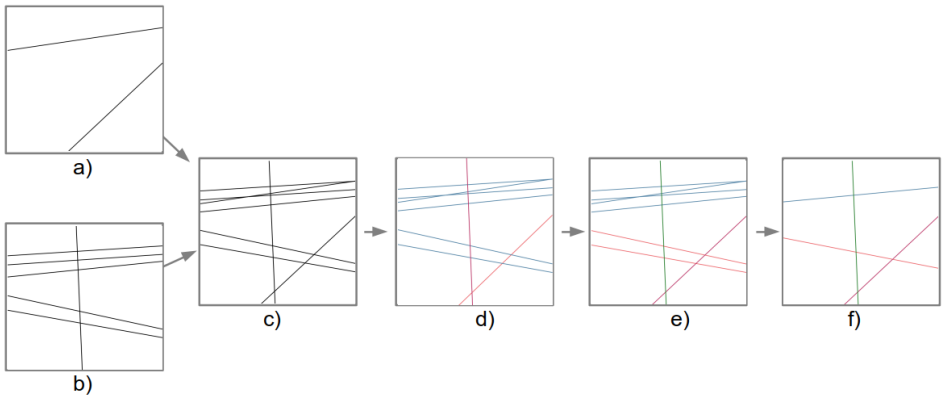



**Figure 4.8:** From fibers to fiber medians: a) fibers from the bounded shapes; b) fibers from the unbounded shapes; c) set of representative fibers d) first clustering on the orientation; e) second clustering on the position; f) reduction to fiber medians.

At this stage, our objective is to combine fibers from both bounded and unbounded elements, while ensuring that fibers with the same orientation and close positions will be grouped and thus remain close by at the synthesis stage.

As depicted in Figure 4.8, we apply a two-step clustering analysis, where the first step is a simple orientation-based clustering (like the one we described above to reduce support segments into fibers), and the second one makes use of a new perceptual distance between fibers to refine clusters as follows.

Our goal was to find a distance that takes into account both the position and orientation of the lines, in the sense that: the more parallel and close two lines are in position, the smaller the distance should be. As shown in Figure 4.9, such a distance can be defined by computing the intersection points of each line with the axes $X=0$, $X=1$, $Y=0$, and $Y=1$ and taking the minimum distance between the two intersection points that belong to the same axis. We use this distance to refine the orientation-based clusters into sub-clusters where fibers are perceived as

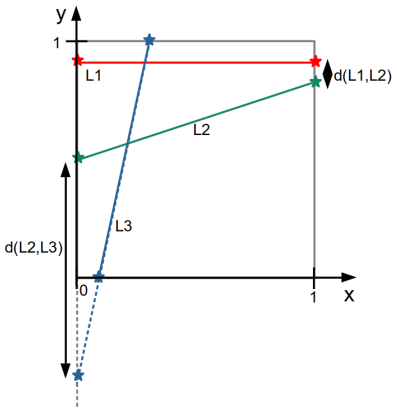



**Figure 4.9:** We compute the "perceived distance" between two fibers in a normalized input domain. It is defined as the minimal distance between their intersection points on any of the lines bordering the domain ($X = 0, X = 1, Y = 0, Y = 1$), which is extremely fast to compute (for each fiber, only the 4 values $y_{X=0}$, $y_{X=1}$,$x_{Y=0}$, $x_{Y=1}$ are needed). This distance accounts for both position and orientation and is defined even if the lines intersect in the domain. Here, $d(L1,L2) < d(L2,L3)$, which matches our perception.

similar since also located nearby. Each sub-cluster of fibers is finally stored as a *fiber median* defined as the average of of the parameters of the fibers grouped both in orientation and in position (see Figure 4.8). We also store the circular standard deviation of fibers belonging to the same fiber median for further use in the synthesis.

### Level 3: from Fiber Medians to Lead Directions

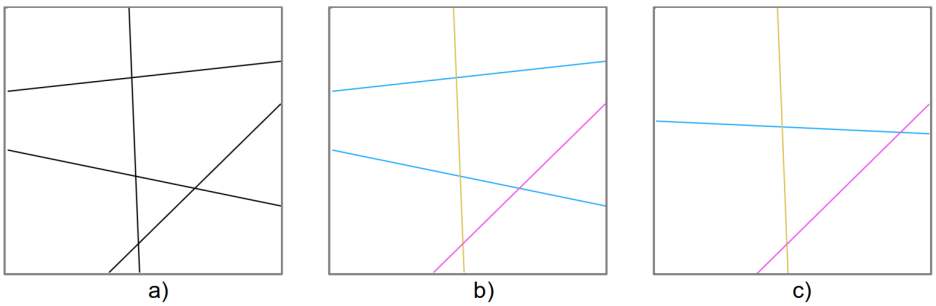



**Figure 4.10:** From fiber medians to lead directions analysis: a) fiber medians; b) orientation clustering retrieved from the previous reduction; c) reduction to lead directions.

To better capture the underlying structure of the input and reflect the user's desired pattern repetitions in the output, we consider a final reduction in our hierarchical analysis to group similarly oriented fiber medians. To this end, we retrieve the cluster from previous orientation-based clustering to determine the fiber medians belonging to the same orientation cluster. Each cluster is simplified into a *lead direction* defined as the average of the parameters in both orientation and position as depicted in Figure 4.10.

### Input domain partitioning

The last step of the analysis stage is to calculate the available space around each clustered shape, or *ribbon*, within their parent structure in the hierarchy. We call this space the *allowed displacement area*, as it will be used at the synthesis stage to add random displacements to repeated structures, providing visual diversity while avoiding unwanted overlap between shapes.

**Lead ribbons around lead directions** We first separate lead directions carrying a singularity, i.e., clusters consisting of only a single fiber median, such as the vertical seaweed or purple fish lead direction in Figure 4.11. These directions will not be replicated during the synthesis, according to our $H2$ design hypothesis on shape repetitiveness (see Section 4.2.1). Instead, we will simply extend them as well as their child structures to entirely span the output domain.

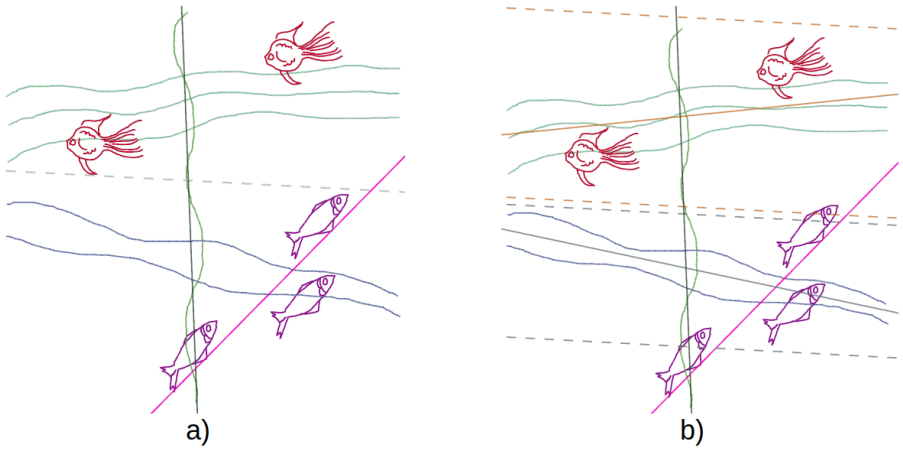



**Figure 4.11:** The creation of lead ribbons: a) Lead directions in dotted; b) Fiber medians (plain) and lead ribbons (dotted).

All other lead directions indicate a perceived repetitiveness in the input since they carry more than one fiber medians of similar orientation. We first partition $IS$ into *lead ribbons* parallel to the lead directions that guide the replication of child structures in a coarse-to-fine manner. We associate each non-singular lead direction with a local frame on which we can determine the height of its fiber medians' content in $IS$. These values are then sorted by increasing order to compute the distance between two adjacent ribbons. We thus generate lead ribbons around each fiber median by letting a gap of half the previously computed distance between two neighboring ribbons, as illustrated in Figure 4.13. In addition, we store all these gap values and two variables representing the extreme heights of the lead ribbons in $IS$. Finally, we associate with each new lead ribbon, both an index and a width defined by the difference between its maximum and minimum heights in the lead direction's local frame.

**Ribbons around support structures:** Then, we compute *ribbons* around fiber medians (see Figure 4.12 a)) by delimiting the area, which covers all the input strokes that belong to the same fiber median. We then partition these ribbons into *sub-ribbons*, computed around the fibers based on unbounded strokes to delimit the strokes' extension during the synthesis and thus avoid overlaps, as illustrated in Figure 4.12 b).

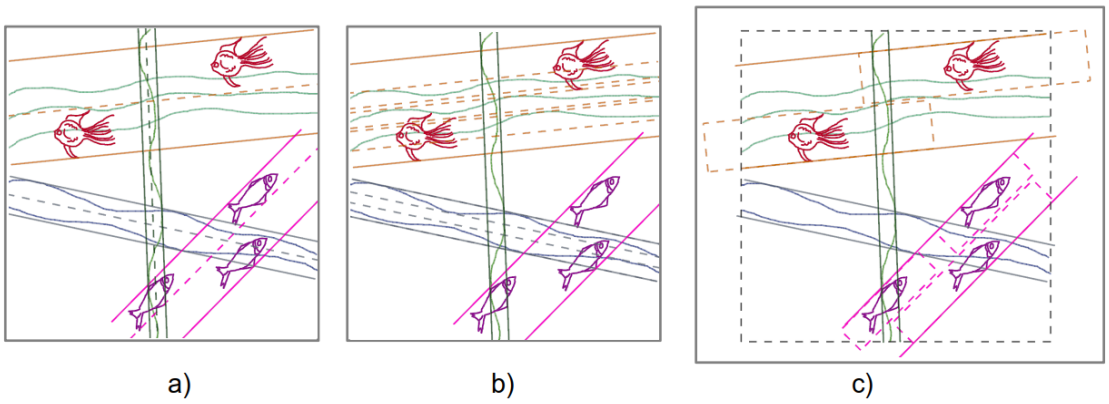



**Figure 4.12:** Input domain partitioning: a) Fiber medians (dotted lines) and their corresponding ribbons; b) Ribbons are partitioned into sub-ribbons (delimited by dotted lines) corresponding to fibers; c) Ribbons and displacement areas (delimited by dotted lines in color) corresponding to bounded shapes. The black dotted lines delimit the input space.

**Displacement areas for bounded shapes:** For bounded shapes, being limited to its fiber's sub-ribbon can entail unwanted empty areas during synthesis, as well as unwanted overlaps during the replication of bounded shapes that belong to the same fiber. To counter these issues, we partition the fiber median's ribbon (and not the sub-ribbons) in areas within which the bounded shapes can move without overlapping neighboring ones. We represent each bounded shape by its oriented bounding box in the fiber median's frame: its two axes $(x, y)$ correspond respectively to the direction of the associated fiber median and its orthogonal direction.

We compute the displacement areas as follows. We first sort the bounding boxes by position along the fiber median's orthogonal direction $(y)$ before applying a *sweeping algorithm* to successively determine, for each box, the neighboring ones that lie on the path along the $(x)$ direction. To ensure a more natural aspect during the planar extension, we consider a variation of a toroidal topology for the input space. Instead of considering our output domain as a collage of patches of the input space (which would be irrelevant for a circular input domain), we consider the toroidal topology at the ribbon level by generating two "ghost" copies of a ribbon, one forward and one backward along the fiber median's direction. With this, we can first guarantee that each box will have two adjacent neighbors (real or not), but also that the displacement areas along $x$ can extend outside the input domain. For each box, we set its displacement area along $x$, both forward and backward, to one-third the distance of the nearest neighboring box. We apply the

same technique to determine the displacement areas along $y$. The extended bounding boxes are first sorted along $x$ and we also use a *sweeping algorithm* to compute distances between the nearest boxes. As the displacement of the boxes along $y$ is limited to the width of the ribbon, the toroidal property is not applied here. The authorized perturbation along $y$ is therefore set to one-third of the distance between neighboring bounding boxes or two-thirds of the distance to the edge of the ribbon. These one-third and two-thirds rates were determined empirically to ensure that no overlap will happen during the synthesis as well as to respect a minimum distance between two bounded shapes. Figure 4.12 c) shows the displacement areas for our detailed example, in particular, the top ones are expanding outside of $IS$ due to our toroidal property.

## 4.4 2D Distribution Synthesis

In this section, we detail our planar synthesis method for anisotropic distributions, based on the previously computed Support Structure Hierarchy and the partitioning of the input domain. To allow for a seamless exploration of a larger 2D domain by simply zooming out after sketching, our goal is to preserve the user-drawn strokes in the input space $IS$ while extending them to a larger output space ($OS$), defined as an expansion of $IS$ by a scale factor $k$, with $k > 1$. This is achieved by a coarse-to-fine replication of the Support Structure Hierarchy, from lead ribbons to the entire hierarchy.

**Lead ribbons replication:** After extending the lead ribbons to the extremities of $OS$, we generate new lead ribbons in the remaining space by an efficient and randomized replication procedure as illustrated in Figure 4.13. For each lead direction, we start from one of the border lead ribbons and we randomly generate both a new gap from the previously recorded range and an index to determine which lead ribbon will be replicated at this position. We then update the position of the corresponding extreme height variable using the newly created lead ribbon's width. We apply this technique from top to bottom to gradually fill the output space. The randomness in the gap values results in different configurations of lead ribbons and thus different outputs from the same input. These replicated lead ribbons can be seen as displacement areas in which fiber medians (more precisely, their corresponding ribbons) will be replicated, at the cost of a slight blend of them and their child structures to avoid any unwanted overlap among the generated strokes.

**Fiber medians and ribbons replication:** For each newly generated lead ribbon, we synthesize its fiber median by first copying the parameters of its original ribbon. We then use the circular standard deviation on the median orientation computed in the analysis stage (Section 4.3) to perturb its orientation. If necessary, we moves the position of its midpoint to place it in the middle of its lead ribbon. This operation adjusts the position of the fiber median and its original ribbon relative to the lead ribbon.

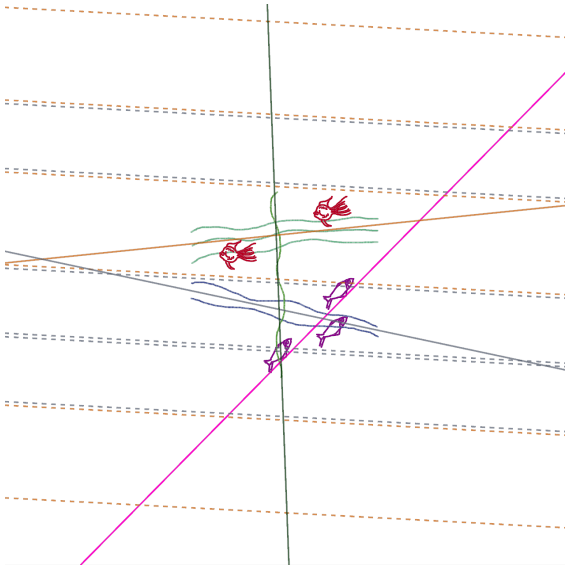


**Figure 4.13:** Depiction of the fiber medians (in plain), in addition to the lead ribbons replication (dotted) on *OS*.

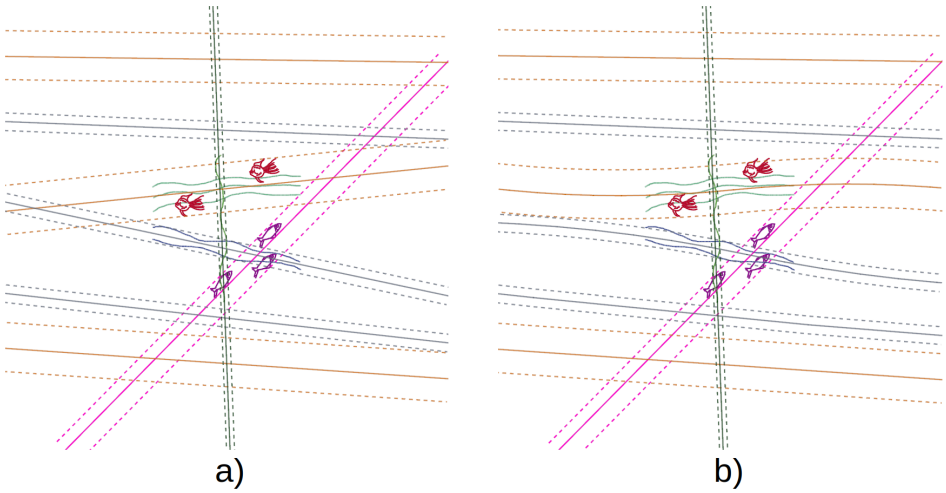



**Figure 4.14:** Fiber medians and ribbons replication on *OS*: a) without curvature; b) with curvature.

While the original ribbons are guaranteed to fit inside their lead ribbon in *IS*, as well as the middle part of generated ribbons are inside their lead ribbon, as illustrated in Figure 4.14 a), this is not necessarily the case when they extend to *OS*. When this occurs, we slightly curve the ribbon and its fiber median (see Figure 4.14 b)) to fit entirely inside its lead ribbon.

**Avoiding overlaps by bending structures:** Inspired by the physical properties of real fibers, we consider the following assumption: the thinner the ribbon is, the most flexible it is supposed to be. This can be formalized through the equation $R_t = \tau w_t$, relating the curvature radius $R_t$ to the ribbon width $w_t$ and a stiffness parameter $\tau \in \mathbb{R}^+$.

In the case of intersections, each ribbon will overlap twice with its lead ribbon. For symmetry reasons, we then bend both sides of the ribbon, even if one of the intersection regions is out of the output domain. From the four intersections points of the ribbon on the lead ribbon, we focus on the two that are the closest to the fiber median midpoint ($I_1$ and $I_2$ in red in Figure 4.15). Among them, one belongs to the ribbon's upper border while the other one is on the lower border, which thus gives us the direction towards which the ribbon should be bent.

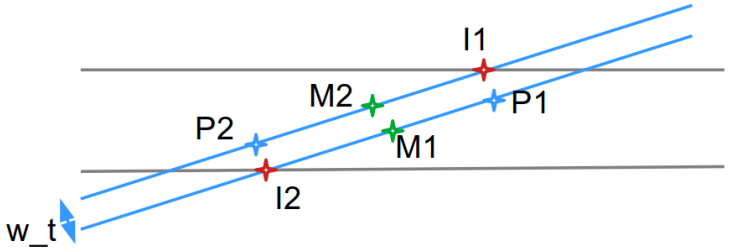



**Figure 4.15:** Example of a case of intersection between the fiber median ribbon of width $w_t$ (in blue) and a lead ribbon (in gray); $I_1$ and $I_2$ are the closest intersection points (in red); $P_1$ and $P_2$ their projection on the other border (in blue); $M_1$ and $M_2$ are the midpoints between an intersection and a projection point on a border (in green).

We start by projecting the intersection points on the other side of the ribbon ($P_1$ and $P_2$ in blue in Figure 4.15) as it is where the curvature radius will be the lowest. In the following algorithm, we focus on only curving the ribbon border with the lowest curvature radius, the other, as well as the fiber median, are determined using the ribbon width. To preserve the continuity between the original borderlines of the ribbon and their curved version, we consider the midpoints ($M_1$ and $M_2$ in green on Figure 4.15) between a projected point ($P1$ or $P2$) and the other intersection point as inflection points.

For each of these inflection points (say $M$), the key idea is to find the arc of circle $C$ that passes through $M$ and remains inside the lead ribbon as illustrated in Figure 4.16.

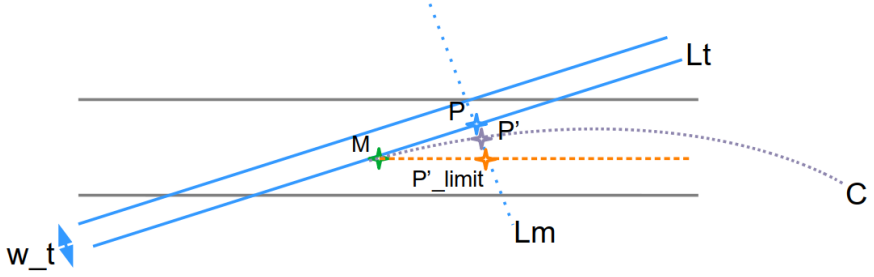



**Figure 4.16:** Illustration of our objective: find the circle $C$ that verifies our hypothesis.

By symmetry, we will only detail the curvature of one border, the other being treated similarly. The objective is to shift $P$ along the ribbon's secondary axis ($L_m$ in Figure 4.16) and toward the interior of the lead ribbon.

To achieve this, we introduce $P'$ as the translation of $P$ by a value of $\alpha$ along $L_m$. From the context, we have $\alpha \in ]0; \alpha_{limit}[$, with $P' = P$ for $\alpha = 0$, and $P' = P'_{limit}$ for $\alpha = \alpha_{limit}$. In particular, $P'_{limit}$ is defined as the intersection point between $L_m$ and the line passing by $M$ and parallel to the lead ribbon principal axis (in dotted orange in Figure 4.16). Taking an $\alpha$ value closer to 0 will result in a lower curvature from the fiber median's direction, however it can be insufficient to avoid the intersection, especially in the case of a thick ribbon. In contrast, an $\alpha$ value near $\alpha_{limit}$ can imply an important and undesirable flattening of the fiber median towards its lead direction.

In the following, we use $\overrightarrow{n_u}$ to represent the normalized vector of a vector $\vec{u}$ and $(x_A, y_A)$ for the coordinates of a point $A$. To find the radius and the center of $C$, we have the following properties:

1. $M \in Lt$, $M \in C$ and $Lt$ is tangent to C on $M$,
2. $P \in Lt$, $P' \in C$ and $P' = P + \alpha\overrightarrow{n_{Lm}}$.

Let $ax + by + c = 0$ be $Lt$'s line equation with $a, b, c \in \mathbb{R}$ and $a^2 + b^2 = 1$.
From the first property, we have:

$$\overrightarrow{OM} = (x_M - x_O, y_M - y_O) \tag{4.1}$$

$$\left\|\overrightarrow{OM}\right\| = R_C \qquad \overrightarrow{n_{OM}} = \pm(a, b) \tag{4.2}$$

The sign before the value of $\overrightarrow{n_{OM}}$ depends on the curvature turn side, or more precisely, whether $\overrightarrow{n_{OM}}$ is in the same direction as the normal of the fiber median direction.

From these equations, we obtain:

$$O = M + R_C\overrightarrow{n_{OM}} \iff \begin{cases} x_O = x_M \pm R_C a \\ y_O = y_M \pm R_C b \end{cases} \tag{4.3}$$

From the definition of $P'$, we have:

$$P' = P + \alpha\overrightarrow{n_{OM}} \iff \begin{cases} x_{P'} = x_P \pm \alpha a \\ y_{P'} = y_P \pm \alpha b \end{cases} \tag{4.4}$$

with $\alpha \in \mathbb{R}^{+*}$ and:

$$(x_{P'} - x_O)^2 + (y_{P'} - y_O)^2 = R_C^2 \tag{4.5}$$

By replacing $(x_{P'}, y_{P'})$ by Eq.4.4, $(x_O, y_O)$ by Eq.4.3, we obtain:

$$((x_P \pm \alpha a) - (x_M \pm R_C a))^2 + ((y_P \pm \alpha b) - (y_M \pm R_C b))^2 = R_C^2 \tag{4.6}$$

By developing the equation and using the following properties:

- $a^2 + b^2 = 1$,
- $x_P^2 + y_P^2 + x_M^2 + y_M^2 - 2 * x_P x_M - 2 y_P x_M = \left\|\overrightarrow{MP}\right\|^2$,
- $P \in Lt$ and $M \in Lt$ thus, $a(x_M - x_P) + b(y_M - y_P) = 0$.

we can reduce Equation 4.5 into:

$$\alpha^2 - 2\alpha R_C + \left\|\overrightarrow{MP}\right\|^2 = 0. \tag{4.7}$$

Moreover, by replacing $R_C$ by $\tau w_t$ we have:

$$\alpha^2 - 2\alpha\tau w_t + \left\|\overrightarrow{MP}\right\|^2 = 0 \tag{4.8}$$

Through this equation, we have a model to curve our fiber median through the use of two parameters ($\tau$ and $\alpha$) and the ribbon data.

We also note that Equation 4.8 has a solution if $\tau^2 w^2 \geq \left\|\overrightarrow{MP}\right\|^2$. Indeed, from $\tau$, $w$ and $\left\|\overrightarrow{MP}\right\| \in \mathbb{R}^+$, we can define $\tau_{min} = \frac{\left\|\overrightarrow{MP}\right\|}{w}$ and therefore $\alpha_{\tau=\tau_{min}} = \left\|\overrightarrow{MP}\right\|$.

A value of $\alpha_{\tau=\tau_{min}}$ leads to an arc of a circle of an angle close to $\Pi/2$ and can result in a point $P'$ out of the lead ribbon.

In practice, we have experimented with different values for $\alpha$ as illustrated in Figure 4.17. A good compromise between high straightening, the preservation of the ribbon's direction, and the no guarantee of non-overlap, is to get an $\alpha$ value between 0.23 and 0.3 depending on how close we want the ribbons to be while avoiding collisions. This parameter choice is put by default at 0.23 and the user can tune it using a slider.

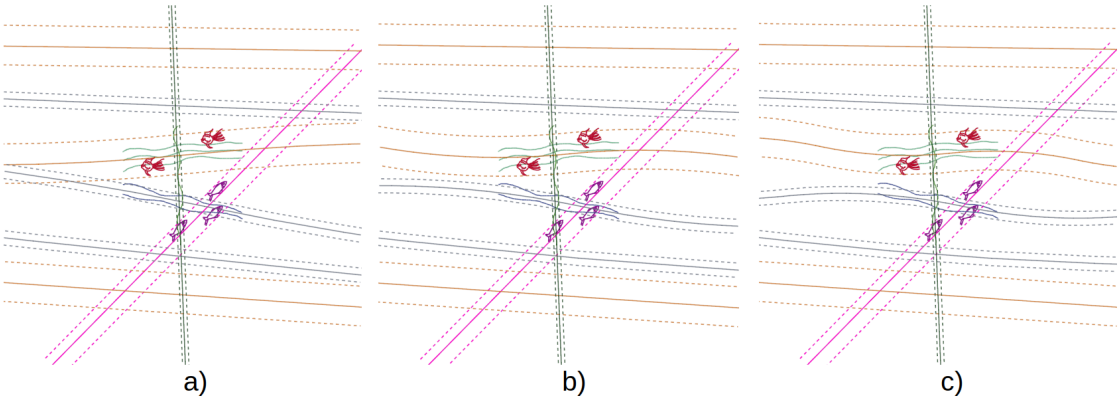



**Figure 4.17:** The impact of the value of $\alpha$ on the curvature: a) 0.1; b) 0.3; c) 0.5.

By setting a value to $\alpha$ and from Eq.4.8, we can deduce $\tau$ with the following relation:

$$\tau = \frac{\alpha^2 + \left\|\overrightarrow{MP}\right\|^2}{2w\alpha} \tag{4.9}$$

Now that both $\alpha$ and $\tau$ are known, we can determine the parameters of circle $C$ using the definition of the curvature radius ($R_t = w\tau$) and Equation 4.3. From this circle, we can then compute the intersections of the limit line (in dotted orange in Figure 4.16). This corresponds to the point ($M'$) at which we need to stop the curve, initially starting from $M$. In other words, we bend the corresponding ribbon border until reaching $M'$. If the latter is out of the domain, we crop the extra part, otherwise, we successively copy and paste this curve and a reversed curve (obtained by a half-turn rotation) until reaching the output domain borders. This oscillation is applied to all the ribbons to extend them to *OS* without overlap.

The same bending process is applied to the child sub-ribbons to fit them inside their parent curved ribbon.

**Shape distribution synthesis:** The final step is to synthesize the distribution of shapes in the output domain.

*a) Replication of unbounded shapes:* We define three categories of unbounded strokes: lines, arcs, and curves, which respectively stand for perfectly linear unbounded strokes, unbounded strokes with a single curvature extremum in *IS*, and any unbounded strokes with more than one curvature extrema. We also consider each stroke as full or half (ray) depending on whether both or only one of its extremities is on the borders of *IS*. We start by extending these unbounded shapes to *OS* along their fiber direction, which is trivial for lines and ray lines. Arcs are extended through an alternative mirror duplication which leads to a smooth sinusoidal curve (see Figure 4.18 bottom).

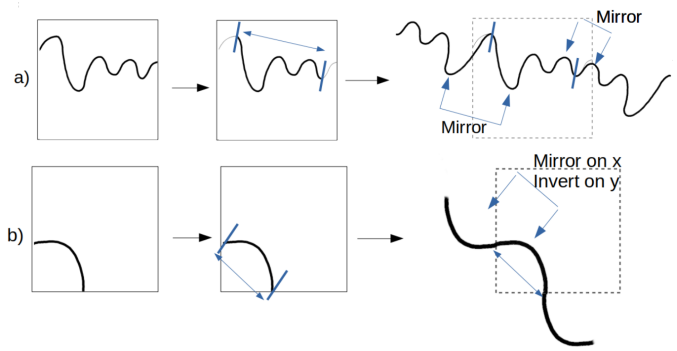



**Figure 4.18:** Extension of unbounded strokes: a) a curve; b) an arc.

In contrast, the curves are first cut at their first and last extrema to avoid artifacts that may appear at the beginning or end of a hand-drawn stroke. Next, we alternatively duplicate the mirror version or the original version of the curve segment to extend it to *OS*, as shown in Figure 4.18 top. These extended strokes are stored in the local frame of their corresponding fiber. They will therefore be automatically replicated and curved, if necessary, by the replication process of their parent structures in the hierarchy.

*b) Replication of bounded shapes:* We process the bounded shapes as follows. We start by replicating their representative support segments or central points along the fiber median and perturbing their positions using the previously computed displacement areas. We then retrieve the strokes in the resulting local frames. We reuse their local positions relative to their fiber median to replicate them on the replicated fiber medians but with randomly modified positions within the authorized displacement areas.

**Avoiding residual overlaps:** Since replications along the different lead directions are performed independently, lead ribbons belonging to two different lead directions will naturally overlap. If two or more lead directions contain bounded shapes, unwanted overlaps may occur and bring perceptual artifacts. To counter this issue, we apply an additional step before generating new bounded shapes. The main idea is to use a data structure to partition *OS* and then be able to delimit the displacement area of a bounded shape to a free-overlap zone. If this condition cannot be met, for example, in the case of a too large overlap between two displacement areas, we choose to randomly pick only one shape from the two and display only this one.

As a first simple solution, we took a 2D grid of fixed size as partition structure. However, the balance between the size of the grid and the ability to provide good accuracy in the positions of the displacement areas while avoiding high computational cost was hard to find. Thus, we adopted an *AABB* tree structure. To take into account the potential curvature of the ribbon, we start by determining, if necessary, the new position of the bounding box by taking the mean tangent of the curve around the shape as the new main direction. We then compute, for each

displacement area, a coarser bounding box in the world frame that we insert in our tree structure. Our coarse-to-fine overlap avoidance is defined as follows. We start by shuffling all the bounded shapes we want to generate. Then, we detect each shape that overlaps with neighboring (and already not visited) shapes using our distance between boxes (defined in Section 4.3). We apply this detection first at the coarser AABB boxes level and then at the oriented displacement area ones. In the case of overlap in the last detection, we directly compute the overlapping area between the two displacement areas using the intersection between their boxes segments. From this area, we can determine whether both perturbation rates can be adjusted, only one or neither. We either update the perturbation rate of the shape or remove this shape from the displayed ones.

Figure 4.19 presents the results from our 2D distribution synthesis, from the lead ribbons creation (left) to the final synthesis with the underlying structure (right). A series of other results are given and discussed in the results section 4.6.

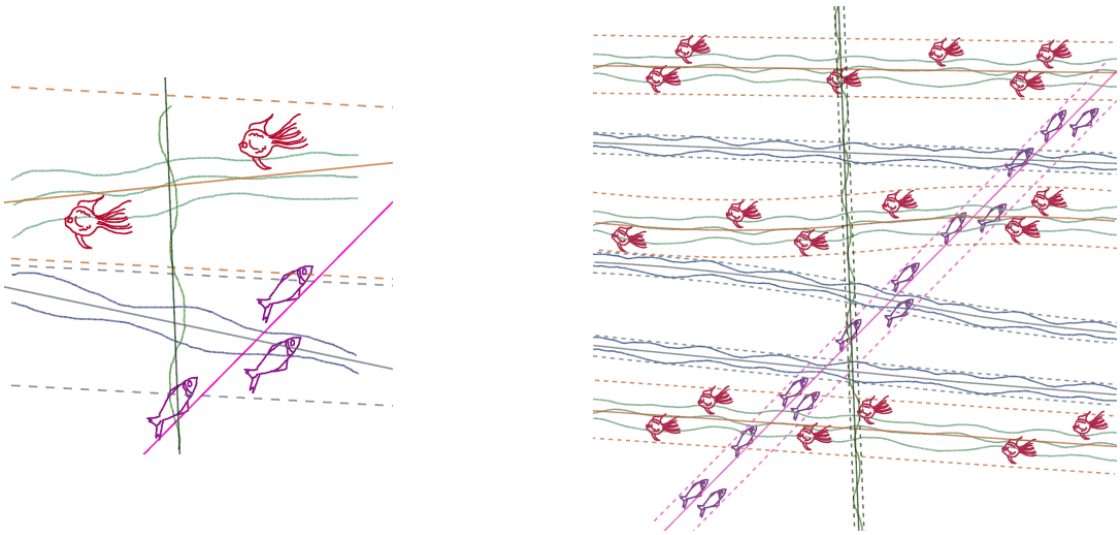


**Figure 4.19:** From the analysis (left) to the planar synthesis (right). Note that both the oblique line with purple fish and the central seaweed were not duplicated in the output.

## 4.5 3D Distribution Synthesis

For this synthesis, our goal is to generate, from the same sketched input, a repetitive 3D distribution, visually similar to this input and within which the user can navigate. Determining depth parameters from a 2D input is a challenging task in general since the number of possible solutions is infinite. Based on perceptual studies in already existing work [WBCG09] as well as our hypotheses (see Section 4.2.1), we based our solution on a few predefined and user-adjustable extension parameters. In particular, our design hypothesis $H1$, stating that any groupings or alignments in the input needs to be maintained in 3D, highlights the use of our Support Structure Hierarchy to immerse clustered strokes using quite similar depths.

To this end, our method first computes a simplified planar extension of the input (a lateral expansion of scale factor $k$) before immersing the resulting 2D distribution in 3D (using an extension of depth $h$). This 3D environment block carrying the shape distribution is then simply replicated in-depth, to allow unlimited exploration.

**Simplified 2D distribution synthesis:** Since shapes will be distributed in 3D, there is no need to avoid the potential 2D overlaps as they can be avoided by using different depths. In particular, there is no need to bend the ribbons during the lateral extension nor avoid residual overlaps between bounded shapes. However, to preserve perceptual similarity with the input, we still impose shapes to stay within their ribbons, and, in particular, unbounded shapes can bend to remain within their sub-ribbons. Using the method presented in Section 4.4, we synthesize the lead directions as well as fiber medians as shown previously in Figure 4.14 a), before replicating the unbounded and bounded shapes.

**From 2D to 3D representation:** This intermediate extended 2D distribution can be seen as the planar projection of our target 3D distribution. The strokes are immersing in 3D, using two additional values: their midpoint depth $d_e$, and their slope angle $\delta$ with respect to the $XY$ plane.

To create a high-level representation of such a 3D distribution, we introduce two different 3D immersions depending on the level of our hierarchy—lead directions or fiber medians—at which we set the 3D parameters. While the former can be more suitable for a homogeneous distribution, the latter avoids unwanted overlaps in the case where the ribbons of fiber medians from the same lead direction overlap in the extended 2D domain. By default, we set the 3D immersion on the fiber medians but the user can use the menu to try the other one. Without any user input, we randomly set the 3D parameters of the chosen structure between $[0,h]$ for $d_e$ and $[-\pi/4,\pi/4]$ for the slope. These range of values have been determined through perceptual studies [WBCG09].

From the slope, we can determine the depth of the extremities of each support line, as illustrated by Figure 4.20 below.

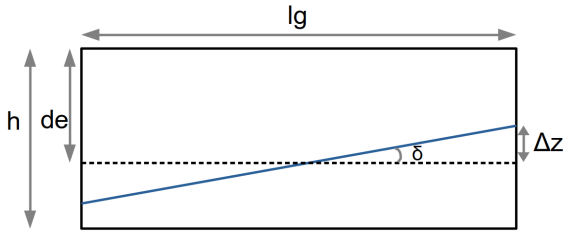



**Figure 4.20:** 3D immersion of a line in a layer of depth $h$: $d_e$ represents the midpoint depth, $\delta$ the slope, and $lg$ the 2D line length. $\Delta_z$ is determined using Equation 4.10.

Indeed, we can compute the depth gap $\Delta_z$ between the midpoint and the extremity points as follows:

$$\Delta_z = \frac{lg}{2}\tan(\delta) \tag{4.10}$$

Therefore, to achieve the required slope $\delta$, we simply increase the depth value of one extremity by $\Delta_z$ while decreasing the other one by the same amount. Moreover, to further improve the feeling of depth, we choose to immerse our 2D shapes on 2D planes positioned in the 3D volume. To do so, we locally apply Equation 4.10 on each stroke point coordinate to determine its 3D position.

For the specific case of unbounded strokes which fully lie inside the input domain *IS* and therefore span the full depth of the 3D domain *OS*, the stroke length in *IS* is used to determine its 3D embedding. The shorter the input stroke, the more its 3D version is aligned with the depth axis, the extreme case being a point that would represent an infinite line parallel to the depth axis.

**3D immersion:** To create an infinite 3D environment within which the user can navigate, we need to account for repetitions along the depth axis. From the first layer of 3D distribution computed from the user's input, we generate slightly different layer copies using a planar offset on the strokes from a predefined range that we position in depth every layer's height ($h$). This allows us to create a virtually infinite field of anisotropic elements. In practice, three layers are enough to create this illusion at a low cost. The user can also tune the number of layers. Note that these layers of strokes may overlap in depth. We display them using the perceptual assumptions that distant objects fade and that 3D space has a toroidal topology in depth.

**3D navigation:** In our implementation, we use a dolly zoom to smoothly transition from the interactive sketching session where the user sketches the input strokes to the exploration of the 3D output scene.

*Path navigation*

During the navigation, the camera translates by a fixed depth step $d_s$ along $z$ and also follows two different sinusoidal paths along the $x$ and $y$ axis of period $\frac{d_s \pi}{h}$. Using this period enables the camera to return to the same position at the beginning of each layer. In addition, the user can, at any time, speed up or slow down the navigation by using the menu to adjust the depth step parameter.

*Expressive rendering*

We chose to characterize each 3D stroke by three rendering parameters: its transparency, thickness, and color. During navigation, each of these parameters is updated to adjust to the camera distance, to fade out background strokes while highlighting nearby ones. For each stroke, we computed its transparency and thickness steps from its original parameters, the number of displayed layers, and the depth step $d_s$. For the color, we use the *HSL* format to update the luminance parameter from 1 (white and thus invisible) to the input stroke color value.

During navigation, we display each stroke the same number of times, that is the number of layers. When a version of a stroke ends up behind the camera, we shift it directly into depth and assign it the lowest visibility, which is the maximum transparency, the minimum thickness, and a luminance parameter of 1. Using this expressive rendering gives the illusion of infinite space in depth and results in better visual results during exploration than applying the same transformation layer per layer while maintaining real-time performances, as illustrated by Figure 4.21. Note that the central seaweed and the oblique fish will be duplicated this time but at different depth levels. Indeed, we assume that we want to synthesize a block of a 3D environment which will be repeated infinitely as we explore. If this repetitiveness is annoying, we could arrange it by adjusting the transparency so that the following oblique fish appear only when the first ones are no longer visible on the screen.

**Figure 4.21:** Immersion in the 3D virtual world created from a 2D sketch.

## 4.6 Validation & Results

### 4.6.1 Visual Results

**2D distribution synthesis** As illustrated in Figure 4.22, our planar synthesis method respects our design guidelines for tightening strokes (as the ones in white) but also preserve boundary constraints between bounded shapes and unbounded ones. Note that bounded shapes may appear to intersect but this is due to the stroke resolution and not to overlaps.

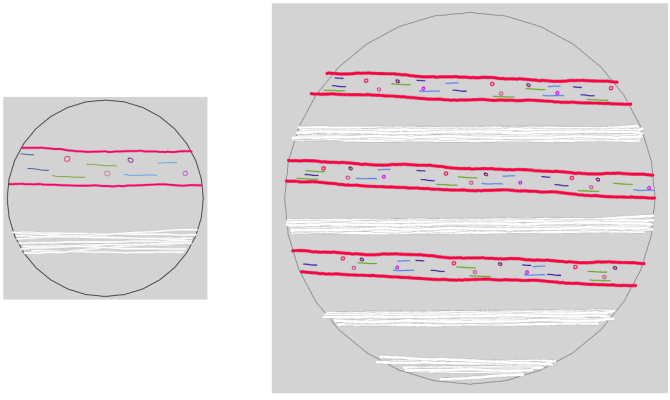



**Figure 4.22:** Our planar synthesis method maintains groupings of strokes as the ones in white here and boundaries constraints: a) input; b) our output.

Moreover, as can be seen in Figures 4.23 and 4.29, our method maintains the regularity of structured distributions for both bounded and unbounded elements, although, such distributions are often failure cases of previous methods (see limitation Figures of [DSJ19] and Figure 13 of [ENMGC19]). In particular, Figures 4.23 right and 4.29 show challenging examples of not instantiating all the bounded shapes during the residual collision avoidance step.

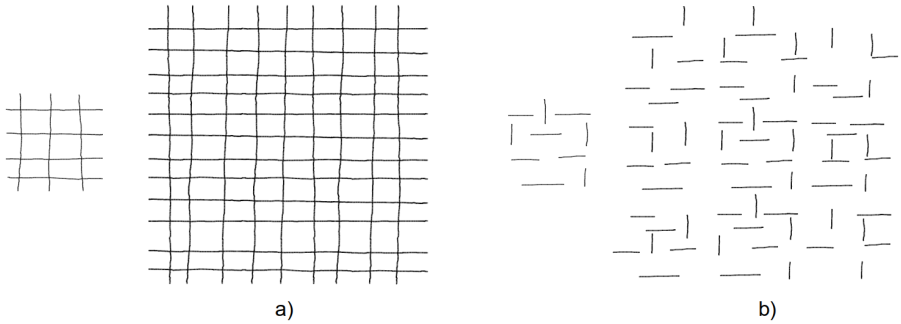



**Figure 4.23:** Our planar synthesis method maintains the perceived regularity of structured distributions (known to be hard to handle) in both cases of unbounded and bounded strokes.

*Comparisons with state of the art methods*

We compared our planar synthesis results with the state-of-the-art methods for both classical approaches on shape distributions, as well as deep learning ones on point distributions such as [TLH19].

For point distributions, we first compare our planar synthesis with the current state-of-the-art methods, using outputs from Tu et al. [TLH19]. As illustrated in Figure 4.24 f), our method is the only one to provide both regularity and variation during the synthesis. In addition, in the case of anisotropic distribution of points such as in Figure 4.25, contrary to [TLH19], we can guarantee that each synthesized points column is complete. Moreover, our synthesis is the only one to be performed in real-time (see Table 4.1).

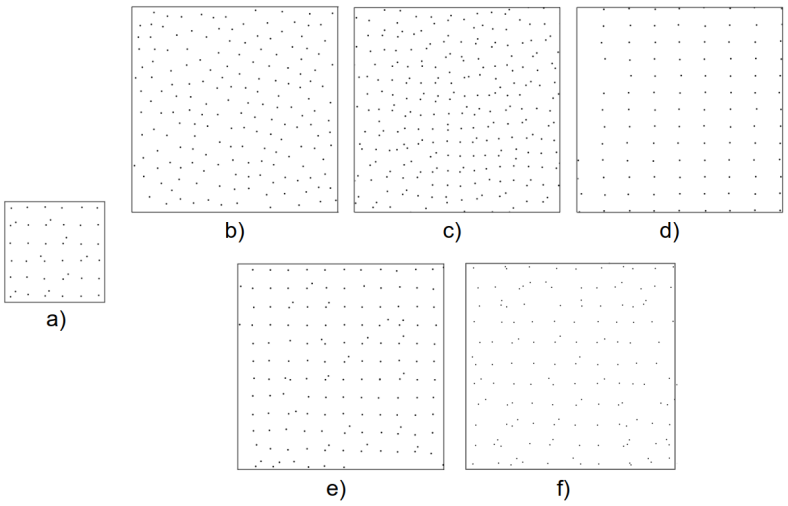



**Figure 4.24:** Comparison with the state of the art methods: (a) image input; b) [ZHWW12], c) [MWT11], d) [ROM*15], e)[TLH19], f) our planar synthesis.

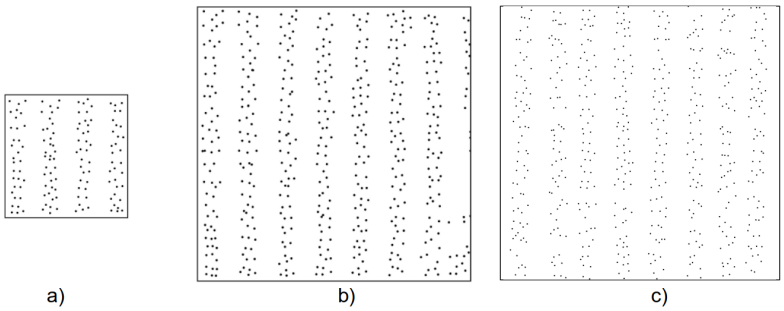



**Figure 4.25:** Comparison with a Deep Learning-based method: (a) image input; b)[TLH19], f) our planar synthesis.

For bounded shapes, we compared our planar synthesis with existing methods using outputs provided by Landes et al. [LGH13] and Davison et al. [DSJ19]. For Figures 4.26 and 4.27, the results from our user study highlight that users preferred our output to the others (namely [LGH13] output which seems the most relevant).

Finally, we generated outputs from inputs containing more than one (here two) lead directions but also thick bounded shapes to check the robustness of our residual collision avoidance step. Figures 4.28 and 4.29 present the comparison with state-of-the-art methods for this specific case. Whereas our method provides a better result than the one from Davison et al. [DSJ19], it is more questionable for the Landes et al. [LGH13] one as described in the Discussion in Section 4.6.4.

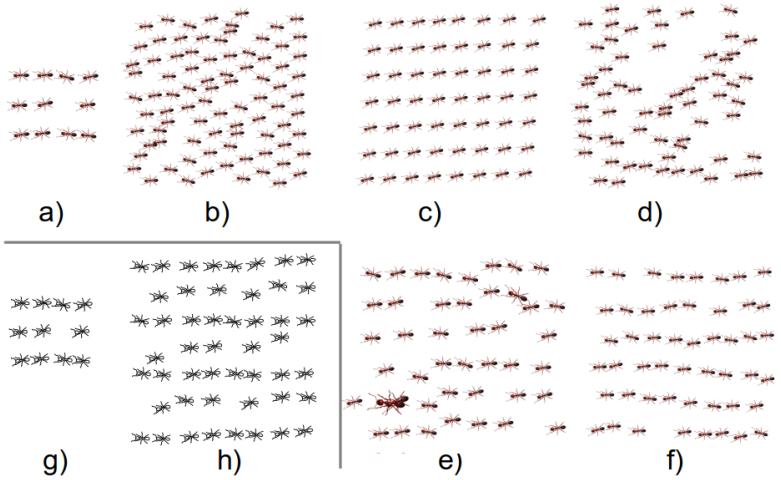



**Figure 4.26:** Comparison with the previous methods for vector synthesis: (a) image input; b) [BBT*06], c) [IMIM08], d) [HLT*09], e)[MWT11], f) [LGH13], g) our corresponding sketched input, h) our result.

**3D distribution** Figures 4.1, 4.3, 4.30, and 4.31 show some views of our 3D distribution results from various 2D sketches. Note that, as such 3D synthesis from a sketch is a new concept, we find no prior work to compare with. However, our user study allowed us to perceptually validate our 3D results (see Appendix B).

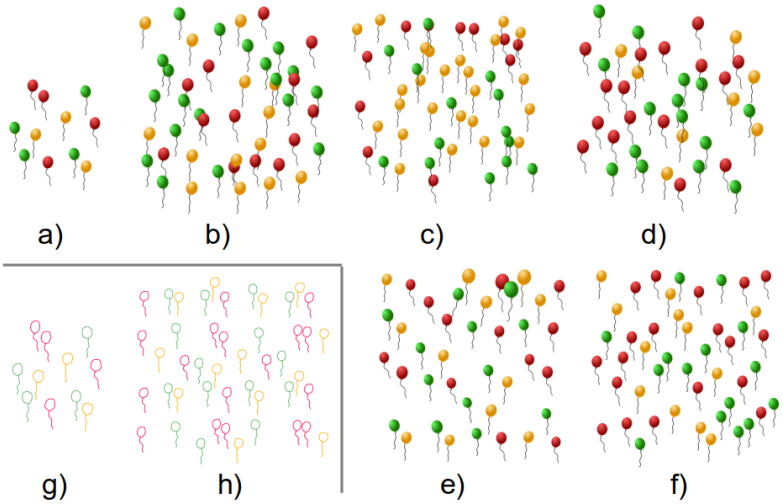



**Figure 4.27:** Comparison with the state-of-the-art methods: (a) image input; b) [BBT*06], c) [IMIM08], d) [HLT*09], e)[MWT11], f) [LGH13], g) our corresponding sketched input, h) our result.

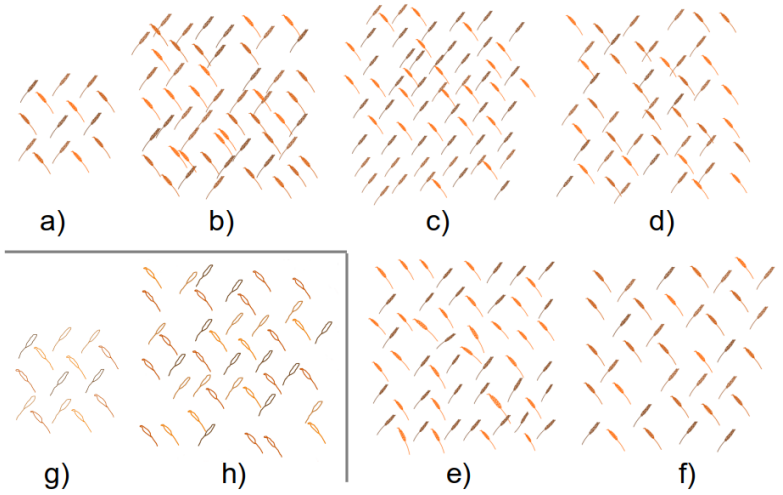



**Figure 4.28:** Comparison with the state-of-the-art methods: (a) image input; b) [BBT*06], c) [IMIM08], d) [HLT*09], e)[MWT11], f) [LGH13], g) our corresponding sketched input, h) our result.

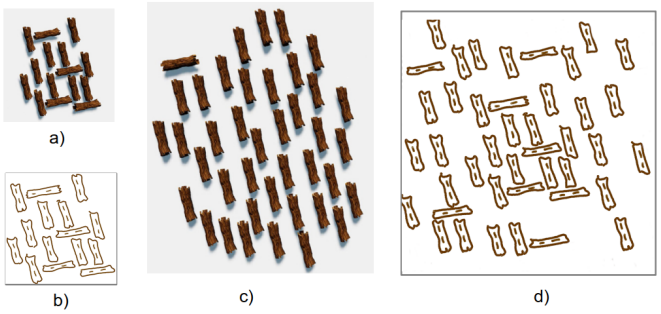



**Figure 4.29:** Challenging structured distributions: (a) input; (b) sketched representation of the input; (c) result of [DSJ19]; d) ours.

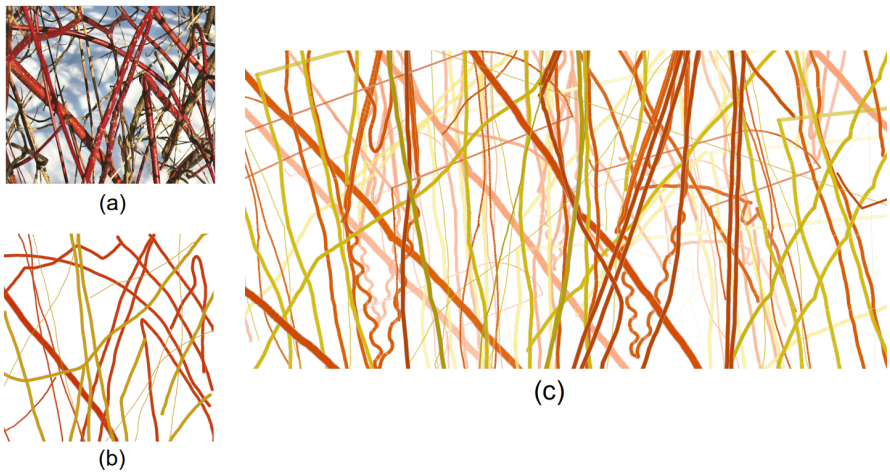



**Figure 4.30:** Stems: a) inspiration picture; b) sketched input; c) a snapshot of the 3D distribution created from the input.

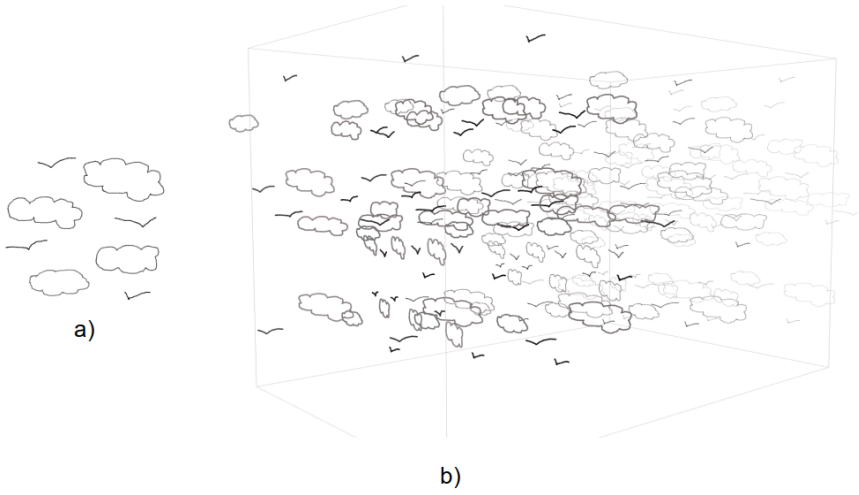



**Figure 4.31:** 3D distribution synthesis of a set of bounded elements.

### 4.6.2 User study for perceptual validation

*Experiment setup*

We carefully designed an online user study aimed at a broad public to validate the perceptual hypotheses presented in Section 4.2.1, as well as the perceived quality of our results. We provide screenshots of this user study in Appendix B. In this study, we first ask users to draw shapes to complete an area around an existing sketch. These examples ranged from simply replicating bounded shapes to extending (or not) unbounded shapes, avoiding collisions, and maintaining the anisotropic distributions of both bounded and unbounded shapes. Providing a drawing session allows us to obtain the most intuitive answers from users without any influence on the desired result. Then, thanks to a comparison session, they were able to confirm or deny their first instinct but also compare the resulting 2D planar synthesis for bounded shapes ([LGH13] and ours), as

well as 3D immersion alternatives both on bounded and unbounded shapes, presented in `GIF` formats.

*Results of the user study*

This study involved 35 users, ranging in age from 19 to 61 years, including 22 males, 9 females, and 4 unspecified genders. Most users had no experience with paper-based (26) or computer-based (21) design. Figure 4.32 presents a summary of the user profiles. Each session lasted about 10 minutes, most of which was spent in the drawing session.

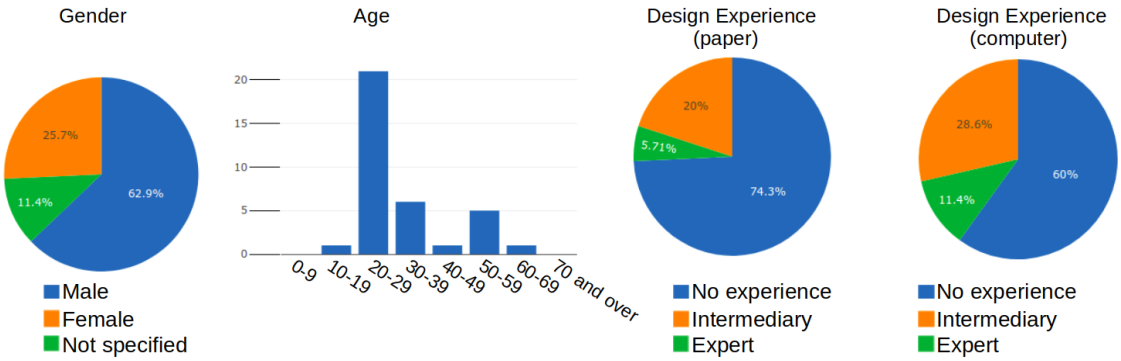



**Figure 4.32:** Summary of the user's profile for our user study.

We validated our design hypotheses using both the drawing and comparison sessions. For $H1$ (groupings and alignments are meaningful), we mostly rely on the drawing session. In the latter, all users preserved bounded stroke clustering when replicating bounded shapes in the extended domain, and 97% maintained the grouping of fiber-like shapes. In addition, 75.9% of users respected the anisotropy directions of bounding shapes. Finally, alternative results from the second part of the study that did not preserve stroke clustering (such as 3D environments assigning a different depth to each stroke) were never retained as correct.

In contrast, $H2$ (repetitiveness is explicit) was primarily evaluated during the comparison session, although this guideline was considered the most questionable one. During the drawing session, only one input contained both repetition and singularity, and it was only for unbounded strokes. We could not process this case as 63% of the users only extended the unbounded strokes to the output domain without any replication (probably showing that the concept of texture was not clear for beginners). In the comparison session, 56.7% of users preferred the output where the isolated fiber was not duplicated, against 40% for repetition and 3.33% neutral.

$H3$ (avoiding overlaps when not present in the input) was validated through both sessions. Of all the drawn outcomes, 73% were overlap-free drawings. In addition, 91.4% of users maintained

groupings and overlap avoidance between unbounded strokes when they share the same anisotropy direction and a consistent gap with other strokes or groups. This proportion dropped to 76.6% when unbounded strokes or groups of strokes were separated by different gaps. During the comparison session, we let users decide how important our curvature step was by displaying two alternative outputs: one with our additional curvature; the other without it, showing brand-new overlaps. Among all user answers, 54.3% preferred the overlap-free output, 31.4% chose the straighter lines with more overlaps, and 14% chose none of the proposed solutions. For this example, we had used an $\alpha$ value of 0.5 which could have resulted in too much straightening of our curves and may have led users to not be satisfied with our curvature algorithm.

In addition, we take the opportunity of this user study to let users compare our results with Landes et al. [LGH13], which is the only existing solution focused on anisotropic distributions of bounded shapes, to determine the best planar synthesis in terms of proximity to a provided input. As we are the first method to use sketches as input and to facilitate the comparisons between the two, we use our software to redraw over Landes et al. [LGH13] for both the inputs and the planar synthesis for the ants (Figure 4.26) and balloons (Figure 4.27) examples, as illustrated in Appendix B. For this comparison, we allow users to choose both methods if they considered them to be perceptually equal. Our 2D synthesis results were considered the best for the two selected inputs by more than 80% of users, as depicted in Figure 4.33.

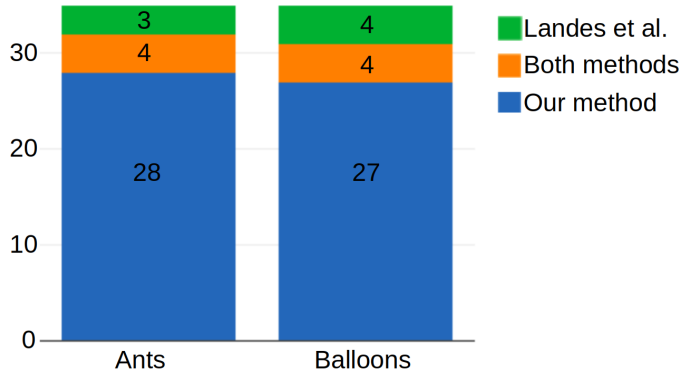



**Figure 4.33:** Sum up of the comparison results with our method and Landes et al. [LGH13], for the ants and balloons examples.

Finally, we exploited this study to determine at which level of our hierarchy the 3D parameters should be set (see Section 4.5). To do this, we separated the unbounded and bounded shapes into two different inputs and proposed three 3D immersions (in the form of `GIF` clips) depending on whether the 3D parameters were set on the lead directions, the fiber medians, or directly on the individual strokes. Based on the results, the lead directions were preferred for the immersion of unbounded shapes (60%) over the fiber medians (34.29%) and the stroke level (5.71%). In

contrast, it was a tied score between the lead directions and fiber medians for bounded shapes. However, with the exception of the 3D immersion directly at the stroke level, the groupings and alignments in 3D may not have been as clear on our `GIF` animations.

### 4.6.3 Computation times and real-time performance

The table 4.1 presents the timing of our method. This was calculated using the Performance DevTools panel of Google Chrome on an Intel(R) Core(TM) i7-7920HQ CPU at 3.10 GHz. The first number is the number of points in the input example, followed by the time in milliseconds for analysis, planar synthesis, and 3D immersion, respectively. As can be observed, the overall computation time of all the presented results is less than a second. This confirms that we are targeting 3D immersion applications that require real-time performance.

Note that unlike learning-based methods, our method requires little memory space and no pre-computation time.

| Example | Points | A. Time | P.S. Time | 3D Time |
|---|---|---|---|---|
| Teaser | 7699 | 73 | 114 | 183 |
| Biology (Fig.4.3) | 4574 | 35 | 102 | 133 |
| Ants (Fig.4.26) | 9447 | 134 | 233 | 413 |
| Balloon (Fig.4.27) | 4034 | 42 | 68 | 100 |
| Wheat (Fig.4.28) | 4813 | 27 | 47 | 104 |
| Trunks (Fig.4.29) | 3164 | 68 | 83 | 155 |
| Rods (Fig.4.30) | 3224 | 36 | 328 | 356 |

**Table 4.1:** Computational times in milliseconds: Points being the number of input in the example, A. Time stands for the analysis time, P.S Time as Planar Synthesis Time, and 3D Time for 3D immersion.

### 4.6.4 Discussion and limitations

In comparison with existing shape distribution methods, our approach does not require any neighborhood matching or learning during the synthesis stage, given that our hierarchical representation already captures correlations. This leads to real-time performance, adapted to our application context.

However, as illustrated in Figures 4.28 and 4.29, our current implementation of the residual collision avoidance may require some adjustments to improve our current results on dense bounded shapes distributions along more than one lead direction. For example, using a relaxation method to adjust the final positions, as done in PCF-based synthesis, might be a good option.

One can also observe in Figure 4.30 some curved stems were generated when they were not in the input. This is probably the result of a curvature analysis near the edge of the input space.

In the current version of this chapter, we have focused on anisotropic distributions through computations of linear directions. An interesting way to generalize this work to isotropic distributions would be to introduce a default direction, such as the horizontal or the vertical ones as the leading direction to synthesize such distributions of bounded shapes.

Another useful extension would be to increase user control during both the analysis and synthesis stages. For instance, this could go from choosing perceptual hypotheses to deciding how to handle singular patterns (i.e., repeating them or not) or even adjusting parameters regarding repetitions and the amount of perturbation during synthesis. As illustrated by the input in Figure 4.3, even though they had different main directions, the fibroblasts (depicted in pink/red) should remain attached to fibers. This is prevented by our current choice of proximity during the successive clustering (see Section 4.3). We could also let the user decide what distance to apply when grouping bounded strokes into shapes. For example, Figure 4.34 presents a failure case for our bounding boxes approach since each stroke should be considered as an individual shape, while their bounding boxes unfortunately overlap, which would currently result in their grouping.

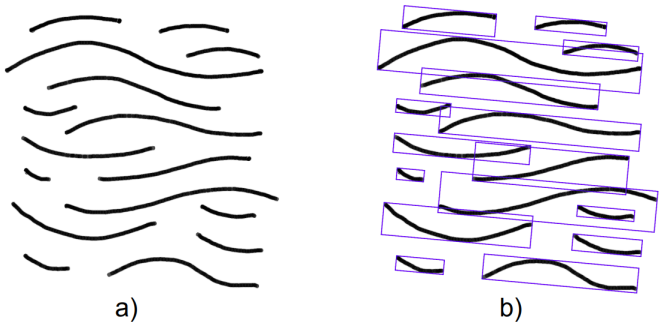



**Figure 4.34:** a) Input example from [MWT11], in which a point-based approach to cluster bounded strokes into shapes can be more relevant than the bounding box representation in b).

Lastly, many perceptual improvements in terms of 3D immersion could be considered. One of them would be to vary the stroke's opacity and thickness in a continuously way to emphasize their inclination. In addition, an authoring system such as the one in [TWY*20] could be used to interactively allow any local adjustment desired by the user in the final synthesized distribution.

## 4.7 Conclusion

Motivated by the development of an interactive sketching tool to draw, extract structure, and give depth to simple sketches, we have designed a method to efficiently extract key anisotropic properties in a 2D input sketch and then generate visually similar distributions into a larger 2D or 3D domain. Our method compares well with the state-of-the-art results in the 2D case and is the first one, to our best knowledge, that addresses the generation of a repetitive 3D environment visually similar to an input 2D sketch.

The construction of our new representation, the Support Structure Hierarchy, is crucial to our method. Extracted at the analysis stage, it allows us to efficiently capture and replicate the multi-scale anisotropic structures while maintaining a good level of visual diversity in the synthesized distribution of elements.

Indeed, one of the advantages of this framework is that no particular expertise in graphics nor sketch-based design is required.

**Future work:** Although perceptually similar to the input sketches, the 3D illustrations we generate are not fully 3D, in the sense that each stroke remains planar. Inferring 3D geometry for the strokes from their 2D depiction, e.g., using 3D helices instead of planar sinusoids for curves as in [WBC07] would be an interesting avenue for future work.

Furthermore, curves with complex branching structures often appear in organic distributions, for instance, in the biological illustration applications we are targeting. Since we address distributions of individual shapes, our method cannot handle such branching. Procedural techniques such as [MM10] can enrich our efficient replication framework to generate complex curves both in 2D and 3D. In this context, synthesizing patterns along curves or ribbon-like surfaces may also be tackled more directly compared to classical optimizers or dynamic programming techniques [ZJL14].

Our system can be used by artists to quickly design new distributions and test new patterns using simple sketches. It can also be used by scientists to quickly convey the vision of the object of study they have in mind. Indeed, while biologists often use sketches as simplified representations, the tissues they represent are inherently 3D. Our exploration of a 3D distribution of fibers from a sketch was motivated by their needs. The next chapter presents how such 3D distributions can be used to create a 3D environment inside which shapes can be created and animated, using only sketching input. Applied to cell biology, we illustrate the potential of our method using two narrative scenarios of real phenomena to point out the potential of our tool for exploring and communicating a dynamic phenomenon.

# 5

# Sketching dynamic environments: the example of biology

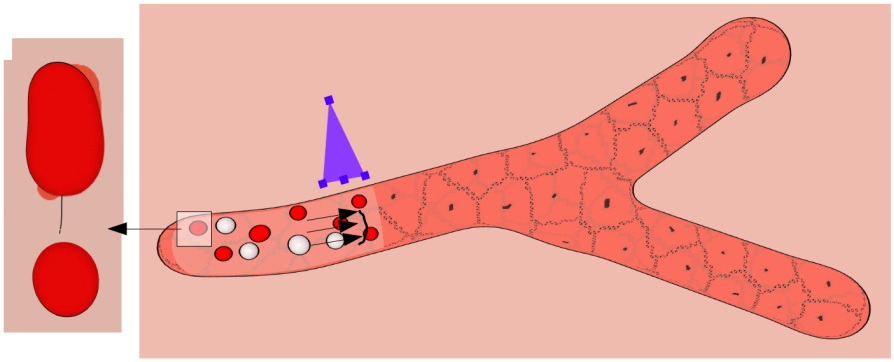


**Figure 5.1:** Preliminary results of our tool: the user sketches to model surfaces but also to define motions through schematic representations, and deformations with our keyframe snippets.

As a second case study and to explore phenomena related to life sciences, we focus this application on cell biology. This allows us to study, on the one hand, the creation of organic shapes from sketches and, on the other hand, how animated sketches could be created and controlled. In the targeted system, scientists are not only able to represent their understanding of a static situation but also their hypotheses about relative motion, and ideally, they can tell a story about it. In this work, we address the problem of proposing a new progressive sketching paradigm for both geometric modeling and animation without any predefined and rigid editing pipeline and dedicated to satisfying the needs of biologists to communicate or explore phenomena. To support narrative design in cell biology, we present *Narrative Sketches*, a new 3D sketching environment focused on the representation, exploration, and communication of dynamic phenomena. Our model allows for the progressive creation of nested structures that evolve according to the sketch inputs. In particular, our sketch-based animation solution relies on a set of schematic vocabulary inspired by the standard depictions in cell biology but also on keyframe snippets.

## Résumé en français

Comme deuxième cas d'étude et pour explorer les phénomènes liés aux sciences de la vie, nous avons axé cette application sur la biologie cellulaire. Cela nous a permis d'étudier d'une part, la création de formes organiques à partir de croquis et d'autre part, comment des croquis animés pourraient être créés et contrôlés. Par conséquent, les scientifiques sont non seulement en mesure d'illustrer leur compréhension d'une situation statique, mais aussi leurs hypothèses sur le mouvement associé, et éventuellement leur capacité à raconter une histoire à ce sujet. Dans ce travail, nous présentons un nouveau paradigme progressif de croquis, à la fois pour la modélisation et l'animation, sans pipeline d'édition prédéfinie et rigide, et destiné à répondre aux besoins des biologistes en termes de communication ou d'exploration des phénomènes. Pour favoriser le design narratif en biologie cellulaire, nous introduisons les *Croquis Narratifs* (Narrative Sketches), un nouveau type de croquis 3D dédié à la représentation, l'exploration et la communication de phénomènes dynamiques. Notre modèle permet la création progressive de structures imbriquées évoluant en fonction des indications de l'utilisateur. En particulier, notre solution d'animation par le croquis s'appuie sur un vocabulaire schématique inspiré des représentations standard pour indiquer le mouvement dans les croquis en biologie cellulaire, mais aussi permet de spécifier des déformations par images clés. Nous prévoyons de valider l'applicabilité de notre système par une étude utilisateur qui sera réalisée auprès d'utilisateurs biologistes et artistes.

## Contents

## 5.1 Motivation

While architecture involves creative design limited by some practical rules, biology is concerned with the scientific study and understanding of life. From the evolution of a population to the composition of an individual to the close analysis of molecular and atomic structures, this field covers a wide range of phenomena and especially multiple levels of organization.

Even though biological data provide 2D or 3D images, these are often quite difficult to understand, if not impossible to control, because only the motion that took place in the actual data can be observed. Therefore, biologists are looking for intuitive and creative tools to clarify, visualize or even communicate on their representation of some shapes but also explore some hypotheses. For this reason, we would like to offer a tool that not only allows to represent nested 3D shapes in a simplified way, but also to create and control any animation scenario involving both motion and deformations. In this chapter, we have focused our application on the cellular level, which examines the structure, function, and behavior of cells living in an environment. As illustrated in Figure 5.2, in response to a specific situation (here an infection), the cells can be induced to strongly deform to escape their default environment.

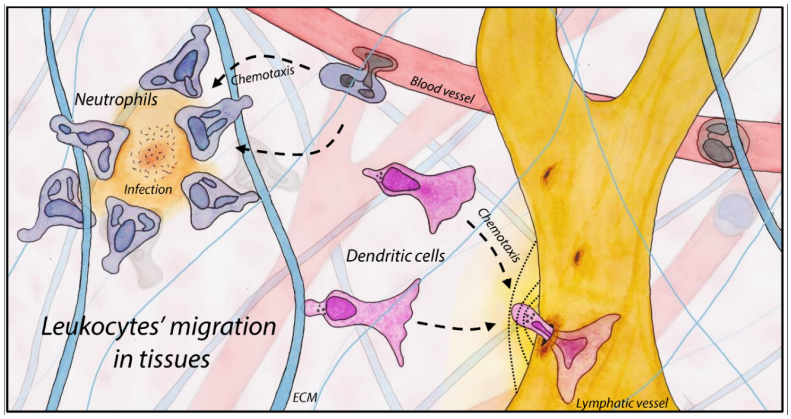



**Figure 5.2:** Illustration of the human body's response to infection: neutrophils (in blue) escape the blood vessel to fight the infection while dendritic cells leave the site to enter the lymphatic vessel. ©Betsy Goic, Ph.D., DrawInScience.

Biologists often represent their phenomenon of study by images, ranging from real data pictures, for instance, captured by putting a camera on a microscope, or illustrations varying from highly schematic to more representative and artistic (as in Figure 5.2). In addition, such representations can require the help of professional artists, which typically implies some long discussions to communicate the biologists' ideas and a series of trials and errors.

While many methods have focused on the representation and visualization of biological data from the molecular level to the mesoscale (interior of cells), only few have targeted the cellular level. As presented previously (see Related work 2.1.1 and 2.3.3), existing methods either target the modeling of vascular systems from sketching conventions, or present a system combining both the creation and animation of a biological environment. In particular, these last methods are specific to either vascular diseases or blood circulation. In particular, none proposes a system enabling the creation and animation of a complex 3D scene inside which cells can navigate and evolve.

In this chapter, we tackle the problem of proposing a new progressive sketching paradigm both for modeling and animation, without any predefined and rigid editing pipeline and dedicated to satisfying the biologists' needs to communicate or explore phenomena. In particular, this work involves a collaboration with the biologist Jean-Luc Coll from IAB Grenoble, who studies cancer targets and experimental therapeutics; and with the professional artist Renaud Chabrier, who has a long-lasting experience in the field of scientific illustration, especially in the domain of cell biology.

We conducted a pre-study among biologists of different specialties to learn the depiction and narration in biology. This study helped us understand that biologists not only need to visualize information, such as through microscopic imagery but also to depict in a simplified way their understanding of their animated phenomena of study. We therefore started by analyzing three case studies to identify the types of shapes, motions, and deformations to consider. In complement, our collaborator from biology introduced us to the schematic vocabulary used to represent dynamics in biological illustrations. Finally, we describe the phenomenon represented in each of our case studies through a narrative scenario. Creating such scenario helped us first identify the sequences of actions we need to reproduce virtually and dynamically but also to experiment how such a tool could be used by biologists to directly represent their understanding of a phenomenon of study.

Based on these guidelines, we introduce *Narrative Sketches*, a new type of 3D sketch dedicated to the exploration and communication of dynamic phenomena. Without any specific editing pipeline, the biologist can progressively create an animated 3D sketch representing a phenomenon of study by using simple sketched input to model new shapes but also embed animation cues representing constraints on the dynamics or the deformation process.

In addition to the concept of *Narrative Sketches* itself, our main technical contributions include:

- a sketch-based modeling method dedicated to the illustration of a biological phenomenon,
- an interactive solution to insert dynamics in a 3D sketch based on a simple vocabulary inspired by standard representations in biology and keyframe snippets.

**Note:** At the time of writing this thesis, the implementation of the contributions of this Chapter was still under development. Therefore, we presented below only the preliminary results and we plan to improve some algorithms at the time of publication. In addition, we plan to conduct a user study with both biologists and artists to validate the applicability of our system.

## 5.2 Depiction and narration in biology

### Pre-user study

We conducted a pre-study with researchers specialized in different subtopics of the field of biology to learn more about their study phenomena. We were particularly interested in how they characterize their phenomenon in terms of cell types, behaviors, and constraints. In addition, we asked them how they would sketch such features. Finally, we questioned them about the usefulness of an interactive and animated illustrative system.

To widen the range of phenomena, we interviewed our collaborator from IAB Grenoble in addition to a professor specialized in cell mechanics (Ecole Polytechnique), a research director on cell polarity and division (Institute Curie Paris), and another one who studies membrane and cytoskeleton dynamics in the context of breast cancer (Institute Curie Paris). They were all able to provide us simple and clear descriptions of the main steps of their study phenomenon, referring to analogies or visual vocabulary when necessary ("cutting similarly as scissors", "the cells hook to the collagen fibers to squeeze through it"). In particular, they present their phenomenon as a story in which a default animated environment evolves in response to some triggering events. Although their oral explanations were clear, they had no visual medium to highlight the motions and deformations they were describing and support their talks. Indeed, the visualization of only static representations is not sufficient for biologists because it does not allow them to depict in a simplified way their understanding of their mental vision of the dynamics of a phenomenon. Moreover, while they can easily express their knowledge orally, it remains quite complicated for them to do the same with the help of sketches. Indeed, providing a legible drawing of an animated phenomenon requires good artistic skills that they usually do not possess. Therefore, as explained in the introduction, they usually use different sources of representations, from real data to schematic or more explicit drawings, which may also potentially require the use of professional artists.

### Analysis of illustrations

Based on these insights and discussions with our collaborators, we selected some illustrations to emphasize their characteristics. The main idea was to use them as case studies. Therefore, we analyzed these sketches to identify the models we could use to create the elements of interest,

which are often part of a complex visual environment, and also which types of motions and deformations we should target.

We relied on the following three illustrations (two of which were created by Renaud Chabrier), which seemed to us sufficiently representative to give us an overview of cell biology phenomena and the way they are usually illustrated:

- Figure 5.3 depicts the immune response in the context of a cut,
- Figure 5.4 focuses on the escape of cancer cells from a bladder to a collagen field,
- Figure 5.5 proposed by Jean-Luc Coll, is taken from a biology article [SMJ18] and represents the propagation of a tumor and the consequence of the surrounding environment.

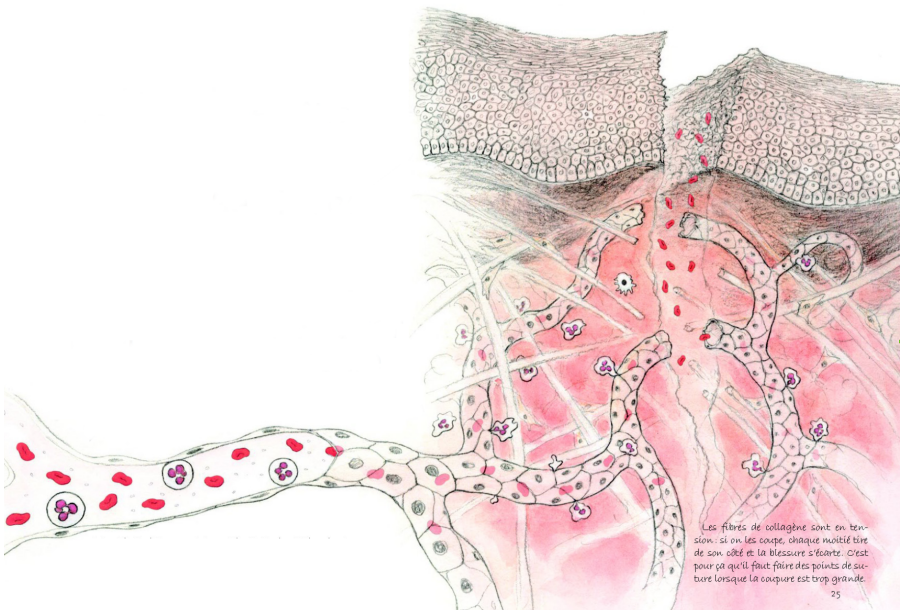



**Figure 5.3:** Immunology illustration from the magazine Globule, by our collaborator. © Renaud Chabrier.

From our observations, we first roughly classified elements according to their level of organization:

- Upper level (everything upper than intermediary): skin (Figure 5.3) and bladder (Figure 5.4),
- Intermediary: vessel and fiber network,
- Cellular,
- Mesoscale (from inside of cells and lower): inside of shapes.

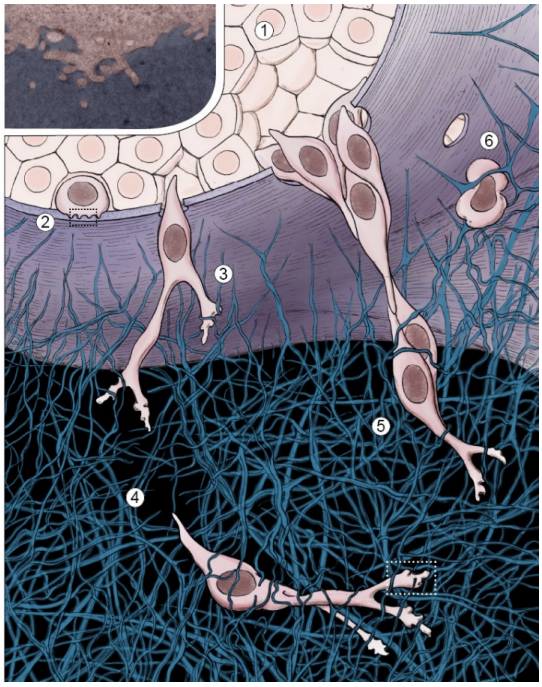



**Figure 5.4:** Illustration of the cancer cells propagation and escaping the bladder. © Renaud Chabrier.

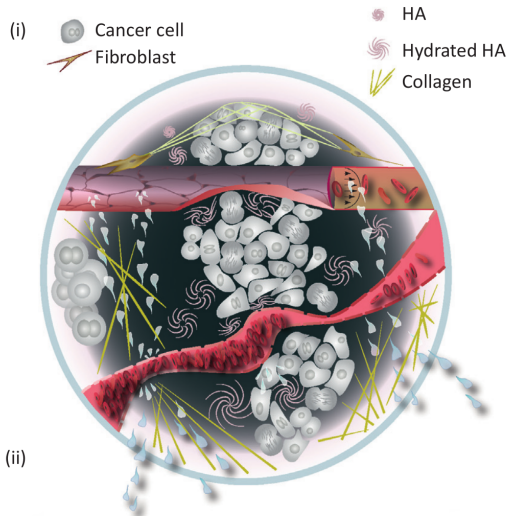



**Figure 5.5:** Illustration of the propagation of a tumor from Figure 1 of [SMJ18].

Then, when looking at their representation, we had organic (bladder, vessels, cells, and inside cells), linear (fibers) shapes, and large surfaces or volume (skin). We also noted the variety of rigidity parameters needed to depict various types of fibers: soft for Figure 5.4 and more rigid for Figures 5.3 and 5.5.

For the motion part, we noticed that in Figures 5.3 and 5.5, the cells are navigating inside the vessels, or more exactly, they let themselves be carried by a flow.

Finally, we identified four different deformation behaviors:

- Deformation of a cell during a change of context and adapted to the escape hole's size (Figure 5.3 and top of Figure 5.4),
- Cellular division (Figure 5.5),
- Adaptive deformation of a cell to navigate in a cluttered environment (Figure 5.4 bottom),
- Deformation of a vessel due to the tumor growing around it (Figure 5.5).

### Schematic vocabulary

The biologist working with us completed these observations by introducing the schematic vocabulary used to represent the dynamics in biological illustrations (see Figure 5.6):

- for the individual motion of an element,

  a single arrow representing the trajectory path,

- for the flow inside a vessel,

  three arrows followed by a straight line (homogeneous flow) or by a curve line (friction forces increasing towards the vessel borders and decreasing in the middle),

- for gradient concentration,

  from the environment, an isosceles triangle that partitions its surrounding environment into isolines to first attract elements towards its basis and whose amount of gradient is represented by its height,

  from a specific element, dispersion waves propagating in space following the element's isovalues and expressing a decreasing concentration.

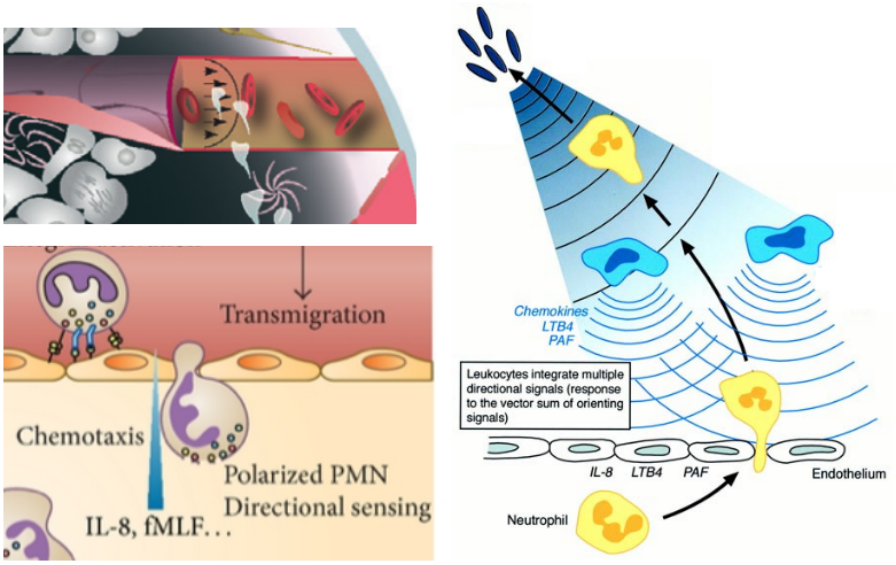



**Figure 5.6:** Schematic vocabulary in biological illustrations, examples of use in three different situations.

### Narrative scenarios

The three previous sections have respectively described our objective, the basic guidelines on representations, the motions and deformations we are targeting, and the schematic vocabulary we can use to insert motions and/or control them through attraction forces within an environment. Thus, the final step is to describe our target phenomena as narrative scenarios, which involves a series of key steps.

For the narrative scenarios presented below, we have mostly focused on the main steps of creation and animation to reproduce the main idea of a phenomenon. Indeed, since our objective is to depict a phenomenon in a simple way, we did not address all the small events that can occur. Furthermore, throughout the design pipeline, the first step is always to create the default, "stable" environment within which one or a series of triggering events will bring the phenomenon into existence.

For the phenomenon described in Figure 5.3, the triggering event is the cutting of the tissue, which separates the tissue into two parts, and also impacts the fibers and the vascular network. In addition to cutting the structures, this event directly results in blood cells near the cut escaping through it. On the other hand, the white cells, actors of the immune response, are attracted by the edge of the vessel containing them. They then roll on the vessel's inner border before escaping by deforming their shape to pass through holes. These escaped cells are then drawn to the cut to help the wound heal.

For our second case study (Figure 5.4), the cancer cells inside the bladder continue to divide, resulting in increasing the local pressure. The triggering event can be seen when the cells manage to escape the bladder by making holes in it. Some cells may even group together to force their way out of the bladder. They then have to adapt their shapes to navigate the cluttered environment made of a fiber network.

Finally, for our last phenomenon in Figure 5.5, the triggering event can be seen primarily as the division of cancer cells. Indeed, successive divisions first increase the pressure of the environment and the permeability of the vessels. Secondly, as tumor cells are spread out in space, they can interact with the surrounding entities when they approach them. For example, they deform a vessel by pushing its surface, which can also help these cells obtain more nutrients that promote the process of cell division. One case not desired in real-life situations is to see cancer cells enter a vessel and then navigate the vascular systems, with the prospect of infecting another area. Finally, when these cells are in contact with a set of fibers, they deform the fibers (elastic force) to a certain point before breaking them.

## 5.3 Overview

### 5.3.1 Our concept: Narrative sketches

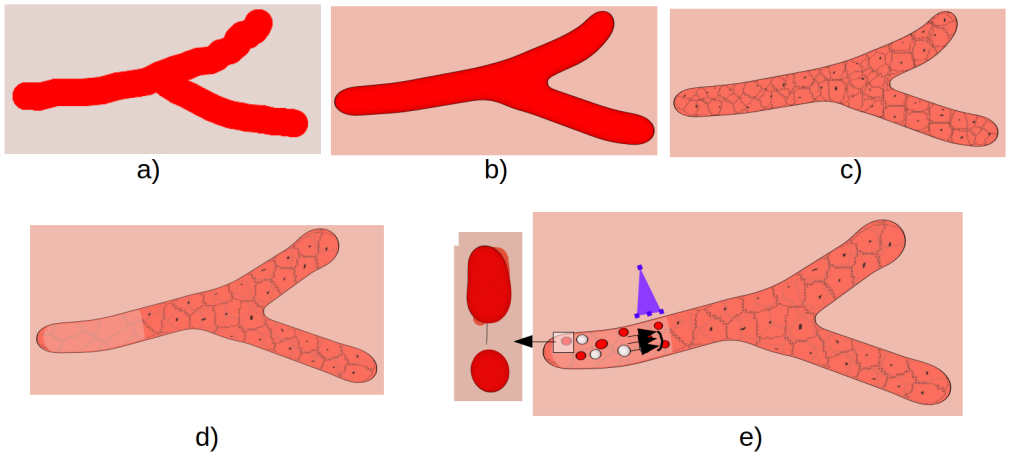



**Figure 5.7:** Overview of the preliminary state of our tool. a) The user paints a region to define a 2D silhouette; b) The sketch is inflated into an implicit surface (a 3D surface, here visualized using expressive sketch-like rendering); c) The user can sketch cell center-points to texture the vessel; d) The vessel could be made partially transparent to insert nested structures (e) The user can add cells, schematic symbols, and even defined keyframe snippets.

Based on representation and narration in biology, we propose a new type of 3D sketching targeting the exploration and communication of animated phenomena. During the interactive session, a biologist can progressively convey the understanding of a phenomenon using sketch inputs. In particular, as illustrated in Figure 5.7, these inputs remain simple and representative enough to let the user easily model a shape or a distribution of elements but also add dynamics constraints or define new deformation processes without any specific editing pipeline. We call this framework *Narrative Sketches*.

In addition to this concept, our main technical contributions include:

- a solution to create a 3D illustration,

  defined as a set of evolving and nested virtual worlds,

  and the tools to create it,

- an interactive solution to insert animation in a 3D sketch,

  based on a simple vocabulary inspired by standard representations in biology,

  and keyframe snippets.

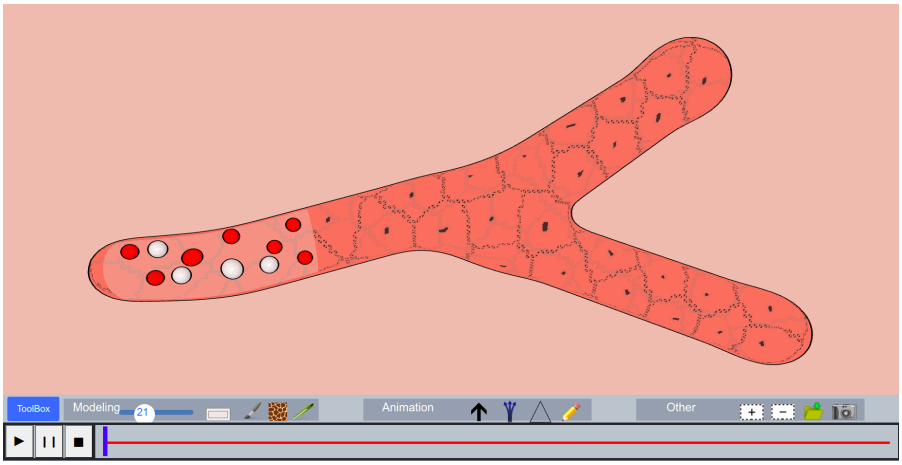



**Figure 5.8:** Overview of our interface with the sketching environment, the toolbox, and the timeline.

### 5.3.2 Presentation of our interface

On our interface (see Figure 5.8), the user can create and animate a 3D scene using the different modes proposed in the toolbox. We render the scene using a perspective camera whose parameters can be updated at any time to navigate in the 3D environment.

We have divided our menu into two main parts: a toolbox and a timeline. The timeline is played automatically each time a new element is animated and no matter the editing order. In addition, the user can click on the stop button to reset the positions of all the animated shapes. We have decomposed the timeline into regular time samples.

Following the left to right order, our toolbox is composed of:

- two brush parameters: a width slider and a color picker;
- four modeling modes: the painting brush to model organic shapes,

  the leopard patch to draw a Voronoi-style texture on a surface,

  the cutter to partially open a vessel around the cutting stroke;
- four animation modes: the single arrow to either create a trajectory path,

  the branching arrow for a flow field (inside a vessel),

  the triangle to create a global concentration gradient,

  the pencil to sketch keyframe snippet;
- two local zoom in and out and the classical load and save buttons.

Figure 5.8 presents the current functionalities of our interface. The final one will contain a feather to sketch distribution sample, which would be automatically synthesized in 3D (using the method in Chapter 4) and the radiation mode for the local concentration gradient around an element.

## 5.4 Creation of a biological environment

Our goal is to let biologists construct a complex 3D environment serving as support for their phenomenon of study.

### 5.4.1 Evolving and nested virtual worlds

*Nested structure*

From our first categorization of shapes (Section 5.2), we can directly notice that a biological environment can be composed of elements at multiple levels of detail. In addition, these elements are usually embedded in larger-scale structures, such as a vessel in which cells navigate or the skin wrapping everything. Therefore, we can first describe our 3D environment as a nested structure of theoretically, an infinite number levels of detail.

*Nested virtual worlds*

Moreover, if we consider our analysis of the motion when a cell moves inside a vessel, its motion is guided first by the flow but also by the shape of the vessel and possibly by some additional constraints imposed on this vessel. The same can be said, for example, for the elements inside a cell.

Therefore, as illustrated by Figure 5.9, we consider the environment of a phenomenon as a set of virtual worlds, the latter consisting of a habitat and virtual sub-worlds. For example, a vessel can be seen as a virtual world whose habitat contains a plasma (and potentially a set of properties or constraints) and a set of virtual worlds that are mainly cells. At the cell virtual world level, the habitat is composed of a cytoplasm and the virtual sub-worlds are even tinier elements. The properties of a habitat, such as density, fluidity, compressibility, impact the displacements of the virtual sub-worlds living inside it. Thus, we extend our previous definition by representing our 3D environment as a set of nested virtual worlds.

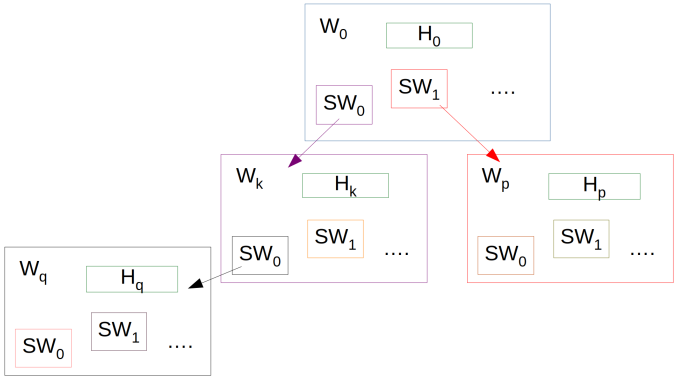



**Figure 5.9:** Overview of our nested virtual worlds model. Notations: $W_i$ and $SW_j$ stand respectively for virtual world $i$ and virtual sub-world $j$, $H_k$ for the habitat of the virtual world $k$.

*Evolving and nested virtual worlds*

Finally, because of their their ability to be guided by the user's strokes and especially deform to escape a habitat, virtual sub-worlds such as cells are not attached to a specific virtual world. In particular, after leaving their current virtual world, such sub-worlds will be connected to another one as depicted by Figure 5.10, thus their displacement and potentially deformations will be updated according to the habitat of new virtual world. Therefore, we can extend our definition again to take into account the evolutionary property of our nested virtual worlds.

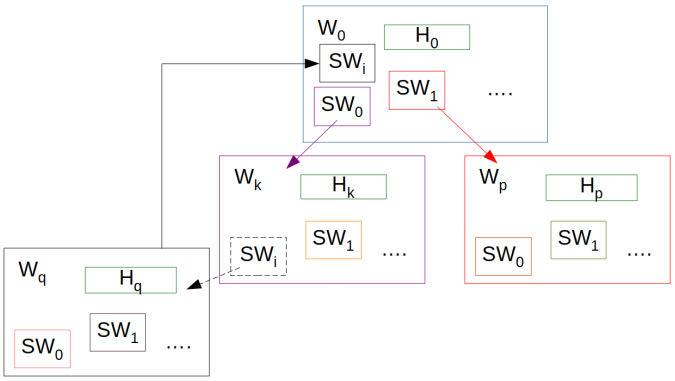



**Figure 5.10:** From nested virtual worlds to evolving and nested virtual worlds. Same notations as before: $W_i$ and $SW_j$ stand respectively for virtual world $i$ and virtual sub-world $j$, $H$ for habitat. Note that the sub-world labeled $SW_i$ in $W_k$ is leaving its current virtual world ($W_k$) to arrive in $W_0$.

This structure, quite simple in appearance, brings a lot of freedom to the creative process. For instance, as each world is defined individually and only its motion is relative to the surrounding habitat, there is no hierarchy of creation. Indeed, through this structure, we also promote multi-scale design without the constraint of creating the container before the content.

### 5.4.2 Modeling tools

In this section, we present the modeling tools we provide to users to create their 3D environment. In particular, as analyzed in Section 5.2, a virtual world can have various shapes, ranging from organic to linear and extruded surfaces. In addition, they can have an outer layer of texture and be part of a distribution. Finally, to match our evolving and nested structure, we support two modeling processes: creating a new virtual world directly in the global environment or defined as a sub-world of an existing world. In the latter case, we took inspiration from illustrative rendering to introduce a cutting tool that allows users to visually open a part of a shape to see its interior and create sub-shapes within it.

In our current implementation, we propose the following sketch-based tools:

- 2D silhouettes drawing to model organic shapes,
- distribution sample to create a 3D field of fibers,
- dot placing for surface texturing,
- cutting gestures to create a cutting window on a surface, similarly to cutter cutting.

**Modeling organic shapes from 2D silhouettes** By their organic nature and their ability to split into pieces, implicit surfaces were chosen as the best model to represent what we defined as organic shapes (bladder, vessel, cell, inner structures in a cell, etc.). To keep the sketching part as simple as possible, we used the work of Bernhardt et al. [BPCB08] and Zanni et al. [ZBQC13] to infer an implicit surface out of a user's 2D painting region (see Related work 2.1.1). To give a brief overview. The user paints a region from which they extract a convolution skeleton. This skeleton is then warped and convolved to generate a scale-invariant field. Finally, they generate the desired surface from this field by an implementation of the Marching Cubes algorithm.

To provide a non-photorealistic but expressive rendering, we adapted the fragment shader of the resulting surface by adding some Phong shading, a Fractional Brownian Motion noise, and also some contour detection and bump recognition, as depicted in Figure 5.11.

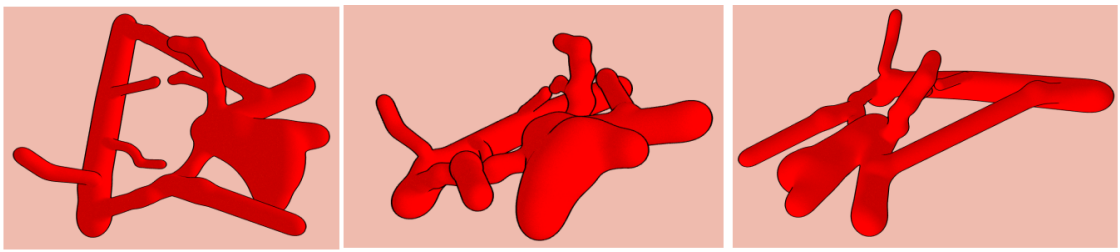


**Figure 5.11:** Example of vessel modeled on our system using Matisse [BPCB08] and enhanced with the Scalis [ZBQC13] field functions.

**Synthesizing a sketched distribution** From our case studies, we estimated that 3D environments are usually filled with arrangements of elements. To cite a few, it can range from field of fibers to distribution of cells, on and inside a vessel and on the skin tissue. Up to now, we have only proposed tools for the generation of a field of 3D fibers and the texturing of vessel-like shapes.

*Creating a 3D field of fibers from a sketch (in process of transfer)*

To let the user model the desired 3D field of elements in a simple way, we rely on an example-based synthesis and especially on the algorithm we proposed in Chapter 4. When this feature is transferred in the narrative sketching system, the biologist will be able to sketch a 2D sample of the desired fiber distributions. From this sketched sample, a Support Structure Hierarchy encoding the distribution parameters will be constructed and guide the real-time generation of a 3D environment inspired by the input. In contrast to the interactive exploration proposed in the last Chapter, we plan to display a 3D distribution block that can be moved and spatially extended by the user through direct manipulation.

*Texturing the outer layer of a vessel*

As illustrated in Figures 5.3 and 5.5, the vessels we target are composed of an external texture partitioning the space into regions. To achieve this and enrich our illustrative rendering, we assume that this partition can be represented by a Voronoi diagram on a free-form surface.

The ideal solution to achieve this would be to let the user define a sample of Voronoi cells on a surface and use a dedicated synthesis algorithm to propagate this distribution over the whole surface. The coarse Voronoi regions would then be defined by computing the geodesic distance between these cells and the surface triangulation. As the last step, a refinement of the mesh would have allowed obtaining a higher resolution of the Voronoi cells delineation. However, finding and computing the geodesic distance and the local refinement/re-meshing step may result in an algorithm that is too computationally expensive for our real-time needs.

On the other hand, we propose a simple yet improvable solution, which consists in letting the user click on the surface to position the Voronoi cells. Currently, the user has to define all the

centers of the Voronoi cells but we plan to improve this step using an existing synthesis algorithm adapted to this task. We create our Voronoi cells by calculating the Euclidean distance between the Voronoi cells and the vertices of the surface to group them with their closest kernel. Indeed, we assume that in the case of close Voronoi cells, the Euclidean distance can be considered as an acceptable approximation of the geodesic distance. However, as explained above, depending on the mesh triangulation, this can lead to a too coarse partitioning. To solve this problem, we apply a single refinement pass locally at the separation of two coarse regions. Figure 5.12 shows the result of our simple approach in the case of a 3D vessel.

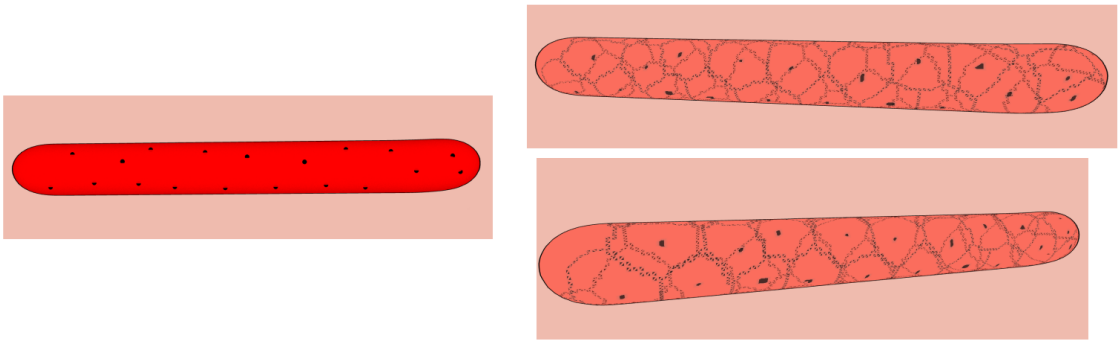


**Figure 5.12:** Example of texturing: (left) original vessel on which the user has positioned dots; (right) results from our texturing and according to two different viewpoints.

**Visually opening a virtual world** To allow the user to model sub-worlds directly within existing worlds, we propose a cutting tool similar to the cutter in that the length of the cutting stroke will determine the degree of openness. Indeed, we have followed the Figure 5.3 in its concept of a cutting window. Furthermore, we especially account for the fact that the structure the user will cut (ex: a vessel) would be in 3D, so it would not be possible to cut everything in the same cutting plane. As this feature is one of those still under study, two different scenarios for the opening of virtual worlds are still being considered: rectangle or eye-shape clipping. This choice is especially undetermined if the surface does not contain any texture; otherwise, the opening should follow the partition stitches. In the same spirit, the type of opening is correlated to the chosen distance function that can range from Gaussian to more complex convolution fields.

In addition, to allow opening and closing at any time and in various places in a virtual world, we decided to preserve the geometry of the surface and rely solely on the fragment shader to remove the unwanted areas. Figure 5.13 shows a simple example of cutting a surface containing a texture.

While this section focused on creating a 3D environment, depicting the underlying phenomenon also requires inserting animation cues that need to be simple to sketch and whose functions are easy to comprehend.

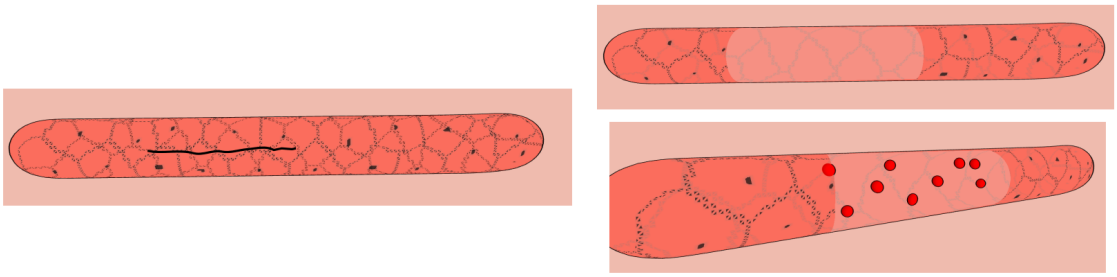


**Figure 5.13:** Example of visual opening: (left) original and textured vessel on which the user has drawn the cutting stroke; (right) results from the cut and after the user has inserted some cells inside.

## 5.5 From motion depiction to animation

To insert animation into a 3D environment, we rely on the schematic vocabulary extracted from our analysis (see Section 5.2) to have the correspondences between the schematic depictions, which biologists are usually familiar with and the expected behavior. In particular, we propose a dynamics model to encode such animation. However, these schematic depictions only act on the motion of animated shapes, from flow to attraction field. Therefore, we complement this model with the possibility to design deformation states through the introduction of our new concept of keyframe snippets.

### 5.5.1 Dynamic model & depiction of the standard representations

Even in the context of schematic representations where the goal is not to simulate motion, the use of simplified physical models, such as those we have used in architecture to deform buildings, can be interesting. Thus, we have adopted the following approach: forces are used for the animated elements to follow the user's gestures while remaining capable of interacting with each other and with obstacles. A force can either represent a motion to follow or a constraint to maintain. Moreover, motions and deformations generally respect intrinsic constraints (e.g., preservation of length, area, or volume for lines, surfaces, and solids, respectively). Finally, adding or removing forces allow the user to test different hypotheses.

From the identified schematic vocabulary (see Section 5.2), we propose the following tools and associated behaviors:

**Individual arrow:** one free-form stroke drawn by the user and represented by an arrow. This tool is only applied to the element closest to the beginning of the stroke and can be seen as a trajectory. Furthermore, as we want to use the speed of drawing as trajectory speed, we defined the underlying force by the gradient of the potential, taken at positions along the trajectory.

**Flow:** one stroke represented by three arrows and then another one for the friction factor. In contrast to the previous mode, the arrow is straight, and its direction is computed only from its first and last points. We compute the speed flow from the difference between these extreme points' time. We assume a flow to always be inserted on a habitat (see Section 5.4.1). The principal challenge of this flow tool is that the user only defines the desired flow field locally, and the latter should propagate throughout the whole structure where this habitat lies. Two behaviors can be used to achieve this. The first one consists of hard coding the flow direction throughout the whole structure, for instance, relying on a graph structure and retrieving all the local flow directions from the surface's skeleton bone directions. However, in the case of large vascular network, this can easily lead to a heavy structure. The alternative is to define a default flow direction as the user sketches it and rely on local collision avoidance with the surface's borders to compute locally the desired flow. While this solution does not require constructing a data structure, this may lead to undesired oscillations on the resulting motion (ping-pong effect). Indeed, we are currently looking for an optimal solution based on the default flow concept but using an intermediary region search local flow process. In addition to this flow, we provide more turbulence in the motion by adding some random noise motion.

**Gradient triangle:** one click to position the base and then drag-and-drop gesture to position the last vertex, and thus, orient and scale the represented triangle. As for the flow, we associate a triangle with a habitat. As a first simplification of this tool, we considered this tool as an local attraction. In particular, the triangle partitions its surrounding environment into isolines parallel to its height direction and whose potential values are computed from the ratio between the height over the base. Therefore, we define the associated force by first computing the potential of the two isolines surrounding the element to animate and taking the gradient of these potentials.

Figure 5.14 illustrates our flow and gradient triangle tools.

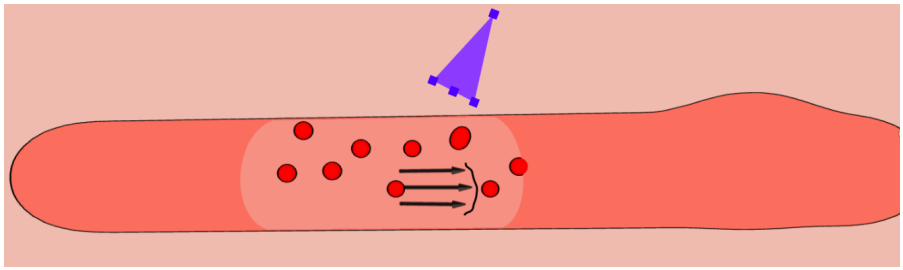


**Figure 5.14:** Representation of the flow and triangle gradient. Note that the gradient triangle can be oriented in any direction.

In addition to these tools, we added some collision detection and avoidance algorithm between animated sub-worlds and virtual world borders.

The schematic motion-design vocabulary we have defined, combined with the collision constraints, allows users to create a default animated environment. However, they are only focused on motion and are not suitable for the deformations that occur during a phenomenon (e.g., cell deformation to escape a habitat, cell division, etc.).

### 5.5.2 Deformations from sketches

To enable easy depiction and control of deformation, we have explored two approaches: deformation to escape from an environment by sketching holes on the surface and keyframe snippets.

*First approach: sketch holes on the surface*

Based on the identified deformations extracted from our case studies, one of the most common deformations taking place were those used to escape from an environment. The main characteristic of this deformation is that the amount of deformation varies with the size of the hole. To specify this hole size and determine the amount of available space for a shape to escape, we let the user sketch either a set of segments (dotted area) or a ring-shaped hole on the contact surface.

Let $t$ be the shape to deform (target) and $h$ the hole. We want to model the following phenomenon:

- if $R_t < R_h$, the target will escape without any deformation,
- if $R_t \ggg R_h$, the target's compression will be too important so the escape will be impossible,
- otherwise, the target will deform at constant volume,

with $R_t$, the target half-width and $R_h$ is the hole's half inner width.

We assume that the target, and the surface carrying holes, are represented by implicit surfaces. Therefore, we simulate the hole on the surface by adding to the potential field of the surface, a deformation field attached to the hole region. To match the desired behavior, we defined our deformation field as follows.

First, we partition the concerned region into three parts: an outer interior ring, a middle ring, and the rest (around the center). Then we have:

- Negative field in the outer interior ring to locally compress the target and makes it fit inside the available space,
- Positive field in the middle ring as to roughly preserve the target's volume,
- No field for the rest as the target has enough room to pass.

As this lead is still in progress, in our current version we did not really look at an accurate model to preserve the volume during the deformation, see Figure 5.15.

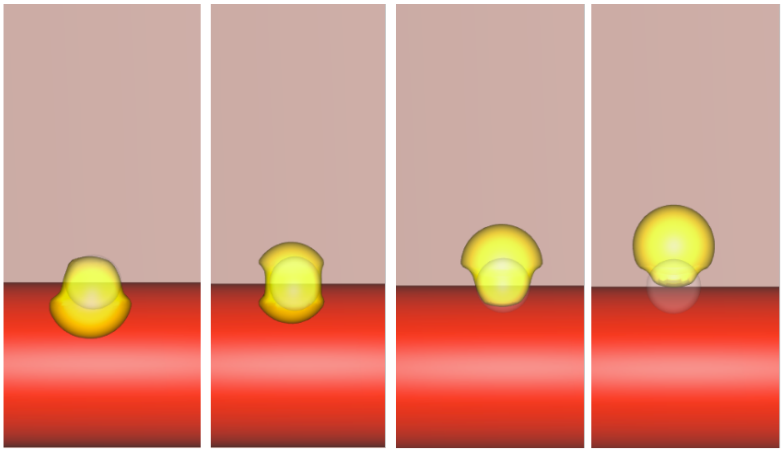


**Figure 5.15:** Early results on our deformation field approach.

While we could extend this approach to provide a valid model for shape deformation in a limited space, this method is specific to implicit surfaces and this kind of deformation. In addition, it would not be easily extendable to a variety of deformations such as cell division.

*Second and chosen approach: keyframe snippet*

To handle a wider variety of deformations with possible topology change while leaving the user in control, we introduce a new concept of keyframe snippets. We define a keyframe snippet by a set of deformation states (or keyframes) and a stroke gesture (for the deformation timing).

As illustrated in Figure 5.16, the specific interface for this process is composed of three parts: sketching, gesture, and visualization. For sketching, we drew on the painting metaphor used to create organic shapes to let the user paint the deformation states. Since the shapes we target are generally relatively simple, this technique allows users to quickly define the desired deformation. The user then sketches gesture strokes to define the timing between two key poses. Finally, the

visualization tab provides direct feedback on the appearance of the deformation. If the user is not satisfied or wishes to refine the deformation, the other modes are still available for editing.

We mainly base our current deformation model on Galin's [Gal97] thesis and in particular, the work on the characterization of the Minkowski sum of two polyhedra and the metamorphosis of BlobTree. To avoid tedious matching, we rely on the Minkowski sum characterization to compute approximate correspondences between vertices of the source and target skeletons. Then, we discretize this formula to obtain a set of vertex positions for each of our intermediate shapes. We also simplify the morphing by displacing only the source vertices while keeping their connectivity. As illustrated in Figure 5.16, the morphing is far from accurate (no volume preservation for example) but it gives us a first model to improve.

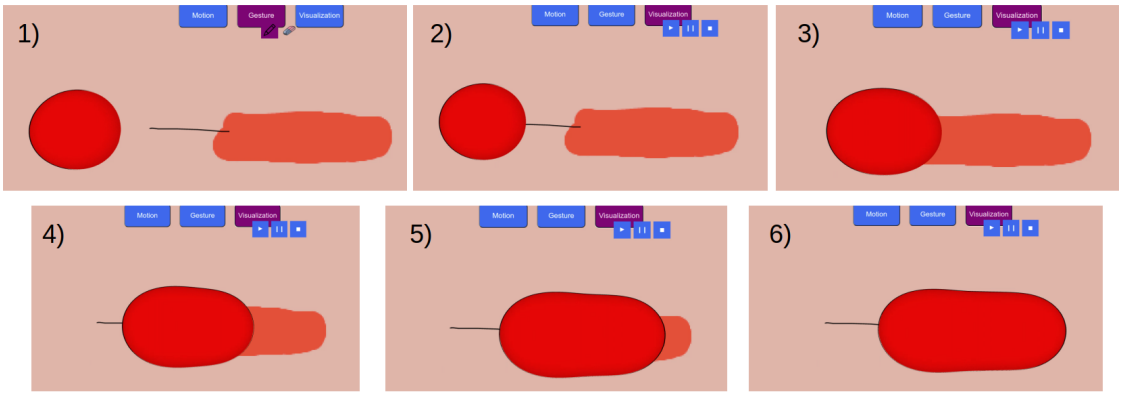



**Figure 5.16:** Early results on our keyframe snippet approach.

A first improvement would be to take all the skeleton features, such as the potential field, during the morphing to make a smooth transition from the source and target and match the desired target pose. In particular, this skeleton-based approach might be even more suitable to our cell division deformation thanks to the implementation of Scalis [ZBQC13] in Matisse [BPCB08], which makes implicit surfaces only blend if they are in contact. In addition, the recent work of these authors has allowed the skeleton to express the topology [ZGC15] but it has not been implemented in Matisse. Finally, we could also consider solutions combining some ideas explored in our two deformation models.

## 5.6 Perspectives of validation & Discussion

As this work is still under implementation and improvement, our results are limited to the ones we presented throughout this chapter; therefore, we can not provide a proper validation of our method.

### 5.6.1 Narrative scenarios

As a short-term objective of validation, we plan to use the narrative scenarios described by our collaborators to test the general applicability of our system but also verify if our choices of models for our sketching tools were relevant.

### 5.6.2 Discussion

Let's discuss the main features we currently offer and how they could be improved.

*Texturing*

As explained in Section 5.4.2 our simplistic model for texturing surfaces (skin, vessels) with a cell texture could be improved. To find a better balance between accuracy and real-time, it would be useful to first rely on a synthesis algorithm to decrease the user's work while maintaining real-time. An alternative could also be to provide a coarse texture in real time and gradually converge toward the accurate one. However, this may cause some visual disruption to the user.

*Visual cut operation*

Although we made the choice to cut only a window of surface, it could be interesting to set up a comparative user study to get more feedback about this choice or propose both possibilities. Indeed, proposing a complete opening along a certain viewpoint can ease the creation of virtual worlds as the access to the interior of this surface would be bigger. However, the concept of opening only a window allows the user to only focus on a small part of the data. Leveraging this window concept, we could also consider a brush that makes the first surface encountered a transparent area. In addition, this local area could also benefit from the example-based synthesis framework as the user will only need to specify a sample distribution in this area, which will be generated all along the surface. In this sense, the user will define the desired sample area while sketching directly in the context of the interior.

*Variety of animation*

This work focused on the ability to reproduce biological scenarios from inspirational illustrations. Based on our prior study and collaboration with a biologist, we tried to offer the most simple and understandable sketching tools. While we mainly focused on deformations of cell shapes, we had identified during our analysis other deformations, such as the deformation of a

vessel due to the pressure of the tumor growing on it. Such an example may be more complicated to represent by our keyframe snippet models because we assume motion during the deformation. One solution to this could be to let the user "pinch" the vessel at the desired location, but it might be complicated to correlate this with the pressure of the tumor. Another extension would be to augment our schematic vocabulary with the ability to insert, for example, pressure cues. As various types of pressure could happen in a system, a color code or other indication could help preserve the sketch readability. With this in mind, the possibility of having indications on some properties like the rigidity of fibers could also increase the range of application of our system.

## 5.7 Conclusion

In this chapter, we proposed a new 3D sketching tool dedicated to the narration of biological scenarios, a problem still little addressed in the literature. Based on a detailed analysis of biological illustrations and a collaboration with a biologist and a professional artist, we introduce the concept of Narrative Sketches allowing both the creation and animation of shapes from a simple sketch and aiming at the representation, communication, and exploration of a biological phenomenon.

Our final tool will be the first one, to our knowledge, to combine sketch-based modeling, sketch-based distribution synthesis, and sketch-based animation. The strength of our approach lies in the fact that both the creation and animation rely on simple sketch inputs, either by sketching linear fibers, relying on painting metaphors, or schematic representation. In addition, we extend the concept of nested structures to evolving nested structures promoting freedom in creative design and we propose a simplified physical model to infer both motion and deformation of shapes.

**Future work:** In addition to the short-term improvement of our system, we see two principal axes of extension.

While we rely on the standard representation of dynamics in biological illustrations completed by our keyframe snippet model, there is still room for other intuitive ways to create deformation or motion from sketches. For example, the schematic vocabulary used in comic strips could provide simple and understandable sketch symbols. Another interesting avenue would be to use a "thin" (in opposition to deep) learning approach and extend example-based synthesis to sketch-based animation when the motion of an element (or a set of elements) is analyzed and synthesized on a specific set of elements, for instance, all the white cells. To increase the potential of this approach, allowing the user to use both direct gestures (such as shaking the cell with the mouse or pen) and sketch-based input could allow for more varied animation samples.

The other direction of future work concerns multi-scale creation, representation, and visualization. While our evolving nested structure promotes an infinite level of details, our current implementation follows our case studies and focuses on only a few levels of representation. Indeed, our shape representation is independent of the camera's zoom, and if a user zooms in on the interior of a cell, some rendering problems may appear. In particular, this level of representation leads to two challenges: how to adapt the rendering of a shape according to the current level of detail; and how to transition between these representations. Such systems supplemented by a sketch rendering would enforce the notion of nested sketches. Some related work has already explored this issue of multi-resolution visualization and abstraction representation but only in the context of molecular structures [vdZLBI11].

# 6
# Conclusion

Throughout this thesis, we have explored sketch-based solutions for modeling shapes and textures, synthesizing shape distributions, and animating shapes. Motivated by the current imbalance between the richness of a paper sketch and its simplified representation when used on digital media, we proposed three real-time, user-friendly approaches for digitally creating augmented sketches:

- a new type of 3D sketch for easy creation and exploration of ideas applied to the preliminary design of man-made shapes and tested in the context of architectural design;
- a new synthesis framework from a sketch representing anisotropic distributions of bounded and unbounded elements, both in an extended 2D domain and in an immersive and exploratory 3D environment;
- a new 3D sketching framework dedicated to the representation, exploration, and communication of dynamic phenomena, tested in the context of cell biology.

For the design of all these three projects, we have relied on the issues identified in the introduction to set up our key contributions, namely the synthesis of anisotropic distributions from a sketch, nested sketches, and finally, time-evolving shapes guided by the user's sketch. In the following, we present how these contributions contribute to enhancing the creativity of the user.

**Synthesis of anisotropic distributions from a sketch** Motivated by anisotropic arrangements of shapes in architecture and biology, we have developed an interactive tool on which the user can quickly sketch anisotropic distributions that our system synthesizes in a larger 2D domain or a 3D environment that can be interactively explored. To interpret a sketched input, we first introduce a set of design guidelines expressing constraints on the desired synthesis. We then introduce a nested structure (Support Structure Hierarchy) to encode the key anisotropic features present in the user's 2D sketch. Based on these guidelines and this structure, we first propose a coarse-to-fine synthesis method in an extended 2D domain. Then, we extend sketch-based modeling to anisotropic 3D environments by embedding this extended distribution in 3D. Using

simple perceptual adjustments, users can immediately explore a 3D environment inspired by their sketch. The user may also set the default parameters of the synthesis environment and return to the sketching interface to edit the input and perform a new synthesis.

**Nested sketches:** While the creative process of designing a complex building may seem very different from synthesizing anisotropic distributions or even illustrating a cell biology phenomenon, all of these projects involve nested elements structured in a hierarchy. By exploiting the nested property of architectural models, we have been able to provide users with visualization and editing tools to display or hide part of a sketch. More importantly, this contributes to an easy and consistent design throughout the editing process, which is essential to promote user creativity in both the creation and exploration of alternative options. On the other hand, as our Support Structure Hierarchy points out, it can also be used to extract, encode, and replicate the multi-scale anisotropic structures present in a sketch and in an efficient manner. Furthermore, this allows to maintain a good level of visual diversity in the synthesized distribution of elements. Finally, in the context of cell biology phenomena, we have extended the notion of nested structures to **evolving nested structures** in which virtual worlds are defined as individuals and their motion is constrained by their parent but they can deform to leave the latter and become the child of a new one. The strength of this structure is that it promotes multi-scale creative design without imposing a hierarchy on the creation process.

**Time-evolving shapes guided by the user's sketch** The detailed analysis of architectural and cell biology sketches allows us to identify a set of visual cues essential for artists to convey motion, deformation, and even alternative options. These features range from stroke density (lighter stroke areas and over-sketching) to the schematic vocabulary used as a standard representation of dynamics in biological illustrations. Identifying these visual characteristics enables us to first propose sketching tools that allow users to insert the cues they are familiar with into a 3D sketch and then use their functions/actions to extend this static sketch into a dynamic sketch. The variation in stroke density in architectural drawings was indicating the confidence of presence which we exploited as a confidence field upon which curves, and then surfaces, would be interactively attracted and deformed promoting the exploration of alternative options. In contrast, the schematic vocabulary used in cell biology represents the dynamics of shapes. This helps us define the desired motion by a simplified physical model based on the forces associated with each symbol. In addition, we introduce the concept of a key-frame snippet, which can be viewed as an additional deformation symbol composed of a set of simple sketches interactively painted by the user and a gesture stroke defining the deformation timing. By relying on morphing techniques, the shape will be interactively deformed into the states sketched by the user. Therefore, whether for refinement or dynamics, **the evolution of the shape is always guided by the user's sketch**. Combined with

our nested structure, this property is even more important as the evolution of a shape affects those of its children.

While allowing for the synthesis of anisotropic distributions present in a sketch can be an interesting feature to add to the toolbox of a more general system, nested structures and time-evolving shapes introduce the concept of our common and general methodology.

## Perspectives

In addition to the future work specific to each chapter, we provide, in what follows, some more general challenges whose solution would be essential to extend our contributions to more general situations. They thus contribute to our main objective of establishing a general methodology.

**Specific challenges** This thesis explored solutions in sketch-based modeling, distributions synthesis, and sketch-based animation. While our primary focus has been on combining these creative processes, it might be interesting to look at improving each one separately. For sketch-based modeling, one perspective might be to focus on the representation at different levels of detail. For example, why not let the user refine a model by using its representation at different levels of detail. In shape distributions, only few works have addressed nested distributions, in the sense that the user can define a sample, and then this distribution is stored as a pipette that can be used to create a higher distribution. In particular, this can also be associated with the previous challenge regarding levels of detail. Finally, sketch-based animation brings some challenges in how to provide in simple sketches the desired animation and for a wide variety of shapes.

**Exploration of other applications** To test the ability of the methodology introduced in this thesis to generalize to more diverse situations, a first direction would be to choose another specific application (other than architecture and biology) and apply the same main steps: analyze its characteristics (from sketches or other types of data) and extract the features or constraints to be added when transferring into a 3D sketch. The choice of application can be to address a combination of sketching paradigms as we did with sketch-based modeling, sketch-based synthesis, and sketch-based animation, or even to focus on one. For instance, for 3D characters, distributions syntheses would not be the focus but there is room for improvement on the creation and more specifically on the animation. In other words, many sketch-based modeling methods can provide tools to represent a coarse model, however, it can be harder to sketch a detailed one. On the animation side, while this thesis tackles only simple shapes, such as cells, providing the ability to sketch the pose of more complex models to deform them on the fly remains a challenge as the correspondences are particularly difficult to determine.

**Towards a general methodology** Although we have limited our projects to specific applications, this thesis has highlighted the potential of a future common and general system. Indeed, as outlined in the introduction as our long-term vision, we are looking for tools that allow users to start the creative process from coarse structures and then refine them. Then, the expressions of motion and deformation could directly set the shape into animation while the geometric model is still in process. In addition, indications of alternate options or prior knowledge could help users clarify their mental vision. Finally, the user could iteratively refine the representation and animation without going through a specific editing pipeline. This ambitious future system would offer the most creative process possible, as it would follow the mental process of creation, from one idea to its refinement and then to the final result.

# Appendices

# A

# Expressive Modeling for Architecture

For this project, all the studies have been conducted physically and the architects needed to fill out a paper survey. We described below the questions that were asked to the architects during both the pre-study and the final user study.

## A.1 Pre-study

We conducted a pre-study to understand the architects' creative workflow and the needs and constraints on their ideal tool. Below are the questions asked during our pre-study. In the following, sketching refers to using the paper medium.

1. Is the sketch always the first step in designing a building? Can we start directly on digital tools? How does this choice affect the rest of the design process?
2. How is the sketching part important for you? How does it help you?
3. How important is the part of digital modeling, whether in 2D or 3D, to you? How does it help you?
4. Do you use digital media as a support for sketching or vice versa?
5. What is for you the ideal part between the sketch/drawing on paper and the digital modeling 2D or 3D?
6. What are, for you, the limitations of digital software?
7. What would be the perfect digital design tool for you?
8. In what way does, for you, digital software formalize the thought?
9. **[Sketch Vision Part]** What does uncertainty in a sketch mean to you? Is it important for you to keep the memory of all the sketched strokes, of the hierarchy in order of importance? In short, why is it important to keep the "draft" side of a sketch?

10. **[Rendering Part]** Do you find that the rendering of 2D and 3D models in digital software restricts your imagination/creativity? Would you like a more pencil/sketch rendering, e.g., a scanned sketch that is represented "naturally" and can be digitally modified, or even seen in 3D?

11. **[Animation Part]** Would you use animation in your building design work? Sketch overlay? Past or future uses of the building? History of creation with the different stages of design?

## A.2 User study

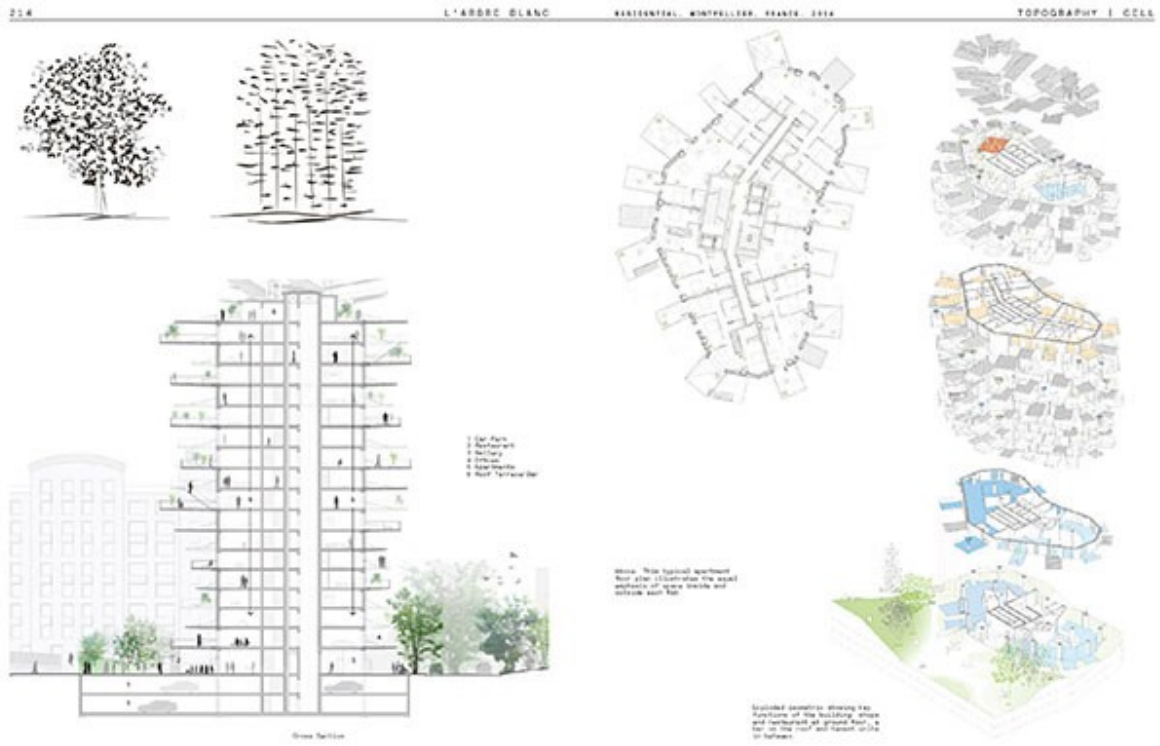
As explained in Section 3.6.1, our user study was composed of two parts: a creative part during which the architects were experimenting with our prototype on a WACOM tablet and a survey part in which they needed to answers to the following questions.

The following two pages present the two examples of creative architectural projects that we took as inspiration. In particular, we believe these examples of free-form building shapes, to be hard to model using the standard industrial software: Sou Fujimoto's White Tree (© Sou Fujimoto architects) and the Cottbus library from Herzog and de Meuron (© Herzog and de Meuron agency).

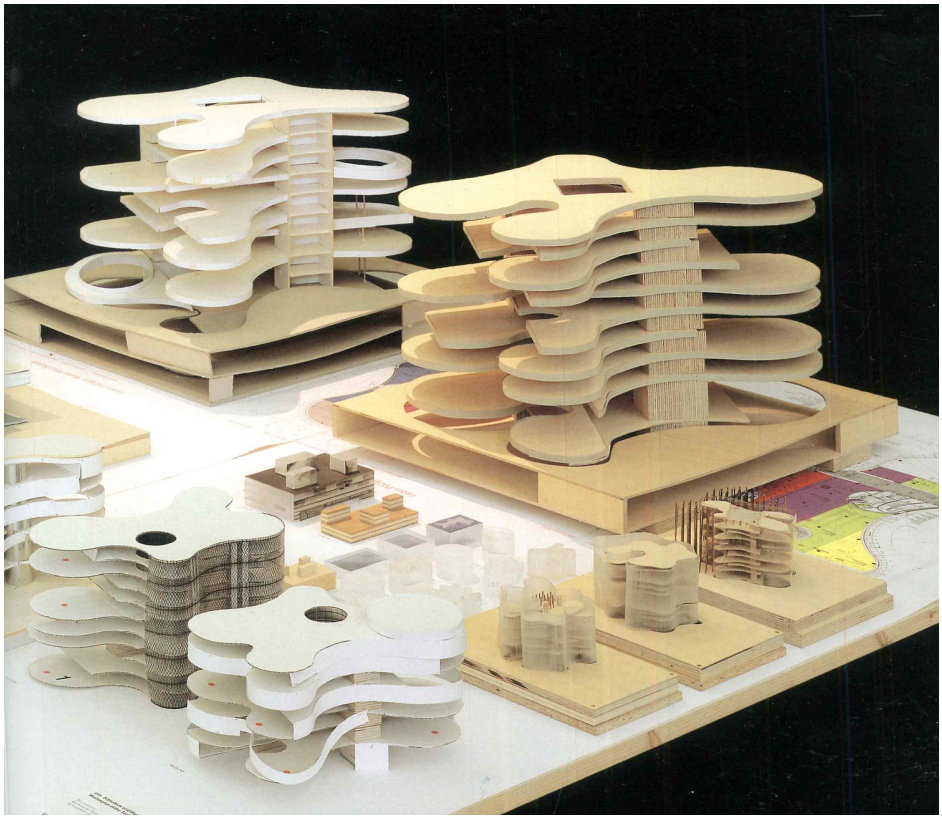
In two next pages after, the reader can find the paper sheet that was provided to each user during the study.

User study – Visual References

Model 1: Sou Fujimoto's White Tree



Model 2: Cottbus library from Herzog and de Meuron



# Nested Explorative Maps

## I. User Profile

Name:
Gender:
Age:
Years of experience in architecture:
Drawing hand:
Specialized software used:

## II. Evaluating NEM

a) Learning curve :

Need for specific training — Immediate usability

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

b) Free-form shapes:

Straight strokes and limited shapes — Free-form strokes and shapes

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

c) Editing pipeline:

Imposes a specific pipeline — Promotes creativity

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

d) Progressive creation, from coarse-to-fine:

Impossible — Promoted

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

e) Simultaneous design of the exterior and interior of a building:

Impossible — Promoted

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

g) Representation of uncertainty and exploration of possible options:

Impossible of representing uncertainty — Possible by over-tracing

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

No exploration of options — Excellent exploration

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

## II/ Creating from a model

a) Sketching an imposed model:

Difficulty — Easiness

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

b) Sketching an imaginary model

Difficulty — Easy, facilitate the creation

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

### III) Comparison with the frequently used digital software (precise the software)

- Software 1:
- Software 2 (to answer under the results from Software 1):

a) Slower learning curve for NEM — Faster learning curve for NEM

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

b) NEM is less suitable for the creation — NEM is better for the creation

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ | ☐ |

**IV) Remarks and free comments** (what I like, what was missing, possible use)

# B

# Anisotropic distributions from a sketch

## B.1 User study

This appendix completes Section 4.6.2 with screenshots of our online study as well as some results from users during the drawing session.

### B.1.1 Drawing session

During the drawing session, the user was asked to draw to complete a provided sketch in an extended domain delimited by a black square. The ratio between the input sketch and the output domain was $1:2$. For this interactive session, we sorted the tasks by increasing orders of complexity. Using a toolbox composed of a pencil and an eraser, the user had no timer to achieve the tasks. In the following Figures B.1, B.2, B.3 and B.4, the title and the left part (in a)) was what the user was seeing, in addition to a small toolbox menu. In addition, most of the examples were done with a mouse by non-artistic skills users.

For the first example, no anisotropic distribution was presented thus the task was just to preserve the bounded shapes during their replication. The idea was mainly to let the user be familiar with the presented interface and the main goal. Figure B.1 presents on the left, the provided input, and on the right, four results from users.

For the second task, we first introduced the notion of unbounded shapes without giving any indication to the user. We expected users to extend and replicate the unbounded shapes to the extended domain while avoiding intersections both inside a group and with another group. As depicted by Figure B.2, users globally respected the anisotropic distributions, while some decided to create loops (see the bottom right result).

We completed the previous example's goal by adding a singularity with the two vertical lines. We supposed that users would extend all the curves but only replicate the horizontal ones. From

all the results and as presented in the sample in Figure B.3, most of the outputs consist in just extending the bounded strokes to the output domain thus, this task has not been considered for our validation.

Finally, we closed this drawing session with an input composed of bounded shapes aligned along with two directions. The objective was to check whether users would preserve the anisotropic distributions. From the results in Figure B.4, the users saw different patterns regarding the anisotropic distributions of these shapes. Considering the limited available space, users usually favored either one shape or one direction over the other.

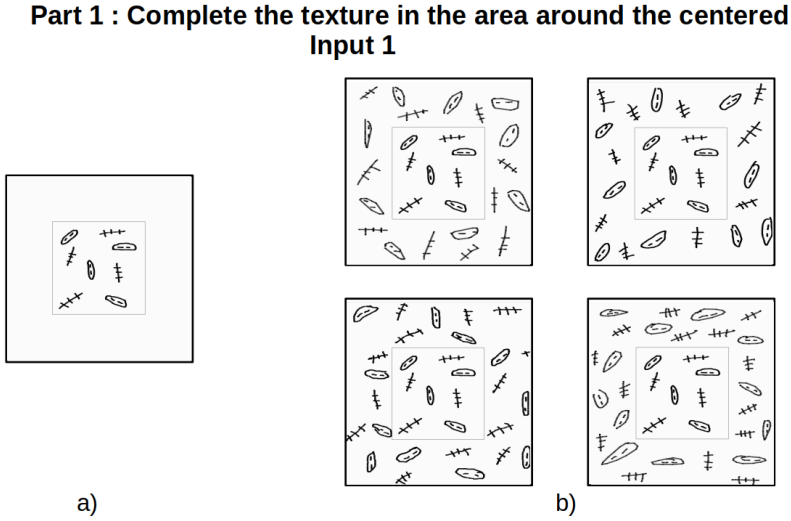



**Figure B.1:** Replication of bounded shapes: a) provided input with an empty area around it; b) set of four outputs taken from the users answers.

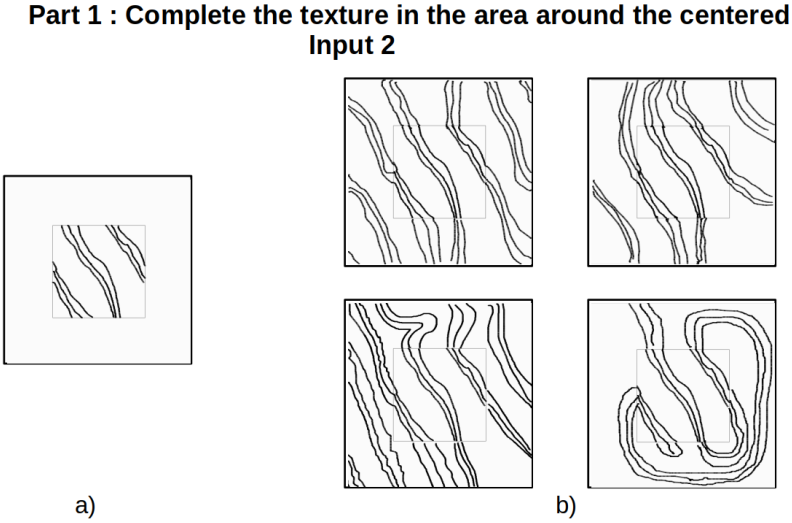



**Figure B.2:** Unbounded shapes extension and replication without collision: a) provided input with an empty area around it; b) set of four outputs taken from the users answers.

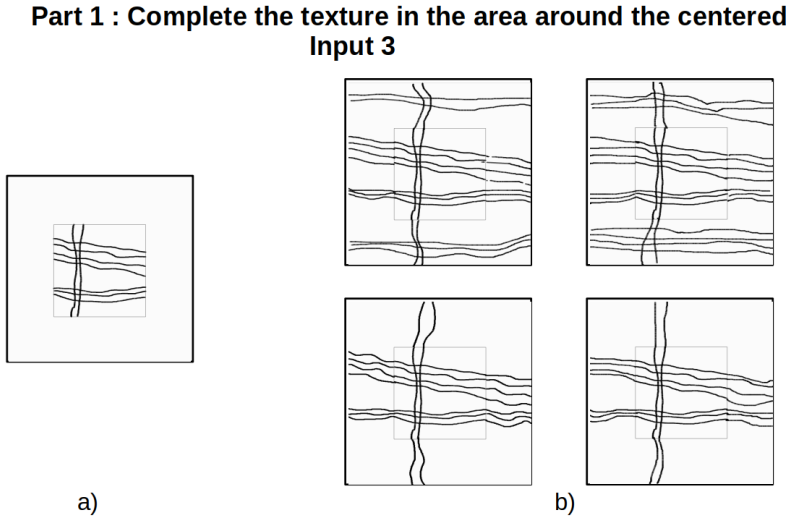



**Figure B.3:** No replication of singularity and no collision for unbounded shapes: a) provided input with an empty area around it; b) set of four outputs taken from the users answers.

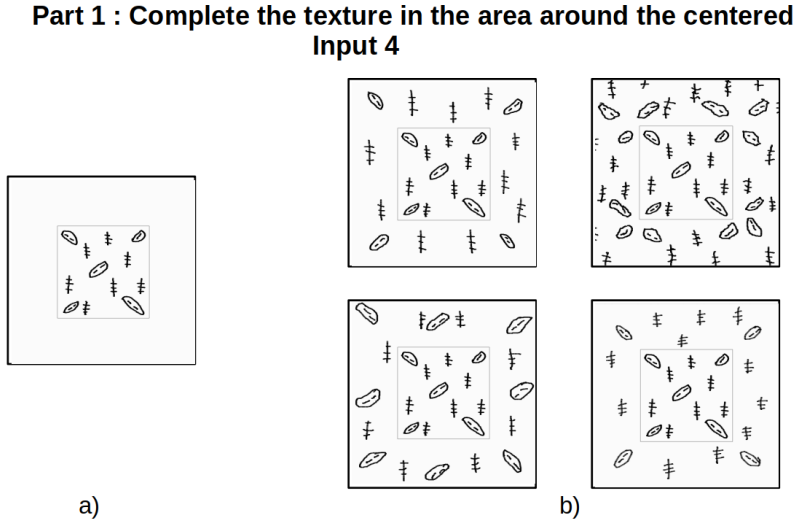



**Figure B.4:** Bounded shapes alignments: a) empty data; b) set of four outputs taken from the users answers.

### B.1.2 Comparison session

After letting users complete a provided input, they were asked to choose one or several outputs (depending on the task) from a provided 2D input. The goal was to pick the ones they thought to be the closest from the example. We believed that it was important for them to see at least two alternative options, no matter what they had drawn before. The number presented under each choice presents the proportion of users that picks this option. For the last task, as users could pick all the 3D immersions, we affected either 1 if the choice was unique, 0.5 if two choices were picked, and 0.3 if all the solutions were considered correct.

For the first two examples illustrated in Figure B.5, the user was asked to choose a unique solution between two alternative methods or none of them. We focused these examples on unbounded shapes and on our two less common design guidelines, which are the non-repetition of singularity, and no overlap of unbounded shapes in the output if they are not present in the input. The first input presented two groups of curves that are getting closer towards the end of the input space. The choice was between no overlap avoidance, a solution to overlap avoidance, or none of these outputs. Depending on the user choice at this step, we adapted the outputs for the second example to only focus on the replication of singular elements. The numbers below the repetitiveness task represent the proportion of users who picked no repetition on singular elements both for the case of with and without overlap.

Our second set of comparisons focused on evaluating our method with the method of Landes et al. [LGH13], after rendering the outputs from Landes et al. with the same style as ours. At this step of comparisons, we let the user pick both outputs if he or she considered that they were perceptually equal. Figure B.6 presents the outputs from Landes et al. and our method and under them the proportion of users who picked which choices.

Finally, to fix our depth choices for 3D immersion, we presented to the user `GIF` clips showing different groups of depth parameters for unbounded shapes (top) and bounded ones (bottom) (see Figure B.7). For both shape type, we assigned different 3D parameters (mid-point depth and slope) to lead directions, fiber medians, support segments or individual strokes.

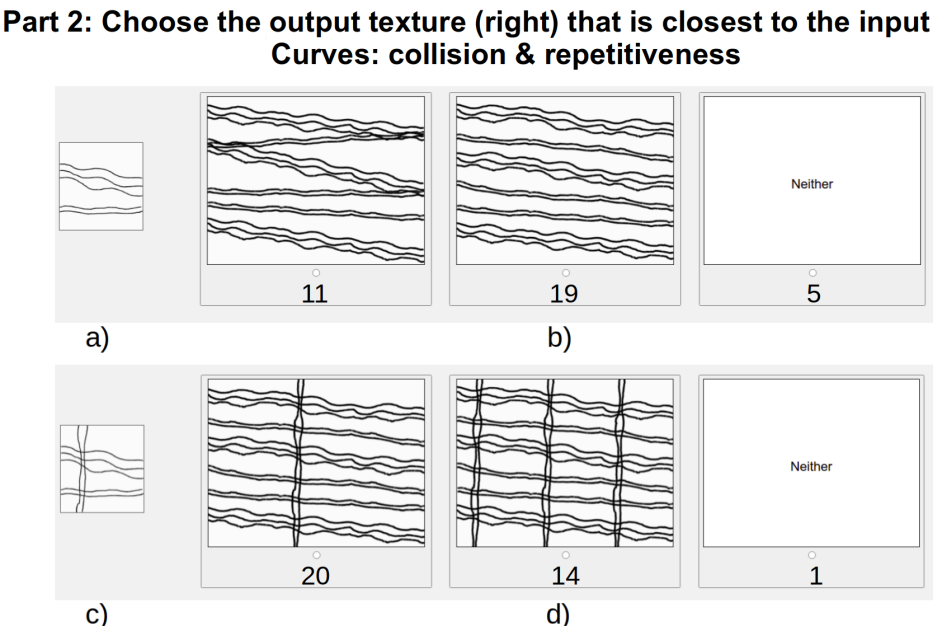



**Figure B.5:** (top) Comparison between overlaps or not: a) input, b) choice. (bottom) Comparison between repetitiveness or not: a) input b) choice.

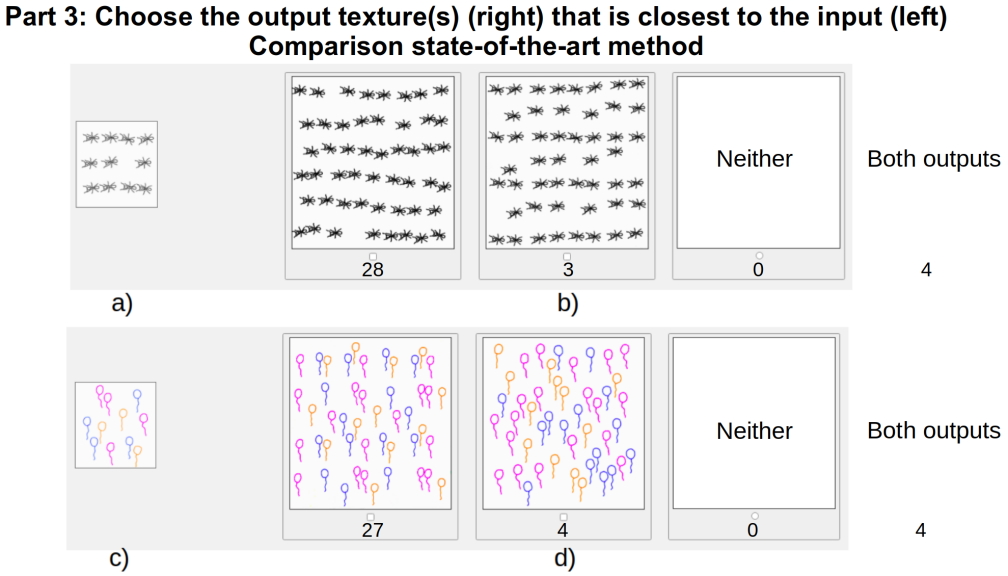



**Figure B.6:** Comparison with the best state-of-the-art method ([LGH13]).

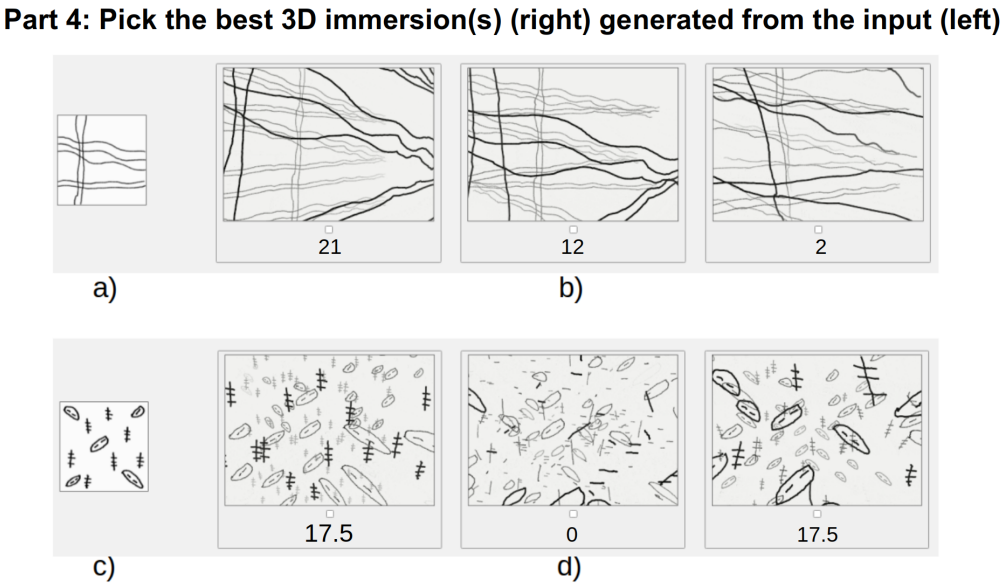



**Figure B.7:** (top) Comparison between 3D immersion for the unbounded strokes: a) input, b) 3D parameters determined at the lead direction, the fiber median or the fiber level. (bottom) Comparison between 3D immersion for unbounded strokes: c) input, d) 3D parameters determined at the fiber median, individual stroke or support segment level.

# Bibliography


[ABL*21] ABDRASHITOV R., BANG S., LEVIN D., SINGH K., JACOBSON A.: Interactive Modelling of Volumetric Musculoskeletal Anatomy. *ACM Trans. Graph. 40*, 4 (jul 2021). doi:10.1145/3450626.3459769.

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-Rigid-As-Possible Shape Interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 157–164. doi:10.1145/344779.344859.

[ADN*17] ARORA R., DAROLIA I., NAMBOODIRI V. P., SINGH K., BOUSSEAU A.: Sketchsoup: Exploratory Ideation Using Design Sketches. *Computer Graphics Forum 36*, 8 (2017), 302–312. doi:10.1111/cgf.13081.

[AF69] ATTNEAVE F., FROST R.: The determination of perceived tridimensional orientation by minimum criteria. *Perception & Psychophysics 6*, 6 (1969), 391–396. doi:10.3758/BF03212797.

[AF02] ARIKAN O., FORSYTH D. A.: Interactive Motion Generation from Examples. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, Association for Computing Machinery, p. 483–490. doi:10.1145/566570.566606.

[AGB04] ALEXE A., GAILDRAT V., BARTHE L.: Interactive Modelling from Sketches Using Spherical Implicit Functions. In *Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (New York, NY, USA, 2004), AFRIGRAPH '04, Association for Computing Machinery, p. 25–34. doi:10.1145/1029949.1029953.

[AHKG*18] ARORA R., HABIB KAZI R., GROSSMAN T., FITZMAURICE G., SINGH K.: *SymbiosisSketch: Combining 2D & 3D Sketching for Designing Detailed 3D Objects in Situ*. Association for Computing Machinery, New York, NY, USA, 2018, p. 1–15. doi:10.1145/3173574.3173759.

[AKA*17] ARORA R., KAZI R. H., ANDERSON F., GROSSMAN T., SINGH K., FITZMAURICE G.: Experimental Evaluation of Sketching on Surfaces in VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, Association for Computing Machinery, p. 5643–5654. doi:10.1145/3025453.3025474.

[Ale02] ALEXA M.: Recent Advances in Mesh Morphing. *Computer Graphics Forum 21*, 2 (2002), 173–198. doi:10.1111/1467-8659.00575.

[AS21] ARORA R., SINGH K.: Mid-Air Drawing of Curves on 3D Surfaces in Virtual Reality. *ACM Trans. Graph. 40*, 3 (jul 2021). doi:10.1145/3459090.

[Aut02] AUTODESK: Revit, built for Building Information Modeling (BIM), 2002. URL: https://www.autodesk.com/products/revit/overview.

[BA88] BERGEN J. R., ADELSON E. H.: Early vision and texture perception. *Nature 333*, 6171 (1988), 363–364. doi:10.1038/333363a0.

[BBB*97] BLOOMENTHAL J., BAJAJ C., BLINN J., WYVILL B., CANI M.-P., ROCKWOOD A., WYVILL G.: *Introduction to implicit surfaces*. Morgan Kaufmann, 1997.

[BBRF14] BROWNING M., BARNES C., RITTER S., FINKELSTEIN A.: Stylized Keyframe Animation of Fluid Simulations. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2014), NPAR '14, Association for Computing Machinery, p. 63–70. doi:10.1145/2630397.2630406.

[BBS08] BAE S.-H., BALAKRISHNAN R., SINGH K.: Ilovesketch: As-Natural-As-Possible Sketching System for Creating 3D Curve Models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2008), UIST '08, Association for Computing Machinery, p. 151–160. doi:10.1145/1449715.1449740.

[BBS09] BAE S.-H., BALAKRISHNAN R., SINGH K.: EverybodyLovesSketch: 3D Sketching for a Broader Audience. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2009), UIST '09, Association for Computing Machinery, p. 59–68. doi:10.1145/1622176.1622189.

[BBT*06] BARLA P., BRESLAV S., THOLLOT J., SILLION F., MARKOSIAN L.: Stroke Pattern Analysis and Synthesis. *Computer Graphics Forum 25*, 3 (2006), 663–671. doi:10.1111/j.1467-8659.2006.00986.x.

[BCD01] BOURGUIGNON D., CANI M.-P., DRETTAKIS G.: Drawing for Illustration and Annotation in 3D. *Computer Graphics Forum 20*, 3 (2001), 114–123. doi:10.1111/1467-8659.00504.

[BCV*15] BESSMELTSEV M., CHANG W., VINING N., SHEFFER A., SINGH K.: Modeling Character Canvases from Cartoon Drawings. *ACM Trans. Graph. 34*, 5 (nov 2015). doi:10.1145/2801134.

[BFKB99] BALAKRISHNAN R., FITZMAURICE G., KURTENBACH G., BUXTON W.: Digital Tape Drawing. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 1999), UIST '99, Association for Computing Machinery, p. 161–169. doi:10.1145/320719.322598.

[BKLP16] BAI Y., KAUFMAN D. M., LIU C. K., POPOVIĆ J.: Artist-Directed Dynamics for 2D Animation. *ACM Trans. Graph. 35*, 4 (jul 2016). doi:10.1145/2897824.2925884.

[BMSA19] BARRERA MACHUCA M. D., STUERZLINGER W., ASENTE P.: The Effect of Spatial Ability on Immersive 3D Drawing. In *Proceedings of the 2019 on Creativity and Cognition* (New York, NY, USA, 2019), C&C '19, Association for Computing Machinery, p. 173–186. doi:10.1145/3325480.3325489.

[BMV*11] BERNHARDT A., MAXIMO A., VELHO L., HNAIDI H., CANI M.-P.: Real-Time Terrain Modeling Using CPU-GPU Coupled Computation. In *2011 24th SIBGRAPI Conference on Graphics, Patterns and Images* (2011), pp. 64–71. doi:10.1109/SIBGRAPI.2011.28.

[BN92] BEIER T., NEELY S.: Feature-Based Image Metamorphosis. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1992), SIGGRAPH '92, Association for Computing Machinery, p. 35–42. doi:10.1145/133994.134003.

[BPCB08] BERNHARDT A., PIHUIT A., CANI M.-P., BARTHE L.: Matisse : Painting 2D regions for Modeling Free-Form Shapes. In *SBM'08 - Eurographics Workshop on Sketch-Based Interfaces and Modeling* (Annecy, France, June 2008), Alvarado C., Cani M.-P., (Eds.), SBM'08 Proceedings of the Fifth Eurographics conference on Sketch-Based Interfaces and Modeling, Eurographics Association, pp. 57–64. doi:10.2312/SBM/SBM08/057-064.

[Bri15] BRIDSON R.: *Fluid simulation for computer graphics*. CRC press, 2015.

[BVS16] BESSMELTSEV M., VINING N., SHEFFER A.: Gesture3D: Posing 3D Characters via Gesture Drawings. *ACM Trans. Graph. 35*, 6 (nov 2016). doi:10.1145/2980179.2980240.

[CA06] CANI M.-P., ANGELIDIS A.: Towards Virtual Clay. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 67–83. doi:10.1145/1185657.1185676.

[CGNS17] CICCONE L., GUAY M., NITTI M., SUMNER R. W.: Authoring Motion Cycles. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, Association for Computing Machinery. doi:10.1145/3099564.3099570.

[CH01] CANI M.-P., HORNUS S.: Subdivision-curve primitives: a new solution for interactive implicit modeling. In *Proceedings International Conference on Shape Modeling and Applications* (2001), pp. 82–88. doi:10.1109/SMA.2001.923378.

[CHZ00] COHEN J. M., HUGHES J. F., ZELEZNIK R. C.: Harold: A World Made of Drawings. In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2000), NPAR '00, Association for Computing Machinery, p. 83–90. doi:10.1145/340916.340927.

[CiRL*16] CHOI B., I RIBERA R. B., LEWIS J. P., SEOL Y., HONG S., EOM H., JUNG S., NOH J.: SketchiMo: Sketch-Based Motion Editing for Articulated Characters. *ACM Trans. Graph. 35*, 4 (July 2016). doi:10.1145/2897824.2925970.

[CKX*08] CHEN X., KANG S. B., XU Y.-Q., DORSEY J., SHUM H.-Y.: Sketching Reality: Realistic Interpretation of Architectural Designs. *ACM Trans. Graph. 27*, 2 (may 2008). doi:10.1145/1356682.1356684.

[CMH*10] CHEN X., MORVAN Y., HE Y., DORSEY J., RUSHMEIER H.: An integrated image and sketching environment for archaeological sites. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops* (2010), pp. 35–42. doi:10.1109/CVPRW.2010.5543514.

[CMZ*99] COHEN J. M., MARKOSIAN L., ZELEZNIK R. C., HUGHES J. F., BARZEL R.: An interface for sketching 3d curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), pp. 17–21.

[Coo86] COOK R. L.: Stochastic Sampling in Computer Graphics. *ACM Trans. Graph. 5*, 1 (jan 1986), 51–72. doi:10.1145/7529.8927.

[CS07] CORDIER F., SEO H.: Free-Form Sketching of Self-Occluding Objects. *IEEE Computer Graphics and Applications 27*, 1 (2007), 50–59. doi:10.1109/MCG.2007.8.

[CSGC16] CORDIER F., SINGH K., GINGOLD Y., CANI M.-P.: Sketch-based modeling. In *SIGGRAPH ASIA 2016 Courses* (New York, NY, USA, 2016), SA '16, Association for Computing Machinery. doi:10.1145/2988458.2988504.

[CSMS13] CORDIER F., SEO H., MELKEMI M., SAPIDIS N. S.: Inferring mirror symmetric 3D shapes from sketches. *Computer-Aided Design 45*, 2 (2013), 301–311.

[CYI*12] CHOI M. G., YANG K., IGARASHI T., MITANI J., LEE J.: Retrieval and Visualization of Human Motion Data via Stick Figures. *Computer Graphics Forum 31*, 7 (2012), 2057–2065. doi:10.1111/j.1467-8659.2012.03198.x.

[DAC*03] DAVIS J., AGRAWALA M., CHUANG E., POPOVIC Z., SALESIN D.: A Sketching Interface for Articulated Figure Animation. In *Symposium on Computer Animation* (2003), Breen D., Lin M., (Eds.), The Eurographics Association. doi:10.2312/SCA03/320-328.

[DAI*18] DELANOY J., AUBRY M., ISOLA P., EFROS A. A., BOUSSEAU A.: 3D Sketching using Multi-View Deep Volumetric Prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 1 (2018), 1–22. doi:10.1145/3203197.

[Dee95] DEERING M. F.: HoloSketch: A Virtual Reality Sketching/Animation Tool. *ACM Trans. Comput.-Hum. Interact. 2*, 3 (sep 1995), 220–238. doi:10.1145/210079.210087.

[DGK*20] DREY T., GUGENHEIMER J., KARLBAUER J., MILO M., RUKZIO E.: VRSketchIn: Exploring the Design Space of Pen and Tablet Interaction for 3D Sketching in Virtual Reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2020), CHI '20, Association for Computing Machinery, p. 1–14. doi:10.1145/3313831.3376628.

[DLCB11] DELAME T., LÉON J.-C., CANI M.-P., BLANCH R.: Gesture-based design of 2D contours: an alternative to sketching? In *SBIM 2011 - International Symposium on Sketch-Based Interfaces and Modeling* (Vancouver, Canada, Aug. 2011), Hammond T., Neale A., (Eds.), Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, Eurographics, Eurographics, pp. 63–70. Session: Sketching & design. doi:10.1145/2021164.2021176.

[DPS15] DE PAOLI C., SINGH K.: SecondSkin: Sketch-Based Construction of Layered 3D Models. *ACM Trans. Graph. 34*, 4 (jul 2015). doi:10.1145/2766948.

[DRvdP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector Graphics Animation with Time-Varying Topology. *ACM Trans. Graph. 34*, 4 (jul 2015). doi:10.1145/2766913.

[DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image Melding: Combining Inconsistent Images Using Patch-Based Synthesis. *ACM Trans. Graph. 31*, 4 (jul 2012). doi:10.1145/2185520.2185578.

[DSC*20] DVOROŽŇÁK M., SÝKORA D., CURTIS C., CURLESS B., SORKINE-HORNUNG O., SALESIN D.: Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation. *ACM Trans. Graph. 39*, 6 (nov 2020). doi:10.1145/3414685.3417805.

[DSJ19] DAVISON T., SAMAVATI F., JACOB C.: Interactive example-palettes for discrete element texture synthesis. *Computers & Graphics 78* (2019), 23 – 36. doi:10.1016/j.cag.2018.10.016.

[DXS*07] DORSEY J., XU S., SMEDRESMAN G., RUSHMEIER H., MCMILLAN L.: The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)* (2007), pp. 201–210. doi:10.1109/PG.2007.64.

[EBC*15] ENTEM E., BARTHE L., CANI M.-P., CORDIER F., VAN DE PANNE M.: Modeling 3D animals from a side-view sketch. *Computers & Graphics 46* (2015), 221–230. Shape Modeling International 2014. doi:10.1016/j.cag.2014.09.037.

[EF01] EFROS A. A., FREEMAN W. T.: Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, Association for Computing Machinery, p. 341–346. doi:10.1145/383259.383296.

[EL99] EFROS A., LEUNG T.: Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* (1999), vol. 2, pp. 1033–1038 vol.2. doi:10.1109/ICCV.1999.790383.

[ENMGC19] ECORMIER-NOCCA P., MEMARI P., GAIN J., CANI M.-P.: Accurate Synthesis of Multi-Class Disk Distributions. *Computer Graphics Forum 38*, 2 (2019), 157–168. doi:10.1111/cgf.13627.

[ERB*12] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-Based Shape Retrieval. *ACM Trans. Graph. 31*, 4 (jul 2012). doi:10.1145/2185520.2185527.

[EVC*15] EMILIEN A., VIMONT U., CANI M.-P., POULIN P., BENES B.: WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *ACM Transactions on Graphics 34*, 4 (Aug. 2015), 11. doi:10.1145/2766975.

[FBR*17] FONDEVILLA A., BOUSSEAU A., ROHMER D., HAHMANN S., CANI M.-P.: Patterns from photograph: Reverse-engineering developable products. *Computers & Graphics 66* (2017), 4–13. Shape Modeling International 2017. doi:10.1016/j.cag.2017.05.017.

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A Search Engine for 3D Models. *ACM Trans. Graph. 22*, 1 (jan 2003), 83–105. doi:10.1145/588272.588279.

[FRH*21] FONDEVILLA A., ROHMER D., HAHMANN S., BOUSSEAU A., CANI M.-P.: Fashion Transfer: Dressing 3D Characters from Stylized Fashion Sketches. *Computer Graphics Forum 40*, 6 (2021), 466–483. doi:10.1111/cgf.14390.

[FS89] FOGEL I., SAGI D.: Gabor filters as texture discriminator. *Biological cybernetics 61*, 2 (1989), 103–113. doi:10.1007/BF00204594.

[Gal97] GALIN E.: *Métamorphose et visualisation de blobs à squelettes*. PhD thesis, Lyon 1, 1997.

[GBK*02] GROSSMAN T., BALAKRISHNAN R., KURTENBACH G., FITZMAURICE G., KHAN A., BUXTON B.: Creating Principal 3D Curves with Digital Tape Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2002), CHI '02, Association for Computing Machinery, p. 121–128. doi:10.1145/503376.503398.

[GBS03] GROSSMAN T., BALAKRISHNAN R., SINGH K.: An Interface for Creating and Manipulating Curves Using a High Degree-of-Freedom Curve Input Device. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2003), CHI '03, Association for Computing Machinery, p. 185–192. doi:10.1145/642611.642645.

[GCR13] GUAY M., CANI M.-P., RONFARD R.: The Line of Action: An Intuitive Interface for Expressive Character Posing. *ACM Trans. Graph. 32*, 6 (Nov. 2013). doi:10.1145/2508363.2508397.

[GD11] GU Q., DENG Z.: Formation Sketching: An Approach to Stylize Groups in Crowd Simulation. In *Proceedings of Graphics Interface 2011* (Waterloo, CAN, 2011), GI '11, Canadian Human-Computer Communications Society, p. 1–8. doi:10.1.1.225.8318.

[GDG*17] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive Example-Based Terrain Authoring with Conditional Generative Adversarial Networks. *ACM Trans. Graph. 36*, 6 (nov 2017). doi:10.1145/3130800.3130804.

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: Texture Synthesis Using Convolutional Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Cambridge, MA, USA, 2015), NIPS'15, MIT Press, p. 262–270.

[GHL*20] GRYADITSKAYA Y., HÄHNLEIN F., LIU C., SHEFFER A., BOUSSEAU A.: Lifting Freehand Concept Sketches into 3D. *ACM Trans. Graph. 39*, 6 (Nov. 2020). doi:10.1145/3414685.3417851.

[GIZ09] GINGOLD Y., IGARASHI T., ZORIN D.: Structured Annotations for 2D-to-3D Modeling. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 148. doi:10.1145/1618452.1618494.

[GJS18] GIUNCHI D., JAMES S., STEED A.: 3D Sketching for Interactive Model Retrieval in Virtual Reality. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2018), Expressive '18, Association for Computing Machinery. doi:10.1145/3229147.3229166.

[GLA00] GALIN E., LECLERCQ A., AKKOUCHE S.: Morphing the BlobTree. *Computer Graphics Forum 19*, 4 (2000), 257–270. doi:10.1111/1467-8659.00462.

[GLX*16] GUO X., LIN J., XU K., CHAUDHURI S., JIN X.: CustomCut: On-demand Extraction of Customized 3D Parts with 2D Sketches. *Computer Graphics Forum 35*, 5 (2016), 89–100. doi:10.1111/cgf.12966.

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. *Advances in neural information processing systems 27* (2014).

[GRGC15a] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Adding dynamics to sketch-based character animations. In *Proceedings of the Workshop on Sketch-Based Interfaces and Modeling* (Goslar, DEU, 2015), SBIM '15, Eurographics Association, p. 27–34. doi:10.5555/2810210.2810213.

[GRGC15b] GUAY M., RONFARD R., GLEICHER M., CANI M.-P.: Space-Time Sketching of Character Animation. *ACM Trans. Graph. 34*, 4 (July 2015). doi:10.1145/2766893.

[GSV*17] GORI G., SHEFFER A., VINING N., ROSALES E., CARR N., JU T.: FlowRep: Descriptive Curve Networks for Free-Form Design Shapes. *ACM Trans. Graph. 36*, 4 (jul 2017). doi:10.1145/3072959.3073639.

[Hal89] HALL R. W.: Fast Parallel Thinning Algorithms: Parallel Speed and Connectivity Preservation. *Commun. ACM 32*, 1 (jan 1989), 124–131. doi:10.1145/63238.63248.

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-Based Texture Analysis/Synthesis. In *Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3 - Volume 3* (USA, 1995), ICIP '95, IEEE Computer Society, p. 3648. doi:10.5555/839284.841332.

[HGA*10] HNAIDI H., GUÉRIN E., AKKOUCHE S., PEYTAVIE A., GALIN E.: Feature based terrain generation using diffusion equation. *Computer Graphics Forum 29*, 7 (2010), 2179–2186. doi:10.1111/j.1467-8659.2010.01806.x.

[HGY17] HAN X., GAO C., YU Y.: DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Trans. Graph. 36*, 4 (July 2017). doi:10.1145/3072959.3073629.

[HKYM17] HUANG H., KALOGERAKIS E., YUMER E., MECH R.: Shape Synthesis from Sketches via Procedural Models and Convolutional Networks. *IEEE Transactions on Visualization and Computer Graphics 23*, 8 (2017), 2003–2013. doi:10.1109/TVCG.2016.2597830.

[HLT*09] HURTUT T., LANDES P.-E., THOLLOT J., GOUSSEAU Y., DROUILLHET R., COEURJOLLY J.-F.: Appearance-Guided Synthesis of Element Arrangements by Example. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2009), NPAR '09, Association for Computing Machinery, p. 51–60. doi:10.1145/1572614.1572623.

[HMC*15] HAHN F., MUTZEL F., COROS S., THOMASZEWSKI B., NITTI M., GROSS M., SUMNER R. W.: Sketch Abstractions for Character Posing. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, Association for Computing Machinery, p. 185–191. doi:10.1145/2786784.2786785.

[HYNP20] HARVEY F. G., YURICK M., NOWROUZEZAHRAI D., PAL C.: Robust Motion In-Betweening. *ACM Trans. Graph. 39*, 4 (jul 2020). doi:10.1145/3386569.3392480.

[IBB15] IARUSSI E., BOMMES D., BOUSSEAU A.: BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Trans. Graph. 34*, 3 (may 2015). doi:10.1145/2710026.

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-Rigid-As-Possible Shape Manipulation. *ACM Trans. Graph. 24*, 3 (jul 2005), 1134–1141. doi:10.1145/1073204.1073323.

[IMIM08] IJIRI T., MÊCH R., IGARASHI T., MILLER G.: An Example-based Procedural System for Element Arrangement. *Computer Graphics Forum 27*, 2 (2008), 429–436. doi:10.1111/j.1467-8659.2008.01140.x.

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A Sketching Interface for 3D Freeform Design. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 1999), SIGGRAPH '99, Association for Computing Machinery, p. 11–es. doi:10.1145/1185657.1185772.

[Inc21] INC. O. I.: Boox Max Lumi2, 2021. URL: https://shop.boox.com/products/maxlumi2.

[Ise15] ISENBERG T.: A Survey of Illustrative Visualization Techniques for Diffusion-Weighted MRI Tractography. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data* (Cham, 2015), Hotz I., Schultz T., (Eds.), Springer International Publishing, pp. 235–256. doi:10.1007/978-3-319-15090-1_12.

[JHR*15] JUNG A., HAHMANN S., ROHMER D., BEGAULT A., BOISSIEUX L., CANI M.-P.: Sketching folds: Developable surfaces from non-planar silhouettes. *ACM Trans. Graph. 34*, 5 (nov 2015). doi:10.1145/2749458.

[JK16] JACKSON B., KEEFE D. F.: Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *IEEE Transactions on Visualization and Computer Graphics 22*, 4 (2016), 1442–1451. doi:10.1109/TVCG.2016.2518099.

[JT81] JOHNSTON O., THOMAS F.: *The illusion of life: Disney animation*. Disney Editions New York, 1981.

[Jul62] JULESZ B.: Visual Pattern Discrimination. *IRE Transactions on Information Theory 8*, 2 (1962), 84–92. doi:10.1109/TIT.1962.1057698.

[JZF*21] JIANG Y., ZHANG C., FU H., CANNAVÒ A., LAMBERTI F., LAU H. Y. K., WANG W.: *HandPainter - 3D Sketching in VR with Hand-Based Physical Proxy*. Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3411764.3445302.

[KALB18] KIM Y., AN S.-G., LEE J. H., BAE S.-H.: *Agile 3D Sketching with Air Scaffolding*. Association for Computing Machinery, New York, NY, USA, 2018, p. 1–12. doi:10.1145/3173574.3173812.

[KB16] KIM Y., BAE S.-H.: SketchingWithHands: 3D Sketching Handheld Products with First-Person Hand Posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (New York, NY, USA, 2016), UIST '16, Association for Computing Machinery, p. 797–808. doi:10.1145/2984511.2984567.

[KCG*14] KAZI R. H., CHEVALIER F., GROSSMAN T., ZHAO S., FITZMAURICE G.: Draco: Bringing Life to Illustrations with Kinetic Textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2014), CHI '14, Association for Computing Machinery, p. 351–360. doi:10.1145/2556288.2556987.

[KCGF14] KAZI R. H., CHEVALIER F., GROSSMAN T., FITZMAURICE G.: Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2014), UIST '14, Association for Computing Machinery, p. 395–405. doi:10.1145/2642918.2647375.

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture Optimization for Example-Based Synthesis. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 795–802. doi:10.1145/1186822.1073263.

[KFM*01] KEEFE D. F., FELIZ D. A., MOSCOVICH T., LAIDLAW D. H., LAVIOLA J. J.: CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, Association for Computing Machinery, p. 85–93. doi:10.1145/364338.364370.

[KG05] KHO Y., GARLAND M.: Sketching Mesh Deformations. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), I3D '05, Association for Computing Machinery, p. 147–154. doi:10.1145/1053427.1053452.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion Graphs. In *ACM SIGGRAPH 2008 Classes* (New York, NY, USA, 2002), SIGGRAPH '02, Association for Computing Machinery. doi:10.1145/1401132.1401202.

[KH06] KARPENKO O. A., HUGHES J. F.: SmoothSketch: 3D Free-Form Shapes from Complex Sketches. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 589–598. doi:10.1145/1179352.1141928.

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (2002), 585–594. doi:10.1111/1467-8659.t01-1-00709.

[KHR04] KARPENKO O., HUGHES J. F., RASKAR R.: Epipolar methods for multi-view sketching. In *Proceedings of the First Eurographics Conference on Sketch-Based Interfaces and Modeling* (Goslar, DEU, 2004), SBM'04, Eurographics Association, p. 167–173. doi:10.5555/2386249.2386273.

[KIZD12] KAZI R. H., IGARASHI T., ZHAO S., DAVIS R.: Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-Ink Illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2012), CHI '12, Association for Computing Machinery, p. 1727–1736. doi:10.1145/2207676.2208302.

[KNL*15] KASPAR A., NEUBERT B., LISCHINSKI D., PAULY M., KOPF J.: Self Tuning Texture Optimization. *Computer Graphics Forum 34*, 2 (2015), 349–359. doi:10.1111/cgf.12565.

[Kof55] KOFFKA K.: *Principles of Gestalt psychology*. Routledge & K. Paul, 1955.

[Kor02] KORT A.: Computer Aided Inbetweening. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2002), NPAR '02, Association for Computing Machinery, p. 125–132. doi:10.1145/508530.508552.

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Trans. Graph. 22*, 3 (jul 2003), 277–286. doi:10.1145/882262.882264.

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International Journal of Computer Vision* (1988). doi:10.1007/BF00133570.

[KZL07] KEEFE D., ZELEZNIK R., LAIDLAW D.: Drawing on Air: Input Techniques for Controlled 3D Line Illustration. *IEEE Transactions on Visualization and Computer Graphics 13*, 5 (2007), 1067–1081. doi:10.1109/TVCG.2007.1060.

[Las87] LASSETER J.: Principles of Traditional Animation Applied to 3D Computer Animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, Association for Computing Machinery, p. 35–44. doi:10.1145/37401.37407.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive Control of Avatars Animated with Human Motion Data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, Association for Computing Machinery, p. 491–500. doi:10.1145/566570.566607.

[LF08] LEE J., FUNKHOUSER T.: Sketch-based search and composition of 3d models. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling* (Goslar, DEU, 2008), SBM'08, Eurographics Association, p. 97–104.

[LGH13] LANDES P.-E., GALERNE B., HURTUT T.: A shape-aware model for discrete texture synthesis. *Computer Graphics Forum 32*, 4 (2013), 67–76. doi:10.1111/cgf.12152.

[LGK*10] LEE D., GLUECK M., KHAN A., FIUME E., JACKSON K.: A survey of modeling and simulation of skeletal muscle. *ACM Transactions on Graphics 28*, 4 (2010), 1–13. doi:10.1.1.157.287.

[LGK*17] LUN Z., GADELHA M., KALOGERAKIS E., MAJI S., WANG R.: 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 67–77.

[LLN*14] LIAO J., LIMA R. S., NEHAB D., HOPPE H., SANDER P. V., YU J.: Automating Image Morphing Using Structural Similarity on a Halfway Domain. *ACM Trans. Graph. 33*, 5 (sep 2014). doi:10.1145/2629494.

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. doi:10.1109/TIT.1982.1056489.

[LLX*01] LIANG L., LIU C., XU Y.-Q., GUO B., SHUM H.-Y.: Real-Time Texture Synthesis by Patch-Based Sampling. *ACM Trans. Graph. 20*, 3 (jul 2001), 127–150. doi:10.1145/501786.501787.

[LLZ*17] LI Y., LUO X., ZHENG Y., XU P., FU H.: SweepCanvas: Sketch-Based 3D Prototyping on an RGB-D Image. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), UIST '17, Association for Computing Machinery, p. 387–399. doi:10.1145/3126594.3126611.

[LPL*17] LI C., PAN H., LIU Y., TONG X., SHEFFER A., WANG W.: BendSketch: Modeling Freeform Surfaces through 2D Sketching. *ACM Trans. Graph. 36*, 4 (jul 2017). doi:10.1145/3072959.3073632.

[LPL*18] LI C., PAN H., LIU Y., TONG X., SHEFFER A., WANG W.: Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling. *ACM Trans. Graph. 37*, 6 (dec 2018). doi:10.1145/3272127.3275051.

[LSM*19] LEIMKÜHLER T., SINGH G., MYSZKOWSKI K., SEIDEL H.-P., RITSCHEL T.: Deep Point Correlation Design. *ACM Trans. Graph. 38*, 6 (Nov. 2019). doi:10.1145/3355089.3356562.

[LVPI18] LAWONN K., VIOLA I., PREIM B., ISENBERG T.: A Survey of Surface-Based Illustrative Rendering for Visualization. *Computer Graphics Forum 37*, 6 (2018), 205–234. doi:10.1111/cgf.13322.

[LWSF10] LI H., WEI L.-Y., SANDER P. V., FU C.-W.: Anisotropic Blue Noise Sampling. In *ACM SIGGRAPH Asia 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, Association for Computing Machinery. doi:10.1145/1866158.1866189.

[LZC21] LIU Z., ZHANG F., CHENG Z.: BuildingSketch: Freehand Mid-Air Sketching for Building Modeling. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (2021), pp. 329–338. doi:10.1109/ISMAR52148.2021.00049.

[LZLS21] LI X., ZHANG B., LIAO J., SANDER P.: Deep Sketch-guided Cartoon Video Inbetweening. *IEEE Transactions on Visualization and Computer Graphics* (2021), 1–1. doi:10.1109/TVCG.2021.3049419.

[MCC09] MIN J., CHEN Y.-L., CHAI J.: Interactive Generation of Human Animation with Deformable Motion Models. *ACM Trans. Graph. 29*, 1 (dec 2009). doi:10.1145/1640443.1640452.

[MGC*16] MILLIEZ A., GUAY M., CANI M.-P., GROSS M., SUMNER R. W.: Programmable Animation Texturing using Motion Stamps. *Computer Graphics Forum 35*, 7 (2016), 67–75. doi:10.1111/cgf.13004.

[MHM*09] MAHAJAN D., HUANG F.-C., MATUSIK W., RAMAMOORTHI R., BELHUMEUR P.: Moving Gradients: A Path-Based Method for Plausible Image Interpolation. *ACM Trans. Graph. 28*, 3 (jul 2009). doi:10.1145/1531326.1531348.

[MM10] MERRELL P., MANOCHA D.: Example-Based Curve Synthesis. *Comput. Graph. 34*, 4 (aug 2010), 304–311. doi:10.1016/j.cag.2010.05.006.

[MNB*14] MILLIEZ A., NORIS G., BARAN I., COROS S., CANI M.-P., NITTI M., MARRA A., GROSS M., SUMNER R. W.: Hierarchical Motion Brushes for Animation Instancing. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2014), NPAR '14, ACM, pp. 71–79. doi:10.1145/2630397.2630402.

[MP90] MALIK J., PERONA P.: Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. Am. A 7*, 5 (May 1990), 923–932. doi:10.1364/JOSAA.7.000923.

[MQW05] MAO C., QIN S. F., WRIGHT D. K.: A Sketch-Based Gesture Interface for Rough 3D Stick Figure Animation. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2005), Jorge J. A. P., Igarashi T., (Eds.), The Eurographics Association. doi:10.2312/SBM/SBM05/175-183.

[MWCS13] MILLIEZ A., WAND M., CANI M.-P., SEIDEL H.-P.: Mutable elastic models for sculpting structured shapes. *Computer Graphics Forum 32*, 2pt1 (2013), 21–30. Special Issue: Proc. Eurographics, May 2013, Girona, Spain. doi:10.1111/cgf.12022.

[MWLT13] MA C., WEI L.-Y., LEFEBVRE S., TONG X.: Dynamic Element Textures. *ACM Trans. Graph. 32*, 4 (July 2013). doi:10.1145/2461912.2461921.

[MWT11] MA C., WEI L.-Y., TONG X.: Discrete Element Textures. *ACM Trans. Graph. 30*, 4 (July 2011). doi:10.1145/2010324.1964957.

[NGDA*16] NISHIDA G., GARCIA-DORADO I., ALIAGA D. G., BENES B., BOUSSEAU A.: Interactive Sketching of Urban Procedural Models. *ACM Trans. Graph. 35*, 4 (July 2016). doi:10.1145/2897824.2925951.

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Trans. Graph. 26*, 3 (July 2007). doi:10.1145/1276377.1276429.

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836. doi:10.1111/j.1467-8659.2006.01000.x.

[NSACO05] NEALEN A., SORKINE O., ALEXA M., COHEN-OR D.: A Sketch-Based Interface for Detail-Preserving Mesh Editing. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 1142–1147. doi:10.1145/1186822.1073324.

[OBP*13] ÖZTIRELI A. C., BARAN I., POPA T., DALSTEIN B., SUMNER R. W., GROSS M.: Differential Blending for Expressive Sketch-Based Posing. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, Association for Computing Machinery, p. 155–164. doi:10.1145/2485895.2485916.

[OCB*16] OVSJANIKOV M., CORMAN E., BRONSTEIN M., RODOLÀ E., BEN-CHEN M., GUIBAS L., CHAZAL F., BRONSTEIN A.: Computing and Processing Correspondences with Functional Maps. In *SIGGRAPH ASIA 2016 Courses* (New York, NY, USA, 2016), SA '16, Association for Computing Machinery. doi:10.1145/2988458.2988494.

[OG12] ÖZTIRELI A. C., GROSS M.: Analysis and Synthesis of Point Distributions Based on Pair Correlation. *ACM Trans. Graph. 31*, 6 (Nov. 2012). doi:10.1145/2366145.2366189.

[OOI06] OKABE M., OWADA S., IGARASHI T.: Interactive Design of Botanical Trees Using Freehand Sketches and Example-Based Editing. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 18–es. doi:10.1145/1185657.1185779.

[OS18] ÖZTIRELI A. C., SINGH G.: Sampling Analysis Using Correlations for Monte Carlo Rendering. In *SIGGRAPH Asia 2018 Courses* (New York, NY, USA, 2018), SA '18, Association for Computing Machinery. doi:10.1145/3277644.3277783.

[OSSJ09] OLSEN L., SAMAVATI F. F., SOUSA M. C., JORGE J. A.: Sketch-based modeling: A survey. *Computers & Graphics 33*, 1 (2009), 85–103. doi:10.1016/j.cag.2008.09.013.

[PCP10] PIHUIT A., CANI M.-P., PALOMBI O.: Sketch-Based Modeling of Vascular Systems: a First Step Towards Interactive Teaching of Anatomy. In *SBIM 2010 - Sketch-Based Interfaces and Modeling* (Annecy, France, June 2010), Marc Alexa E. Y.-L. D., (Ed.), Eurographics Association, pp. 151–158. doi:10.2312/SBM/SBM10/151-158.

[PKM*11] PACZKOWSKI P., KIM M. H., MORVAN Y., DORSEY J., RUSHMEIER H., O'SULLIVAN C.: Insitu: Sketching architectural designs in context. *ACM Trans. Graph. 30*, 6 (dec 2011), 1–10. doi:10.1145/2070781.2024216.

[PS00] PORTILLA J., SIMONCELLI E. P.: A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision 40*, 1 (2000), 49–70. doi:10.1023/A:1026553619983.

[PS18] PREIM B., SAALFELD P.: A survey of virtual human anatomy education systems. *Computers & Graphics 71* (2018), 132–153. doi:10.1016/j.cag.2018.01.005.

[PWKK20] PENG M., WEI L.-Y., KAZI R. H., KIM V. G.: Autocomplete Animated Sculpting. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2020), UIST '20, Association for Computing Machinery, p. 760–777. doi:10.1145/3379337.3415884.

[PXW18] PENG M., XING J., WEI L.-Y.: Autocomplete 3D Sculpting. *ACM Trans. Graph. 37*, 4 (July 2018). doi:10.1145/3197517.3201297.

[reM17] REMARKABLE: remarkable, 2017. URL: https://remarkable.com/.

[RLT*17] RUDAKOVA V., LIN N., TRAYAN N., SEZGIN T. M., DORSEY J., RUSHMEIER H.: Cher-ish: A sketch- and image-based system for 3d representation and documentation of cultural heritage sites. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage* (Goslar, DEU, 2017), GCH '17, Eurographics Association, p. 195–199. doi:10.2312/gch.20171314.

[ROG17] ROVERI R., ÖZTIRELI A. C., GROSS M.: General Point Sampling with Adaptive Density and Correlations. *Computer Graphics Forum 36*, 2 (2017), 107–117. doi:10.1111/cgf.13111.

[ROM*15] ROVERI R., ÖZTIRELI A. C., MARTIN S., SOLENTHALER B., GROSS M.: Example Based Repetitive Structure Synthesis. *Computer Graphics Forum 34*, 5 (2015), 39–52. doi:10.1111/cgf.12695.

[Roo95] ROOSENDAAL T.: Blender, 1995. URL: https://www.blender.org/.

[RRS19] ROSALES E., RODRIGUEZ J., SHEFFER A.: SurfaceBrush: From Virtual Reality Drawings to Manifold Surfaces. *ACM Trans. Graph. 38*, 4 (jul 2019). doi:10.1145/3306346.3322970.

[RSW*07] ROSE K., SHEFFER A., WITHER J., CANI M.-P., THIBERT B.: Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Goslar, DEU, 2007), SGP '07, Eurographics Association, p. 163–172. doi:10.5555/1281991.1282014.

[Saa] SAAS: Anoto. URL: http://www.anoto.com/.

[SBSS12] SHAO C., BOUSSEAU A., SHEFFER A., SINGH K.: CrossShade: Shading Concept Sketches Using Cross-Section Curves. *ACM Trans. Graph. 31*, 4 (jul 2012). doi:10.1145/2185520.2185541.

[SCCS13] STANCULESCU L., CHAINE R., CANI M.-P., SINGH K.: Sculpting multi-dimensional nested structures. *Computers and Graphics 37*, 6 (Oct. 2013), 753–763. Special Issue: Shape Modeling International (SMI) Conference 2013. doi:10.1016/j.cag.2013.05.010.

[SCO17] SENDIK O., COHEN-OR D.: Deep Correlations for Texture Synthesis. *ACM Trans. Graph. 36*, 5 (July 2017). doi:10.1145/3015461.

[SCSI15] SASAKI N., CHEN H.-T., SAKAMOTO D., IGARASHI T.: Facetons: face primitives for building 3d architectural models in virtual environments. *Computer Animation and Virtual Worlds 26*, 2 (2015), 185–194. doi:10.1002/cav.1603.

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering* (2009), pp. 25–33.

[SDY*18] SU W., DU D., YANG X., ZHOU S., FU H.: Interactive sketch-based normal map generation with deep neural networks. *Proc. ACM Comput. Graph. Interact. Tech. 1*, 1 (jul 2018). doi:10.1145/3203186.

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 106–es. doi:10.1145/1275808.1276510.

[SI07] SHIN H., IGARASHI T.: Magic Canvas: Interactive Design of a 3-D Scene Prototype from Freehand Sketches. In *Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), GI '07, Association for Computing Machinery, p. 63–70. doi:10.1145/1268517.1268530.

[SKSK09] SCHMIDT R., KHAN A., SINGH K., KURTENBACH G.: Analytic drawing of 3d scaffolds. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, Association for Computing Machinery. doi:10.1145/1661412.1618495.

[SKv*14] SÝKORA D., KAVAN L., ČADÍK M., JAMRIŠKA O., JACOBSON A., WHITED B., SIMMONS M., SORKINE-HORNUNG O.: Ink-and-Ray: Bas-Relief Meshes for Adding Global Illumination Effects to Hand-Drawn Characters. *ACM Trans. Graph. 33*, 2 (apr 2014). doi:10.1145/2591011.

[SMJ18] STYLIANOPOULOS T., MUNN L. L., JAIN R. K.: Reengineering the Physical Microenvironment of Tumors to Improve Drug Delivery and Efficacy: From Mathematical Modeling to Bench to Bedside. *Trends in Cancer 4*, 4 (2018), 292–319. Special Issue: Physical Sciences in Oncology. doi:10.1016/j.trecan.2018.02.005.

[SPS01] SCHKOLNE S., PRUETT M., SCHRÖDER P.: Surface drawing: Creating organic 3d shapes with the hand and tangible tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2001), CHI '01, Association for Computing Machinery, p. 261–268. doi:10.1145/365024.365114.

[SR95] SHAPIRA M., RAPPOPORT A.: Shape blending using the star-skeleton representation. *IEEE Computer Graphics and Applications 15*, 2 (1995), 44–50. doi:10.1109/38.365005.

[SRAIS10] SHECHTMAN E., RAV-ACHA A., IRANI M., SEITZ S.: Regenerative morphing. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 615–622. doi:10.1109/CVPR.2010.5540159.

[SRS91] SACHS E., ROBERTS A., STOOPS D.: 3-draw: A tool for designing 3d shapes. *IEEE Computer Graphics and Applications 11*, 06 (nov 1991), 18–26. doi:10.1109/38.103389.

[SSGS11] SCHMID J., SENN M. S., GROSS M., SUMNER R. W.: Overcoat: An implicit canvas for 3d painting. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, Association for Computing Machinery. doi:10.1145/1964921.1964923.

[SSPOJ16] SAALFELD P., STOJNIC A., PREIM B., OELTZE-JAFRA S.: Semi-Immersive 3D Sketching of Vascular Structures for Medical Education. In *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine* (Goslar, DEU, 2016), VCBM '16, Eurographics Association, p. 123–132. doi:10.5555/3061507.3061528.

[SWSJ07] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: ShapeShop: Sketch-Based Solid Modeling with BlobTrees. In *ACM SIGGRAPH 2007 Courses* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 43–es. doi:10.1145/1281500.1281554.

[SXY*11] SHAO T., XU W., YIN K., WANG J., ZHOU K., GUO B.: Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching. *Computer Graphics Forum 30*, 7 (2011), 2011–2020. doi:10.1111/j.1467-8659.2011.02050.x.

[SZF*21] SHEN Y., ZHANG C., FU H., ZHOU K., ZHENG Y.: DeepSketchHair: Deep Sketch-Based 3D Hair Modeling. *IEEE Transactions on Visualization and Computer Graphics 27*, 7 (2021), 3250–3263. doi:10.1109/TVCG.2020.2968433.

[TBvdP04] THORNE M., BURKE D., VAN DE PANNE M.: Motion Doodles: An Interface for Sketching Character Motion. *ACM Trans. Graph. 23*, 3 (aug 2004), 424–431. doi:10.1145/1015706.1015740.

[TLH19] TU P., LISCHINSKI D., HUANG H.: Point Pattern Synthesis via Irregular Convolution. *Computer Graphics Forum 38*, 5 (2019), 109–122. doi:10.1111/cgf.13793.

[TO99] TURK G., O'BRIEN J. F.: Shape Transformation Using Variational Implicit Functions. SIGGRAPH '99, Association for Computing Machinery, p. 13–es. doi:10.1145/1198555.1198639.

[Tri00] TRIMBLE: Google SketchUp, 2000. URL: https://www.sketchup.com/fr.

[TWY*20] TU P., WEI L.-Y., YATANI K., IGARASHI T., ZWICKER M.: Continuous Curve Textures. *ACM Trans. Graph. 39*, 6 (nov 2020). doi:10.1145/3414685.3417780.

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. *Computer Graphics Forum 23*, 1 (2004), 71–83. doi:10.1111/j.1467-8659.2004.00006.x.

[vdZLBI11] VAN DER ZWAN M., LUEKS W., BEKKER H., ISENBERG T.: Illustrative Molecular Visualization with Continuous Abstraction. *Computer Graphics Forum 30*, 3 (2011), 683–690. doi:10.1111/j.1467-8659.2011.01917.x.

[VGH*05] VIOLA I., GRÖLLER M. E., HADWIGER M., BÜHLER K., PREIM B., EBERT D.: Illustrative Visualization. In *Eurographics 2005 - Tutorials* (2005), Lin M., Loscos C., (Eds.), The Eurographics Association. doi:10.2312/egt.20051052.

[WBC07] WITHER J., BERTAILS F., CANI M.-P.: Realistic Hair from a Sketch. In *SMI '07 - IEEE International Conference on Shape Modeling International* (Lyon, France, June 2007), IEEE, pp. 33–42. doi:10.1109/SMI.2007.31.

[WBCG09] WITHER J., BOUDON F., CANI M.-P., GODIN C.: Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum 28*, 2 (Apr. 2009), 541–550. Special issue: Eurographics 2009. doi:10.1111/j.1467-8659.2009.01394.x.

[WC11] WEI X., CHAI J.: Intuitive Interactive Human-Character Posing with Millions of Example Poses. *IEEE Computer Graphics and Applications 31*, 4 (2011), 78–88. doi:10.1109/MCG.2009.132.

[WGG99] WYVILL B., GUY A., GALIN E.: Extending the CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum 18*, 2 (1999), 149–158. doi:10.1111/1467-8659.00365.

[WI04] WATANABE N., IGARASHI T.: A Sketching Interface for Terrain Modeling. In *ACM SIGGRAPH 2004 Posters* (New York, NY, USA, 2004), SIGGRAPH '04, Association for Computing Machinery, p. 73. doi:10.1145/1186415.1186500.

[WKL15] WANG F., KANG L., LI Y.: Sketch-based 3D shape retrieval using Convolutional Neural Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, jun 2015), IEEE Computer Society, pp. 1875–1883. doi:10.1109/CVPR.2015.7298797.

[WL00] WEI L.-Y., LEVOY M.: Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 479–488. doi:10.1145/344779.345009.

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the Art in Example-based Texture Synthesis. In *Eurographics 2009 - State of the Art Reports* (2009), Pauly M., Greiner G., (Eds.), The Eurographics Association. doi:10.2312/egst.20091063.

[WNS*10] WHITED B., NORIS G., SIMMONS M., SUMNER R. W., GROSS M., ROSSIGNAC J.: BetweenIT: An Interactive Tool for Tight Inbetweening. *Computer Graphics Forum 29*, 2 (2010), 605–614. doi:10.1111/j.1467-8659.2009.01630.x.

[Wol98] WOLBERG G.: Image morphing: a survey. *The visual computer 14*, 8 (1998), 360–372.

[WSI07] WEXLER Y., SHECHTMAN E., IRANI M.: Space-Time Completion of Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 3 (2007), 463–476. doi:10.1109/TPAMI.2007.60.

[XCF*13] XU K., CHEN K., FU H., SUN W.-L., HU S.-M.: Sketch2Scene: Sketch-Based Co-Retrieval and Co-Placement of 3D Models. *ACM Trans. Graph. 32*, 4 (jul 2013). doi:10.1145/2461912.2461968.

[XCS*14] XU B., CHANG W., SHEFFER A., BOUSSEAU A., MCCRAE J., SINGH K.: True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization. *ACM Trans. Graph. 33*, 4 (jul 2014). doi:10.1145/2601097.2601128.

[XCW14] XING J., CHEN H.-T., WEI L.-Y.: Autocomplete Painting Repetitions. *ACM Trans. Graph. 33*, 6 (Nov. 2014). doi:10.1145/2661229.2661247.

[XFZ*19] XU P., FU H., ZHENG Y., SINGH K., HUANG H., TAI C.-L.: Model-Guided 3D Sketching. *IEEE Transactions on Visualization and Computer Graphics 25*, 10 (Oct 2019), 2927–2939. doi:10.1109/TVCG.2018.2860016.

[XKG*16] XING J., KAZI R. H., GROSSMAN T., WEI L.-Y., STAM J., FITZMAURICE G.: Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (New York, NY, USA, 2016), UIST '16, Association for Computing Machinery, p. 755–766. doi:10.1145/2984511.2984585.

[XNC*19] XING J., NAGANO K., CHEN W., XU H., WEI L.-Y., ZHAO Y., LU J., KIM B., LI H.: HairBrush for Immersive Data-Driven Hair Modeling. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2019), UIST '19, Association for Computing Machinery, p. 263–279. doi:10.1145/3332165.3347876.

[XSS08] XIN M., SHARLIN E., SOUSA M. C.: Napkin Sketch: Handheld Mixed Reality 3D Sketching. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2008), VRST '08, Association for Computing Machinery, p. 223–226. doi:10.1145/1450579.1450627.

[YAS*21] YU E., ARORA R., STANKO T., BÆRENTZEN J. A., SINGH K., BOUSSEAU A.: CASSIE: Curve and Surface Sketching in Immersive Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2021), CHI '21, Association for Computing Machinery. doi:10.1145/3411764.3445158.

[YDSG21] YU X., DIVERDI S., SHARMA A., GINGOLD Y.: *ScaffoldSketch: Accurate Industrial Design Drawing in VR*. Association for Computing Machinery, New York, NY, USA, 2021, p. 372–384. doi:10.1145/3472749.3474756.

[YGW*15] YAN D.-M., GUO J.-W., WANG B., ZHANG X.-P., WONKA P.: A survey of blue-noise sampling and its applications. *Journal of Computer Science and Technology 30*, 3 (2015), 439–452. doi:10.1007/s11390-015-1535-0.

[YH21] YUAN Q., HUAI Y.: Immersive sketch-based tree modeling in virtual reality. *Computers  Graphics 94* (2021), 132–143. doi:10.1016/j.cag.2020.12.001.

[YLL16] YE Y., LI B., LU Y.: 3D sketch-based 3D model retrieval with convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), pp. 2936–2941. doi:10.1109/ICPR.2016.7900083.

[YSSK10] YOON S. M., SCHERER M., SCHRECK T., KUIJPER A.: Sketch-Based 3D Model Retrieval Using Diffusion Tensor Fields of Suggestive Contours. In *Proceedings of the 18th ACM International Conference on Multimedia* (New York, NY, USA, 2010), MM '10, Association for Computing Machinery, p. 193–200. doi:10.1145/1873951.1873961.

[ZBQC13] ZANNI C., BERNHARDT A., QUIBLIER M., CANI M.-P.: SCALe-invariant Integral Surfaces. *Computer Graphics Forum 32*, 8 (2013), 219–232. doi:10.1111/cgf.12199.

[ZGC15] ZANNI C., GLEICHER M., CANI M.-P.: N-ary implicit blends with topology control. *Computers & Graphics 46* (2015), 1–13. Shape Modeling International 2014. doi:10.1016/j.cag.2014.09.012.

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: SKETCH: An Interface for Sketching 3D Scenes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 163–170. doi:10.1145/237170.237238.

[ZHWW12] ZHOU Y., HUANG H., WEI L.-Y., WANG R.: Point Sampling with General Noise Spectrum. *ACM Trans. Graph. 31*, 4 (July 2012). doi:10.1145/2185520.2185572.

[ZIH*11] ZHU B., IWATA M., HARAGUCHI R., ASHIHARA T., UMETANI N., IGARASHI T., NAKAZAWA K.: Sketch-Based Dynamic Illustration of Fluid Systems. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, Association for Computing Machinery. doi:10.1145/2024156.2024168.

[ZJL14] ZHOU S., JIANG C., LEFEBVRE S.: Topology-Constrained Synthesis of Vector Patterns. *ACM Trans. Graph. 33*, 6 (nov 2014). doi:10.1145/2661229.2661238.

[ZLC*21] ZHANG F., LIU Z., CHENG Z., DEUSSEN O., CHEN B., WANG Y.: Mid-Air Finger Sketching for Tree Modeling. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)* (2021), pp. 826–834. doi:10.1109/VR50410.2021.00110.

[ZLDM16] ZHENG Y., LIU H., DORSEY J., MITRA N. J.: Smartcanvas: Context-inferred interpretation of sketches for preparatory design studies. *Computer Graphics Forum 35*, 2 (2016), 37–48. doi:10.1111/cgf.12809.

[ZLWH16] ZHU H., LIU X., WONG T.-T., HENG P.-A.: Globally Optimal Toon Tracking. *ACM Trans. Graph. 35*, 4 (jul 2016). doi:10.1145/2897824.2925872.

[ZNA07] ZIMMERMANN J., NEALEN A., ALEXA M.: SilSketch: Automated Sketch-Based Editing of Surface Meshes. In *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling* (New York, NY, USA, 2007), SBIM '07, Association for Computing Machinery, p. 23–30. doi:10.1145/1384429.1384438.

[ZPBK17] ZHU Y., POPOVIĆ J., BRIDSON R., KAUFMAN D. M.: Planar Interpolation with Extreme Deformation, Topology Change and Dynamics. *ACM Trans. Graph. 36*, 6 (nov 2017). doi:10.1145/3130800.3130820.

[ZvdP18] ZHANG X., VAN DE PANNE M.: Data-driven autocompletion for keyframe animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (2018), pp. 1–11.

[ZZB*18] ZHOU Y., ZHU Z., BAI X., LISCHINSKI D., COHEN-OR D., HUANG H.: Non-Stationary Texture Synthesis by Adversarial Expansion. *ACM Trans. Graph. 37*, 4 (July 2018). doi:10.1145/3197517.3201285.

ECOLE
DOCTORALE

**Titre :** Croquis dynamiques : Modélisation hiérarchique de scènes complexes et évolutives

**Mots clés :** Systèmes d'aide à la création, Informatique graphique, Modélisation par esquisse, Animation par ordinateur, Modélisation géométrique

**Résumé :** Les représentations visuelles sont essentielles pour explorer et communiquer une idée ou un phénomène. Alors que les logiciels numériques pour la modélisation 3D et l'animation restent complexes et spécialisés, ils ne favorisent généralement pas la créativité. En particulier, ils ne permettent pas l'ébauche rapide d'options alternatives. Ainsi, le croquis sur un support physique reste jusqu'à maintenant, la manière la plus simple et générale pour créer de telles représentations. Récemment, les techniques de modélisation par esquisse ont été intensément étudiées pour la création de modèles 3D mais seulement peu d'entre elles se servent du croquis comme entrée pour créer et immerger les utilisateurs dans un environnement 3D, guider le mouvement ou encore explorer des hypothèses. Cette thèse se concentre sur la modélisation temps-réel de scènes complexes et évoluant dans le temps à partir d'entrées croquis. Plus précisément, la vision à long terme serait de fournir aux utilisateurs une sorte de crayon 'augmenté' leur permettant de créer interactivement une scène 3D composée de formes qui peuvent être mises en mouvement ou déformées tout en permettant le raffinement à la fois sur la création et le mouvement, sans se voir imposer d'ordre spécifique dans ce processus de création. Grâce à une collaboration avec des architectes, nous avons tout d'abord mis en place *Nested Explorative Maps*, un nouveau type de croquis 3D dédié à la création rapide et l'exploration d'idées pour le design préliminaire de formes architecturales. Notre modèle permet d'esquisser du grossier aux détails des structures imbriquées afin de donner forme à un bâtiment en 3D, du plan de sol aux détails d'intérieurs et de façade, tout en gardant les traits de l'utilisateur et permettant une navigation interactive à travers les designs alternatifs suggérés visuellement par le croquis.
Nous avons ensuite abordé la synthèse de distributions anisotropes à partir d'une esquisse comme un outil général de création de contenu à la fois en 2D et en 3D. À partir d'une analyse multi-résolution sur les distributions de formes présentes dans un croquis, nous proposons une méthode efficace pour la synthèse de ces distributions dans un domaine 2D étendu. Une intégration 3D de cette nouvelle distribution a également développée, complétée par une illusion de profondeur afin de permettre aux utilisateurs une immersion immédiate dans un environnement 3D qui s'inspire de leur croquis.
Enfin, nous avons collaboré avec des biologistes afin d'explorer les croquis 3D animés, dans lesquels les mouvements et déformations de formes organiques peuvent être exprimés et raffinés à travers l'utilisation d'un vocabulaire schématique inspiré des représentations standards de leur domaine et d'encarts d'image clés.

**Title :** Dynamic sketches : Hierarchical modeling of complex and time-evolving scenes

**Keywords :** Creative artificial intelligence, Computer graphics, Sketch-based modeling, Computer Animation, Geometric modeling

**Abstract :** Visual representations are essential to explore and communicate an idea or a phenomenon. As digital software for 3D modeling and animation are still complex and specialized, they usually do not favor creativity. In particular, they offer no easy way to quickly draft a series of alternative options. Thus, up to now, sketching on a physical medium remains the only simple and general means to create such representations. Recently, sketch-based modeling techniques were intensively studied to create 3D models but only few techniques use sketching as input to create and immerse the users into a 3D environment, guide the motion of shapes or explore hypotheses.
In this thesis, we focused on the real-time modeling of complex and time-evolving scenes using only sketching as input. More precisely, the long-term vision would be to provide users with an augmented pen enabling them to interactively create a 3D scene composed of shapes that can be put into motion or deformed while enabling refinement both on the creation and motion without any editing pipeline.
Through a collaboration with architects, we first introduce *Nested Explorative Maps*, a new type of 3D sketch for the easy creation and exploration of ideas applied to the preliminary design of man-made shapes. Our model enables coarse-to-fine sketching of nested structures to progressively shape a 3D building from the floor plan to interior design while keeping the original strokes and allowing interactive navigation through the alternative design that the sketch visually suggests.
We then tackle the synthesis of anisotropic distributions from a sketch as a means for the general creation of content both in 2D and 3D. From a simple multi-resolution analysis of the shape distributions, we propose an efficient method to synthesize the input distribution into an extended 2D domain but also a 3D embedding of this extended distribution in addition to an illusion of depth to enable users to immediately explore a 3D environment inspired from their sketch. Finally, we collaborated with biologists to explore animated 3D sketches, where motions and deformations of organic shapes can be expressed and refined through the use of a simple depiction vocabulary inspired from standard representations in their field, and key-frame snippets.

**Institut Polytechnique de Paris**
91120 Palaiseau, France