Interactive Animation and Modeling by Drawing – Pedagogical Applications in Medicine

David Bourguignon

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE

pour obtenir le titre de

DOCTEUR DE L'INPG

spécialité Modèles et instruments en médecine et biologie

préparée au sein du projet iMAGIS, laboratoire GRAVIR, INRIA Rhône-Alpes
dans le cadre de l'école doctorale Ingénierie pour le vivant

présentée et soutenue publiquement

par

David Bourguignon

le 8 janvier 2003

Titre

Interactive Animation and Modeling by Drawing

Pedagogical Applications in Medicine

Directrice de thèse

Marie-Paule Cani

JURY

James Crowley,	Président
Ronen Barzel,	Rapporteur
Christophe Chaillou,	Rapporteur
Jean-Daniel Fekete,	Rapporteur
Marie-Paule Cani,	Directrice de thèse
George Drettakis,	Examinateur

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE

pour obtenir le titre de

DOCTEUR DE L'INPG

spécialité Modèles et instruments en médecine et biologie

préparée au sein du projet iMAGIS, laboratoire GRAVIR, INRIA Rhône-Alpes
dans le cadre de l'école doctorale Ingénierie pour le vivant

présentée et soutenue publiquement

par

David Bourguignon

le 8 janvier 2003

Titre

Interactive Animation and Modeling by Drawing

Pedagogical Applications in Medicine

Directrice de thèse

Marie-Paule Cani

JURY

James Crowley,	Président
Ronen Barzel,	Rapporteur
Christophe Chaillou,	Rapporteur
Jean-Daniel Fekete,	Rapporteur
Marie-Paule Cani,	Directrice de thèse
George Drettakis,	Examinateur

To see this opposition between Hugh's science and ours more clearly, perhaps we should stick to Hugh's term, and with Dindimus, speak about it as *philosophia* – as "the caring pursuit of truth, motivated not by that love which cherishes the well-known, but driven by the desire to pursue further what has been tasted and has been found pleasing", as Dindimus says.

Ivan Illich[1]

[1] From Illich (1981). The excerpt is from Hugh of Saint Victor, *Epitome Dindimi in philosophiam*, circa 1129 (of Saint Victor, 1966).

Abridged Contents

Preface

From Molecular Biology to Computer Graphics

Before stepping into the subject of this thesis, I would like to talk a little bit about its ontogenesis, or in other terms, the circumstances of its development. Above all, I would like to explain what motivated the quite bold curricula move of mine from molecular biology to computer graphics.

When I graduated in 1997, I realized that my interest in biology, initially motivated by the pleasure found in the contemplation of living things, had been seriously challenged by years of manipulating test tubes filled with transparent liquids. I had to find something else to do. By an incredible stroke of luck, at the same moment I was looking for a research laboratory interested in computer modeling of biological forms, Marie-Paule Cani proposed a research subject on human heart motion simulation. This was the ideal occasion to bridge the gap between the two distant fields of biology and computer science. But this also entailed bridging my own knowledge gap: a few classes on Ada language for biological simulation purposes were obviously not sufficient, and my programming inexperience was only the emerging tip of the immense iceberg of my ignorance in computer matters (so to speak).

Sometimes, it helps to know that you don't know, because you avoid thinking that you have the solution even before looking at the problem. Nevertheless, the pessimistic diagnosis of Joe Marks (Marks et al., 1994) on the capacity of a single individual to embrace computer graphics as a whole was not very encouraging. His words were resonating in my (empty) head as a menacing thunder:

> Just a few years ago, all one needed to be a competent researcher or practitioner in computer graphics was a solid background in geometry, algebra, calculus, topology, probability, mechanics, electromagnetism, signal processing, image processing, electrical engineering, mechanical engineering, optics, information theory, structured programming, basic algorithms and data structures, complexity theory, computer architecture, human factors, perceptual psychology, colorimetry, graphic design, industrial design, semiotic, and art! Unfortunately, the list is growing [...].

Ignoring these warnings, and some might say, foolishly, I decided to embark on a three-year voyage on the wide open sea of image synthesis. Even if winds were not

always with me, I finally reached a few islands, and found there strange fruits I hope you will find tasteful.

Acknowledgements

Without the help of many people, for fixing a torn sail or a broken helm, or simply to share some thoughts about the best route between fringing reefs, I wouldn't have been able to travel as far as I did.

First, I would like to thank Claude Puech for accepting me in his lab when it was obvious that my computer science background was scarce, and Marie-Paule Cani for being my advisor during this thesis, letting me freely explore research areas of my own choice, without loosing her patience when results were long to come.

I would also like to thank the people of the iMAGIS project for their joyfulness, their helpfulness, and their ability to cope with my idiosyncratic tendencies to talk too much at lunch time (and therefore eat slowly). Special thanks to Fabrice Neyret for many OpenGL debugging tips and supercool demos, to François Faure for a few months of common work on the AnimAL library, to Gilles Debunne for his guidance in the validation of my deformable model using his animation software.

During the three years of my doctoral studies, many researchers generously shared their expertise with me. I thank them all, by chronological order: Jacques Ohayon (TIMC lab), for his collaboration in biomechanics; Isabelle Magnin and Denis Friboulet (CREATIS lab), for their collaboration in the Beating Heart research project; George Drettakis (REVES project), for his continuous support and collaboration; Pierre Bessière and Emmanuel Mazer (SHARP project), for discussions on Bayesian inference; Dominique Attali (LIS lab), for discussions on computational geometry; Mariette Yvinec (PRISME project) and Lutz Kettner (MPI – Saarbrücken), for extensive coding of the CGAL library; Fred Cazals and Andreas Fabri (PRISME project), for discussions on computational geometry; Jean-Paul Chirossel and Olivier Palombi (Anatomy lab, Grenoble CHU), for their collaboration (and the recording of an entire anatomy course on the blackboard).

Finally, I would like to thank members of my jury, James Crowley (PRIMA project), George Drettakis (REVES project), and especially the reviewers Ronen Barzel (Pixar Animation Studios), Christophe Chaillou (LIFL lab) and Jean-Daniel Fekete (IN SITU project), for accepting this supplementary workload, in a very busy time of the year!

Chapter 1

Introduction

1.1 User-centered Computer Graphics

Computer graphics is a burgeoning field that goes far beyond its sensu stricto definition as "the branch of science and technology concerned with methods and techniques for converting data to or from visual presentation using computers" (Hapeman et al., 2001). By its interdisciplinary nature, it tends to absorb nearly all the computer-related fields under its broad paradigm of visual data processing. This enormous diversity of subjects of interest is the common fortune and also the common fate of computer graphics researchers: the creative bursts in every imaginable direction also have their drawbacks when it comes to defining pertinent criteria to evaluate research or simply finding structuring principles to teach what the field is about.

This thesis presents contributions ranging from animation to modeling, and in that respect it is faithful to the eclectic tradition of computer graphics. However, we would like to take a step back and think about one of its underlying motivations as a key to understanding the choices that have been made. In fact, throughout our work, we have been mostly interested in user-centered computer graphics, that is an approach of the process of image synthesis where the point of view of the user prevails over all the others. We will make the case for this position by starting with a few reminders on computer graphics history.

1.1.1 A Brief History of Computer Graphics

Computer graphics is a young field. It takes its roots in computer-aided design (CAD), which emerged in the 1960's due to the increasing need for three-dimensional prototyping in the automotive and aerospace industries. Thus, computer graphics was built upon the discoveries of engineers during a decade, for example in terms of surface representations and object visibility calculations, but the goal was much more ambitious than simply providing three-dimensional blueprints of mechanical parts: it was about a new medium for creating mirror images of Nature.

Rapidly growing in possibilities and scope during the 1970's and 1980's, computer graphics explored more and more complex problems in modeling, rendering and animation. Therefore, computers invested domains where they were previously absent: either traditional image industries, such as publishing, television and movies, or fields demanding visualization such as medicine and engineering. Each time, this caused profound changes in the ways the work was done because this was not a mere improvement over previous practice but a complete redesign of the pipeline. So much that people could clearly distinguish before and after computers entered their professional lives.

Now, achievements speak for themselves: from highly detailed images rendered at interactive rates on commodity graphics hardware, leading a booming computer game industry, to special effects allowing movie directors to nearly forget limitations of traditional film making, thanks to virtual sets and digital crowds, computer graphics is as pervasive as computers in many aspects of our lives. Moreover, the historical goal of physical realism has been met in some cases (Alias|wavefront, 2002a).

Nevertheless, the fully computer-generated movie *Final Fantasy: The Spirits Within* (Sakaguchi and Sakakibara, 2001), whose tag line was "fantasy becomes reality", and can be considered as state-of-the-art in physically realistic computer graphics imagery, was still carefully crafted by hand for more than 100 million USD. Actually, the ever increasing complexity of scenes and the raw economic fact that silicon chips are much cheaper than human brains call for innovative solutions to handle the workload.

This could be an explanation for the recent rise of the "capture" paradigm, or the measurement of an augmenting number of real world properties in order to feed an image synthesis process. Capture technologies have already been used for recording human motion, for recovering objects surface geometry or light interaction behavior, but we expect that many other possibilities are waiting at the corner. Moreover, the fact that most of these measurements are image-based, may foretell the advent of a "super camera" device, where everything could take place at the same time, from the same viewpoint. In the long term, this approach could complement or replace proceduralism as a solution for tackling the problem of creating complex scenes.

Thus, one possible answer to the expanding workload is to diminish the amount of human intervention by using even more automatic systems; but another possible answer, which we are interested in, is to present to users better ways for performing their tasks, thereby making their work easier, faster, and more enjoyable. Of course, these answers are not mutually exclusive, and in what follows, we will explore this alternative as one among all the other computer graphics classifications.

1.1.2 A Simple Computer Graphics Taxonomy

Building a taxonomy of a problem is an interesting way to focus on the big picture without getting lost in details, but also a powerful tool to discover fresh ideas (Ivan E. Sutherland, cited by Sun microsystems, 2002). In order to do this, one must define a few axes, each representing a separate dimension of the problem. The combinations

of the axes values provide the different classes, or taxons, of the taxonomy. A careful inspection of these may lead to unexpected insight.

If we are considering the computer graphics input problem, we can define an axis that focuses on the amount of human intervention, and goes from fully automated systems, which do not need user input, to fully controllable systems which require user input from the beginning to the end of the task. To sum up, this axis could also be defined by the following opposing pairs:

with user	$\longleftrightarrow$	without user
controllable	$\longleftrightarrow$	automatic
human-centered	$\longleftrightarrow$	machine-centered

Another interesting axis that can be defined around the input problem concerns how much knowledge of the world[1] is encoded into the system: does the system generate new images because it has been given examples of expected results, or because it has the knowledge of the laws of optics? As before, one can clarify this opposition using a few word pairs:

with knowledge	$\longleftrightarrow$	without knowledge
procedural	$\longleftrightarrow$	statistical
rule-based	$\longleftrightarrow$	data-based

From this simple two-axis taxonomy, it seems obvious that a system requiring user input can be obtained by a wide diversity of computer programs because, theoretically, axes of the taxonomy are orthogonal descriptions of the same problem, and thus all combinations are possible. But it is not what is observed in practice. In fact, some programs lend themselves easily to user interaction while others do not. If our primary concern is the way the user interacts with the system, that is the system interface, we will always face limitations, since points of interaction we can offer are constrained by those offered by the program itself and its underlying model. Therefore, in designing a system, we cannot consider the model and the interface separately, because somehow the model is the interface.

This claim has a direct consequence on the problem solving process in computer graphics. It involves devising the solution to a problem not only in terms of the strict requirements of the end result (e.g., images of trees) but also in terms of the kind of interaction that will be necessary for a user to actually accomplish the task (e.g., model and render trees). This seems trivial but it is not: because the choices made during the model definition phase have already reduced the set of possible interactions, it is too late to change anything if in the end the interface proves wrong. In that respect, computer graphics is a tool making activity, and the tools produced must obey general usability principles, but also specific ones. We will discuss this point of view in the next section.

[1] We mean by knowledge the explicit, formalized knowledge, obtained by the scientific method.

1.2 Computer Graphics as Tool Making

Tool making has been a distinctive activity of the members of the genus *Homo* for at least 2 million years.[2] Amidst today's profusion of tools invented by our industrious species, recently developed information processing systems occupy a position apart, because their "abstract nature [...] poses a particular challenge for the designer" (Norman, 1988, p. 177). Compared to prehistoric tools still in use today (the knife, the needle, etc.), the current usability of computer systems makes some people suggest humorously that we are facing a regression in that matter (see Fig. 1.1).

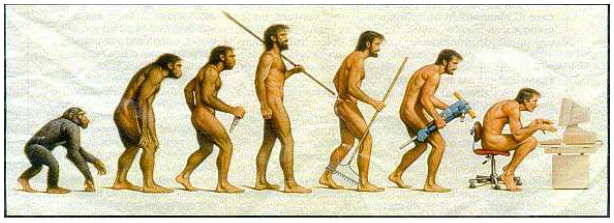


Figure 1.1: A cartoonist's view of evolution. Legend: "Somewhere, something went terribly wrong." (We do not know the author of this satirical drawing).

Computer graphics tools have evolved tremendously over the past thirty years. In the early days, tools were limited to a programming language and its compiler, and eventually a graphics application programming interface (API). The user was the programmer. Then, software with remarkably well-thought graphical user interface (GUI), such as Photoshop (Adobe Systems, 2002), appeared in the 1980's, introducing progressively non-programmer users, such as traditional artists, to computer graphics. Now, tools usability have nearly reached a standstill, even if users tasks have never been so complex and various. However, we believe there is still much room for improvement. In fact, computer graphics system design, as a tool making process, has a lot to learn from ergonomics.

1.2.1 Classic Ergonomics

Ergonomics, from the Greek $\epsilon\rho\gamma o\nu$ (work) and $\nu o\mu o\varsigma$ (laws), is "the scientific discipline concerned with the understanding of interactions among humans and other elements of a system [...]. Ergonomists contribute to the design and evaluation of tasks, jobs, products, environments and systems in order to make them compatible with the needs, abilities and limitations of people. [...] That is, ergonomics promotes a holistic approach in which considerations of physical, cognitive, social, organizational, envi-

[2] Famous stone tools discovered by Mary Leakey at Olduvai Gorge, in Tanzania, are dated at 1.7 My BP, but recent findings could be as old as 2.4 My BP.

ronmental and other relevant factors are taken into account" (International Ergonomics Association, 2000).

The science of ergonomics provides the methods we need to design usable tools. In the best possible world, this knowledge would be used on a large scale, resulting in the best possible tools, up to what we know about user needs. Unfortunately, an objective look at the situation in the computer industry reveals impressive failures. Of course, there are always plenty of possible explanations for them, related to technical, industrial, commercial, historical, etc., constraints. However, there is another possible explanation that sums up the others: the user is too often the last factor to be taken into account (see Fig. 1.2).

Figure 1.2: The need for user-centered tools. *Off the leash* by W. B. Park. Legend: "Darn these hooves! I hit the wrong switch again! Who designs these instrument panels, raccoons?" Cited by Norman (1988, p. 187).

Progress has occurred in computer systems since, for example, the first word processors. They are no longer the primitive tools described by Norman (1988). Thanks to cognitive ergonomics studies, they now have self-modifying menus, contextual help, automatic spell checking, etc. But these beautiful pieces of software are using age-old input devices that are one of the causes of an epidemy of repetitive strain injuries (RSI), due to the neglect of basic ergonomic principles. We consider this as a prototypical example of our previous assertions.

"The RSI, also known as cumulative trauma disorder or overuse syndrome, consists of many different injuries: carpal tunnel syndrome, tendinitis, tynosynovitis, thoracic outlet syndrome, lateral epicondilitis, etc." (Luskin, 1993). They are mainly caused by the excessive usage of the computer mouse and keyboard, two examples of ergonomic

disasters. “We couldn’t find a single mouse or trackball we felt was safe to use for extended periods of time” (Pascarelli, 1994). However, these injuries have other possible causes that go beyond repetitive motions.[3] Physical factors (screen position, etc.) and organizational factors (intensified workload, etc.) are also responsible (Koehoorn et al., 2001). The percentage of RSI among all occupational illnesses has regularly grown since the release of the IBM PC, starting from less than 20% in 1981, passing 50% in 1992, and reaching 70% by the year 2000, according to estimates (Putz-Anderson et al., 1997). Pascarelli evaluates that RSI cost 20 billion USD a year: this is one of the hidden costs of present computer systems.

Thus, it seems that, in many cases, computer system design is trapped in some local optimum that satisfies many constraints, except the usability of the system. This gloomy perspective applies also to computer graphics system design, the problem of usable input devices being even more acute since tasks frequently involve manipulating complex three-dimensional data.[4] Changing this state of facts will imply much more than a few good design ideas.[5] Nevertheless, considering that the goal is still out of reach does not forbid thinking ahead of it. We have emphasized the importance of classic ergonomics factors in general system design, but we consider that they are not sufficient to describe all the requirements for creation tools, such as computer graphics systems. Therefore, we will risk the definition of a new domain of this discipline that could be of great interest in creation tool design.

1.2.2 Creative Ergonomics

In *The Ultimate Design Tool*, Blinn (1990) defines creation as a two-phase process that starts with ideation and finishes with implementation. He makes the point that paper and pencil is the perfect tool for the ideation phase. “Computers help a lot with implementation, but idea generation is still done with pencil and paper. Can computers help here too? I don’t think so. Why try to cram something on a computer when there is a cheap and effective alternative?” We entirely agree with this. Given the current software, hardware, input, and output technology, there is no such thing as paper and

[3] That is why ergonomists prefer to call RSI work-related musculoskeletal disorders of the upper limb (WMSD).

[4] “While the essence of artists are reflected in their work, it is rooted in skill — skill which is hard earned, and therefore worthy of respect by the instrument builder, or ‘luthier’. But it is precisely these same skills which are so poorly captured by most computer-based tools. I maintain that the skills (and therefore needs) of the artist are different from those of, say, an accountant. Yet, based on the tools used, when I walk through Disney Feature Animation, for example, I can hardly tell if I am in the accounting or character animation department.” (Buxton, 1997).

[5] “It will take extra effort to design systems that complement human processing needs. It will not always be easy, but it can be done. If people insisted, it would be done. But people don’t insist: Somehow, we have learned to accept the machine-dominated world. If a system is to accommodate human needs, it has to be designed by people who are sensitive to and understand human needs. I would have hoped such a statement was an unnecessary truism. Alas, it is not.” (Norman, 1993, p. 227).

pencil available on computers: architects still sketch on paper for early design, but the definite building plans are obtained using CAD tools.[6] However, we do not agree with the fact that the ideation phase is the only one to take advantage of creation-friendly tools: after working out his sculpture on paper, a sculptor is not simply following his previous studies as rigid blueprints when he handles the chisel. Implementation is as creative as ideation.

As a result, it is necessary to evaluate tools used during the implementation phase of the creative process, not only for their classic ergonomics properties, but also for their creative potential, or in other terms their "creative ergonomics" properties. Before giving a precise definition of what we mean by creative ergonomics, we will illustrate this idea with an example in an unrelated domain: children's toys.

For our demonstration, we classify toys in two categories. On the one hand, there are toys that are multipurpose, such as a doll or a ball. They afford invention of many different sorts of games because they offer a good support to children's imagination. On the other hand, there are "monopurpose", single-game toys, such as a jigsaw puzzle. This toy obeys to a fixed set of rules and evolves from a defined initial state (all pieces apart) to a defined final state (jigsaw puzzle completed). The game is over when this state is attained. Of course, one could object that it is also possible to invent games with such toys, that it is only a matter of individual creativity. This is absolutely true. But the point is that some toys are on the average better at stimulating children's fantasy than others. Understanding this property is what creative ergonomics is about. In the spirit of the definitions of physical, cognitive, and organizational ergonomics given by the International Ergonomics Association (see Appendix A), here is a tentative definition of the domain:

Creative ergonomics is concerned with mental processes, such as inspiration, and association, as they affect production of original work through interactions among humans and other elements of a system. Relevant topics include input device expressiveness, output device stimulating quality, abstract/ambiguous/imprecise representation handling (Goel, 1995; Gross and Do, 1996), and non-verbal interaction.

To begin with, the creative ergonomics properties of tools can be analyzed from the point of view of objects affordances.[7] In fact, we distinguish between two kinds of affordance. *Cognitive affordance* is the kind of affordance considered by Norman (1988), it is about the tool's cognitive usage. Successful design makes the tool as strictly defined as possible, so that there is no possibility of misunderstanding the use intended by the designer. *Creative affordance* is a different kind of affordance, it is

[6] We heard recently of a software design tool named SketchUp (@Last Software, 2002) that could modify architects work habits, by filling the gap between sketching with paper and pencil, and modeling with CAD tools. This tool seems inspired by the work of Zeleznik et al. (1996).

[7] "There already exists the start of a psychology of materials and of things, the study of affordances of objects. When used in this sense, the term *affordance* refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used [...]. A chair affords ('is for') support and, therefore, affords sitting. A chair can also be carried." (Norman, 1988, p. 9)

about the tool's creative usage. Successful design makes the tool as loosely defined as possible, so that there is a wide range of possibilities of inventing uses intended or not by the designer. These two kinds of affordance have sometimes opposite goals and the resulting tool design is in fact a trade-off between them.

Of course, this framework explores only a small part of the complexity of creative ergonomics, and we are aware of the necessity of other approaches (Boden, 1992). In the future, creative ergonomics could lead to a new success metric for computer graphics systems, different from the technology-centered metric in use today. However, with current difficulties for evaluating it (Eysenck, 1994), we cannot expect anything but qualitative results for the moment.

After this discussion on the underlying motivations of this thesis, we will precise how the previous considerations influenced our contributions. For this, we will describe briefly the application contexts, the problems posed, and the solutions chosen.

1.3 Interactive Animation and Modeling by Drawing

The possibilities offered by information technologies for teaching are enormous, and they now define a new field of scholarly research, with a growing academic community (e.g., the Stanford Center for Innovations in Learning was established in 2002). Among the disciplines, biology has long been an obvious choice because of the complexity of the field of study: living beings have a three-dimensional, dynamic structure that poses difficult teaching challenges. Moreover, ethical problems, practical availability, etc., are also to be considered. To tackle these issues, various projects have proposed virtual dissection kits, frogs being by far the most popular (SUMMIT, 2002b; Hill, 2002).

Medicine is a discipline where visualization is an essential component of learning. Anatomy is a visual discipline in essence, but dissection of cadavers conveys only a portion of the necessary information to understand spatial relationships between organs: the heart does not beat, the diaphragm does not move, etc. The time dimension is missing. Moreover, when three-dimensional imaging data are at hand, there are no editing tools available that are as easy to use as white chalk and blackboard. Thus, teacher and student are maintained in a passive observer attitude with respect to these data. The existing teaching tools are either based on anatomical image databases (Walsh et al., 2002), or multimedia documents (SUMMIT, 2002a). There is a need for truly interactive teaching tools that will enable teacher and students to create and manipulate computer models, not just watch them. With that goal in mind, we have proposed different approaches, all having pedagogical applications in medical education.

First, we were interested in interactive physically-based animation of anisotropic elastic materials. The envisioned application scenario is a physiological anatomy course on the human cardiac muscle. Using our model, teacher and students can build interactive samples of cardiac muscular tissue in order to demonstrate organ function,

and experiment effects of various pathologies. Possible exercises include: qualitative influence of fibers orientation on the mechanical behavior of the cardiac pump, quantitative influence of ischaemic necrosis[8] in selected areas of the tissue on the mechanical efficiency of the muscle, etc.

To achieve this, our model exhibits two key features. The first one is low computational cost that results in high frame rates, a sine qua non for interactivity. Thanks to this, the user can play with the model, picking it with computer mouse-generated forces, and tweaking it to understand its rich behavior, with instant visual feedback. Besides, this also allows simulations to be rerun immediately after changing parameters. The second feature is an intuitive *system image*[9] that ensures easy control by the user. Most of the time, physically-based models lead to crude system images that are simple translations of the model parameters in terms of GUI. Depending on the target user, e.g., a medical student, requiring expert knowledge to control the system, e.g., continuum mechanics, might not be a good design idea.

Next, we were interested in interaction in three dimensions using two-dimensional input, either for annotating existing models, or for creating new models. The fact that drawing practice is still considered a fundamental learning method by some anatomy teachers in French medical school curriculum, presents a rare opportunity to take advantage of a previous know-how for reusing it in a computer-based tool. During a typical course session, the teacher draws on the blackboard three-dimensional anatomical structures with colored chalks, either as perspective projections, or as (sagittal, transverse, or frontal) sections. At the same time, the students, following the examples on the blackboard, draw the same structures in their sketchbooks. This process facilitates memorization of the shapes and spatial relationships, because a structure is better "understood" once it has been implicitly analyzed by drawing.

Thus, a simple application scenario is a human anatomy course where the teacher draws functional schemas using a computer tablet. The image on his computer screen is projected on the wall behind him for the students to see. Using our system for illustration and annotation in 3D, he can draw and visualize each schema under different viewpoints, drawing new information when the inferred schema becomes obviously wrong. If he is presenting an existing model, for example an isosurface of a femur bone extracted from imaging data, he can still annotate this three-dimensional model as he would have for a paper diagram. Finally, if he wants to complement the femur model, as if by sculpting it, for example to indicate effects of partial bone fracture, he

[8] Ischaemic necrosis is the death of cells as a result of diminished blood flow in a tissue or an organ.

[9] "Three different aspects of mental models must be distinguished: the *design model*, the *user's model*, and the *system image* [...]. The design model is the conceptualization that the designer has in mind. The user's model is what the user develops to explain the operation of the system. Ideally, the user's model and the design model are equivalent. However, the user and designer communicate only through the system itself: its physical appearance, its operation, the way it responds, and the manuals and instructions that accompany it. Thus the *system image* is critical: the designer must ensure that everything about the product is consistent with and exemplifies the operation of the proper conceptual model." (Norman, 1988, pp. 189–190)

can do it using our (nearly finished) modeling by drawing tool.

All this is possible because our 3D drawing system has a stroke representation that enables drawing redisplay when the viewpoint changes. Moreover, this representation can be mixed freely with existing polygonal surfaces for annotation purposes. Differently, our modeling by drawing tool strives to offer a sketching interface using both strokes geometry and drawing image information to allow three-dimensional modeling without explicit depth specification.

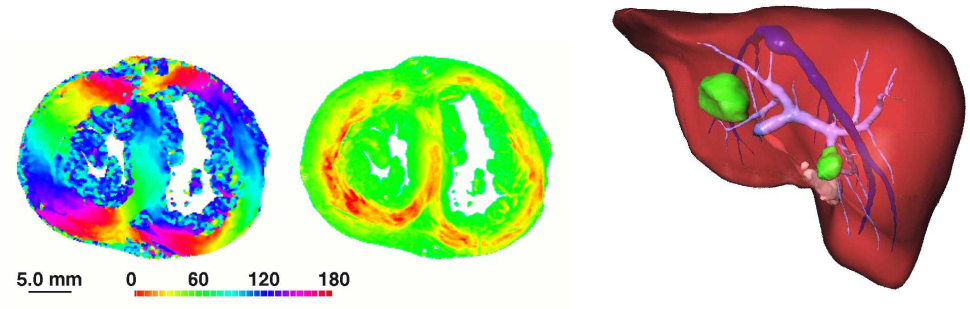
We will present our contributions in the next three chapters: our model of anisotropic elastic materials in chapter 2, our drawing system for illustration and annotation in 3D in chapter 3, and our modeling by drawing tool in chapter 4. We will conclude by presenting the current limitations and the future extensions of these approaches in chapter 5.

Chapter 2

Animating Anisotropic Elastic Materials

2.1 Introduction

Many natural objects are strongly anisotropic, due to their structure (crystalline, fibrous) or the composite materials they are made of (tree trunks, mammal organs). For example, human heart motion simulation is a challenging problem since the heart is a muscle of complex geometry, where anisotropy caused by the varying directions of muscle fibers (see Fig. 2.1a) plays an important role (Hunter, 1995). The human liver is also a good example of anisotropic material, although it has been previously animated using isotropic elastic models (Cotin et al., 1999; Debunne et al., 1999, 2001). In fact, it can be seen as a composite material: the root-like structures of rather rigid vessels are embedded in the liver tissue itself, which is a soft material (see Fig. 2.1b).



(a) (b)

Figure 2.1: Two examples of complex anisotropic materials. In (a), angular maps of muscle fiber direction obtained on a human heart (Usson et al., 1994); left, map of the azimuth angle; right, map of the elevation angle. In (b), a human liver with the main venous system superimposed (courtesy of Épidaure project, INRIA).

We would like to offer a tool for easy creation and interactive animation of anisotropic elastic materials. For example, we envision the use of digital biological samples that could be edited and manipulated interactively during an anatomy course on human liver structure or a physiology course on human heart muscle. After reviewing literature on deformable models, with special emphasis on interactive systems that could allow simple anisotropy specification, we have found that mass-spring systems presented an interesting possibility because of their trade-off between model complexity and expressiveness.

In fact, mass-spring systems have been extensively used in computer graphics over the last fifteen years, and are still very popular. Easier to implement than finite element methods, these systems allow animation of dynamic behaviors. They have been applied to the animation of inanimate bodies such as cloth or soft material (Luciani et al., 1991; Provot, 1995) and to the animation of organic active bodies such as muscles in character animation (Miller, 1988; Chadwick et al., 1989).

However, modeling specific biological material is impossible using mass-spring systems: as will be shown, neither isotropic materials nor anisotropic materials can be generated easily. Another problem is that most of the living tissues, which are essentially made of water, maintain quasi-constant volume during deformations (this is well known for muscles, but also holds for other tissues). Mass-spring systems do not have this property.

Overview

We present an alternative model to classical mass-spring systems that enables one to specify isotropic or anisotropic properties of an elastic material, independently of the 3D mesh used for sampling the object. The approach we use is still related to mass-spring systems in the sense that we animate point masses subject to applied forces. However, the forces acting on each point mass are derived from the anisotropic behavior specified for each of the volume elements that are adjacent to it.

Since there are no springs along the mesh edges, the geometry and topology of the mesh do not restrict the simulated behavior. Moreover, quasi-constant volume deformations can be obtained by adding extra forces acting as soft constraints. We illustrate this on both tetrahedral and hexahedral meshes. Our results show that computation time remains low, while more controllable behaviors are achieved. Finally, we validate our model by testing its ability to simulate nonlinear stress-strain relationships, and by comparing its behavior for different resolutions of the mesh.

Part of this work has been previously published in the *Proceedings of the 11th Eurographics Workshop on Animation and Simulation* (Bourguignon and Cani, 2000).

2.2 Previous Work

Despite extensive computer graphics literature on deformable models, commented in recent surveys (Gibson and Mirtich, 1997; Debunne, 2000), there are, to the best of our knowledge, only a few papers mentioning the possibility of modeling anisotropic materials. Continuum mechanics has been modeling composite materials for a long time and biomechanics is now facing the challenge of modeling living materials (Ohayon et al., 2001), but the complexity of the phenomena involved induces highly detailed mathematical models that are also very demanding in computation time. Interactive deformable models of anisotropic materials thus remain elusive.

We will review the two main kinds of physically based deformable models described in the literature, with an emphasis on interactive systems and volumetric approaches. We will then investigate the advantages and disadvantages they would present for modeling anisotropic nonlinear materials with quasi-constant volume deformations.

2.2.1 Continuous Models

Continuum mechanics and elasticity theory offer a general framework for modeling deformable materials. However, most of the continuous models in computer graphics make assumptions concerning the simulated material: it must be homogeneous, isotropic and follow a linear stress-strain relationship under the small deformations hypothesis (geometrical linearity) in order to simplify equations. Because it is necessary to understand these assumptions before discussing previous work, we present below a very brief survey of the theory of linear elasticity (Hollister, 2002). For a thorough introduction, we refer the reader to the classical work of Fung (1965).

Indicial Notation

In what follows we will use indicial notation. In a Cartesian orthonormed coordinate system $\mathcal{R}(O, \mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3})$, a first order tensor (a.k.a. vector) $\mathbf{a}$ has components a_i, i.e.,

$$\mathbf{a} = a_i\,\mathbf{e_i} = a_1\,\mathbf{e_1} + a_2\,\mathbf{e_2} + a_3\,\mathbf{e_3}$$

according to the rule of summation over the range of the repeated index (a.k.a. Einstein summation convention). Similarly, a second order tensor (a.k.a. tensor) $\mathbf{A}$ has components A_{ij}, i.e.,

$$\mathbf{A} = A_{ij}\,\mathbf{e_i} \otimes \mathbf{e_j}$$

and it is now possible to write equations in a basis-independent manner, e.g,

$$a_i\,\mathbf{e_i} = A_{ij}\,b_j\,\mathbf{e_i} \quad \text{is written} \quad a_i = A_{ij}\,b_j$$

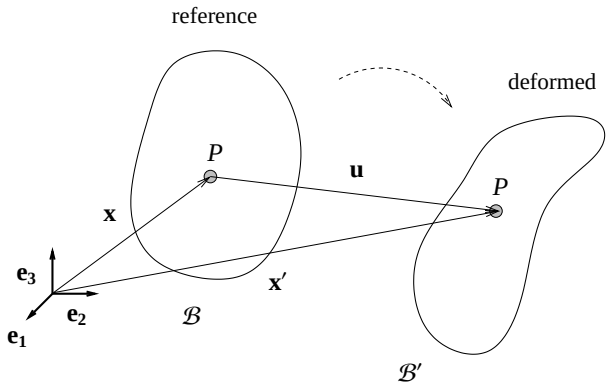


Figure 2.2: Elasticity definitions. A body initially in reference configuration $\mathcal{B}$ at time $t = 0$ undergoes deformation and is now in deformed configuration $\mathcal{B}'$ at time t. Position $\mathbf{x} = x_i\,\mathbf{e_i}$ of material point P in reference configuration is expressed in a Lagrangian coordinate system; position $\mathbf{x}' = x'_i\,\mathbf{e_i}$ of material point P in deformed configuration is expressed in an Eulerian coordinate system. Displacement of material point P at time t is $\mathbf{u} = \mathbf{x}' - \mathbf{x}$

Small Deformation Elasticity

The strain tensor describes, given a point of the material, the deformation w.r.t. the rest state, in every possible direction. It is represented by a 3×3 symmetric matrix with six strain components: three principal strains (diagonal terms of the matrix) and three shear strains. A classical strain metric is the Cauchy strain tensor ϵ_{ij} defined by

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \quad \text{with} \quad \left|\frac{\partial u_i}{\partial x_j}\right| \ll 1 \quad \forall i, \forall j$$

where $\frac{\partial u_i}{\partial x_j}$ is the derivative of the ith coordinate of the displacement u_i w.r.t. the jth coordinate of the reference configuration. The displacement u_i is defined for every point as the vector between reference and deformed positions.

The stress tensor describes, given a point of the material, the force acting on every infinitesimal surface element (stress is thus equivalent to a pressure). It is represented by a 3×3 symmetric matrix with six stress components: three normal stresses (diagonal terms of the matrix) and three shear stresses. A classical stress metric is the Cauchy stress tensor σ_{ij} defined by

$$t_i = \sigma_{ij}\, n_j$$

where t_i is the traction vector, i.e., areal density of force and n_j is the unit normal to the surface.

The simplest stress-strain relationship is Hooke's law, a linear constitutive law first stated in the one-dimensional case of springs. In the general case of three-dimensional bodies, it states that stress is proportional to strain, i.e.,

$$\sigma_{ij} = C_{ijkl}\,\epsilon_{kl}$$

where C_{ijkl} is the tensor of elasticity, a rank four symmetric tensor, with thirty-six different terms, since σ_{ij} and ϵ_{kl} are rank two symmetric tensors with six different terms. It is the constitutive law for a linear, elastic, anisotropic and homogeneous material.

When the material is assumed to be isotropic, i.e., the material behavior is the same in every direction, only two elastic constants are necessary to represent the behavior and the constitutive law becomes

$$\sigma_{ij} = \lambda\,\epsilon_{kk}\,\delta_{ij} + 2\,\mu\,\epsilon_{ij} \quad \text{with} \quad \delta_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

where λ and μ are the Lamé elastic constants (equivalent to pressures) and δ_{ij} is the Kronecker delta. In practice, two other elastic constants are determined by rheological experiments: Young's modulus, that gives a measure of material rigidity, and Poisson's ratio, that gives a measure of its incompressibility. Lamé constants can be expressed using these constants, thus

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2(1+\nu)}$$

where E is Young's modulus (equivalent to a pressure) and ν is Poisson's ratio (without dimension). Since ν varies from 0 to $\frac{1}{2}$ with the incompressibility of the material, λ is infinite when the material is perfectly incompressible.

Volume conservation is achieved by introducing an extra variable called the Lagrangian term. In fact, for perfectly incompressible materials, we have

$$\lim_{\nu \to \frac{1}{2}} \lambda = +\infty \quad \text{and} \quad \lim_{\nu \to \frac{1}{2}} \epsilon_{kk} = 0$$

then,

$$\sigma_{ij} = P\,\delta_{ij} + 2\,\mu\,\epsilon_{ij}$$

is the new formulation for the constitutive law given previously, where $P = \lambda\,\epsilon_{kk}$ is an unknown finite quantity (the Lagrangian term). However, a material is generally considered sufficiently incompressible when $\lambda > 100\,\mu$, and no Lagrangian term is introduced. But increasing λ leads to stiff equations and numerical instability.

Anisotropy can be introduced straightforwardly using an approximate formulation, e.g., for incompressible material with unidirectional fibrous reinforcement, the constitutive law becomes

$$\sigma_{ij} = P\,\delta_{ij} + 2\,\mu\,\epsilon_{ij} + T b_i b_j$$

where T is the fiber constraint and $b_i b_j$ is the tensor product of the fiber direction vector b_i with itself. Of course, an exact formulation would use the tensor of elasticity with as many elastic constants as necessary to represent the anisotropic behavior.

Newton's second law is expressed as

$$\frac{\partial \sigma_{ij}}{\partial x_j} + f_i = \rho \gamma_i$$

where f_i is the sum of external forces, ρ is the mass density of the material and γ_i is its acceleration.

Rewriting Hookean isotropic constitutive law using displacement, we have

$$\sigma_{ij} = \lambda \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

and using Newton's second law, neglecting external forces, one obtains the Navier equation

$$\begin{aligned}
\frac{\partial \sigma_{ij}}{\partial x_j} &= \lambda \frac{\partial^2 u_k}{\partial x_j \partial x_k} \delta_{ij} + \mu \left(\frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{\partial^2 u_j}{\partial x_j \partial x_i} \right) \\
\frac{\partial \sigma_{ij}}{\partial x_j} &= \lambda \frac{\partial^2 u_k}{\partial x_i \partial x_k} + \mu \left(\frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{\partial^2 u_j}{\partial x_i \partial x_j} \right) \quad \text{since} \quad a_j \delta_{ij} = a_i \\
\rho \gamma_i &= (\lambda + \mu) \frac{\partial^2 u_j}{\partial x_i \partial x_j} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j}
\end{aligned}$$

that can be rewritten, for perfectly incompressible materials, as

$$\rho \gamma_i = \frac{\partial P}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \quad \text{with} \quad \frac{\partial u_k}{\partial x_k} = 0$$

these two equations being solved for the two unknown variables P and u_i.

Large Deformation Elasticity

We define the deformation gradient tensor F_{ij} as

$$F_{ij} = \frac{\partial x'_i}{\partial x_j} = \delta_{ij} + \frac{\partial u_i}{\partial x_j}$$

where $\frac{\partial x'_i}{\partial x_j}$ is the derivative of the ith coordinate of position x'_i in the deformed configuration w.r.t. the jth coordinate of the reference configuration. It has to be noted that F_{ij} is not a symmetric tensor.

A classical strain metric is the Green-Lagrange strain tensor E_{ij} defined by

$$E_{ij} = \frac{1}{2}\left(F_{ki}\,F_{kj} - \delta_{ij}\right)$$
$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{\partial u_k}{\partial x_i}\frac{\partial u_k}{\partial x_j}\right)$$

This tensor is nonlinear because second order terms (geometrical nonlinearities) are not neglected. We also note that Cauchy strain tensor is in fact a first order approximation of the Green-Lagrange strain tensor.

A classical stress metric is the second Piola-Kirchoff stress tensor S_{jr} defined by

$$S_{jr} = F^{-1}_{ij}\,\sigma_{ik}\,J\,F^{-1}_{rk}$$

where $J = \det F_{ij}$ is the determinant of the deformation gradient tensor and its third invariant.

This tensor is used in the reference configuration and thus one avoids the problem of using the Cauchy stress tensor for materials undergoing large deformation when the area in the deformed configuration is unknown. Moreover, it is energetically consistent with the Green-Lagrange strain tensor, i.e., the strain energy density calculated with these tensors is equal to the one calculated with the Cauchy stress tensor and the Cauchy strain tensor:

$$S_{ij}\,E_{ij} = \sigma_{ij}\,\epsilon_{ij}$$

The second Piola-Kirchoff stress tensor has very little physical meaning and is mainly used to compute the Cauchy stress tensor. However, if straight Cauchy stresses are used, incorrect results will be obtained, because they do not take into account the effect of rigid body rotations.

Papers Overview

A classic way to solve elasticity equations is to use finite elements method (FEM) in the static case, the simulation consisting of a series of equilibrium states, thus with no dynamic behavior. Gourret et al. (1989) propose one of the first applications of FEM to computer animation by simulating deformations of a hand grasping and pressing a ball. The material is considered isotropic and obeying Hooke's law. To take into account geometrical nonlinearities, the second Piola-Kirchoff stress tensor and Green-Lagrange strain tensor are used. Classical FEM is thus rigorously employed, but as a result the method is very time consuming.

Chen and Zeltzer (1992) model human skeletal muscle using a classic biomechanical model of muscle contraction to apply nonlinear forces to finite elements mesh nodes. Active and passive muscle force generators and tendons force generators act

along the longitudinal direction of the muscle. The FE model then simulates the dynamic behavior of an isotropic linear elastic passive material submitted to external forces. Even though this model is a rough approximation of the behavior of a living muscle tissue, it gives good results when validated using classic muscle physiology experiments.

Sagar et al. (1994) present a model of the eye and surrounding face for use in a surgical simulator. Mechanical properties of the cornea are simulated in the static case using a large deformation finite element model of an homogeneous, orthotropic and nonlinear elastic material with a J-shaped uniaxial stress-strain behavior. Using a high-end machine with multiple processors, the authors achieved update rates significantly greater than 10 Hz, which is barely sufficient for interactive use, but is still impressive considering this is a very complex biomechanical model.

Interactivity is achieved by precomputing matrix inverses, assuming they are constant when the elements geometry does not change too much, but thus preventing any change in mesh topology. Bro-Nielsen and Cotin (1996) simulate an isotropic material obeying Hooke's law. Under the small deformation hypothesis, three simplifications are made to speed up computations. First, the stiffness matrix obtained with FEM is rewritten: it takes only surface nodes into account, but still displays the volumetric behavior of the object. Second, the matrix inverse is precomputed. Third, a selective matrix vector multiplication is used to exploit the sparse structure of the force vector. However, it is impossible to impose displacements constraints and only a static equilibrium position is obtained after application of forces.

James and Pai (1999) use the boundary elements method (BEM) to solve the Navier equation for compressible materials in the static case. They propose a fast update method of the stiffness matrix when few boundary conditions change, which corresponds to the application scenario where a virtual tool is tickling an object, resulting in only a few nodes being affected. By exploiting the coherence of the stiffness matrix inverse, modification is achieved at the expense of precomputation and memory usage.

Zhuang and Canny (1999) propose a real-time simulation of isotropic linear elastic material under the large deformation hypothesis. The use of a nonlinear strain tensor prevents employing the preprocessing step previously used for achieving real-time performance. Since internal force vector is no longer linear w.r.t. nodal displacement vector, there is no constant stiffness matrix inverse to be precomputed. Instead of a sequence of static equilibria, an explicit time integration scheme allows dynamic behavior. To simplify the Lagrange equation of motion, global mass and damping matrices are diagonalized. A graded mesh is used instead of a uniform mesh to speed up computations, at the expense of limited object geometry due to hexahedral elements.

Explicit FEM avoids stiffness matrix inversion by considering only a superposition of local solutions. O'Brien and Hodgins (1999) simulate initiation and propagation of cracks in 3D objects. Linear elastic fracture mechanics is used, with homogeneous, isotropic material, under the large deformation hypothesis. Following fracture development requires local remeshing, and brittle materials involve stiff equations that are solved using small time steps with an explicit integration scheme; thus, computation

is very slow. Debunne et al. (2001) reuse the previous approach to simulate soft materials but speed it up and reach interactivity with multiresolution techniques. However, this requires precomputing a mesh hierarchy, and thus prevents any change in object topology during simulation.

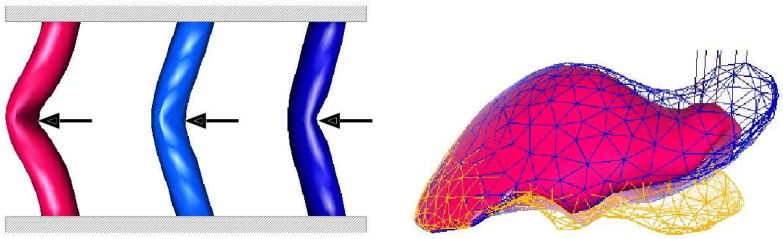
Related to explicit FEM, the "tensor-mass" model of Cotin et al. (1999) simulates dynamic deformations of an isotropic material with a linear constitutive equation, under the small deformation hypothesis. The main idea is to split, for each tetrahedron, the force applied at a vertex in two parts: a force created by the vertex displacement (w.r.t. rest position) and a force produced by the displacement of its neighbors, i.e., vertices that are linked to it by mesh edges. This is possible using local tensors that relate force to displacement. Evaluation of these tensors requires evaluation of the stiffness matrix associated to each tetrahedron adjacent to one of the edges linked to the vertex. New positions of the vertices are obtained from forces using a dynamic explicit integration scheme.

Picinbono et al. (2000, 2001) have improved this model to handle anisotropic behavior and nonlinear elasticity under the large deformation hypothesis. For this, strain is measured using Green-Lagrange strain tensor (called Green-St Venant by the authors) and elastic energy is rewritten from a quadratic function into a fourth order polynomial of the displacement gradient (St Venant-Kirchoff elasticity). Transversally isotropic materials are supported. This a special case of anisotropy where a material has a different behavior in one given direction (see Fig. 2.3a). To optimize computation time, nonlinear elasticity is used only for nodes whose displacement is larger than a given threshold, otherwise linear elasticity is considered as a sufficiently good approximation (see Fig. 2.3b).

Finally, for the sake of completeness, we mention continuous models that simulate only global deformation without using FEM for solving elasticity equations. Pentland and Williams (1989) simulate global deformation by modal analysis. The Lagrange equation of motion is rewritten by diagonalizing the global mass, damping and stiffness matrices. Thus, the previous system is now composed of independent equations, each describing a different vibration mode of the object. Linear superposition of these modes determines how the object responds to a given force; thus, neglecting the high frequency, low amplitude modes, it allows interactive simulation. Terzopoulos et al. (1987) propose a model based on minimization of deformation energy that is more concerned with differential geometry properties of objects (either curves, surfaces or solids) and use only the analysis of deformations part of the linear elasticity framework. Dynamic differential equations are solved using finite differences method and implicit integration, thus limiting the model to regular meshes and non-interactive applications.

Discussion

In continuum mechanics, hyperelasticity theory is used for nonlinear elastic materials under the large deformation hypothesis. Constitutive equations are written us-



(a) (b)

Figure 2.3: Continuous models. Results from Picinbono et al. (2000, 2001). In (a), deformation of tubular structures with top and bottom faces fixed, submitted to the same forces (from left to right, isotropic model, anisotropic models along the height direction, with rightmost being twice as stiff as the middle one in the anisotropic direction). In (b), a force is applied to the right lobe of the liver with linear (wireframe, top) and nonlinear (solid) liver models. The linear model displays an unrealistic volume increase due to the linear strain tensor. The rest shape is indicated (wireframe, bottom).

ing Green-Lagrange strain tensor and the second Piola-Kirchoff stress tensor. This framework could be used for non-interactive simulations of such materials in computer graphics. O'Brien and Hodgins noted that there is nothing that would preclude switching the linear isotropic formulation used in their paper to anisotropic formulation or even to a nonlinear stress-strain relationship.

Among all the papers presented, only the work of Picinbono et al. focused specifically on anisotropic materials under large deformation hypothesis. However, this does not imply that other previous interactive approaches could not be adapted to simulate these materials. Handling anisotropic Hookean elastic material is straightforward, since anisotropy is encoded in the tensor of elasticity. However, it is not clear whether or not the framerate would remain acceptable for interactive applications, because the constitutive law would now contain matrix-matrix multiplication, that may slow computations down in the case of explicit FEM. The approaches of Bro-Nielsen and Cotin and James and Pai, based on processing of the global stiffness matrix could still be valid, but only under the small deformation hypothesis. Concerning nonlinear materials, an entirely new constitutive law is needed, which might be computationally intensive. FEM or BEM solutions of Bro-Nielsen and Cotin and James and Pai would have to be completely rewritten. But a first approximation could use piecewise linear formulations, as suggested by Debunne (2000).

The modal analysis approach of Pentland and Williams (1989) could handle anisotropic material, since anisotropy would be encoded in the global stiffness matrix. According to the authors, nonlinear materials may be modeled by summing the modes at the end of each time step to form the material stress state, which can then be used to drive nonlinear material behavior.

Incompressible materials remains an unsolved problem for all the papers presented. Exact solution using a Lagrangian term is never used and instead they increase the

Lamé constant λ, since a material is generally considered sufficiently incompressible when $\lambda > 100\,\mu$ (Debunne, 2000), but at the risk of an increased numerical instability. Picinbono et al. address the problem by adding an incompressibility constraint using a penalty method: volume variation is penalized by applying to each vertex of the tetrahedron a force directed along the normal of the opposite face, the norm of the force being the square of the relative volume variation. Their method is quite effective: they obtain on their example 0.2 to 1% volume variation with incompressibility constraint, as opposed to 0.3 to 2% without this constraint.

This last example emphasizes the complexity of recent interactive models based on elasticity theory; however, parameter setting is simple when data are available in continuum mechanics literature. When this is not the case, it is possible to conduct experiments according to standardized protocols, but not always without encountering difficulties. In fact, obtaining these measurements on biological materials is currently an active research area.

Finally, continuous models are particularly adapted to multiresolution in animation. Debunne et al. proposed a space and time adaptive refinement method to simulate linear isotropic visco-elastic materials in real time, using a non nested multiresolution hierarchy of tetrahedral meshes. In such a model, regions simulated at different resolutions must vibrate at the same frequency, with the same amplitude, otherwise discrepancies at the interfaces between regions will cause numerical instability. A linear isotropic elasticity formulation using the Green-Lagrange strain tensor meets such requirements.

2.2.2 Discreet Models

Discreet models approximate a continuous material by a set of primitives (point masses) linked by constraints. Each primitive neighborhood is either limited to the nearest nodes in the graph of primitives, as in mass-spring systems, or potentially extended to all primitives, but generally function of distance, as in particle systems.

The animation algorithm consists in integrating w.r.t. time, for each point mass, the Lagrange equation of motion, written as

$$m\,\frac{\partial^2 \mathbf{x}}{\partial t^2} + d\,\frac{\partial \mathbf{x}}{\partial t} + k\,\mathbf{x} = \mathbf{f}$$

where m is the mass of the primitive, d and k are respectively the damping and stiffness constants and $\mathbf{x}$ is point position. Moreover, $m\,\frac{\partial^2 \mathbf{x}}{\partial t^2}$ is the inertial force, $d\,\frac{\partial \mathbf{x}}{\partial t}$ is the damping force, $k\,\mathbf{x}$ is the internal elastic force and $\mathbf{f}$ is the sum of the external forces, such as gravity and constraint forces.

Papers Overview

Mass-spring systems are intuitive, simple to implement, fast and easily amenable to parallelization. They are less physically accurate than continuum mechanics models

but they do not introduce any geometric distortion due to linear strain metric. They are generally built by discretizing the deformable object with a given 3D mesh and setting point masses on the mesh nodes and damped springs on the mesh edges (for a tetrahedral mesh). This simplicity explains why mass-spring systems proposed in literature do not differ a lot from paper to paper except for the geometry of mesh elements, the number of springs per element (due to a varying number of diagonal springs for an hexahedral element) and slightly different force formulations. However, there is a great diversity of applications of these systems and also of solutions given to mass-spring systems drawbacks. The latter point will be analyzed in the discussion section.

Cloth simulation has been one of the most popular applications of mass-spring systems, with either quadrilateral or triangle meshes. Provot (1995) improves the classical cloth mass-spring model in order to take into account non-elastic properties of woven fabrics. In fact, when linear springs are used, concentration of high stresses in small regions of the surface produce unrealistic deformations. Increasing spring stiffness is a straightforward but computationally expensive solution to this problem. Provot's method, inspired from inverse dynamic procedures, projects overstretched springs back into the acceptable range of elongation and thus simulates nonlinear cloth behavior without stiff equations.

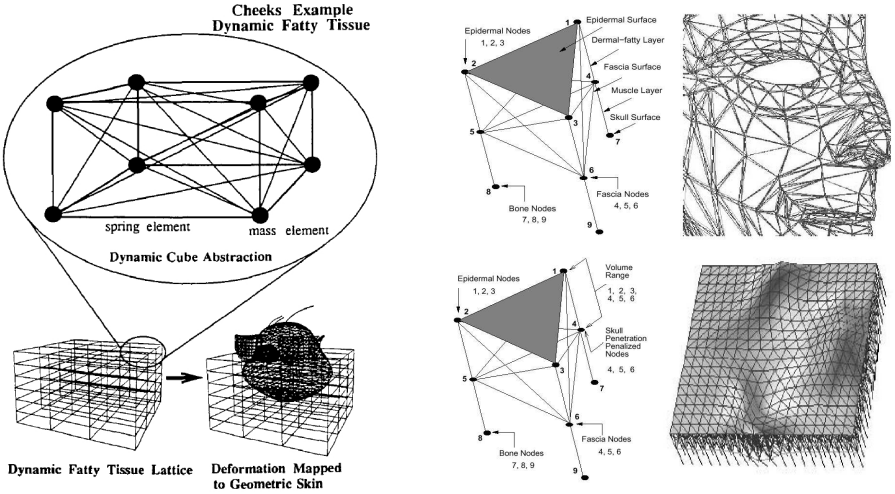
Complex living materials, such as muscles, have also been simulated with mass-spring systems, using nonlinear, active springs and volume conservation constraints. Miller (1988) animates snakes and worms by simulating muscle contractions; directional friction is included and thus locomotion results. Each segment of the creature is modeled as a cube of masses with springs along each edge and across the diagonal of each face. Spring rest length is animated as a function of time to simulate muscle contractions. Rest length for the springs around the circumference of the creature is scaled to keep the total volume of the creature constant.

Chadwick et al. (1989) propose a complete system for creating and animating characters defined with three layers: skeleton, muscle and fatty tissue, and skin. The skeleton layer is an articulated hierarchy which controls the motion of the character. The muscle and fatty tissue layer is composed of free-form deformation (FFD) lattices and is attached to the skeleton. These deformations act on the geometric skin. Deformations are based on either kinematic, dynamic or sculpted constraints. With kinematic constraints, the kinematic joint angles of the skeleton control an abstract muscle behavior. Muscle deformation is thus caused by muscle contraction and tendon influence on the shape of the joint. With dynamic constraints, FFD lattices are mapped to hexahedral mass-spring systems of the same topology. Dynamic simulation of the point masses is mapped back onto the FFD control points, thus resulting in damped oscillations of the fatty tissue layer caused by character motion (see Fig. 2.4a). With sculpted constraints, the animator entirely controls the deformation by moving control points of the FFD lattices.

The face animation system of Lee et al. (1995) gives a good idea of the simulation complexity one can achieve using mass-spring systems. Functional models of

the heads of human subjects are built automatically from laser-scanned range and reflectance data. The skull is covered by deformable tissue which has five distinct layers. Each layer has its own mechanical properties modeled by linear or piecewise linear spring laws, some of them with time varying parameters, to simulate muscles. Incompressibility of real human skin is enforced for each triangular prism element using a volume constraint force based on the change of volume of the element and displacements of its nodes (see Fig. 2.4b).

Finally, the muscle model of Ng-Thow-Hing and Fiume (1997) demonstrates that nonhomogeneous anisotropic material can be simulated using mass-spring systems. It uses a B-spline solid for muscle geometry and attaches to its sample points a lattice of springs, partitioned into radial and longitudinal groups. Thus, a set of dependency constraints exists between sample points and allows easy adjustment of many control points simultaneously. Since sample points exist within the solid as well as on the surface, the B-spline solid can model microstructures within the solid, such as fiber bundles, using the mass-spring lattice.



(a) (b)

Figure 2.4: Discreet models. In (a), results from Chadwick et al. (1989). Dynamic constraints for fatty tissue deformation: FFD lattices are mapped to hexahedral mass-spring systems of the same topology to simulate the damped oscillations of the fatty tissue layer caused by character motion. In (b), results from Lee et al. (1995). Skin tissue triangular prism element (top left) with a close-up view of a face mesh (top right); volume conservation and skull nonpenetration element (bottom left) and assembled layered tissue elements under multiple muscle forces (bottom right).

Classical particle systems are well adapted for unstructured materials, because they are not affected by topology changes. Terzopoulos et al. (1989) model melting material by decreasing stiffness of a mass-spring system and then replacing it by a particle system beyond melting point. Most of the time, interparticular forces are based on long range attraction – short range repulsion laws, with an equilibrium distance where

force disappears; e.g., Miller and Pearce (1989) use a Lennard-Jones force formulation that is used in physics to model interatomic forces. But arbitrarily complex force formulations can be used; e.g., Szeliski and Tonnesen (1992) simulate particles that model surface elements instead of volume elements and which undergo forces according to the orientation of their neighbors. However, classical particle systems, with their "soft" particle's neighborhood definition has been mostly employed for unstructured rather than deformable materials.

The smoothed particles hydrodynamics (SPH) method has been used to model gaseous phenomena, such as smoke, or unstructured materials, such as mud (Desbrun and Gascuel, 1996). But particles are rather considered as sample points of continuous fields than point masses discretizing an object. They allow evaluation of the values of these fields and their derivatives by local filtering. Thus, they enable simulation of state equations that describe the physical behavior of various materials.

Discussion

We will not discuss particle systems and SPH models here because they are tailored for unstructured materials. The difference between mass-spring and particle approaches is in fact quite analogous to the difference between continuum mechanics and fluid mechanics: each theory is grounded on the basic hypothesis concerning the material simulated.

Simulating anisotropic materials with mass-spring systems is very difficult. Since springs are positioned along the edges of a given mesh, the geometrical and topological structure of this mesh strongly influences the material's behavior. Whatever the spring law used, the mesh geometry may generate uncontrolled anisotropy, as shown in Fig. 2.5a for a tetrahedral mesh. However, the undesired behavior disappears when hexahedral elements aligned with the forces directions are used, as shown in Fig. 2.5b. Of course, if the tiling of the object volume was computed from the tetrahedrization of random uniformly-distributed sample points, the unwanted anisotropy problem would tend to disappear when the density of the mesh increases. However, using an extremely dense mesh would reduce efficiency.

The most common approach to controlling the behavior of a mass-spring system, at least along a few "directions of interest", is to specifically design the mesh in order to align springs on these specific directions. This was done for instance in Miller's models of snakes and worms and in the muscle model of Ng-Thow-Hing and Fiume, where some of the springs were aligned with the muscle fibers and the rest were set perpendicular to them. Unfortunately, creating such meshes manually would be time consuming in a general case, where fiber directions generating anisotropy vary in an arbitrary way inside the object (see Fig. 2.1). We are rather looking for an approach that takes as input a 3D volume mesh obtained with a volume meshing package, such as Simulog's TetMesh-GHS3D, fed with a 3D surface mesh, and outputs the deformable model behavior with specified properties in specific directions. As a result, this approach would decouple the simulation model from the graphical model (Barzel, 1992).

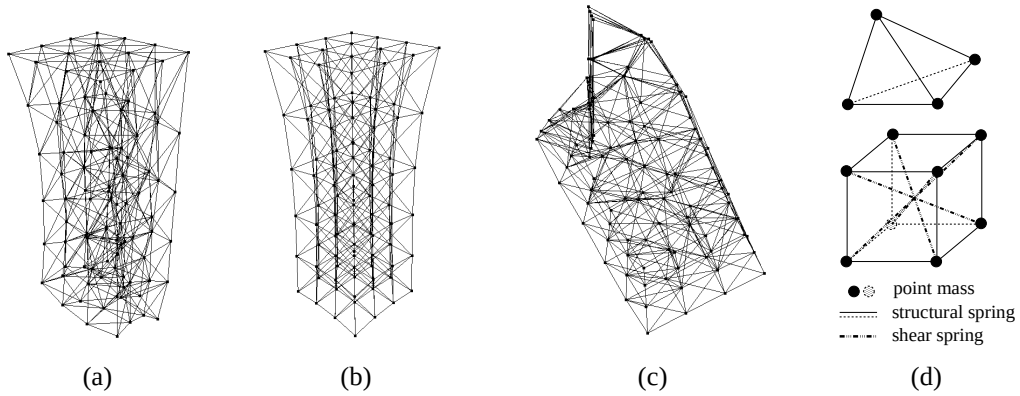


Figure 2.5: Mass-spring systems drawbacks. Left, comparison between two meshes, under the action of gravity, undergoing a downward pull at their bottom end while their top end is fixed. We observe uncontrolled anisotropy in the tetrahedral mesh (a), but not in the hexahedral mesh with springs aligned in the gravity and pull force directions (b). Right, equilibrium state of a cantilever beam, which left end is fixed, under the action of gravity (c). Obviously, the mass-spring system considered (tetrahedral mesh) is unable to sustain flexion. The spring configurations used for tetrahedral and hexahedral meshes are given in (d).

As opposed to what is observed for anisotropy, it is easy to design arbitrary non-linear stress-strain relationship with springs, e.g., by designing a piecewise linear or a nonlinear spring law or even by imposing a constraint on maximum spring length, as Provot does. Concerning volume conservation, the solutions proposed are based on the penalty method. These soft constraints are either very specific (the case of tube-like shapes animated by hexahedral mass-spring systems for Miller) or more general (triangular prism element for Lee et al.). In both cases, we cannot evaluate the efficiency of the constraint since no indication of volume variation is given. Meseure and Chaillou (2000) give an intermediate solution to the problem of enforcing constant volume deformation. They simulate human organs using a deformable and a rigid component: a mass-spring system fixed on the surface of a virtual rigid body (virtual because its only purpose is to provide rigid body behavior). Thus, the mass-spring system is constrained to have only one rest position, and the volume variation of the object is kept within boundaries, depending on the stiffness of the zero-length springs tying the surface deformable component and the virtual rigid component.

Among the intrinsic limitations of mass-spring systems, one of the main problems remains parameter setting. In the case of a tetrahedral mesh, computing the masses in order to set up a homogeneous material can be done approximately by computing each point mass according to the volume of the Voronoï region around it. However, there is no easy solution for spring parameters. Approximating a desired behavior using a given mesh can be achieved using optimization to tune individual spring stiffnesses. Louchet et al. (1995) apply an evolutionary genetic technique to the identification of internal parameters of a physical model of fabrics that uses a mass-spring mesh and Provot's method to model nonlinear elastic behavior. Identification of the internal pa-

rameters from geometric data is based on a cost function which measures the difference in behavior between the reference and the model, and an evolutionary minimization algorithm. On a 17×17 mesh, convergence is obtained after about 50 to 100 generations.

Deussen et al. (1995) use simulated annealing to obtain optimal mass-spring systems approximations of deformable bodies obeying to linear elasticity. It is a two-step process: first, find positions and masses of the points that approximate the mass distribution, second, define the topology of the connections and optimize their spring constants. Four test configurations were used for optimizing elasticity on a 2D deformable body: two with stretching loads and two with shearing loads. The quality criterion used is the standard deviation between actual and reference displacements of all points. This method allows approximation of homogeneous as well as inhomogeneous and anisotropic materials. Two-dimensional mass-spring systems containing up to some hundred points are optimized successfully and an extension to 3D with nine basic loads is suggested but not tested, due to the large computational cost.

These two optimization methods prove the possibility of approximating mechanical behaviors with mass-spring systems but van Gelder (1998) demonstrates the impossibility of setting the stiffnesses of a mass-spring system to obtain an exact simulation of the elastic material properties of a continuous model. However, this doesn't mean that global behavior, i.e., stress-strain relationships, cannot be reproduced with a mass-spring system. In fact, Boux de Casson (2000) simulated linear and nonlinear stress-strain relationships using linear and nonlinear spring laws, proving that the behavior at the spring level is conserved at the object level.

Finally, since mass-spring system behavior changes when topology or geometry of the mesh is modified, dynamic behavior at different resolutions is different. Thus, very few papers address the issue of physical simulation at multiple levels of detail with mass-spring systems. Hutchinson et al. (1996) propose a scheme for adaptively refining portions of mass-spring systems to a required accuracy, producing visually more realistic results at a reduced computational cost. Detection of inaccuracy is performed using an angle criterion between springs joining a mass from opposite directions. The response is the addition of masses and springs around the area where the discontinuity occurs: point masses keep the same value but spring stiffnesses double at each level of refinement to prevent regions of increased mass from behaving differently. However, this approach is possible only for quadrilateral or hexahedral meshes (here, for simulating a deformable sheet) because they lend themselves naturally to regular subdivision. Debunne has submitted a mass-spring system to the same multiresolution "aptitude test" that was used for continuous models. Results demonstrated without ambiguity that the motion of an oscillating mass-spring system cannot have the same frequency and amplitude at different mesh resolutions.

2.3 Modeling Anisotropy

Our aim is to specify the mechanical properties of the material independently from the mesh geometry and topology. In usual mass-spring systems, internal forces acting inside the material are approximated exclusively by forces acting along the edges of the mesh (i.e., along the springs). This is the reason for the uncontrolled anisotropy problem described earlier, and for the difficulty in specifying desired anisotropic properties.

The basic idea of our method is to let the user define, everywhere in the object, mechanical characteristics of the material along a given number of axes corresponding to orientations of interest at each current location, such as fiber and cross-fiber directions in a muscular tissue. All internal forces will be acting along these axes instead of acting along the mesh edges. For instance, in the case of organic materials such as muscles, one of the axes of interest should always correspond to the local fiber orientation. Since the object is tiled using a mesh, axes of interest and the associated mechanical properties are specified at the barycenter of each volume element inside the mesh. We currently use three orthogonal axes of interest.

General Scheme

During deformations of the material, the three axes of interest, of given initial orientation, evolve with the volume element to which they belong. In order to be able to know their position at each time step, we express the position of the intersection point of one axis with one of the element faces as a linear combination of the positions of the vertices defining the face. The corresponding interpolation coefficients are computed for each face in the rest position (see Figures 2.7 and 2.8).

Given the position of the point masses of a volume element, we are thus able to determine the coordinates of the six intersection points and consequently the three axes that constitutes the local frame, up to the precision of our linear interpolation. From the deformation of the local frame, we can deduce the resulting forces on each intersection point. Then, for a given face, we can compute the force value on each point mass belonging to this face by “inverse” interpolation of the force value at the intersection point. The interpolation coefficients previously defined are therefore also considered as weighting coefficients of the force on each point mass.

2.3.1 Forces Calculations

Damped springs with associated stiffness and damping coefficients are used to model stretching characteristics along each axis of interest. In order to specify shearing properties, angular springs are added between each pair of axes. Rest lengths and rest angles are pre-computed from the initial position of the object that defines its rest shape. The equations we use for these springs are detailed below.

Axial damped spring

The spring forces $\mathbf{f_1}$ and $\mathbf{f_2}$ between a pair of intersection points P_1 and P_2 at positions $\mathbf{x_1}$ and $\mathbf{x_2}$ with velocities $\mathbf{v_1}$ and $\mathbf{v_2}$ are

$$\mathbf{f_1} = -\left[k_s \left(\|\mathbf{l_{21}}\| - r \right) + k_d \frac{\dot{\mathbf{l}}_{\mathbf{21}} \cdot \mathbf{l_{21}}}{\|\mathbf{l_{21}}\|} \right] \frac{\mathbf{l_{21}}}{\|\mathbf{l_{21}}\|} \qquad \mathbf{f_2} = -\mathbf{f_1}$$

where $\mathbf{l_{21}} = \mathbf{x_1} - \mathbf{x_2}$, r is the rest length, $\dot{\mathbf{l}}_{\mathbf{21}} = \mathbf{v_1} - \mathbf{v_2}$ is the time derivative of $\mathbf{l_{21}}$, k_s and k_d are respectively the stiffness and damping constants (see Fig. 2.6a). This is the classical formulation for a Hooke's law spring (Witkin, 1999b), but we could easily extend the approach to model nonlinear spring behavior, as we did in Section 2.6.1.

Angular spring

The spring forces $(\mathbf{f_1}, \mathbf{f_2})$ and $(\mathbf{f_3}, \mathbf{f_4})$ between two pairs of intersection points (P_1, P_2) and (P_3, P_4) are

$$\begin{aligned} \mathbf{f_1} &= -\left[k_s \left(\frac{\mathbf{l_{21}} \cdot \mathbf{l_{43}}}{\|\mathbf{l_{21}}\| \, \|\mathbf{l_{43}}\|} - c \right) \right] \frac{\mathbf{l_{43}}}{\|\mathbf{l_{43}}\|} \qquad \mathbf{f_2} = -\mathbf{f_1} \\ \mathbf{f_3} &= -\left[k_s \left(\frac{\mathbf{l_{21}} \cdot \mathbf{l_{43}}}{\|\mathbf{l_{21}}\| \, \|\mathbf{l_{43}}\|} - c \right) \right] \frac{\mathbf{l_{21}}}{\|\mathbf{l_{21}}\|} \qquad \mathbf{f_4} = -\mathbf{f_3} \end{aligned}$$

where $\mathbf{l_{21}} = \mathbf{x_1} - \mathbf{x_2}$ and $\mathbf{l_{43}} = \mathbf{x_3} - \mathbf{x_4}$, c is the cosine of the rest angle between $\mathbf{l_{21}}$ and $\mathbf{l_{43}}$ (equal to zero for orthogonal axes) and k_s is the stiffness constant (see Fig. 2.6b). We have chosen this formulation because it is related to a classic distortion metric in continuum mechanics: the "loss of right angle". We consider two vectors $\mathbf{u}$ and $\mathbf{v}$ such as the angle $(\mathbf{u}, \mathbf{v}) = \frac{\pi}{2}$. After distortion, let the angle $(\mathbf{u}, \mathbf{v}) = \frac{\pi}{2} - \theta$. The "loss of right angle" is equal to θ.

Here, two approximations are made since we assume a small variation of the angle around $\frac{\pi}{2}$, i.e., θ small. First, we have $\sin(\theta) = \cos\left(\frac{\pi}{2} - \theta\right) \approx \theta$, and we take as metric the variation of the $(\mathbf{u}, \mathbf{v})$ angle's cosine. Second, we consider sufficient to use as unit vector the other vector of the pair, instead of a vector normal to the one considered, in the plane where the angle is measured. These two approximations gave good results in practice. Furthermore, we found no necessity to use damped angular springs.

2.3.2 Application to Tetrahedral Meshes

Many objects in computer graphics are modeled using triangular surface meshes. Generating a 3D mesh from such a description, using tools like Simulog's TetMesh, yields tetrahedral volume meshes. This section details our method in this case.

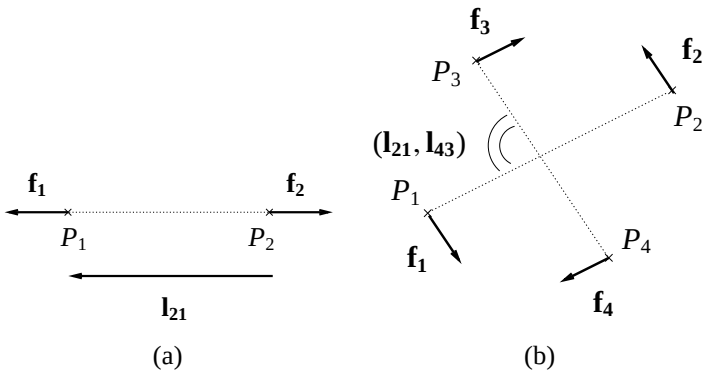


Figure 2.6: In (a), spring forces $\mathbf{f_1}$ and $\mathbf{f_2}$ between a pair of intersection points 1 and 2 for our axial damped spring. In (b), spring forces $(\mathbf{f_1}, \mathbf{f_2})$ and $(\mathbf{f_3}, \mathbf{f_4})$ between two pairs of intersection points (P_1, P_2) and (P_3, P_4) for our angular spring (b). See corresponding paragraphs for equations.

Figure 2.7 depicts a tetrahedral element, with the associated frame defining the three axes of interest. We express the position $\mathbf{x_P}$ of point P as a function of the positions of vertices A, B and C of the given face, using barycentric coordinates:

$$\mathbf{x_P} = \alpha\,\mathbf{x_A} + \beta\,\mathbf{x_B} + \gamma\,\mathbf{x_C}$$

for example, if $\alpha = 1$ and $\beta = \gamma = 0$, we get $\mathbf{x_P} = \mathbf{x_A}$. Therefore, a force $\mathbf{f_P}$ applied to point P is split into forces $\alpha\,\mathbf{f_P}$, $\beta\,\mathbf{f_P}$ and $\gamma\,\mathbf{f_P}$, respectively applied on points A, B and C. We can note that since the elementary volume has four faces, and since there are three axes of interest defining six intersection points, two such points may lie on the same face of the volume element. This was not a problem in practice, since forces applied on mesh nodes are correctly weighted.

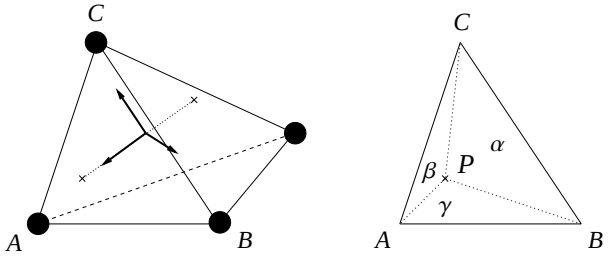


Figure 2.7: Tetrahedral element. A point mass is located at each vertex. A local frame is defined at the barycenter of the element (left). Each axis is characterized by the barycentric coordinates α, β and γ (with $\alpha + \beta + \gamma = 1$) of its two intersection points (right, for a given face). These coordinates are easily obtained using an area ratio.

2.3.3 Application to Hexahedral Meshes

The use of hexahedral meshes is not as common as tetrahedral ones, since the range of geometries they can define is more limited. However, these meshes may be useful for animating objects modeled using free form deformation lattices (Chadwick et al., 1989) or voxels (Chen et al., 1998). This kind of data, with information about material characteristics specified in each voxel (possibly including anisotropy), may be provided by medical imaging applications.

Applying the general method presented in Section 2.3 to hexahedral meshes is straightforward. Figure 2.8 depicts an hexahedral element, with the associated frame defining the three axes of interest. We express the position $\mathbf{x_P}$ of point P as a function of the positions of vertices A, B, C and D of the given face, using bilinear interpolation coordinates:

$$\mathbf{x_P} = \zeta\,\eta\,\mathbf{x_A} + (1-\zeta)\,\eta\,\mathbf{x_B} + (1-\zeta)\,(1-\eta)\,\mathbf{x_C} + \zeta\,(1-\eta)\,\mathbf{x_D}$$

for example, if $\zeta = 1$ and $\eta = 1$, we get $\mathbf{x_P} = \mathbf{x_A}$. Therefore, a force $\mathbf{f_P}$ applied to point P is split into forces $\zeta\,\eta\,\mathbf{f_P}$, $(1-\zeta)\,\eta\,\mathbf{f_P}$, $(1-\zeta)\,(1-\eta)\,\mathbf{f_P}$ and $\zeta\,(1-\eta)\,\mathbf{f_P}$, respectively applied on points A, B, C and D. Here, there is only one intersection point per face of the volume element. Since the element has eight vertices, the system is under-constrained instead of being over-constrained, as in the tetrahedral case. As a consequence, each elementary volume may have several equilibrium states, corresponding to the same rest position of the three axes of interest but to different positions of the vertices, if volume conservation forces are not applied.

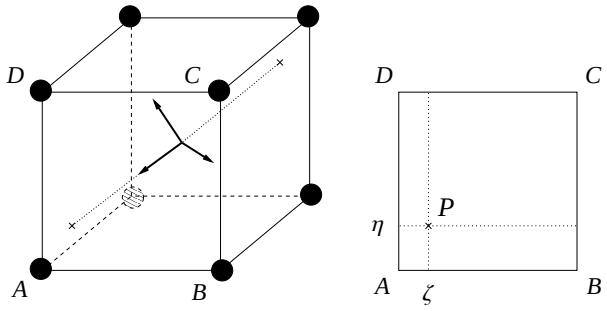


Figure 2.8: Hexahedral element. A point mass is located at each vertex. A local frame is defined at the barycenter of the element (left). Each axis is characterized by the bilinear interpolation coordinates ζ and η (with $0 \leq \zeta \leq 1$ and $0 \leq \eta \leq 1$) of its two intersection points (right, for a given face).

2.4 Volume Conservation

Living tissues such as muscles undergo quasi-constant volume deformations. Thus, it is important to be able to simulate this property with our model. Classical mass-spring systems have no constraint on volume, thus the amount of volume variation depends largely on mesh geometry. Animating constant volume deformations with a classical mass-spring system is not simple: forces are only applied along the edges of each volume element, while maintaining a constant volume basically requires adding radial forces. We have devised a solution for tetrahedral and hexahedral meshes that uses soft constraints to ensure volume conservation on each element in order to simulate more or less compressible materials.

2.4.1 Tetrahedral Meshes

To ensure volume conservation of a tetrahedral mesh, a straightforward solution would be to set a constraint using a force proportional to the variation of volume of each tetrahedron. After conducting experiments similar to the ones presented in Fig. 2.9, we can conclude that this solution does not work as well as one could expect. In fact, constraint forces applied on vertices of each tetrahedron are competing with other forces to modify the position of each vertex in order to satisfy the constraint. If the metric chosen for the constraint is a nonlinear function of the position of the vertices, as the volume is, a nonconstant stiffness coefficient will be needed to compensate for the variations of the stiffness of the constraint itself. If this is not the case, the constraint becomes very sloppy for small displacements of the vertices, resulting in oscillations.

Even if it is not sensu stricto a constraint on volume, a linear function of the position of the vertices would be a better choice for the metric. After several attempts, the sum of the distances between each vertex and the barycenter was considered a good compromise. Moreover, since in any case we need to compute barycenter-vertex vectors for the directions of application of our radial constraint forces, it is much more efficient to use these already computed vectors to evaluate our metric. This constraint force formulation is loosely related to the soft volume conservation constraint of Lee et al. (1995).

We define $\mathbf{x_B}$ as the position of the barycenter of the tetrahedral element, with

$$\mathbf{x_B} = \frac{1}{4} \sum_{i=0}^{3} \mathbf{x_i}$$

where $\mathbf{x_i}$ is the position of the ith vertex. We introduce our metric as the sum of the distances between each vertex and barycenter, i.e., $\sum_{i=0}^{3} \|\mathbf{x_i} - \mathbf{x_B}\|$, then, we define the force applied on the jth vertex as

$$\mathbf{f_j} = -\left[k_s \left(\sum_{i=0}^{3} \|\mathbf{x_i} - \mathbf{x_B}\| - \sum_{i=0}^{3} \|\mathbf{x_i} - \mathbf{x_B}\|_0 \right)\right] \frac{\mathbf{x_j} - \mathbf{x_B}}{\left\|\mathbf{x_j} - \mathbf{x_B}\right\|}$$

where k_s is the constraint stiffness and $\sum_{i=0}^{3} \|\mathbf{x_i} - \mathbf{x_B}\|_0$ is the rest length of this "volume spring". It was not necessary to add damping forces with this constraint.

This method gives satisfactory results in practice: we get less than 1.5% volume variation in our experiment (see Fig. 2.9); however, results depend on the material parameters chosen and the type of experiment conducted. Since our model is much more mesh-independent than a mass-spring system, it displays larger volume variation than the corresponding mass-spring system when no volume conservation forces are used. But our model displays five times less volume variation than mass-spring systems when we enforce constant volume with a soft constraint.

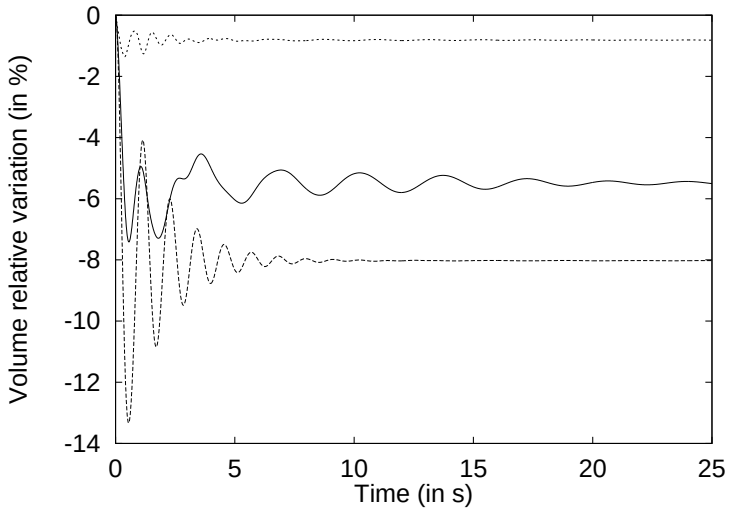


Figure 2.9: Volume conservation experiments using the same tetrahedral mesh lying on a table under force of gravity. In our model, one axis of interest is set to the vertical direction (the direction of application of gravity) and the two others in horizontal directions. Parameters are chosen identical along the 3 axes. The same stiffness and damping values are used in all experiments. Bottom graph (dashed line): our model without volume conservation forces. Middle graph (solid line): mass-spring system (using the same mesh). Top graph (dotted line): our model with volume conservation forces.

2.4.2 Hexahedral Meshes

Given the characteristics of hexahedron geometry, we can use the same expression for volume conservation forces, acting in radial directions with respect to the volume element.

We define $\mathbf{x_B}$ as the position of the barycenter of the hexahedral element, with

$$\mathbf{x_B} = \frac{1}{8} \sum_{i=0}^{7} \mathbf{x_i}$$

where $\mathbf{x_i}$ is the position of the ith vertex. Then, we define the force applied on the jth vertex as

$$\mathbf{f_j} = -\left[k_s\left(\sum_{i=0}^{7}\|\mathbf{x_i}-\mathbf{x_B}\| - \sum_{i=0}^{7}\|\mathbf{x_i}-\mathbf{x_B}\|_0\right)\right]\frac{\mathbf{x_j}-\mathbf{x_B}}{\left\|\mathbf{x_j}-\mathbf{x_B}\right\|}$$

where k_s is the constraint stiffness and $\sum_{i=0}^{7}\|\mathbf{x_i}-\mathbf{x_B}\|_0$ is the rest length of this "volume spring". As in the tetrahedral case, it was not necessary to add damping forces with this constraint.

2.4.3 Alternative Formulations

The volume conservation constraints presented in Sections 2.4.2 and 2.4.1 are quite coarse, to say the least. They were considered sufficient at the time this work was published; however, we see two ways of improving them, perhaps at the cost of an increase in computation time.

First, we could rewrite our constraints using Witkin's method to design energy functions, formulated as positive definite functions on the model parameter space with zeroes at points satisfying the constraints, and to derive constraint forces from them (Witkin et al., 1987; Witkin, 1999a). But this solution will still have the drawbacks of penalty methods, i.e., constraints are not fulfilled precisely and lead to numerical instability due to stiff equations. Second, volume conservation could be enforced directly as a hard constraint, i.e., a constraint verified at any time during the simulation. There is a rich body of literature on the subject.

Promayon et al. (1996) propose a method for constraining physically-based deformable objects described by a set of mass points on their surface. All constraints that can be defined as a region in the position or velocity space, and that are differentiable, can be used in this method: constant volume constraints, fixed or moving position constraints, and velocity constraints. For each point, the computed position using all applied forces is projected to the nearest position in the region of parameter space that satisfies the constraint. However, this is a projection method that manipulates physical state variables (position, velocity) directly and we would prefer force-based constraint methods that are more appropriate for physically-based animation.

Platt and Barr (1988) present two methods that fulfill constraints exactly: reaction constraints and augmented Lagrangian constraints. Reaction constraints are based on projection method but manipulate forces only. A reaction constraint simply adds a force to cancel parts of the applied forces that will not maintain the constraint, instead of using displacements to trigger restoring forces that will fight with applied forces to meet the constraint criterion, as in penalty methods. Augmented Lagrangian constraints are a differential version of the method of multipliers from optimization theory. They require extra differential equations. Both approaches might be useful (see also Witkin and Welch, 1990).

Of course, we face the problem of competing constraints if we enforce volume conservation on an element-by-element basis using either soft or hard constraints. Handling this issue will require further research.

2.5 Results for Hexahedral and Tetrahedral Meshes

All the experiments presented in this section have been computed by setting point masses to the same value. Thus, objects sampled using tetrahedral meshes are generally heavier than those sampled using hexahedral meshes. Moreover, objects are slightly inhomogeneous in the former case, since mesh nodes are not evenly distributed. Better results would be obtained by computing the mass values according to the density of the simulated material and to the volume of the Voronoï region associated with each point mass. However, we found the results quite demonstrative as they are.

Numerical simulation of all experiments was achieved using Stoermer's explicit integration method (Press et al., 1992) with no adaptive time step, and therefore might be improved. In some experiments, viscous drag was used to limit numerical drift. The viscous drag force $\mathbf{f}$ for a point mass with velocity $\mathbf{v}$ is $\mathbf{f} = -k_{drag}\,\mathbf{v}$ where k_{drag} is the drag constant (Witkin, 1999b). We added gravity to simulate realistic load conditions for the model. The gravitational force $\mathbf{f}$ for a point mass with mass m is $\mathbf{f} = m\,\mathbf{g}$ where $\mathbf{g}$ is a constant vector (presumably pointing down) whose magnitude is the gravitational constant (Witkin, 1999b).

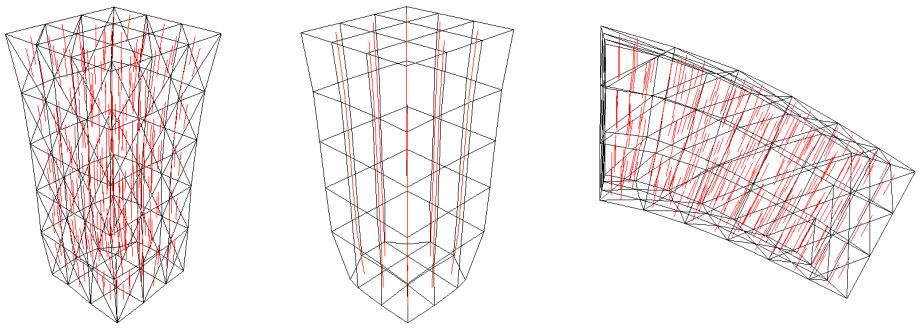
Each figure depicts outer mesh edges and one of the three axes of interest inside each elementary volume. In Fig. 2.11 this axis represents the orientation along which the material is the stiffest. Stiffness has been set to the same value in the two other directions.

Comparison with Mass-Spring Systems

The same experiments as in Fig. 2.5 are performed using our model instead of a classic mass-spring system (see Fig. 2.10). Here, one axis of interest is set to the vertical direction (the direction of application of gravity and pull forces) and the two others in horizontal directions. The same stiffness and damping values are used in each direction. The equilibrium states observed are the consequence of the interplay of forces opposing stretch and shear, and forces conserving volume of the material.

Controlling Anisotropy

A set of experiments with different anisotropic behaviors using the same tetrahedral mesh is presented in Fig. 2.11. Anisotropy is tuned by changing the stiffest direction in the material. All dynamic behaviors obtained correspond to what we intuitively expected when we designed these materials. Moreover, it is interesting to note that an



(a) (b) (c)

Figure 2.10: Experiments similar to those of Fig. 2.5, but computed with our model. As expected, we do not observe uncontrolled anisotropy in both the tetrahedral (a), and the hexahedral (b) meshes. All things being equal, with the same mesh and material parameters as in Fig. 2.5, our tetrahedral model is perfectly able to sustain flexion, as shown by its equilibrium state (c).

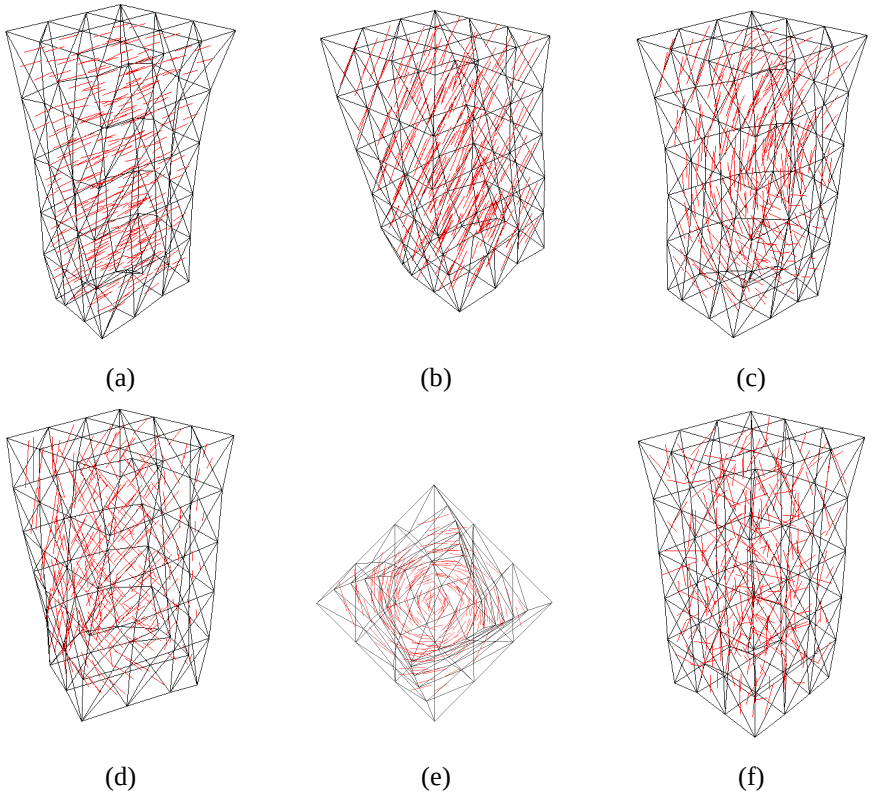
isotropic material can be modeled using a random orientation for the stiffest axis in each volume element.

Performance Issues

Our benchmarks are on an sgi O2 workstation with a MIPS R5000 CPU at 300 MHz with 512 MB of main memory. Experiments use tetrahedral and hexahedral meshes lying on a table under force of gravity. Other conditions are similar to those of volume conservation experiments (see caption of Fig. 2.9). Note that material stiffness strongly influences computation time since we use an explicit integration method.

The maximum number of springs per element varies between the different models. Since it gives a rough indication of the number of operations performed at each simulation step, it is also related to computational load. For a classical mass-spring system, a tetrahedral element has 6 structural springs along its edges, and an hexahedral element has 12 structural springs along its edges plus 4 shear springs along its main diagonals. Bending springs between hexahedral elements (Chen et al., 1998) were not used. This has to be compared with 3 axial springs, 3 angular springs and 4 volume springs (undamped), making approximately 10 springs for our tetrahedral element, and 3 axial springs, 3 angular springs and 8 volume springs, making 14 springs for our hexahedral element.

We can conclude from the results displayed in Table 2.1 that simulating anisotropic behavior and ensuring volume conservation are not very expensive in our model. These properties make it suitable for interactive applications. However, the cost of our method is directly related to the number of elements. Thus, unlike mass-spring systems, our benchmark experiment using the tetrahedral mesh is slower than the one using the hexahedral mesh.



(a) (b) (c)

(d) (e) (f)

Figure 2.11: Different anisotropic behaviors were obtained using the same tetrahedral mesh undergoing a downward pull at its bottom end while its top end is fixed. Anisotropy is tuned by changing the stiffest direction in the material. This direction is: horizontal (a), as a result, the material tends to get thinner and longer; diagonal (b), with angle of $\frac{\pi}{4}$, which constrains the material to bend in this manner; semicircular (c), as a C shape, which causes a snake-like undulation of the material; concentric helicoidal, side (d) and top view (e), the material successively twists and untwists upon itself; random (f), the material exhibits an isotropic behavior.

		Masses	Elements	Springs	Springs/Element	Time (in s)
Mass-Spring	Tetra	222	804	1175	1.461	0.129
	Hexa	216	125	1040	8.320	0.117
Our model	Tetra	222	804	≈ 8040	≈ 10	1.867
	Hexa	216	125	1750	14	0.427

Table 2.1: Benchmarks results for classical mass-spring system and our model with tetrahedral and hexahedral meshes. See explanations in the text concerning the estimated number of springs per element in our model. Time: time spent to compute one second of animation, with a time step of 0.01 s.

2.6 Validation for Tetrahedral Meshes

The results presented in Section 2.5 concerned the qualitative behavior of our model with respect to a mass-spring system. Now, we would like to answer quantitatively two questions in order to validate our model. First, are we able to simulate a wide range of stress-strain relationships using our model? It is necessary to have a versatile model for simulating nonlinear materials undergoing large deformations, i.e., more than 10% strain. Second, what is the behavior of our model when we change the sampling of the object simulated, i.e., the number of points and tetrahedra? It is important to maintain a consistent behavior across levels of detail for managing multiresolution in animation. This way, the user will not notice sudden changes when simulation quality or computation time constraints impose a transition from one level to another.

For the experiments in this Section, point masses were set in order to obtain a homogeneous material, as opposed to what has been done for the experiments in Section 2.5. In our model, one axis of interest is set to the vertical direction (the direction of application of gravity) and the two others in horizontal directions. Parameters are chosen identical along the 3 axes.

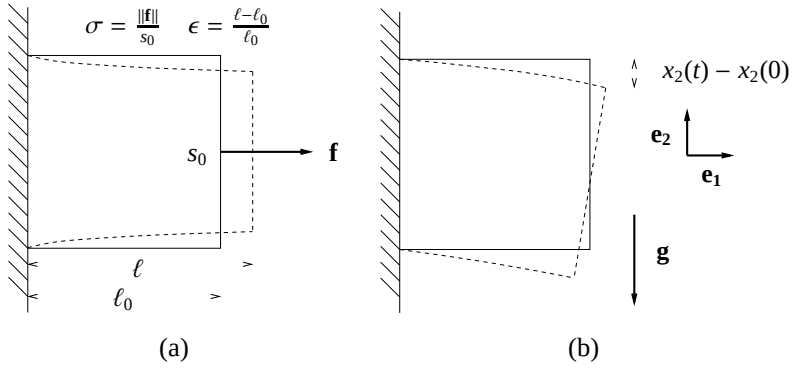


Figure 2.12: Validation experiments for stress-strain relationship (a), and multiresolution behavior (b). In (a), a force **f** is applied to one side of the cube (initial surface s_0); lengths before and after deformation (ℓ_0 and ℓ) are measured; Lagrangian stress σ and Lagrangian strain ϵ are computed in the direction of application of the force (see Section 2.6.1). In (b), a cube sustains flexion under force of gravity **g**; the cube oscillates and the position of one of its free corners is recorded (see Section 2.6.2). The cube drawn with solid lines represent the cube before deformation; the cube drawn with dashed lines represent the cube after deformation.

2.6.1 Stress-Strain Relationship

Here we use variants of the experiments conducted previously for mass-spring systems (Boux de Casson, 2000). A cube of homogeneous material composed of 346 points and 1350 tetrahedra has one of its sides fixed; we apply a constant force to the opposite side and then measure the strain of the cube at equilibrium in the direction of

application of the force. We repeat this measure by varying the intensity of the force (see Fig. 2.12a). This is equivalent to an uniaxial traction experiment conducted in continuum mechanics to determine stress-strain relationships of various materials.

We adopted Boux de Casson's spring model, which is different from the one used in Section 2.3.1. The spring forces $\mathbf{f_1}$ and $\mathbf{f_2}$ between a pair of intersection points P_1 and P_2 at positions $\mathbf{x_1}$ and $\mathbf{x_2}$ with velocities $\mathbf{v_1}$ and $\mathbf{v_2}$ are

$$\mathbf{f_1} = -\left[\,\varphi\left(d_{21}\right)\,\right]\frac{\mathbf{l_{21}}}{\|\mathbf{l_{21}}\|} \qquad \mathbf{f_2} = -\mathbf{f_1} \qquad \text{with} \quad d_{21} = \frac{\left(\,\|\mathbf{l_{21}}\| - r\,\right)}{r}$$

where $\mathbf{l_{21}} = \mathbf{x_1} - \mathbf{x_2}$, r is the rest length and $\varphi\left(d_{21}\right)$ can be

$$\text{linear:} \qquad \varphi\left(d_{21}\right) = k_{s_1} d_{21}$$

$$\text{quadratic:} \quad \varphi\left(d_{21}\right) = k_{s_2} d_{21}^2$$

$$\text{cubic:} \qquad \varphi\left(d_{21}\right) = k_{s_3} d_{21}^3 + k_{s_2} d_{21}^2 + k_{s_1} d_{21}$$

where k_{s_i} is the stiffness constant of the ith order term in $\varphi\left(d_{21}\right)$.

In the first experiment, we used linear and quadratic spring models to simulate the corresponding behavior. We fixed the stiffness parameters k_{s_1} and k_{s_2} to the same value. In the second experiment, we used a cubic polynomial spring model to simulate the nonlinear stress-strain relationship of a continuum mechanics model (Ohayon et al., 2001). To obtain the value of the parameters k_{s_i} we fitted a cubic polynomial to reference data points obtained with the continuum mechanics model. The `fit` command, from the `gnuplot` package (Williams et al., 2001), fits a user-defined function to a set of data points using an implementation of the nonlinear least-squares Marquardt-Levenberg algorithm.

Results for the first experiment are shown in Fig. 2.13a. As we expected, the stress-strain relationship exhibits a linear or quadratic behavior. Moreover, fit results displayed on Table 2.2 confirm that higher-order stiffness parameters are equivalent for both the linear and quadratic case ($a = 69.062$ versus $a = 72.752$). This indicates that properties at the spring level are conserved at the object level. Results for the second experiment are shown in Fig. 2.13b. We obtain a stress-strain relationship very similar to the one obtained with the continuum mechanics model, thus demonstrating the versatility of our model.

All these results are comparable to those obtained previously by Boux de Casson for mass-spring systems. However, it should be noted that we had to apply a scale factor to the parameters to obtain quantitatively similar stress-strain relationships. This is mainly due to the interplay of two phenomena. We know that equivalent stiffness k_s for two springs in parallel is such as $k_s = k_{s_1} + k_{s_2}$ and that for two springs in series is such as $\frac{1}{k_s} = \frac{1}{k_{s_1}} + \frac{1}{k_{s_2}}$. Since the parallel case seems preponderant in our model, the more elements you have, the stiffer the object becomes. But, because this has only a linear effect on the stress-strain relationship, it is easy to correct by scaling parameters. We

consider this "tweaking" process more usable than brute force optimization techniques previously used for mass-spring systems (Deussen et al., 1995; Louchet et al., 1995).

Fit Function	Parameters Value (in kPa)	Asymptotic Standard Error
$f(x) = ax$	$a = 69.062$	± 0.126 (0.183%)
$f(x) = ax^2 + bx$	$a = 72.752$	± 0.199 (0.273%)
	$b = 2.645$	± 0.094 (3.555%)

Table 2.2: Fit results with linear and quadratic polynomials on the data sets from our first experiment, displayed on Fig. 2.13a.

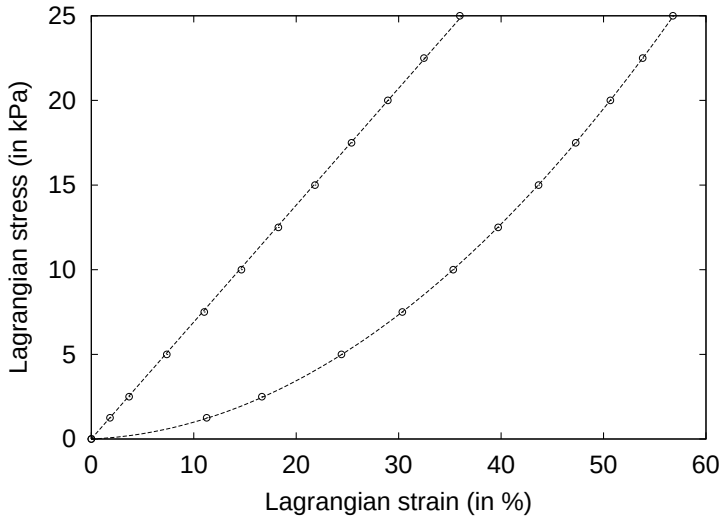
2.6.2 Multiresolution Behavior

We performed the experiments presented in this section using Debunne's software testbed, used for evaluating multiresolution behavior of various deformable models (Debunne, 2000; Debunne et al., 2001). The example studied has been described in detail elsewhere (Debunne, 2000) but we recall its main features here. A cube of elastic material (10 cm edge length, 1 kg mass) is attached by one of its faces to a vertical surface. At time zero the cube is released from its horizontal undeformed position, with no initial velocity. Then, we measure position of one of its free corners when the cube oscillates under the action of gravity (see Fig. 2.12b). Neither internal nor external dissipative forces are used, in order to easily compare the vibration modes obtained with the different resolutions.

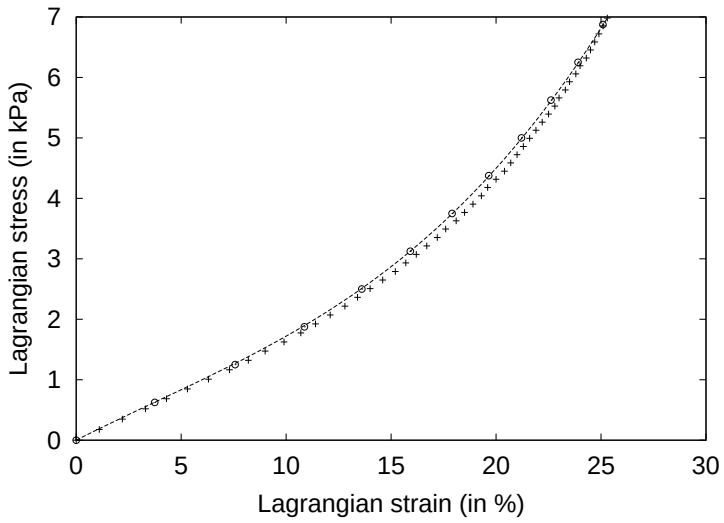
Numerical simulation was achieved using the Desbrun et al. (1999) integration method, which allows to double the integration time step, thus dividing by two the time needed to compute one second of animation. This method introduces an additional "numerical drag" effect but this was not considered an issue. Experiments are conducted for three resolution levels of the cube, with 27, 57 and 135 points (corresponding to 48, 122 and 448 tetrahedra, respectively). We used spring formulation from Section 2.3.1, but without damping term.

Results obtained with the mass-spring system are displayed on Fig. 2.14a. It clearly demonstrates that mass-spring systems fail to ensure the same frequency and amplitude of oscillations at different resolutions. This is comparable to Debunne's observations. Results obtained with our model are displayed on Fig. 2.14b. Oscillation curves are smooth and have nearly the same frequency for all resolutions (5.882, 5.769 and 6.122 Hz for 27, 57 and 135 points, respectively). For the first level, the variation in amplitude is comparable to the one observed for the mass-spring system: a main oscillation modulated by a lower frequency oscillation (only a half period of this "carrier" oscillation is shown here).

The decrease of amplitude between levels in our model is due to the increased damping effect of the integration method. It is purposely more pronounced than for



(a)



(b)

Figure 2.13: Stress-strain relationships obtained with our model: linear and quadratic behaviors are simulated (a); cubic behavior has been obtained using a cubic polynomial interpolation of reference data points obtained with a continuum mechanics model (b). In abscissa, Lagrangian strain (in %); in ordinate, Lagrangian stress (in kPa). Lagrangian strain is the relative extension in the considered direction; Lagrangian stress is the force per unit of undeformed surface. Circles (◯) are data points obtained with our experiments; crosses (+) are reference data points obtained with the continuum mechanics model; dashed lines are polynomials fitted to the experimental data sets.

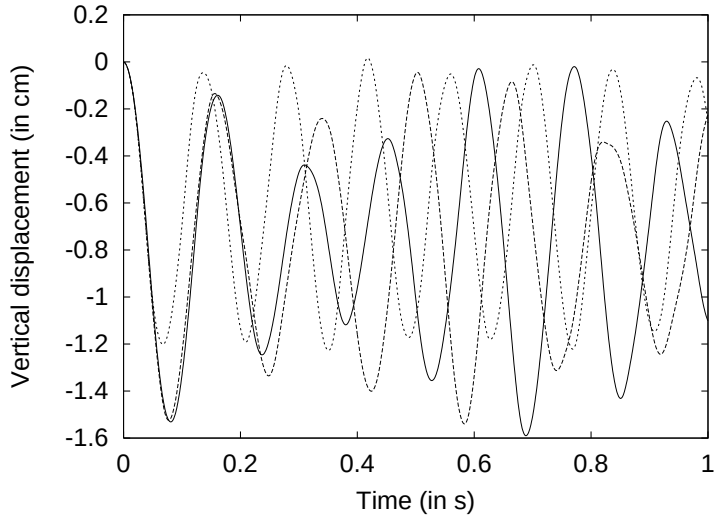
the mass-spring system to avoid numerical instability problems, because our model has been set much stiffer than the mass-spring system in order to obtain the same amplitude of oscillations. However, results for our model show multiresolution behavior that compares favorably with Debunne's approaches, since the frequency of oscillations is similar at different resolutions.

2.7 Conclusion and Future Work

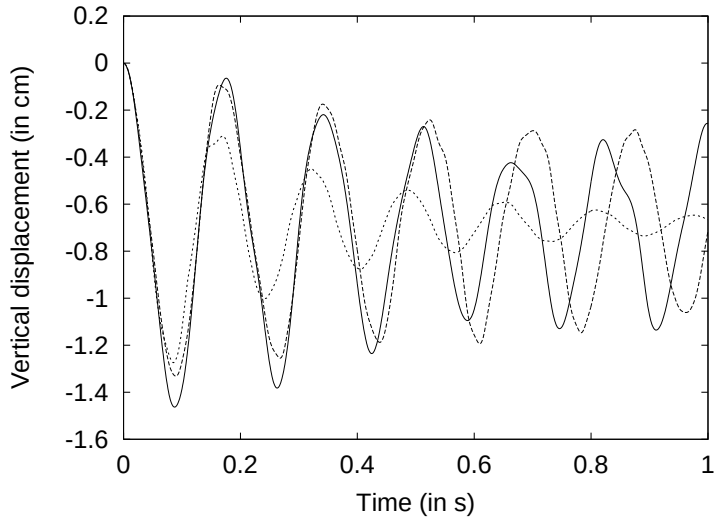
We have presented an alternative formulation for mass-spring systems, where anisotropy of a deformable volume is specified independently from the geometry of the underlying mesh. There is no special requirement for the mesh, which may be built from either tetrahedral or hexahedral elements. Moreover, a method for generating quasi-constant volume deformations is provided. The new model stays very close to mass-springs systems, since it is as easy to implement and still efficient in computation time. It also benefits from the ability of mass-spring systems to animate deformations without a priori hypothesis of geometrical or physical linearity.

Future work includes possible generalization to surface materials, such as cloth, which exhibits interesting anisotropic behavior in garments, e.g, the wrists of a sweat shirt. To do so, extra parameters controlling bending will have to be added to the current volume model. Otherwise, interesting possibilities could arise by combining different volume element types to obtain an hybrid mesh which better approximates the shape of the object; or by using elements of different orders (linear versus quadratic interpolation, etc.), in the same mesh.

Finally, we envision that our model would lend itself easily for creating "animated sketches" or "interactive illustrations" (Beall et al., 1997), exploring a new way of specifying animation parameters. In this application, the user could define by drawing both the shape of the object and its anisotropy, described for example by fiber directions. The shape and anisotropy information would be processed separately, then used to animate the sketch and manipulate it at user's will.



(a)



(b)

Figure 2.14: Multiresolution experiments: a cube, fixed on one side to a vertical surface, oscillates under force of gravity, with a mass-spring system (a) or our model (b). In abscissa, time (in s); in ordinate, vertical displacement relative to initial position of one of the free corners of the cube (in cm). The cube is sampled at different resolutions, with 27, 57 or 135 points (solid line, dashed line, dotted line, respectively).

Chapter 3

Drawing for Illustration and Annotation in 3D

3.1 Introduction

Drawing has long been an intuitive way to communicate complexity in a comprehensible and effective manner, due to visual abstraction. Compared to a photograph of a real object, extraneous details can be omitted, and thus attention can be focused on relevant features. The impression a drawing produces on a viewer is very different from the one a solid model produces: strokes mark the presence of a surface or a contour, but they can be "heavier" or "lighter", giving an indication of uncertainty where needed. The viewer's imagination is immediately engaged.

This kind of communication is useful in educational applications such as teaching, but also in the early stages of design, because drawing a sketch is much faster than creating a 3D model, and definitely more convenient to express ideas. However, the obvious drawback of 2D sketches is their limitation to a single viewpoint. The user cannot move around the object drawn, nor view it from different angles. Adding the ability to render a single sketch from multiple viewpoints has evident advantages, since the work of the artist is significantly reduced. As an example, consider Figure 3.1, which shows two views of a single rough landscaping sketch, generated by our system. It is imperative that such a system be interactive, since a slow, non-interactive system would interfere with the natural artistic creation process.

The aim of this work is to provide a system that enhances classical sketching with 3D capabilities. Here, as in traditional 2D drawing systems, the user draws strokes that may represent either surface silhouettes or 1D features. These strokes may either belong to a single object or to several different ones, embedded in a complex scene. Both open and closed strokes may be used. As in 2D drawing, the user can draw several neighboring strokes to accentuate a given contour. The main difference is that the viewpoint can be changed while drawing, thus creating a 3D sketch.

Figure 3.1: A single landscaping sketch, which can also be seen as an annotation of an existing 3D model; the two different views are automatically generated by our system.

Overview

The central idea of our approach is to represent strokes in 3D space, thus promoting the idea of a stroke to a full-fledged 3D entity. Even in 3D, we think that strokes are an excellent way to indicate the presence of a surface silhouette: several neighboring strokes reinforce the presence of a surface in the viewer's mind, while attenuated strokes may indicate imprecise contours or even hidden parts.

Finding surface properties of objects from their silhouette is a classic hard problem in computer vision. The algorithms presented here do not address this issue, since our goal is to develop a drawing system rather than to perform geometric reconstruction. As a consequence, we develop approximate solutions that are appropriate in the context of interactive drawing and sketching.

To enable the user to view stroke-based sketches from multiple viewpoints, we interpret 2D silhouette strokes as curves, and use a curvature estimation scheme to infer a local surface around the original stroke. This mechanism permits efficient stroke-based rendering of the silhouette from multiple viewpoints. In addition to stroke deformations, this includes variation of intensity according to the viewing angle, since the precision of the inferred local surface decreases when we move away from the initial viewpoint. It also includes relative stroke occlusion, and additive blending of neighboring strokes in the image. Apart from silhouette strokes, our system also provides line strokes that represent 1D elements. These have the ability to remain at a fixed position in space while still being occluded by surfaces inferred using silhouette strokes. They can be used to add 1D details to the sketches, such as the arrow symbols in the example of annotation (see Fig. 3.21).

Because strokes have to be positioned in space, we present an interface for 3D stroke input. The user always draws on a 2D plane which is embedded in space. This plane is most often the screen plane, selected by changing the viewpoint. The depth location of this plane can be controlled either explicitly via the user interface

or implicitly by drawing onto an existing object. The user may also draw strokes that are not in the screen plane, but that join two separate objects. The combination of fast local surface reconstruction and graphics hardware rendering with OpenGL results in truly interactive updates when using our system.

Finally, we show that our method can be applied to artistic illustration as well as annotation of existing 3D scenes, e.g., for rough landscaping or educational purposes. An existing 3D object can also be used as a guide to allow the design of more involved objects, e.g., using a model mannequin to create 3D sketches for clothing design.

Part of this work has been previously published in the journal *Computer Graphics Forum*, Eurographics conference issue (Bourguignon et al., 2001).

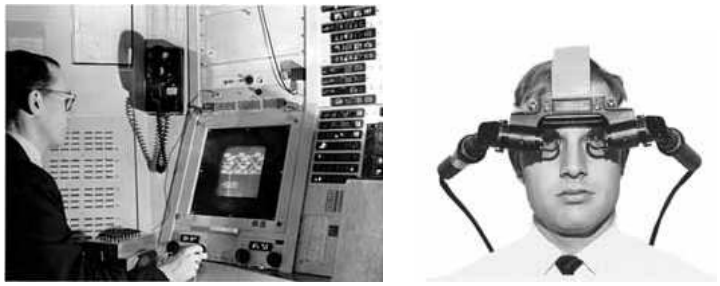
3.2 Previous Work

Our work is a natural continuation of 3D drawing or sketching tools which have been developed in computer graphics over the last few years. Before giving an overview of the related papers, we would like to recall the pioneering work of Sutherland in this area, forty years ago (Sun microsystems, 2002). They shaped the two main trends in 3D drawing interfaces we still see today.

In 1963, using the high-end TX-2 computer, Sutherland invented the first interactive computer graphics application, which he dubbed Sketchpad (see Fig. 3.2a). The TX-2 computer, at the Lincoln laboratory of Massachusetts Institute of Technology (MIT), was one of the few computers of the day that could run on line instead of only crunching batch jobs. It had huge memory capacity, magnetic tape storage and various input and output devices; among them, two extremely important pieces of equipment: a lightpen and a nine-inch cathode-ray tube (CRT) display. Using this simple but powerful interface and the Sketchpad program, precise engineering drawings could be created and manipulated. Many concepts that are now common in GUI were defined by this revolutionary software, e.g., rubber-banding of lines, zoom in and out, automatic beautification of lines, corners and joints, etc.

A few years later, in 1968, Sutherland presented the first computer head-mounted display (see Fig. 3.2b). This work was inspired by early experiments, such as a remote perception project at Bell Helicopter Company, where HMDs were used to control distant cameras. Replacing the real world images by computer-generated images let the user enter the first virtual reality (VR) environment, composed of a single wireframe room with one door and three windows in each of the cardinal directions.

Nowadays researchers have realized the importance of providing usable tools for the initial phase of design, beyond traditional 3D modeling. These tools have taken the form of 3D drawing or sketching systems, using direct 3D input, where the user draws in 3D and the computer gives him the necessary visual feedback, or 2D stroke interfaces, where the user draws in 2D and the computer infers 3D strokes or a 3D object.



(a) (b)

Figure 3.2: Ivan Sutherland's pioneering work. In (a), Ivan Sutherland at the console of the TX-2 using Sketchpad, MIT, 1963 (Sutherland, 1963). In (b), Quint Foster wearing the HMD, Harvard University, circa 1967 (Sutherland, 1968).

3.2.1 3D-to-3D Drawing Systems

A straightforward way of drawing in 3D is to use 3D input devices to draw 3D strokes. Pablo Picasso's light drawings, made with a bright light source and a camera set for long exposure could be seen as drawings in three dimensions (see Fig. 3.3a). With computers, HMDs and six-degrees-of-freedom (6-dof) sensors, full capture of user gestures is possible with immediate and persistent visual feedback (in the case of Picasso, as for most artists, the drawing was in his head even before he started to draw, visual feedback was thus superfluous).

Papers Overview

The systems presented can be classified in two categories according to their use of 3D strokes. Some systems take them as their base primitives to build shapes, and others, generalizing the vector drawing metaphor to three dimensions, use them as gestural commands to create higher level shapes.

Sachs et al. (1991) describe 3-Draw, a CAD system for the initial stages of conceptual design. Three-dimensional freeform shapes are designed directly in 3D using a pair of hand-held, 6-dof sensors. The authors consider that the limitations of the traditional CAD interface come from the way the user communicates information using 2D input devices: more time is spent on deciding how to draw and where to draw a primitive rather than on drawing it. As opposed to this, they propose a natural way of creating and manipulating objects. One hand is in charge of the position of the object in the virtual world; the other hand handles a pen to input precise editing commands; the body position is similar to the one of a painter holding a palette and a brush. This interface takes advantage of the kinesthetic feedback from the hands: the user knows precisely the relative position of his hands without much effort. Designing a shape is done in four steps: first, curves are sketched directly in 3D, they represent either silhouette, reflection lines or "skeletal" features of the 3D model; second, these curves are edited using deformations, i.e., stretching, cutting, bending and erasing; third, surfaces

are fitted to groups of linked curves; fourth, surfaces are edited using deformations to add details. The authors report implementation of the first and second steps only.

With 3DM, Butterworth et al. (1992) propose a 3D surface modeling program that uses a HMD and 3D tracking devices. Manipulation techniques from CAD and 2D vector drawing programs are adapted to a 3D setting. Surface creation is done using a triangle tool for single triangle and triangle strip generation, and an extrusion tool for complex surfaces: a 3D polyline is dragged by the user along an arbitrary extrusion path, undergoing rotations and translations. Standard surface shapes such as box, sphere, or cylinder can be obtained with the appropriate tool and interactively resized. Finally, classical editing tools in 2D programs are available here in 3D: mark-move tool, undo-redo stack, etc.

HoloSketch (Deering, 1995) is a highly accurate 3D object creation and manipulation tool that uses head-tracked stereo shutter glasses and a desktop CRT display. A 3D "wand" manipulator allows the creation of 3D drawings in front of the user, edit them, and even animate them. The system is controlled through the use of a 3D multilevel "fade-up" pie menu that allows switching from one mode to another. Several types of drawing primitives are supported: rectangular solids, spheres, ellipsoids, cylinders, cones, rings, 3D text, lines, polylines, etc. To use them, the user selects the desired primitive as the current drawing mode (through the menu), then depresses the left wand button to create an instance of the primitive. Common 2D drawing operations are translated to 3D: selection, primitive editing operations (movement, grouping, scaling, modifying objects attributes), and other general operations (cut, copy, undo, etc.). Finally, simple animating operations (rotation about an axis, looping by temporal grouping, color oscillation, flight path, etc.) extend even more the possibilities offered by HoloSketch. It should be noted that an informal user study gives precious information concerning ergonomics of the system, details that most of the time are absent in papers. The author mentions that maintaining the body position for long periods of time was not a problem: the user is seated, holding his hands in the air with his elbows resting on the desk. However, it appeared quite difficult to make fine adjustments in this position, thus hindering the user performance.

The 3D Sketch system of Han and Medioni (1997) is not exactly a 3D drawing system but it uses the drawing metaphor for casually digitizing an existing object. The system consists of three distinct modules. In the Prototyper module, the user sketches a few strokes on the surface of the object using the digitizing stylus as a 3D pen. A rough model is generated, as if made of lumps of clay. In the Refiner module, the model surface adapts to the new strokes added by the user, matching the object distinct features, e.g., edges and corners. In the Autotracer module, the system infers smooth surfaces from the user inaccurate and discontinuous stroke input.

Finally, Keefe et al. (2001) have built a fully immersive Cave environment dedicated to 3D painting, i.e., the creation of 3D scenes by layering 3D brush strokes in space (see Fig. 3.3b). The user wears shutter glasses that allow him to see both the real and virtual worlds and has the choice of a large set of stroke types such as line, ribbon, tube, dripping, splat, extrusion, etc. Strokes are created by moving a paint brush prop

around in the Cave, tracked in position and orientation while the single button on the brush is depressed; sampled 3D points are used to define the stroke geometry. Several stroke types interact with Cave walls as if the walls were part of the virtual world. The user assigns a stroke type to the brush by literally dipping the physical brush into a cup that "contains" the desired stroke. Colors are picked using a 3D color picker using hue, lightness and saturation (HLS) color space. Two-handed interaction is possible by wearing a tracked pinch glove on the non-dominant hand to gain access to the color picker, etc. Navigation in the 3D scene is performed using a tracked pinch glove for small translations or a foot pedal for large ones; rotation and scaling widgets are also available. Interestingly, the authors note that programming expressive effects into a stroke was a mistake since users naturally produce expressive strokes when given a simple stroke with sufficient control and visual feedback. Even if the interface was found easy to understand and use, users complained about the lack of accuracy when painting details. Moreover, the painting metaphor was considered poorer than real paint since it doesn't allow to move, mix or scrape off strokes.

(a) (b)

Figure 3.3: 3D-to-3D drawing systems. In (a), Pablo Picasso's light drawing, made with a bright light source and a camera set for long exposure. In (b), *Florentine Vineyard* by Daniel Keefe, a 3D drawing obtained with the system of Keefe et al. (2001).

Discussion

We see three main concerns with the direct 3D input approach: the first and foremost is system ergonomics, the second is modeling limitation, the third is equipment availability.

Sachs et al. claim that the use of both hands makes the modeling task more intuitive and efficient. However, even without performing detailed ergonomics studies, it seems obvious that user body position will be tiring since he is seated, holding a stylus and a palette in the air, with the elbow resting on the desk. Even worse, performing precise movements without any physical feedback from real world objects, e.g., a sheet of

paper on a drawing board or a sculpting material, is very difficult and requires full-body gestural skills closer to those of a dancer than those of a designer or a painter, as Keefe et al. emphasized. The precision problems reported by Deering and Keefe et al. are in fact not due to the hardware-limited resolution of their system but to human intrinsic capabilities.

Are VR techniques adapted to design in the long run? Headache is common for people working with VR glasses when the framerate is too low; and muscular fatigue awaits anybody standing with arms extended, trying to perform precise gestures. However, detailed studies exist on these problems (Pausch and Crea, 1992; Bolas, 1994; Felger, 1995; Stanney and Kennedy, 1997) and may help prevent some of them.

Three-dimensional drawing, in the exact definition, is the production of lines in three dimensions. This is confusing because these strokes are not equivalent to strokes in traditional drawing. In the 3D case, it is a space curve, with absolutely no relationship to a 3D object; thus the stroke is view-independent. In the 2D case, it is most of the time the projection on the image plane of the silhouette, or another 3D curve attached to a 3D object; thus the stroke is view-dependent. Therefore, the representation of 3D shapes using lines is much more simple in 2D (only their projection is depicted) than in 3D. Imagine the number of lines you will need to describe approximatively the surface of a 3D object, as Han and Medioni do! To work around this modeling problem, the authors allow creation of higher-level primitives with a line flavor, such as strips, tubes, etc., or consider lines as support for defining surfaces. In fact, the best solution is to input surfaces directly (Schkolne et al., 2001), but this is more sculpting than drawing.

Finally, all these systems require special equipment (VR glasses, tracking sensors, etc.) and often cannot be used on traditional workstations. The cost varies a lot from a simple system using only a pair of 6-dof trackers to an entire Cave environment with glasses, trackers and gloves; but in any case it will be higher than the price of a personal computer with a graphics board. Poor man's VR systems (Guckenberger and Stanney, 1995) might be an alternative, but we will instead focus on approaches which do not require specific additional equipment and which are more practical than drawing directly in 3D.

3.2.2 2D-to-3D Drawing Systems

A large body of literature is devoted to 3D drawing systems that use traditional 2D strokes as input. As opposed to direct 3D approaches, these systems don't require any special computer equipment beyond classical 2D input devices: a mouse or better, a tablet. We will classify the systems in two categories according to the final output: either 3D strokes or a 3D object (typically, a closed manifold surface).

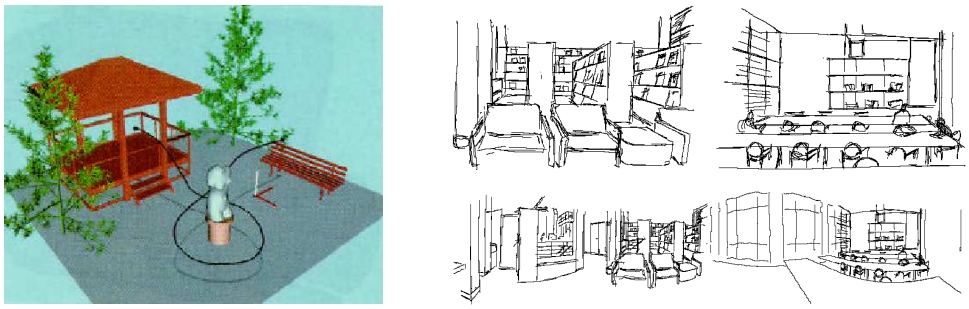
Papers Overview

Some systems use direct 2D drawing, i.e., 2D strokes are transformed into space curves using additional information or simply assuming a common projection surface. Most of the time, the resulting curves behave as 1D objects in the 3D world (limited parallax effects and light reflection properties).

Cohen et al. (1999) present a system to create non planar 3D curves with 2D input. Instead of editing the curve from several viewpoints, as in CAD systems, the user can specify the curve shape from a single viewpoint by successively drawing its screen plane projection (first stroke) and its approximate shadow on the floor plane (second stroke). The definition of the desired space curve is done in four steps. First, the second stroke is projected on the floor plane. The resulting shadow space curve is in fact the projection of the desired space curve along a projection vector, onto the floor plane. Second, this shadow curve is extruded along the projection vector, to obtain a shadow surface. The desired space curve is the projection of the first stroke on this possibly layered surface. Third, each point of the first stroke is assigned a projection layer on the shadow surface. It is not always possible to project the first curve on the shadow surface and to obtain a continuous space curve. Fourth, the first stroke is extruded along the camera's view vector and each part of this surface is intersected with the corresponding layer of the shadow surface. The curve obtained can be refined by overdrawing both input strokes; two drawing modes, projection or shadow, allow to distinguish between editing operations. This approach takes advantage of traditional artists' drawing skills. It can be useful, for example, for describing camera paths (see Fig. 3.4a), but seems potentially unintuitive for drawing shapes.

Tolba et al. (1999) propose a new drawing paradigm based on projective points. Each stroke drawn by the user in the image plane is a collection of image points. These points are projected on the surface of a unit sphere centered at the viewpoint, to get projective points. Rotation and zooming around the viewpoint are obtained by generating new reprojections of the projective points on the image plane. The user has the impression of being immersed in a three-dimensional space described by the drawing, but without parallax effects due to changes in viewing position (see Fig. 3.4b). Vanishing lines and projective grids are provided to help the user in creating accurate drawings in perspective and scale. The user interface also has tools for camera control. Importing a traditional sketch drawn on paper, or even a photograph, is possible using its perspective (vanishing points, etc.) and scale information. The layering of several drawings, or of drawings and photographs, allows easy comparison of different designs. This system seems naturally directed towards architectural applications where the use of perspective drawings is a common practice for representing buildings and their environment. The authors have now extended their system to enable modeling of extrusion surfaces, and to handle shading and projected shadows from infinite light sources (Tolba et al., 2001).

Disney's Deep Canvas (Daniels, 1999) is a painterly renderer that proceeds in three steps. First, 3D models of the objects of the scene are created using a traditional



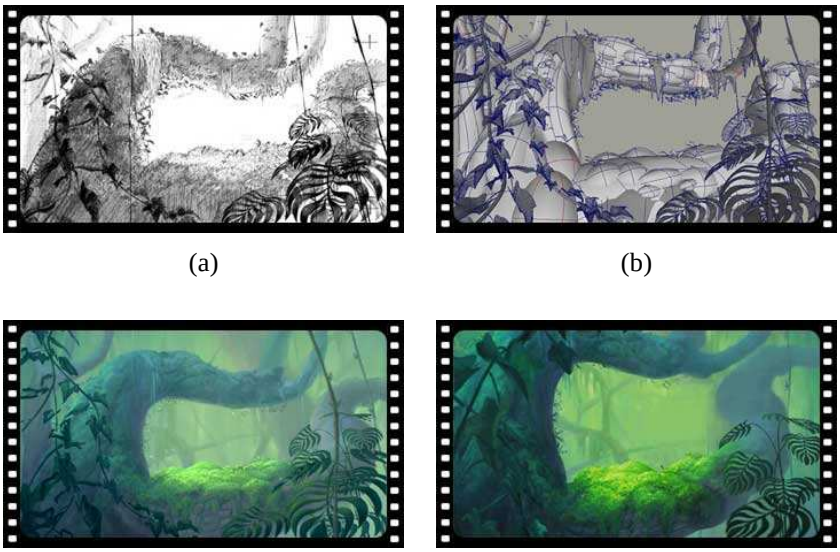
(a) (b)

Figure 3.4: Direct 2D drawing. In (a), results from Cohen et al. (1999). The user has sketched a camera path through the virtual environment. The curve was created from the current viewpoint. In (b), results from Tolba et al. (1999). Two of the four drawings used to create the library interior panorama (top); this panorama shown as an unrolled cylinder (bottom).

modeling package. Second, an artist "paints" these models, the computer recording the sequence and position of the strokes in 3D space (this position is obtained by projecting strokes on the objects of the scene). Third, from a different viewpoint, strokes are redrawn by the computer, the artist filling in holes with additional strokes as needed (see Fig. 3.5).

Paint Effects (Alias|wavefront, 2002c) is a toolbox of the Maya software for painting in 2D and 3D (see Fig. 3.6a). On canvas, it offers a full range of simulated traditional tools, ranging from airbrushes to watercolors, but also brushes that automatically create pictures of objects, such as trees or flowers. Thus, complex images can be obtained with only a few brush strokes. In a three-dimensional scene, strokes are curves positioned on planes or surfaces, and full three-dimensional entities, such as trees or flowers, are instantiated along stroke paths. These entities can be animated, e.g., to give the impression of wind blowing in tree foliage, and are rendered after 3D geometry. Strokes can be painted either on the grid plane (Maya's "floor" plane), on the view plane (parallel to image plane), or directly on other objects (see Fig. 3.6b, top). When painting on objects, two modes are available for setting stroke depth: either depth value of the stroke is equal to the depth value of the object surface (mode Paint At Depth off, the default), or depth value of the stroke is fixed at the first depth value determined when the brush is clicked (mode Paint At Depth on) (see Fig. 3.6b, bottom). In fact, Maya's Paint Effects can be considered as a painting interface for positioning three-dimensional objects. The objects themselves are not created by the painting process and must be defined previously.

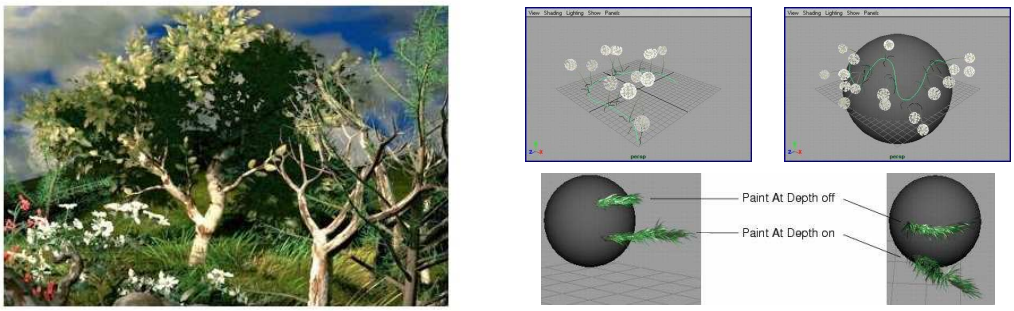
Another family of systems infers 3D models from 2D drawings and a set of constraints. These constraints come from the interpretation of perspective and axonometric drawings or the definition of gestural interfaces. Probably the first constraint-based modeling system was Sutherland's Sketchpad, and it can be considered as the inspiration for the numerous research efforts made in the past forty years. In fact, the problem of reconstructing a 3D object from its 2D projections, and the related 3D ob-



(a) (b)

(c) (d)

Figure 3.5: Disney's Deep Canvas (Daniels, 1999). A pencil sketch of a background scene from the *Tarzan* animated film (a); a simple 3D model of the main objects in the scene (b); the scene painterly rendered by an artist (c); the scene painterly rendered by the computer from a different viewpoint (d).



(a) (b)

Figure 3.6: Alias|wavefront's Maya Paint Effects. In (a), picture of a 3D scene by Duncan Brinsmead. In (b), interface examples. From left to right and top to bottom: painting with a "dandelion brush" on the grid plane; painting with the same brush on 3D geometry (a sphere); two possible modes (Paint At Depth on and off) for defining stroke depth relatively to an object.

ject recognition problem, have implications far beyond 3D modeling systems: they are important research areas in computer vision and artificial intelligence. We will focus on computer graphics papers but Wang and Grinstein (1993) present a complete taxonomy of 3D object reconstruction from 2D projection line drawings algorithms, classified according to the number of input views, the degree of automatism and the data structures employed.

The Viking system (Pugh, 1992) is based on interactive interpretation of polyhedral object drawing. Each time the user draws a new edge, a new three-dimensional object description is generated by the system, consistent with both the drawing and a set of geometric constraints. These constraints are either implicitly derived from the drawing or explicitly specified by the user (see Fig. 3.7). To draw in 3D, i.e., to define the position of each vertex in three dimensions, three mechanisms are provided. First, two-views geometry: straightforwardly, the user positions the vertex in two different views, as in traditional CAD systems. Second, "preferred directions", i.e., 3D vectors, either user-defined or automatically generated according to the context: they help the user in drawing a new edge endpoint by projecting it onto the closest line supported by a preferred direction and passing through the edge origin. Third, cutting planes, i.e., planes defined in object space, also useful for visualizing the three-dimensional structure of the object: the user positions the vertex by moving it parallel to the cutting plane or to the cutting plane normal. Once a new edge has been added to the drawing, the sketch interpretation algorithm generates a new object description by finding a new surface topology (using an extension of Huffman-Clowes line labeling scheme) and by solving for a new geometry that satisfies the constraints.

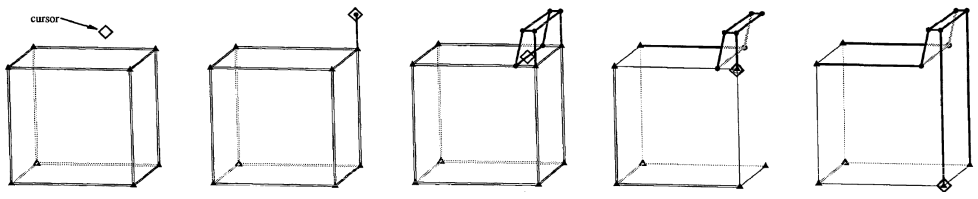


Figure 3.7: Results from Pugh (1992). From left to right: a three-dimensional object is inferred by the Viking system using geometric constraints, either implicitly derived from the drawing or explicitly specified by the user, e.g., hidden edge or redundant edge identification.

Akeo et al. (1994) describe a system that uses cross-section lines on a designer's drawing to generate automatically a three-dimensional model of the object (see Fig. 3.8). The input sketch is analyzed as follows. First, the lines are extracted from the drawing using image processing techniques. Then, three-points perspective information associated with shape cross-sections is used to infer relative position of the lines in three dimensions. Finally, closed loops are detected to create B-spline surfaces. The authors description is not very detailed but they stress the problem of system sensitivity to sketch inaccuracies, e.g., inconsistent vanishing points and varying line widths. The user provides missing data when automatic sketch processing fails.

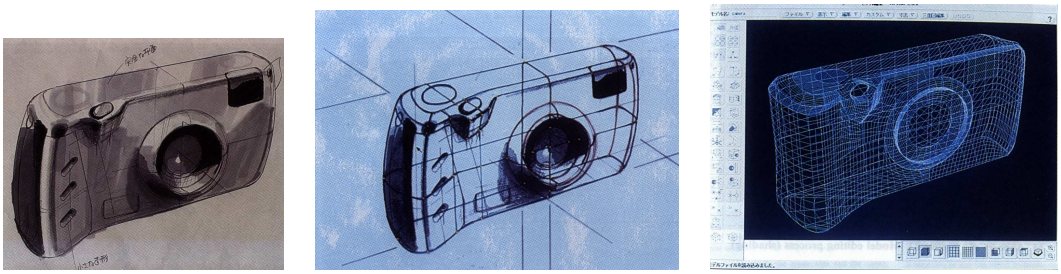


Figure 3.8: Results from Akeo et al. (1994). From left to right: idea sketch, sketch augmented with shape cross-section lines, 3D model editing interface.

The IDeS (intuitive design) system (Branco et al., 1994) combines sketch input with common features of solid modelers, such as constructive solid geometry (CSG) operators. The user can perform four different tasks with the system. First, sketching a 3D model, as in a classic 2D drawing program: the object must be drawn without hidden lines and in "general position", i.e., a position avoiding alignment between edges and vertices. Each time the user draws a line, junctions are analyzed and classified. A junction dictionary stores information on each junction type that will be used by the reconstruction. When the user has finished, the system attempts a reconstruction in two steps: compute the fully and partially visible faces; infer the hidden faces of the model. Second, using a modeling tool: basic shapes such as extruded solids can be constructed directly with the appropriate tool. If information is missing to apply the modeling operation, the system will wait for the user to provide it by drawing. Third, editing a 3D model: various editing operations are available, e.g., direct drawing over the surface of the model ("gluing"). Fourth, "explaining" a sketch to the system: the "is a" operator allows to distinguish between 2D and 3D models, such as a circle and a sphere, or to identify regular shape from imprecise input, such as a straight line segment from a freehand line.

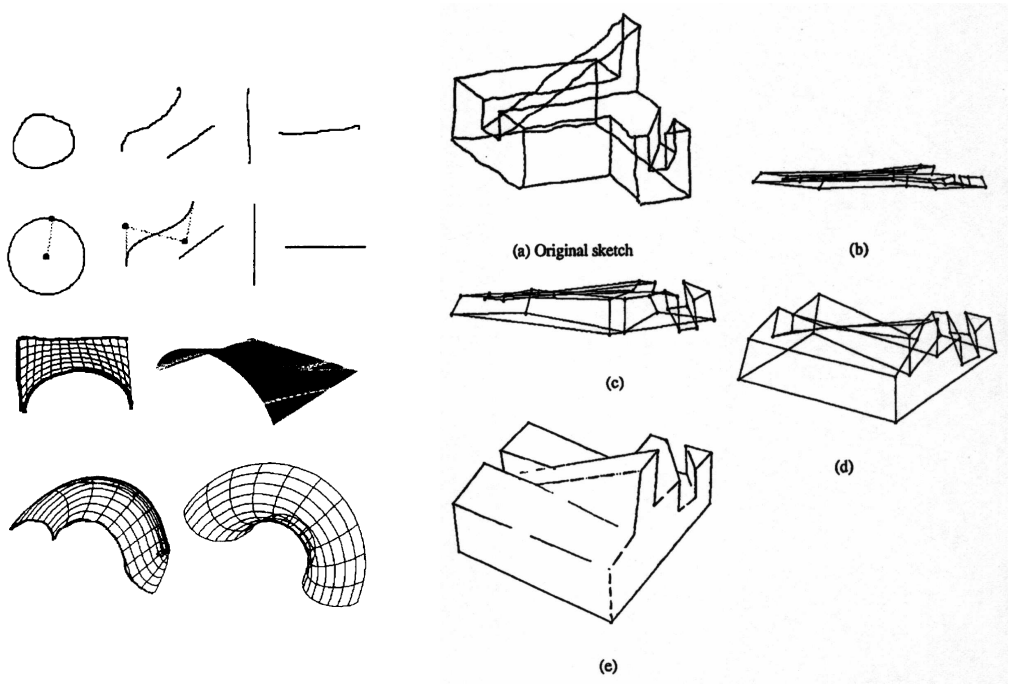
Eggli et al. (1995, 1997) present a 2D and 3D modeling tool that takes simple pen strokes as input. A graph-based constraint solver is used to establish geometrical relationships and to maintain them when objects are manipulated. Two-dimensional shapes, such as line, circle, arc or B-spline curve, and geometrical relationships, such as right angle, tangency, symmetry and parallelism are interpreted automatically from the strokes. This information is used to beautify the drawing and establish constraints (see Fig. 3.9a, top). Since inferring a 3D object from an arbitrary 2D input is impossible in the general case, specific drawing techniques that have an unambiguous interpretation in 3D are used. Extrusion surfaces are generated by sweeping a 2D profile along a straight line; ruled surfaces are defined between two curves; sweep surfaces are created by sweeping a cross-section along a curve; revolution surfaces are determined using two approximately symmetric silhouette lines (see Fig. 3.9a, bottom). The user can also draw lines on faces of existing objects. Tolerance in strokes interpretation is necessary to cope with inexact input. However, if interpretation does

not correspond to user intent, he can easily edit the model with gestures, e.g., to move control points. Soft constraints are introduced to achieve more predictable behavior when underconstrained drawings are manipulated.

Lipson and Shpitalni (1996) describe an optimization-based algorithm for reconstructing a 3D model from a freehand drawing of a 3D object (see Fig. 3.9b). The line drawing is assumed to represent the parallel projection of a general wireframe object (not necessarily manifold, containing flat or cylindrical faces), from an arbitrary viewpoint. Since an infinite number of possible objects can correspond to this projection, implicit information must be extracted from the drawing in order to reconstruct the most probable object. As a preprocessing stage, the 2D sketch is transformed into a 2D line-and-junction graph that assumes a one-to-one relationship between lines and projected edges of the object, and equivalently, junctions and projected vertices of the object. The reconstruction process progressively extracts the spatial information present in the edge-and-vertex graph from three sources: image regularities, i.e., geometrical relationships between entities or groups of entities, face topology, and statistical configuration of entities. Once spatial information is identified and formulated, associated 3D configurations are explored. This process is tolerant to inaccurate vertex positioning and missing entities because it does not rely on exact solution of equations, as previous approaches. In addition, the reconstruction is invariant with respect to small variations in the initial drawing. In fact, it is this fault tolerance that allows to reconstruct the probable object. The authors report reconstruction times between one second and half an hour, according to the complexity of the input drawing. Distorted 3D models are generated when the system fails to distinguish between more or less important sketch inaccuracies. Moreover, objects with curved faces are more difficult to reconstruct correctly since the majority of image regularities concerns only straight line segments.

The SKETCH system (Zeleznik et al., 1996) is a solid modeling tool for initial design, with a purely gestural interface and non-photorealistic rendering (see Fig. 3.10a). A three-button mouse input device is used for specifying operations directly in the 3D scene rather than for menu selection. The desired command is inferred by recognizing two types of gestural elements: strokes and “interactors”. Strokes are sets of sample points on the image plane, made by pressing the first mouse button. They are most of the time axis-aligned with the projection of one of the three main axes. There exists five classes of strokes: dot, axis-aligned line, non-axis aligned line, freehand curve, and freehand curve drawn on object surface. Interactors are made by pressing the second mouse button. There exists two classes of interactors: “click” and “click-and-drag”. Direct manipulation of camera parameters is possible with the third mouse button: pan, zoom, rotate, focus, and “select rendering” are available modes. Sequences of gestural elements result in four main classes of actions: creating geometry, placing geometry, editing, and grouping.

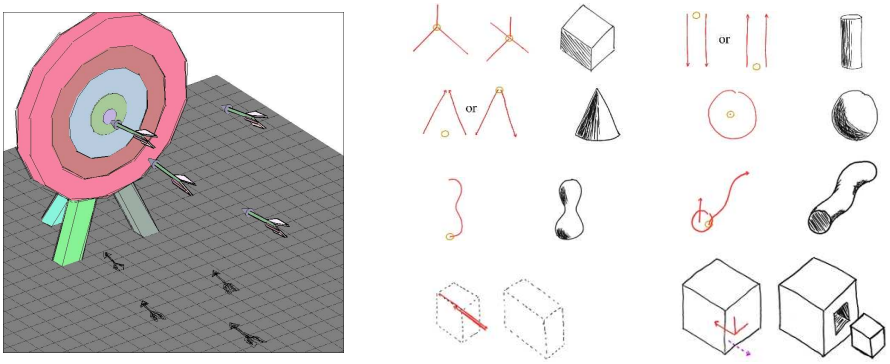
Creating geometry is made using strokes that instantiate primitives such as cuboid, cone, cylinder, sphere, revolution surface, prism, extrusion surface, sweep surface and superquadrics. Primitives are described with an “ideographic language” using visual



(a) (b)

Figure 3.9: Results from Eggli et al. (1995) and Lipson and Shpitalni (1996). In (a), results from Eggli et al. (1995). From top to bottom: 2D input strokes; resulting interpretation of 2D shapes; extrusion surface (at left, the two unambiguous 2D strokes in bold, at right, resulting 3D surface); sweep surface (see previous explanation). In (b), results from Lipson and Shpitalni (1996). A 2D drawing (top left) inflated into a 3D wireframe object (bottom left), with three steps of the optimization process.

features, e.g., a cube is represented by three segments (a corner), or generative properties, e.g., a revolution surface is represented by a profile and an axis (see Fig. 3.10b). Placing geometry is based on four rules: first, geometry features project onto their corresponding creation strokes in the image plane; second, new geometry is in contact with objects of the scene; third, classic line junction invariants provide information on placement and dimensions; fourth, strokes drawn inside an object imply a CSG "subtract" operation. Editing geometry use either strokes or interactors. Resizing an object is performed by "oversketching" over its boundaries to define a new size; moving an object position is made by sketching its shadow on the floor plane; transforming an object (translation or rotation) is obtained by defining a constraint, such as an axis or a plane, with stroke gestures (the rest plane is taken by default), and then select (click) and displace (drag) the object; removing an object is done by clicking on it. Grouping applies a transformation to multiple objects at the same time. By default, objects are grouped with the surface on which they are instantiated. Otherwise, a "lasso" stroke can be used to define groups explicitly. Most of the grouping is unidirectional and thus allows hierarchical manipulation of objects. Finally, orthographic views of 3D scenes are rendered with a "sketchy" appearance that helps preserve the ambiguity and imprecision of traditional drawing, which is important in engaging user imagination beyond approximate models generated by SKETCH.



(a) (b)

Figure 3.10: Results from Zeleznik et al. (1996). In (a), an example of a model created and rendered with SKETCH. In (b), a sample set of gestures available for creating and manipulating shapes. From left to right and top to bottom: cube, cylinder, cone, sphere, revolution surface, and sweep surface, are created with a few strokes following simple rules; scaling, and a CSG "subtract" operation, manipulate existing shapes.

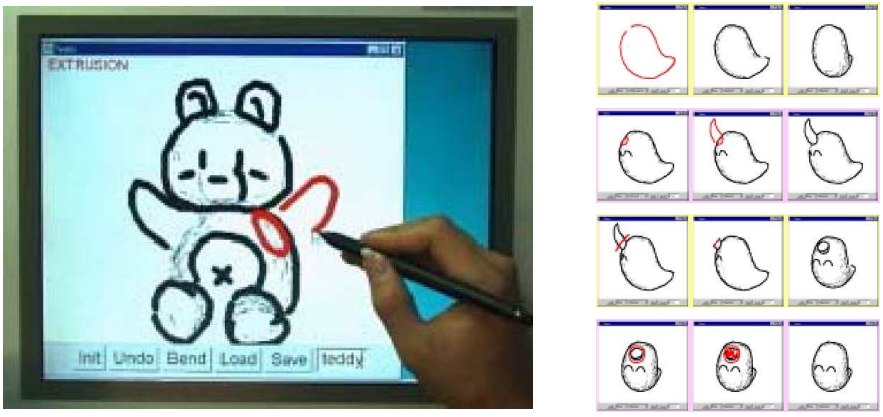
The Teddy system (Igarashi et al., 1999) is an intuitive sketching interface for modeling 3D polygonal surfaces using 2D input devices such as a mouse or a tablet (see Fig. 3.11a). Modeling operations are executed by drawing freeform strokes in the image plane; some actions require only one stroke while, for others, a sequence of strokes is necessary. Three kinds of strokes are recognized by the system: open or closed non self-intersecting strokes, and "scribbling" strokes. The system cannot handle (create,

edit or combine) multiple objects at the same time. More importantly, only closed manifold surfaces with a spherical topology can be generated. Four general modes of interaction exist: creation, painting, extrusion, and bending (see Fig. 3.11b). The first and the last modes are not fully stroke-based since they are entered by pressing a button on the GUI. A mouse click allows escape from the extrusion mode.

Creation requires a single closed stroke on a blank screen. This stroke represents the external silhouette of the object. A 3D shape is inferred from this silhouette in four steps: first, the stroke vertices define a planar closed polygon; second, a constrained Delaunay triangulation of the polygon provides its "chordal axis"[1], and this axis is pruned to obtain the "spine" of the polygon; third, spine vertices are given a height value proportional to their distance to the polygon edges; fourth, a closed surface with oval section is wrapped around the spine and polygon vertices. This "inflates" the external silhouette while preserving the relative size of its different parts. Painting requires a single stroke drawn across the external silhouette of the object created previously. If the stroke is open and stays inside the silhouette, it is transformed into a 3D polyline by projection onto the object surface. This polyline can be erased by scribbling over it. If the stroke is open and simply passes through the silhouette, it defines a cutting surface, extruded along the view vector. The object is cut in two pieces and remeshed (only the right part remains). After a cutting stroke, the system is in extrusion mode. The polyline composed of the sharp edges resulting from the cut is considered as the first extruding stroke. If the stroke is closed and inside the silhouette, it is the first extruding stroke and the system enters extrusion mode. Extrusion requires two strokes: a closed stroke drawn on the object surface (extrusion profile) and an open stroke (extrusion "path", i.e., external silhouette of the extruded surface). The first stroke is swept along the axis of the projection of the second stroke on a plane perpendicular to the object surface. By drawing the second stroke inside the projection of the external silhouette of the object, one obtains an "intrusion", i.e., a cavity. If the user scribbles on the first stroke, a smoothing operation occurs: polygons enclosed by the first stroke are removed and the resulting hole is filled with a smooth surface. Finally, bending requires two strokes: a "reference" stroke and a "target" stroke. Mesh vertices are displaced in directions parallel to the image plane so that their final relative position w.r.t. the second stroke is equal to their initial relative position w.r.t. the first stroke. In this operation the mesh topology remains the same.

Harold (Cohen et al., 2000) is an interactive system for creating three-dimensional worlds by drawing. The entire interface is based on drawing: apart from tools selection (modifying drawing parameters, e.g., pen style and pen width), all objects of the world are edited by clicking on them or drawing strokes, with a 2D input device. The most important 3D primitive is a collection of 2D strokes lying on a 3D plane which changes its orientation to stay as front-facing as possible to the camera, by rotating around a point or an axis. This primitive is also known as a "billboard", and is adapted for objects whose appearance does not change a lot when walking around them, i.e., ob-

[1] The chordal axis is not equivalent to the medial axis.



(a) (b)

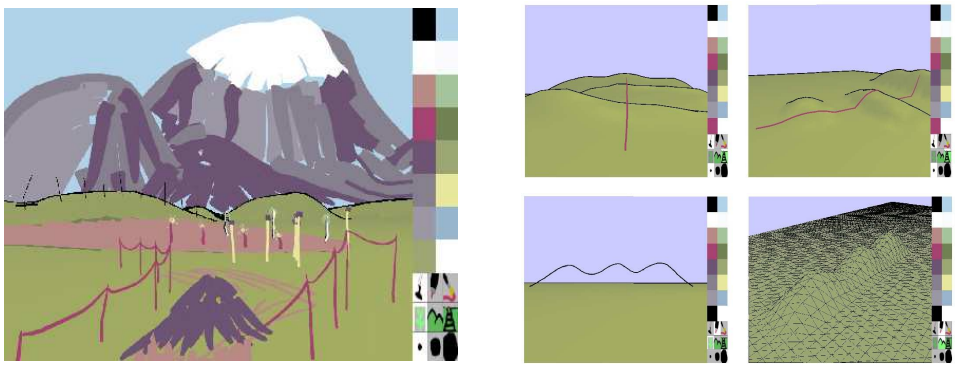
Figure 3.11: Results from Igarashi et al. (1999). In (a), Teddy on a display-integrated tablet, allowing the user to draw directly on the screen, as on an electronic paper sheet. In (b), demonstration of some of the modeling operations. From top to bottom: creation (stroke, result, rotated view); extrusion (painting strokes previously applied and now first stroke, second stroke, result); cutting (stroke, result and enter extrusion mode, click to quit extrusion mode); smoothing (first stroke, second stroke, result).

jects that exhibit a strong radial symmetry (typically trees), but this primitive does not work well for asymmetric objects. Moreover, since billboards have a view-dependent position, billboard intersections may occur when they are close to one another. Three main modes are available through a three-button mouse: drawing mode, camera control mode, and eraser mode. When in drawing mode, three specific submodes are explicitly chosen: ground, billboard, and terrain mode.

In drawing mode, the user draws a stroke on the image plane by moving the cursor, first mouse button pressed. The starting point of a stroke determines whether it is a "stroke on the sky" (the sky is a triangulated sphere) or a "stroke on the ground" (the ground is a triangulated plane located inside the sphere). The stroke on the sky is simply projected onto the sphere while the stroke on the ground is interpreted according to the current drawing submode. In ground mode, the stroke is projected onto the ground to represent roads, etc. If the stroke crosses the projection of a ground silhouette on the image plane (e.g., a hill), parts of the projected stroke are joined by straight line segments on the ground to make it continuous. In billboard mode, the stroke creates a new billboard at the world position corresponding to the projection of the starting point of the stroke on the ground. The rectangular area defined by the new billboard is highlighted, and any stroke whose starting point projects on this area (and ending point does not project onto another billboard) is projected onto the billboard plane (in fact, this is true whatever the current submode). If the ending point is on another billboard, the stroke is projected onto a new "billboard bridge", created between the positions of the projection of the starting and ending points. This primitive allows to define an object which has a view-independent relationship with two other objects, because it is

tied to them. In terrain mode, the stroke, starting and ending on the ground, creates an extrusion of the ground plane so that the projection of the extrusion silhouette on the image plane matches the stroke path. As a result, the height position of all world objects is modified to stay at ground level. Terrain features generated this way are always equivalent to heightfields.

Other gestures are possible in drawing mode: switch off highlighted billboard by clicking anywhere, switch on billboard by clicking on it, paint object by drag-and-dropping a color from the toolbar to the object. In camera control mode, a stroke on the ground defines a camera path and a click anywhere defines the camera point-of-interest (POI). The camera walks along the path, at human height and jogging speed, progressively aiming the POI, and finishes its move looking at the POI. In eraser mode, strokes can be erased with a simple click on them, billboards can be removed with a "scribbling" stroke.



(a) (b)

Figure 3.12: Results from Cohen et al. (2000). In (a), an example scene: background strokes are drawn on the sky, middleground strokes are drawn on billboard bridges, and foreground strokes are drawn on a billboard. In (b), ground and terrain modes. Top row: a ground stroke crossing the projection of several silhouettes is made continuous using line segments. Bottom row: a terrain stroke creates an extrusion of the ground plane whose silhouette projection matches the stroke.

Discussion

The diversity of the existing systems makes comparison difficult because their relative merits are sometimes incompatible, e.g., some automatically reconstruct 3D objects from a single line drawing while others offer a rich interface for interactive modeling under many viewpoints. However, it is possible to focus on the difficulties encountered in order to discern the problems that remain to be solved.

The direct 2D drawing approach is limited by the incomplete information given by 2D input to position strokes in space. Cohen et al. solve this ambiguity by drawing two strokes for each 3D stroke, but at the expense of slowing down user work. However,

as the authors mention, their approach is more appropriate to drawing camera paths or motion curves than to drawing objects in 3D. Tolba et al. avoid this problem by projecting all the strokes on the unit sphere, obtaining panoramas where objects have fixed locations, but without parallax effect since they are supposed to be infinitely far away from the viewer: in fact, it is quite disturbing for close objects and cannot be considered as 3D modeling. Daniels focuses on painterly rendering of 3D models, and stroke positions are obtained by projection on the models. This allows rendering strokes from a different viewpoint, but requires to input a full 3D scene, in addition to the strokes themselves. Maya's Paint Effects elegantly avoids the previous pitfalls by projecting strokes either on fixed planes or on existing objects. We have been inspired by these solutions for our system.

Among the systems that use sophisticated constraint solving mechanisms, some offer interactive modeling possibilities (the systems of Pugh, Eggli et al., and to some extent, Branco et al.) and others are fully automatic (the systems of Akeo et al., and Lipson and Shpitalni). But apart from the system created by Akeo et al. that uses real design drawings as input, all the others require drawings of 3D polyhedral surfaces in wireframe, and without hidden lines. This is a CAD atavism that keeps these solutions out of reach of common drawing practice, but sounds familiar for people accustomed with 2D vector drawing systems. However, Eggli et al. and Lipson and Shpitalni alleviate this burden of unintuitive input by using constraints that tolerate imprecise drawings (line approximations, perspective errors, etc.). In fact, since it is the rough nature of a sketch that makes it a convenient mean for communicating ideas, fault tolerance algorithms are required to preserve this property.

The recent achievements of Zeleznik et al. and Igarashi et al. demonstrated that gesture-based interfaces are powerful and intuitive tools for 3D model design. They trade their simplicity against limitations on the type of models generated. The Zeleznik et al. system offers a restricted set of geometric primitives, compared to most CAD systems. Even if more complex shapes can be obtained by combining simple primitives, models end up looking very similar. Igarashi et al. try to avoid this drawback by using a restricted set of freeform strokes: the inferred shapes are more diverse but still must have plane symmetry at creation (w.r.t. the image plane) and spherical topology. Nevertheless, these works are milestones on the road towards easy-to-use modeling systems that go beyond CAD paradigm: for some researchers, Teddy deserves without hesitation the title of "favorite graphics paper of the last five years".

Harold by Cohen et al. is probably the previous work the most closely related to our system. Its 3D stroke representation and positioning technique have inspired our own. However, it does not handle strokes that correspond to silhouettes of 3D objects, and thus should deform when the viewpoint changes. As we shall see, this is an essential feature of our approach, and the ability to draw silhouettes is much needed in both annotation and initial design applications.

Finally, we would like to raise some questions. In some of the systems presented, the designed model or scene is rendered in a non-photorealistic manner in order to obtain a result that does not inhibit designer imagination. In fact, Strothotte et al. (1994)

have described the effect on the viewer of adjusting the degree of precision in the rendering of a scene, to produce images ranging from rough charcoal sketches to detailed pen-and-ink illustrations. The former are more suitable to convey a "work in progress" feeling than the latter, since information transmitted is less precise. Nonetheless, both are rendered using the same geometric data. Why is it necessary to build a complete model to render a rough sketch? Aren't there weaker forms of knowledge about the geometry that would suffice? We see this as an open problem, involving human cognition issues: how much information about an object is really needed to produce a draft of it? And one of its subproblems concerns mapping from geometry space to drawing space: can all drawings be generated from geometrical information only?

3.3 Drawing and Rendering 3D Strokes

In order to render a sketch from multiple viewpoints, we consider strokes as three-dimensional entities. Two kinds of strokes are used in our system: line strokes that represent 1D detail, and silhouette strokes that represent the contour of a surface. This is the case for both open and closed strokes.

For line strokes, we use a Bézier space curve for compact representation. These strokes are rendered using hardware, and behave consistently with respect to occlusion. Silhouette strokes in 3D are more involved: a silhouette smoothly deforms when the view-point changes. Contrary to line strokes, a silhouette stroke is not located at a fixed space position. It may rather be seen as a 3D curve that "slides" across the surface that generates it. Our system infers the simplest surface, i.e. the same local curvature in 3D as that observed in 2D. For this we rely on the differential geometry properties of the user-drawn stroke, generating a local surface around it. But the degree of validity of this surface decreases when the camera moves. Therefore, we decrease the intensity of the silhouette as the point of view gets farther from the initial viewpoint. This allows the user to either correct or reinforce the attenuated stroke by drawing the silhouette again from the current viewpoint.

3.3.1 Local Surface Estimation from 2D Input

Since the inferred local surface will be based on the initial stroke curvature, the first step of our method is to compute the variations of this curvature along each 2D silhouette stroke drawn by the user.

We start by fitting each 2D silhouette stroke segment to a piecewise cubic Bézier curve. This representation is more compact than a raw polyline for moderately complex curve shapes. The fitting process is based on the algorithm of Schneider (1990a); we briefly review it next. First, we compute approximate tangents at the endpoints of the digitized curve. Second, we assign an initial parameter value to each point using chord-length parameterization. Third, we compute the position of the second and third control points of a Bézier curve by minimizing the sum of the squared distance from

each digitized point to its corresponding point on the Bézier curve. Fourth, we compute the fit error as the maximum distance between the digitized and fitted curves; we note the digitized point of maximum error. Fifth, if this error is above threshold, we try to improve the initial parameterization by a nearest-point-on-curve search using a Newton-Raphson method (see below) and search a new Bézier curve; if this fails, we break the digitized points into two subsets and recursively apply the fit algorithm to the subsets.

Then, each control point V_i of the piecewise cubic Bézier curve Q_3 is associated with a given value of the parameter u along the curve. From the definition of a cubic Bézier curve (see Appendix B), we obtain immediately $u_{V_0} = 0$ and $u_{V_3} = 1$, but u_{V_1} and u_{V_2} are not defined because Bézier curves are approximation splines. However, we can determine a parameter value corresponding to the point on the curve nearest to the control point. For this, we apply the method of Schneider (1990b): we look for the values of u that are roots of the equations

$$[\mathbf{Q}_3(u) - \mathbf{V}_1] \cdot \dot{\mathbf{Q}}_3(u) = 0 \quad \text{and} \quad [\mathbf{Q}_3(u) - \mathbf{V}_2] \cdot \dot{\mathbf{Q}}_3(u) = 0$$

since they define the parameter's value for points on the curve nearest to each control point. These roots can be approximated using the Newton-Raphson method, a classic one-dimensional root-finding iterative routine. The initial estimates for roots are obtained with simple trigonometry (see Fig. 3.13)

$$u^0_{V_1} = \frac{(\mathbf{V}_3 - \mathbf{V}_0) \cdot (\mathbf{V}_1 - \mathbf{V}_0)}{\|\mathbf{V}_3 - \mathbf{V}_0\|^2} \quad \text{and} \quad u^0_{V_2} = 1 - \frac{(\mathbf{V}_0 - \mathbf{V}_3) \cdot (\mathbf{V}_2 - \mathbf{V}_3)}{\|\mathbf{V}_0 - \mathbf{V}_3\|^2}$$

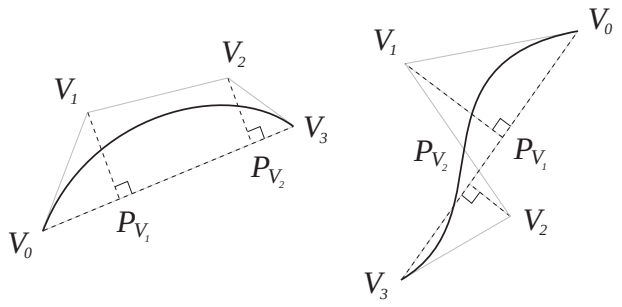


Figure 3.13: Solving the nearest-point-on-curve problem (Schneider, 1990b). Parameter values for points on cubic Bézier curve nearest to control points V_1 and V_2 are obtained using the Newton-Raphson method. Initial estimates for the parameters are $u^0_{V_1} = \left\|\mathbf{P}_{V_1} - \mathbf{V}_0\right\|$ and $u^0_{V_2} = \left\|\mathbf{P}_{V_2} - \mathbf{V}_0\right\|$.

For each parameter value u associated with a control point V, we find the center of curvature $\mathbf{C} = [\xi\ \eta]^T$ by first computing the derivatives of the position coordinates and then solving the following equations (Bronshtein and Semendyayev, 1998):

$$\xi = x - \frac{\dot{y}\left(\dot{x}^2 + \dot{y}^2\right)}{\dot{x}\,\ddot{y} - \dot{y}\,\ddot{x}} \qquad \eta = y + \frac{\dot{x}\left(\dot{x}^2 + \dot{y}^2\right)}{\dot{x}\,\ddot{y} - \dot{y}\,\ddot{x}}$$

where $\dot{x}$ and $\ddot{x}$ are first and second derivatives of x with respect to u. Therefore, we obtain a curvature vector between a point on the curve at parameter u and its associated center of curvature C (see Fig. 3.14a). We will be using these curvature vectors to reconstruct local 3D surface properties. However, if the stroke is completely flat, the norm of the curvature vector, i.e., the radius of curvature, becomes infinite; the method we present next solves this problem.

In order to infer a plausible surface in all cases, we use a heuristic based on the curve's length to limit the radius of curvature. One way of looking at this process is that of attempting to fit circles along the stroke curve. Thus, if we encounter many inflection points, the circles fitted should be smaller, and the local surface should be narrower; in contrast, if the curve has few inflection points, the local surface generated should be broader.

To achieve this, we construct axis-aligned bounding boxes of the control polygon of the curve between each pair of inflection points. Inflection points can be found easily since we are dealing with a well-defined piecewise cubic Bézier curve (see Appendix B). They are either the common control point of two "head-to-foot" cubic Bézier curves of type I (see Fig. 3.13, left) or are located on a cubic Bézier curve of type II (see Fig. 3.13, right). We discard bounding boxes which are either too small or too close to the curve extremities. If the norm of the curvature vector is larger than a certain fraction of the largest dimension of the bounding box computed previously, it is clamped to this value (see Fig. 3.14b). We use a fraction value at most equal to $\frac{1}{2}$, which gives a length equal to the radius of a perfect circle stroke. We also impose a consistent in-out orientation of the curve based on the orientation of the curvature vectors in the first bounding box computed, thus implicitly considering initial user input as giving correct orientation (see Fig. 3.14c). This intuitive choice corresponds to the intent of the user most of the time. If not, a button in the GUI can be used to invert all the curvature vectors along the stroke.

From these 2D strokes, we infer local surface properties, which are then used to create a 3D stroke representation. Each center of curvature embedded in the drawing plane is considered as the center of a circle in a plane perpendicular to the drawing plane and passing by the corresponding control point (see Fig. 3.15a). We consider an arc of $\frac{2\pi}{3}$ radians for each circle, thus defining a piecewise tensor product surface by moving each control point on its circle arc (see Fig. 3.15b). This piecewise Bézier surface is quadratic in one dimension, corresponding to a good approximation of a circle arc, and cubic in the other, which corresponds to the stroke curve. To define the quadratic Bézier curve easily, we express the position of its middle control point as a ratio of the height of the equilateral triangle whose base is defined by the two other control points, of known positions (see Fig. 3.15c). We found the optimal ratio iteratively by measuring the maximum distance between points on the Bézier and on

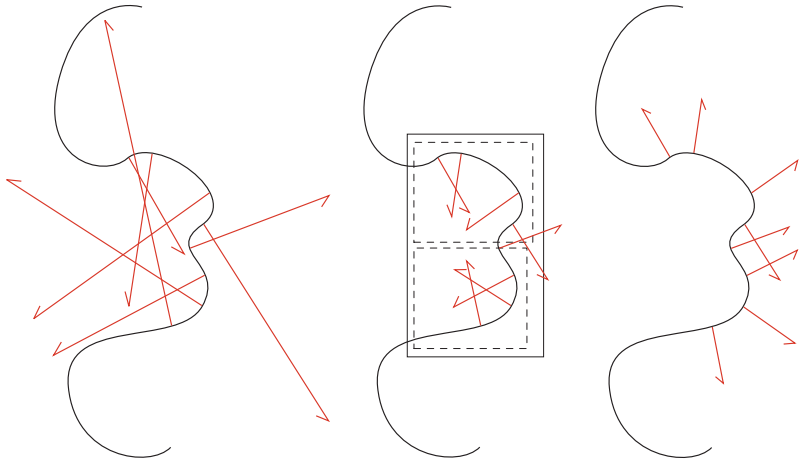


Figure 3.14: Processing vectors of curvature. In (a), curvature vectors before clamping. In (b), curvature vectors after being clamped relative to solid bounding box length (dotted bounding boxes were considered too small to be selected). In (c), curvature vectors after correcting orientation.

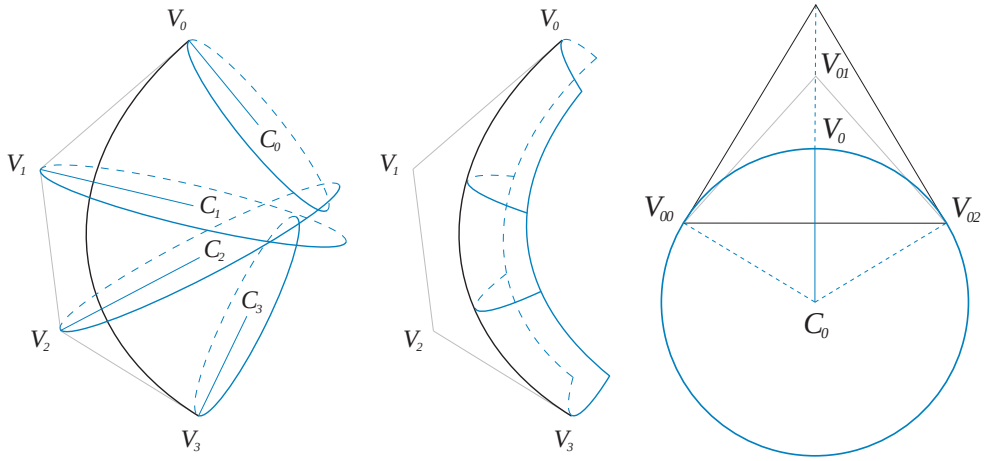
the circle arc.

In practice, the inferred radius of curvature may of course be inaccurate, but as stated earlier, the inferred surface will only be used for generating a probable silhouette when the viewing angle changes slightly. If more information is needed about the 3D surface geometry, the contour will have to be redrawn by the user at another viewpoint. However, this simply add a new stroke and doesn't modify the old stroke and its local surface. For a complete overview of the behavior of our method in a simple "textbook example", see Fig. 3.16.

3.3.2 Rendering in 3D

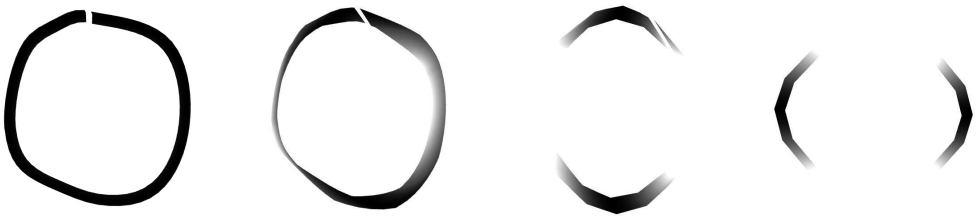
Given a local surface estimation, our goal is to display the initial stroke from a new viewpoint. When the viewpoint changes, we expect the stroke to change its shape, as a true silhouette curve would do. We also expect its color to change, blending progressively into the background color to indicate the degree of confidence we have in this silhouette estimation. Recall that we want our system to be interactive, which imposes an additional computational constraint. In what follows, the term "local surface" refers to the polygonal approximation of the local surface estimation of the stroke.

The solution we adopt is to generate a fast but approximate silhouette based on the local surface generated as described above. We simply render a "slice" of the local surface that lies between two additional clipping planes, parallel to the camera plane and situated in front of and behind the barycenter of the centers of curvature (see Fig. 3.17a). The distance between clipping planes depends on the stroke width value we have chosen. This ensures silhouette-like shape modification, with minimal



(a) (b) (c)

Figure 3.15: Construction of a 3D stroke from a 2D stroke composed of one cubic Bézier curve with control points V_i. In (a), the 2D centers of curvature C_i computed with our method, and the corresponding 3D circles (the dotted lines are hidden by the drawing plane). In (b), the Bézier surface obtained (same remark as above for the meaning of dotted lines). In (c), definition of the quadratic Bézier curve that approximates the circle arc. The position of the control point V_{01} is determined using the positions of the two other control points V_{00} and V_{02} and the ratio of the height of the equilateral triangle.



(a) (b) (c) (d)

Figure 3.16: "Textbook example": a simple circular stroke. In (a), front view; in (b), side view rotated by 30 degrees; in (c), side view rotated by 90 degrees; in (d), top view.

computational overhead.

It is important to note that our approach to silhouette rendering is very approximate: its behavior will be somewhat unpredictable for wide camera angles and very long strokes. A good accurate solution for computing a new silhouette from the estimated surface would be to use one of the existing algorithms (Markosian et al., 1997; Raskar and Cohen, 1999; Hertzmann and Zorin, 2000). However, we have seen that our surface is only a coarse inference of the local surface to which the silhouette belongs, so computing an accurate silhouette would probably be unnecessary in our case.

Initially, we render all geometry other than the silhouette strokes (for example the house in Fig. 3.1). Therefore, the depth and color buffers are correctly filled with respect to this geometry. In the next step, we use different elements to display the silhouette strokes and to perform stroke occlusion. Because of this, we need a multipass algorithm, summarized in Algorithm 3.1.

Rendering Silhouette Strokes

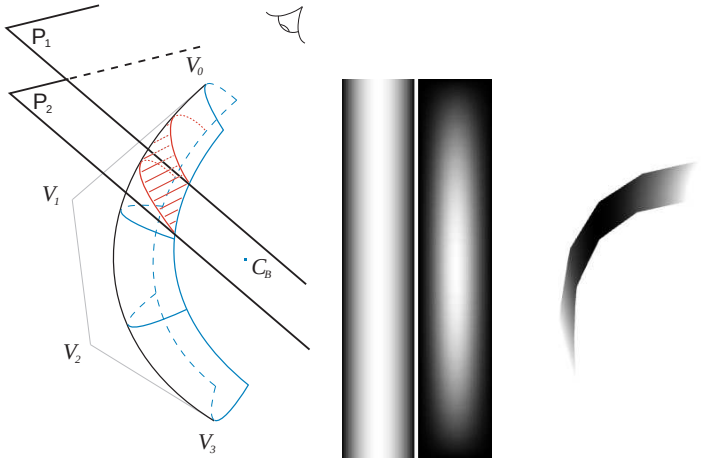
In the first pass, we render the strokes as clipped local surfaces, with the depth test and color blending enabled, but with depth buffer writing disabled. Thus correct occlusion is performed with respect to other (non-stroke) geometry. To represent the confidence in the surface around the initial stroke we apply a "stroke texture" (see Fig. 3.17b, left) as an *alpha* texture to the local surface. This confidence is maximum at the initial stroke position and minimum at left and right extremities of local surface. We use a Gaussian distribution that progressively blends the stroke color into the background color for modeling this confidence function. As a result, the stroke becomes less intense as we move away from the initial viewpoint. This blending also allows two different strokes to reinforce each other by superposition, which corresponds to the behavior of traditional ink brush drawings.

Occlusion by Local Surfaces

In addition to occlusion by other geometry, we also need to handle occlusion by strokes. This required a slightly more sophisticated process, since we do not want local surfaces to produce hard occlusion (such as that created by a depth buffer) but rather to softly occlude using the background color, in a visually pleasing way. To meet these requirements, stroke occlusion is achieved in an additional two passes. Recall that we start with a depth buffer which already contains depth information for other objects in the scene.

In the second pass, we render the local surfaces into the depth buffer with the depth test and depth buffer writing enabled. Local surfaces are textured with a different alpha texture called the "occluder texture" (see Fig. 3.17b, right) and rendered with the alpha test enabled. As a result, occluder shape will be a rounded version of local surface shape.

In the third to fifth passes, we render the local surfaces into the color buffer with the depth test and color blending enabled, but with depth buffer writing disabled. Local surfaces are textured with the same "occluder texture" and colored with the background color. The occluder color thus blends with what is already present in the color buffer: we obtain progressive occlusion from the edge of local surface to the center of initial stroke. Moreover, we use the stencil buffer to mask the stroke rendered during first pass, and this way the occluder does not overwrite it in the color buffer.



(a) (b) (c)

Figure 3.17: Stroke rendering. In (a), the final stroke is a slice of Bézier surface obtained using two clipping planes P_1 and P_2 facing the camera; C_B is the barycenter of the C_i (see Fig. 3.15). In (b), two texture samples, one of "stroke texture" (left) and one of "occluder texture" (right). White corresponds to an alpha value of 1, black to an alpha value of 0. In (c), image obtained from rendering a black stroke against a white background, with the slice position corresponding roughly to (a).

Drawing Style

We can have a different color for the background and the stroke occluder, such as what is shown in the artistic illustration of Fig. 3.20. This gives a subtle indication of local surface around a stroke: it can be seen as equivalent to hatching or pencil pressure variation in traditional drawing. Finally, since "stroke texture" and "occluder texture" are procedurally generated, their parameters can vary freely. This allows the creation of different tools according to specific needs.

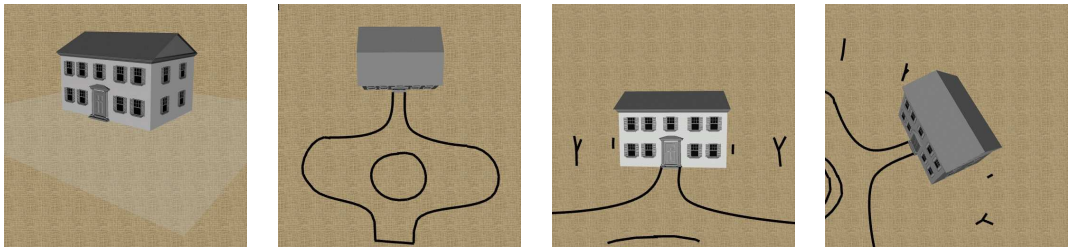
3.4 Interface for Drawing

A drawing session using our system is in many ways similar to traditional drawing. A designer starts with an empty space, or, in the case of annotation, he can add in a pre-existing 3D model, such as the house in Fig. 3.1. For strokes drawn in empty space, we

M R ()
// Pass 0
// Render non-silhouette strokes geometry, i.e., scene and line strokes
// ...
// Pass 1
// Draw clipped local surfaces with stroke texture, stroke color
Enable Depth test
Enable Blend
Disable Depth buffer write
Draw silhouette strokes in **Color buffer**
// Pass 2
// Draw local surfaces with occluder texture
Enable Alpha test
Enable Depth buffer write
Disable Color buffer write
Draw occluders in **Depth buffer**
// Passes 3, 4, 5
// Draw local surfaces with occluder texture, occluder color
Enable Stencil test
Enable Stencil buffer write
Disable Depth buffer write
Disable Color buffer write
For each silhouette stroke:
    Draw clipped local surface in **Stencil buffer**
    Disable Stencil buffer write
    Enable Color buffer write
    Draw occluder in **Color buffer**
    Enable Stencil buffer write
    Disable Color buffer write
    Erase clipped local surface in **Stencil buffer**

Algorithm 3.1: Multipass stroke rendering algorithm.

project onto a reference plane, parallel to the camera plane and containing the world origin (it is possible to choose a fixed offset relative to this position). Typically, the user will place the drawing plane in space using the trackball. An example is shown in Fig. 3.18, where we start drawing the grounds of the house. We want to draw in the plane of the ground corresponding to the house, so we position ourselves in a "top view". We then verify that the position of the plane is as intended (a) and draw the strokes for the grounds in the plane (b). Similarly, tree trunks are drawn in planes parallel to the walls of the house (c).



(a) (b) (c) (d)

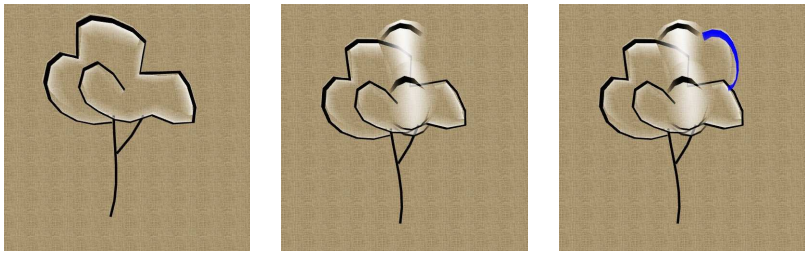
Figure 3.18: Plane positioning. First, position ourselves in a "top view". Then, we verify the plane position colored in semi-transparent grey (a), and we draw the grounds in this plane (b). We next draw trees trunks in planes parallel to the walls of the house (c), and examine the result from another viewpoint (d).

Once such parts of the drawing have been created, we can use the existing entities to position the curves in space. More precisely, if at the beginning or at the end of a 2D stroke the pointer is on an existing object, we use this object to determine a new projection plane. We obtain the depth of the point selected by a simple picking. The picked object can correspond to a geometric object or to (the local surface of) a stroke. There are three possibilities:

- If only the beginning of the stroke is on an object, we project the stroke on a plane parallel to camera plane, which contains the selected point. An example can be seen in Fig. 3.19, where we draw the leaves of a tree in one plane (a) and in another (b).

- If the beginning and the end of the stroke are on an object, we interpolate depth values found at the two extremities by using the parameter u of the piecewise cubic Bézier curve. Each control point is projected on a plane parallel to the camera plane and located at the corresponding depth. See Fig. 3.19c, where this "bridge" mechanism is used to join two parts of a tree.

- If it happens that the stroke extremities are in empty space, it is projected on the same plane as the previous stroke, except if the trackball has been moved. In this case, the reference plane is used.



(a) (b) (c)

Figure 3.19: Different projections using objects of the scene. In (a) and (b), we draw on planes automatically positioned in space with the help of the tree trunk, i.e., planes passing through the trunk. This produces convincing tree foliage. In (c), we use a "bridge" to draw a new branch. It fits correctly in place because of the automatic positioning of the start and the end of the stroke.

Classic 2D computer drawing operations extended to 3D are also very useful. Among them, we have implemented erasing strokes and moving strokes (in a plane parallel to camera plane).

3.5 Applications

We present results for three different application scenarios. The first one is artistic illustration, the second one is annotation of a pre-existing 3D scene, and the third one is "guided design". In our current implementation, drawings can be saved in a custom file format, in world coordinates, but without reference to an annotated object. The initial learning curve for our system is relatively significant, requiring a few hours to get used to the idea of positioning planes and drawing on them. Once this is understood, typical drawings take between ten minutes to one hour to complete.

Illustration in 3D

Figure 3.20 shows an illustration designed with our system. Most strokes are silhouette strokes. They have been rendered on a textured background so that the local surface occluder appears as a "fill" effect. The illustration is displayed from different points of view, showing the effects of occlusion, varying stroke lightness, and silhouette deformation.

Annotation of a 3D Scene

Another application of 3D drawing is to use our system for annotating an existing 3D model. While 2D annotation is widespread, few systems provide a straightforward manner to do this in 3D. In a typical user session, we initially load the 3D model. It is subsequently integrated with the drawing in the same way as for local surfaces: if a

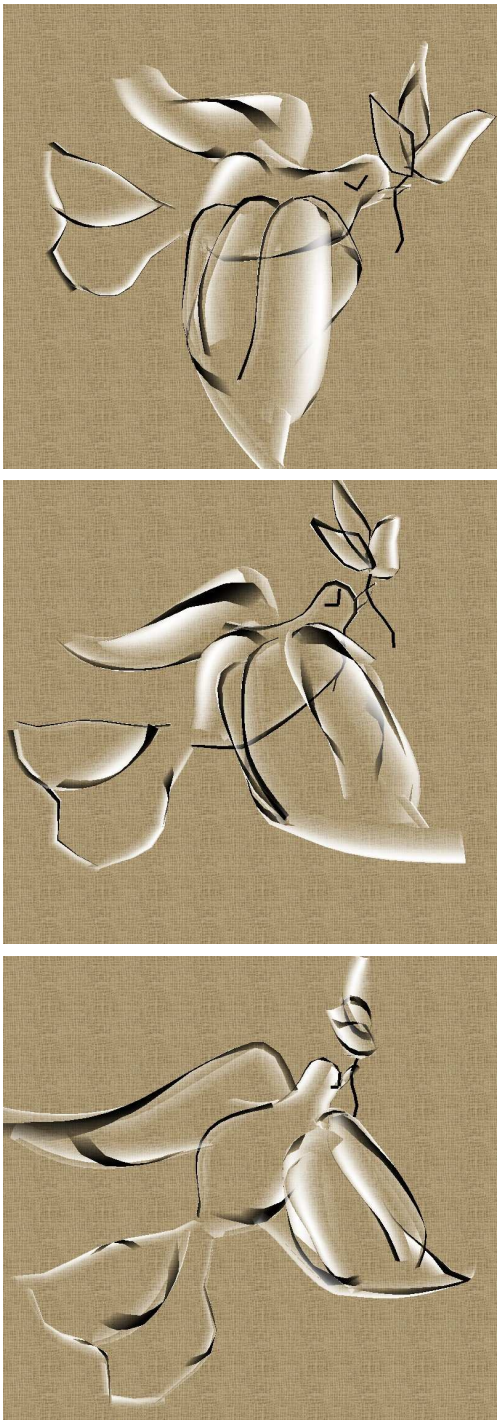


Figure 3.20: An example of artistic illustration. Three views of the same 3D sketch area are shown.

stroke is drawn on the model, it is set to lie on it. Line strokes can be used for adding annotation symbols, e.g., arrows, text, etc.

Figure 3.1 is a simple example of annotation: adding a coarse landscaping sketch around an architectural model. Figure 3.21 shows annotation used for educational purposes: a heart model is annotated during an anatomy course. The use of a well-chosen clipping plane gives an inside view of the model and allows drawing anatomical details inside it. We could also imagine using our system in collaborative design sessions. Annotation would then be employed to coarsely indicate which parts of the model should be changed, and to exchange ideas in a brainstorming context.

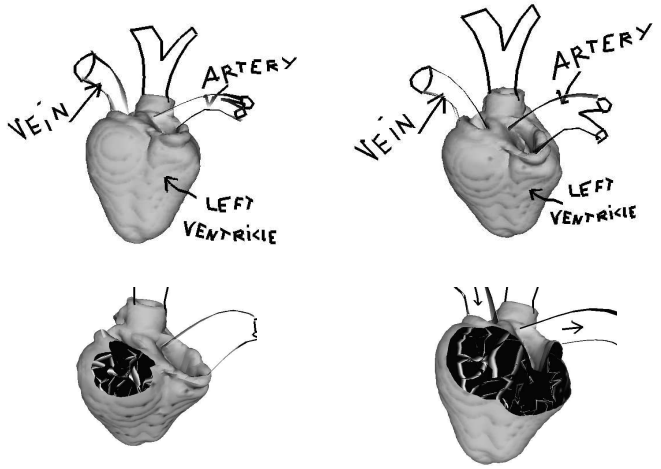


Figure 3.21: An example of annotation in 3D: annotating a heart model during an anatomy course. Anatomic structures have been drawn both inside and outside the heart surface. The text displayed has also been drawn with our system, using line strokes, thus it is view-dependent. A possible improvement would consist of drawing text on billboards, or implementing more sophisticated schemes (Preim et al., 1996; Fekete and Plaisant, 1999).

"Guided Design"

The idea of this third application is to load a 3D model and use it as a guide for designing new objects. When the drawing is complete, the model is removed. A good example of this kind of application is clothes design. A 3D model is used to obtain body proportions (see Figure 3.22).

3.6 Conclusion and Future Work

We have presented a system which enhances the traditional 2D drawing process with 3D capabilities, notably by permitting multiple viewpoints for a single drawing. Instead of attempting a complete 3D reconstruction from 2D strokes, we infer a local

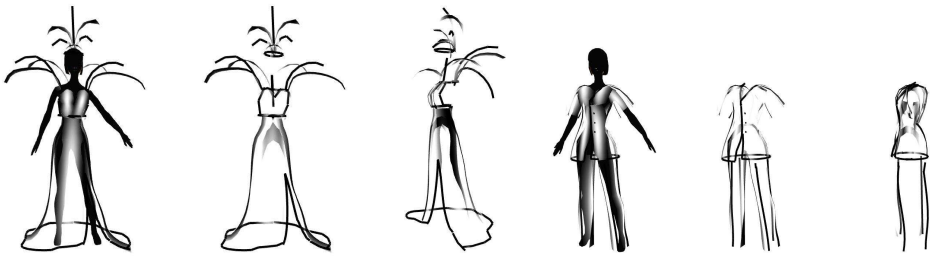


Figure 3.22: Using a 3D model as a guide can be useful in clothes design.

surface around the stroke. This is achieved by assuming that strokes represent planar silhouettes of objects, and by using differential geometry properties of the curve.

The resulting local surfaces are then drawn efficiently using hardware-accelerated rendering of a clipped part of the local surface, corresponding approximately to a silhouette. Color blending is used to gracefully diminish the intensity of the strokes as we move away from the initial viewpoint, and to allow reinforcement of intensity due to multiple strokes. We have also introduced a multipass algorithm for stroke inter-occlusion, which results in a visually pleasing gradual occlusion. Our system provides an interface which retains many of the characteristics of traditional drawing. We help the user in positioning the drawing plane in empty space, and placing it relatively to other objects (strokes or geometry) in the sketch.

Future Work

We have chosen a Bézier surface representation for storing silhouette stroke information. This approach is convenient but it can result in a large number of hardware-rendered polygons. Other representations could be used instead. For instance, a volumetric data structure would allow us to combine information about the local surface: the more strokes are drawn at a given location in space, the more we are certain that it corresponds to a true surface point. To render strokes from a new viewpoint, we would have used a variant of the marching cubes algorithm (Lorensen and Cline, 1987) to produce a surface estimation, associated with a silhouette detection algorithm to generate new silhouette strokes. The obvious drawbacks of this approach are memory usage and data loss due to volumetric sampling of user input. A particle-based approach, i.e., strokes composed of particles that try to satisfy a set of constraints such as "stay on silhouette", etc., would produce interesting stroke transformations. However, numerical stability would undoubtedly be an issue as well as computational overhead, which would impede interactivity.

Our user interface is clearly far from perfect. Plane positioning is not completely intuitive, and alternatives should be considered. For example, a computer vision approach in which the user defines two viewpoints for each stroke and implicitly reconstructs positions in space (within an error tolerance) could potentially prove feasible.

But it is questionable whether such an approach would truly be more intuitive. We are investigating various alternatives in order to find an appropriate combination that will improve our current solution, both in terms of stroke rendering quality and user interface.

Chapter 4

Relief: A Modeling by Drawing Tool

4.1 Introduction

Most people draw. We sketch, doodle, and scribble effortlessly, to keep a trace of our thoughts or communicate ideas to others. We consider drawing as a writing alternative, because it is faster and more precise to describe three-dimensional shapes and spatial relationships with two-dimensional lines than with words. The tool set is minimal: a thin, short stick held in hand (pen), and a flat rough surface (paper). The tool principle is simple: rubbing the stick against the surface produces a mark made of the stick material. And that is all one needs to know. The dextrous use of the tool constitutes a wealth of common knowledge most people have acquired since kindergarten. However, this common knowledge is seldom used in computer graphics for anything but two-dimensional vector or pixel-based drawing applications.

Few people sculpt. Creating forms in three dimensions involves modeling clay, chiseling wood or marble, etc. These materials are difficult to manage, and shaping them generally requires highly specialized tools and skills.[1] Using computers does not make the sculpting process simpler: three-dimensional data are obtained either by scanning an existing sculpture, or by modeling directly with the computer, using 2D or 3D input devices. In practice, designers turn to computers when they need to be definite and precise, while advantages of the digital medium could be used in the initial idea stage.

The aim of this work is to promote drawing as an effective modeling tool for de-

[1] In his posthumous *Treatise on Painting*, Leonardo da Vinci describes the labor of the sculptor. "For his face is smeared and dusted all over with marble powder so that he looks like a baker, and he is completely covered with little chips of marble, so that it seems as if his back had been snowed on; and his house is full of splinters of stone and dust. In the case of the painter it is quite different [...] for the painter sits in front of his work in perfect comfort. He is well-dressed and handles the lightest of brushes which he dips in pleasant colors. He wears the clothes he likes; and his house is full of delightful paintings, and is spotlessly clean. He is often accompanied by music or by men who read from a variety of beautiful works, and he can listen to these with great pleasure and without the din of hammers and other noises." (cited in Freud, 1964, pp. 64–65). See also da Vinci (1956).

signers. By focusing on a 2D-to-3D approach, we will avoid problems inherent to 3D-to-3D solutions that use haptic feedback manipulation devices and other VR props, as we explained in Section 3.2.1. Above all, we want to take advantage of drawing's natural expressiveness to enrich the modeling process. This implies that our system input must consist of "just plain strokes", to stay as close as possible to the traditional drawing experience. Thus, the user draws strokes freely without being disturbed by mode: strokes belong either to silhouettes or sharp features, they convey either texture or shading information; by topology: strokes are either open or closed, they are either self-intersecting or not; by depth: strokes are positioned implicitly in 3D space.

Our 3D drawing system presented in Chapter 3 produces no model at all, but a drawing representation allowing the generation of new drawings when the viewpoint changes. Compared to it, our modeling by drawing tool outputs manifold polyhedral surfaces, i.e., complete models, as a traditional CAD system, and thus can fit in the classic computer graphics images production pipeline.

Overview

Our approach is based on the simple idea that relief sculpting is an appropriate metaphor for the process of modeling by drawing, which is really about making 3D output progressively emerge from 2D input. In fact, a relief is a "sculpture that projects from a background surface rather than standing freely. According to the degree of projection, reliefs are usually classified as high (*alto rilievo*), medium (*mezzo rilievo*), or low (*basso rilievo* or bas-relief)"; *rilievo schiacciato* being "a form of very low relief" (Chilvers et al., 1997). Thus, there is a continuum in three-dimensional object representation that goes from drawing or painting, to sculpture in the round, relief being a transitional art[2] that encompasses all intermediate levels (see Fig. 4.1).

Considering this fact, our system allows the user to create a three-dimensional shape from two-dimensional input, by iteratively modeling by drawing, and changing the viewpoint to examine the resulting model, repeating these two stages until the end of the modeling session. In the modeling by drawing stage, the system proceeds in two steps: first, from 2D stroke input in image space, a 2.5D polyhedral surface is inferred, also in image space; then, this 2.5D surface is used to create (resp. modify) a new (resp. existing) 3D polyhedral surface in world space.

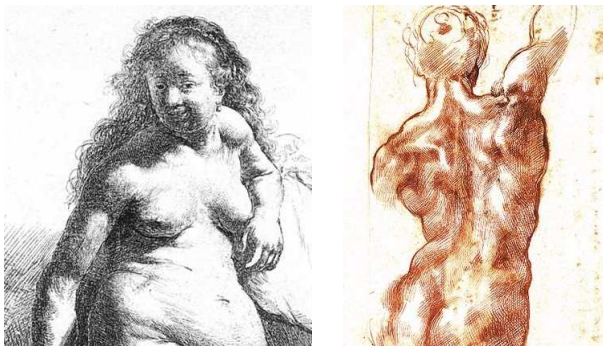
In the first step, strokes are discretized into a point set, and a triangulation of a possible "non-convex hull" of this set is obtained. Independently, the drawn image is processed to infer a height field. Then, starting from the previous triangulation, a polygonal approximation of the height field is computed. Therefore, a relative depth value is associated to each vertex of the triangulation. Consistent results suppose that the drawing follows a few conventions, that have their counterpart in traditional art. To

[2] Leonardo da Vinci compared relief to painting because both pay attention to perspective, and the play of light and shadow. "The sculptor may claim that *basso relievo* is a kind of painting; this may be conceded as far as drawing is concerned because relief partakes of perspective [...] this art is a mixture of painting and sculpture." (cited in Williams, 1990). See also da Vinci (1956).

(a) (b) (c)

Figure 4.1: From low relief to high relief. In (a), relief of Cleopatra as Hathor, temple of Haroeris at Kom Ombo, 1st century B.C., Egypt. This is an example of Egyptian sunken or coelanaglyphic relief, where carved figures are not projecting beyond block surface. In (b), dedicatory relief to Asklepios, 4th century B.C., Piraeus archaeological museum, Athens. In (c), Tullio Lombardo, Saint Mark baptizes Ammianus, circa 1481, marble, 107 × 154 cm, basilica dei Santi Giovanni e Paolo, Venice.

better understand them, we would like to point out a distinction between two classic shading styles: one style is interested in the effect of light and shadow as a way of depicting the illumination of a scene, we will call it "painter shading" (see Fig. 4.2a); the other style is interested in the effect of light and shadow as a way of bringing out the modeling of surfaces, we will call it "sculptor shading" (see Fig. 4.2b). In this last style, light is used in a non-realistic but expressive way, as if each modeling detail of the surfaces was illuminated by a different light source, in order to be as "understandable" as possible by the viewer. Only the second style is meaningful in our system. This way, relative depth is indicated by the user with varying shades of gray, for example the darkest shade corresponding to the maximum relative depth value.



(a) (b)

Figure 4.2: Painter versus sculptor shading. In (a), Rembrandt van Rijn, seated female nude, circa 1631, etching, 177 × 160 mm, British museum, London. In (b), Michelangelo Buonarroti, study for a nude (studies for *The battle of Cascina*), 1504, pen and ink over black chalk, 408 × 284 mm, Casa Buonarroti, Florence.

In the second step, the previous 2.5D surface is "unprojected" in 3D space. There are two possibilities, depending on whether or not the drawing was completed on a "blank page", or on the projection of an existing model. In the first case, the surface is unprojected using an arbitrary depth offset value, added to the relative depth values of the vertices. In the second case, projections of the vertices of the existing model (that are part of the 2.5D surface in the same way strokes vertices are) are used to compute the absolute depth of strokes vertices by propagation. That is, knowing the absolute depth of the vertices of the existing model and the value of relative depth (height field), one can obtain the absolute depth of strokes vertices.

4.2 Previous Work

We have already presented an overview of systems taking two-dimensional strokes as input for modeling three-dimensional objects, using explicit or implicit constraints (see Section 3.2.2). Among them, very few were actually interested in sculpting the surface of objects, their use of strokes being limited to drawing one-dimensional features: three-dimensional paths, edges of wireframe models, silhouettes of objects, commands of gestural interfaces, etc. The Teddy system (Igarashi et al., 1999), with the "polygon inflation" method for the extrusion of arbitrary polygon meshes (van Overveld and Wyvill, 1997), is one of these exceptions, and we refer the reader to our previous description.

We will shortly summarize and discuss works related to surface modeling by two-dimensional input, either academic papers or commercial software. Beforehand, let's cite as a curiosity the paper of Cignoni et al. (1997) which studies the inverse of the process we are interested in. Their system automatically generates low and high relief from arbitrary 3D models, and it can be useful as a computer formalization of the traditional rules of this art.

Papers Overview

To the best of our knowledge, Williams (1990) is the first to present 3D painting as a surface sculpting tool. The system takes advantage of an unusual hardware technique for displaying video rasters, as a set of section lines which defines a surface. This way, an image is displayed as a height field, with the luminance value corresponding to the elevation. Thus, classic 2D graphics and image processing techniques can now be used to create three-dimensional surfaces (see Fig. 4.3a to Fig. 4.3d). However, the painting convention that maps luminance to elevation is not very intuitive since humans naturally interpret shading as indication of local surface orientation or color, not height. After 2D editing is over, 3D paint raster data can be transformed into geometrical surface representation in multiple ways, among them Coons patches offer a useful parameterization for subsequent texturing (see Fig. 4.3e to Fig. 4.3g). The author mentions applications such as retouching depth maps of scanned faces and building

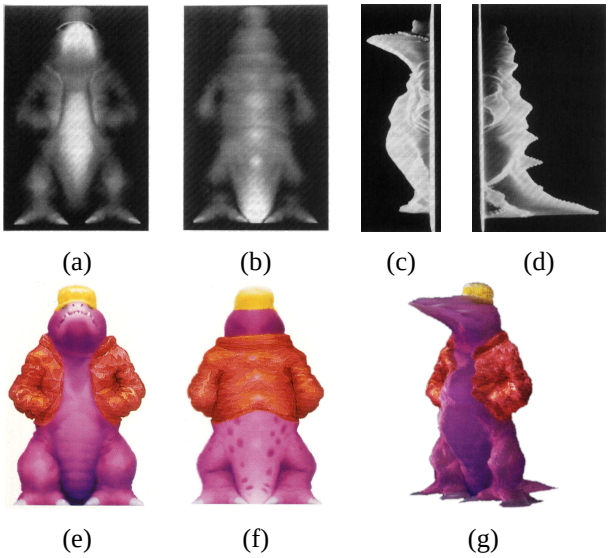


Figure 4.3: Results from Williams (1990). *Working class dinosaur* by Bill Maher: standard video display, front (a) and back (b); raster surface display, front (c) and back (d); texture painting, front (e) and back (f); full 3D model made of the front and back halves (g).

complete models, either from scratch or with the help of an existing image. But he insists on the fact that building models this way remains difficult.

One year later, Williams (1991) reviews techniques for shading in two dimensions. Shading conveys information about three-dimensional shape but is also useful for visually emphasizing regions of an image; it is therefore a tool of choice for incorporating animated characters in live action sequences. One solution to the 2D shading problem is to infer a 3D surface from two-dimensional information and use it to compute shading: in this respect, it is related to the shape from silhouette problem, an active research area in computer vision. However, techniques that do not attempt to infer a 3D shape are still very popular. The most simple 2D shading techniques are shape-independent and thus do not convey any 3D information. More sophisticated, shape-dependent algorithms exist and are available in many paint systems. The "skeleton" fill algorithm fills a region by varying intensities across the levels created with a region thinning algorithm, such as the medial axis transform. However, the resulting shading is not smooth due to derivative discontinuities along skeleton axes. The "2D sweep" algorithm uses two user-specified curves of different colors to define a smoothly shaded region by sweeping, i.e., by interpolating between the curves. Filters can be applied to the object matte[3] in order to approximate shading, but also highlighting from a "local" light source, and even color bleeding from the background. For example, a highlighting matte is built by scaling the blurred matte up depending on the position of the light,

[3] A matte is an "image or signal that represents or carries only transparent information that is intended to overlay or control another image or signal" (Hapeman et al., 2001). An acceptable synonym for a binary matte is a mask.

negating the result, and masking it with the original matte. As opposed to low pass filtering, pyramidal filtering adapts the shading to the scale of the silhouette, so that shading is correct for large regions as well as small ones ("pyramidal airbrushing").

Techniques that infer a 3D model from outlines ("silhouette inflation"), allow 3D shading algorithms to be applied in 2D animation. Using a classic reference image (see Fig. 4.4a), a possible solution to infer a 3D surface is the shape from shading method from computer vision (see Fig. 4.4b). The author suggests several other methods: automatic segmentation into superquadrics (by correlating the image with a family of superquadrics silhouettes), manual segmentation into "symmetry seeking" generalized cylinders (user-defined cylinders are then automatically fit to image segments), and automatic inflation by masked pyramidal convolution. This last technique applies a series of Gaussian filters of decreasing radii to each region considered separately from the others. After each pass, the blurred image is masked with the original image to limit inflation to the inside of silhouette (see Fig. 4.4c and Fig. 4.4d).

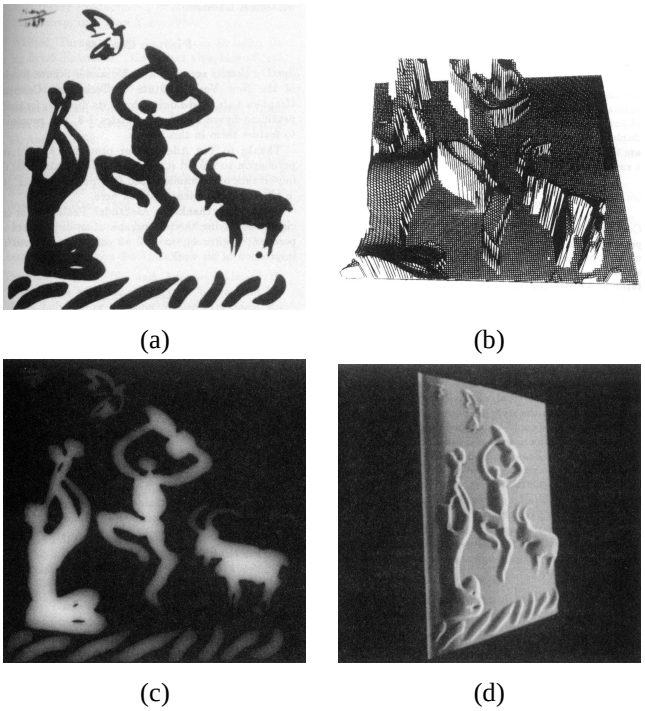
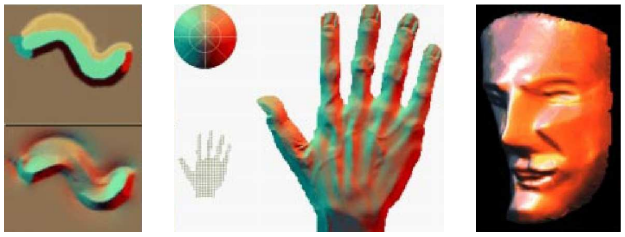


Figure 4.4: Results from Williams (1991). Various inflations of Pablo Picasso's *Rite of Spring* (a), a classic reference image: using the shape from shading algorithm (b); using a masked pyramidal convolution, inflation displayed as an image (c), or as a relief (d), with non-standard video hardware (Williams, 1990).

GRADED (van Overveld, 1996) is an interactive system for designing freeform surfaces using shading information. Interestingly, the author notes that current computer design tools have direct manipulation techniques at the antipodes of those of non-computer tools. In fact, with CAD systems, techniques for designing surfaces are

more frequently 0D (control points) than 1D (boundary curves of Coons and Gordon patches), and are exceptionally 2D (such as Williams's 3D paint), as opposed to what is observed with traditional design tools. In GRADED, surfaces are represented as heightfields and manipulated by editing a depth buffer or a gradient buffer. To compute local illumination, color in each point is considered as a function of the normal vector, obtained using the gradient value. Editing the depth buffer automatically updates the gradient buffer and thus has a visible effect on surface shading. Classic paint system tools and image processing operations are available: direct color editing (tinting), opaque painting, smoothing, etc. But they must be interpreted in a depth image context, e.g., tinting (adding color) corresponds to rising (adding material). Brushes are fully configurable in shape, orientation, size, and profile. Conversely, editing the gradient buffer allows the surface to be modified by modifying its shaded image. Since there is a one-to-one relation between shade color, normal vector and gradient, a shape from shading algorithm is not necessary. The user simply selects a shade color (corresponding to a unique orientation) on an illuminated sphere, and paints it in the gradient buffer. To enforce that the current gradient distribution corresponds to a consistent depth distribution, the conservation constraint (stating that the accumulated depth variation over any closed loop must be equal to zero) is propagated over the surface by an iterative algorithm (see Fig. 4.5a). Surfaces created with GRADED can be converted into gradient maps, for bump mapping, or into triangular meshes. However, the author notes that this system is more adapted to refining existing polygonal surfaces (previously scan converted to depth maps), than to sculpting entire shapes from scratch (see Fig. 4.5b and Fig. 4.5c).

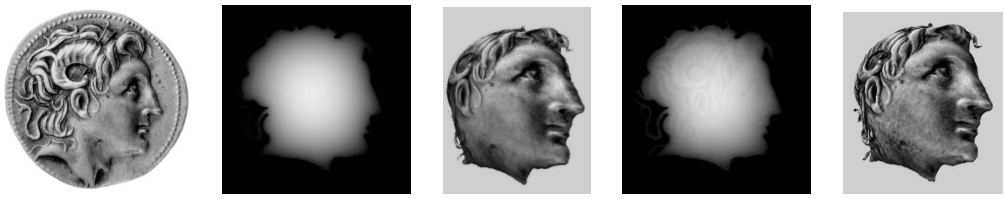


(a) (b) (c)

Figure 4.5: Results from van Overveld (1996). In (a), a gradient distribution painted into the gradient buffer (top), the result of enforcing the conservation constraint (bottom). In (b), a bas-relief of a hand, painted from scratch in 2 hours (in the top left corner, the shade color selection sphere). In (c), a face, obtained in less than 20 minutes by editing in GRADED a simple polygonal face mask.

Williams (1998) proposes a simple shape from silhouette algorithm that infers 3D models from silhouette information only. A 2D matte (a "silhouette") is defined with an image (see Fig. 4.6a), then is converted into implicit form as a signed pseudometric function: the original matte shape is a level set of this function (see Fig. 4.6b). This process is called the "inflation" of the matte. Interpreting the inflated matte as depth information, a geometric model is created, and textured with the original image (see

Fig. 4.6c). For objects exhibiting planar symmetry, such as human faces, a single silhouette can produce a reasonable approximation of the entire object. This technique is very useful for creating approximate shading from two-dimensional input, such as traditional animation where only matte information is available, in the spirit of the toolbox presented previously (Williams, 1991).



(a) (b) (c) (d) (e)

Figure 4.6: Results from Williams (1998). The image of an antique coin with the profile of Alexander the Great (a) is used to define a matte, an inflation of the matte (b), the corresponding textured 3D model (c), an inflation of the matte blended with 5% of the original image (d), and the corresponding textured 3D model (e).

Johnston (2002) presents a method for computing approximate lighting on cel animation drawings. As opposed to other methods (Williams, 1991, 1998), surface normals are inferred without attempting to reconstruct 3D geometry. Given a line drawing (see Fig. 4.7a), the most simple normal approximation scheme is the "simple blobbing". It uses the drawing matte to determine exterior silhouette edges, whose associated normals are known to be perpendicular to the eye vector. Interpolating these normals across the matte gives a complete normal vector field (see Fig. 4.7b). The previous scheme can be refined in "compound blobbing" if the matte is subdivided into different layered regions, and blobbing is applied on each of these regions, before compositing the results (see Fig. 4.7c). Instead of evaluating normals on matte edge, the "quilting" scheme considers ink lines as silhouette edges. Associated normals on each side of the lines are interpolated across the matte (see Fig. 4.7d). However, this is obviously wrong since relative depth information is not taken into account (e.g., the right hand of the character is in front of his body). The final scheme involves the tagging of each side of the ink lines to determine if it is "over" or "under", i.e., front facing or back facing. This step requires human intervention, but it is possible to provide an initial estimate using relative depth of paint regions and line dominant curvature information. Interpolating these tags (encoded as black and white colors) gives a grayscale "confidence" matte. Blending quilted and blobby normals, obtained previously, with the confidence matte as a key, produces a normal vector field that can be used for approximate lighting (see Fig. 4.7e).

Artisan (Alias|wavefront, 2002b) is a set of tools of the Maya software with a common painting-based interface. Among these tools, the Sculpt Polygons Tool for polygonal surfaces offers an intuitive interface for editing vertices. An equivalent tool exists for non uniform rational B-spline (NURBS) surfaces, the Sculpt Surfaces Tool. Editing is achieved by simply "painting" the surface, using one of four different sculpting

(a) (b) (c) (d) (e)

Figure 4.7: Results from Johnston (2002). Normals approximation: original drawing (a), region-based or "simple blobby" normals (b), compound region-based or "compound blobby" normals (c), line-based or "quilted" normals (d), and blended normals (e).

operations: push, pull, smooth, and erase. Pushing translates vertices in the direction of the tool reference vector, of an amount dependent on their current displacement w.r.t. reference surface, and in a way dependent on the brush stamp profile (radius, opacity, shape); while pulling does the same thing, but in the opposite direction (see Fig. 4.8a). The surface subdivision density has an influence over the precision of the result (see Fig. 4.8b). The reference vector is defined in the tool settings, and possible values are: surface normal, surface normal at the beginning of the stroke, camera view vector, x, y or z axis (see Fig. 4.8c). The reference surface is defined as the surface at the beginning of the sculpting session. Its vertices cannot be translated any further than the maximum displacement value in the tool settings. However, a larger displacement is possible if the user updates the reference surface during the session. It can be done automatically after each stroke so that strokes displacements become additive. Smoothing reduces bumps in the surface. It can be set to be applied automatically after each stroke. Erasing resets vertex displacements to their values in the erasing surface (see Fig. 4.8d and Fig. 4.8e). This surface is equivalent to the reference surface of push-pull operations. Initially, they are identical, thereafter, the erasing surface is updated independently. Other operations include: creating masks, to prevent areas of the surface from being affected by sculpting; flooding the surface, to apply the current brush operation to the entire surface.

ZBrush (Pixologic, 2002) is a modeling software with a painting metaphor. The polygon sculpting capabilities of ZBrush are very similar to Maya's Artisan Sculpt Polygons Tool (Alias|wavefront, 2002b), presented previously. Starting with any 3D polyhedron (over a dozen primitives are already proposed, such as spheres, cylinders, cubes, etc., with variable mesh density) the user can sculpt the shape by pushing and pulling vertices, i.e., by painting onto the object with push and pull brushes. Higher level deformation tools are also provided to twist, bend, or inflate the shape, and even simulate gravity (see examples on Fig. 4.9). The polygonal surface can be smoothed, refined or simplified. These modifications can be constrained using axial or radial symmetries, or restricted to a region by masking. Masks can be defined directly by painting on the object. However powerful these sculpting tools are, the real innovation

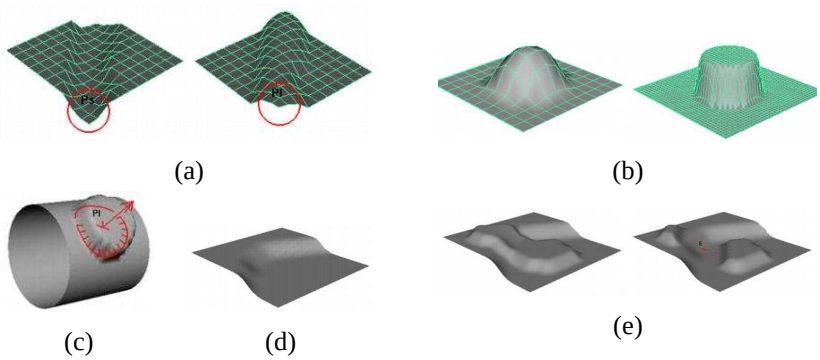


Figure 4.8: Alias|wavefront's Maya Artisan Sculpt Polygons Tool. Various sculpting operations: pushing or pulling the polygonal surface (a); at left (resp. right), result of pulling with low (resp. high) subdivision density (b); result of pulling with the reference vector set to surface normal (c); erasing, updated erase surface (d), sculpted surface, and resulting erased surface (e).

of ZBrush comes from its set of 2.5D painting tools, that allows anyone familiar with 2D paint programs to create 3D models easily. To achieve this, "pixols" in the ZBrush canvas represent not only position and color, but also depth, material, and orientation. Brush strokes can be fully controlled in type (color, depth, or material), dimensions, depth of embedding into the canvas, load (how much color or depth is applied), shape (set by a grayscale alpha texture), etc. They are either constrained to planes or lines, or drawn on other objects and positioned on their surface. Some high-level brushes create complete 3D shapes directly, in the spirit of Maya's Paint Effects (Alias|wavefront, 2002c). For example, the Fiber Brush can paint hairs or blades of grass in a single stroke.

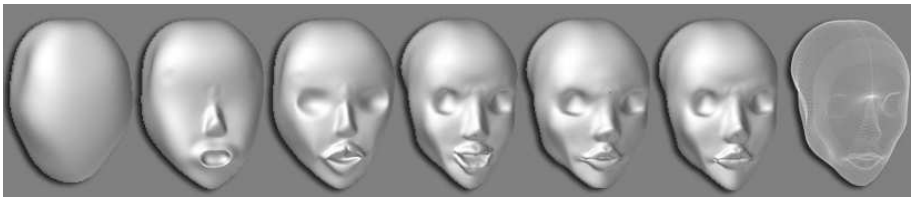


Figure 4.9: Pixologic's ZBrush. Several steps in the creation of a character's head, starting from a sphere with a 128 by 128 horizontal and vertical division count. The general shape of the head is obtained with "gravity" and global deformation tools. The features (eye sockets, nose, etc.) are created by pulling and pushing the sphere vertices with the edit tools.

Discussion

On the small number of works presented, nearly one third are in fact commercial softwares that are not mere implementations of academic papers. This emphasizes

the practical interest of the computer graphics industry for modeling tools using two-dimensional input. Besides, even if we have considered all these approaches together, they do not correspond to the same needs.

Some of these tools are meant to be used as interactive systems while others are automatic processing methods, and this is related to differences in input. In the first case, the user iteratively builds a model by drawing, deciding after each stroke if the result is satisfying, undoing his actions if needed (Williams, 1990; van Overveld, 1996; Alias|wavefront, 2002b; Pixologic, 2002). In the second case, the user has been given finished drawings (and their associated mattes), and wants to obtain rough depth information from them, e.g., for shading traditionally animated characters (Williams, 1991, 1998; Johnston, 2002).

The difference in output echoes the previous distinction on interactivity. A few systems allow creation of full 3D models, and thus fit well in the 3D computer graphics pipeline (Alias|wavefront, 2002b; Pixologic, 2002); they are interactive. The majority of the systems are handling 2.5D slabs (terrains or bas-reliefs), which can be useful for detailing 3D surfaces or 2D cartoon shading (Williams, 1990, 1991, 1998; van Overveld, 1996; Pixologic, 2002); half of them are interactive. Only one system generates 2D bump maps for automatic shading of line drawings (Johnston, 2002); it is not interactive.

Nevertheless, the most important difference between systems lies in the mapping of two-dimensional input to three-dimensional depth. Systems closest to sculpting (Alias|wavefront, 2002b; Pixologic, 2002) use solutions derived from direct manipulation texturing interfaces (Hanrahan and Haeberli, 1990; Daily and Kiss, 1995), but here surfaces are painted with depth variations instead of being painted with colors or normals values. Systems for cartoon shading mainly rely on drawing mattes, i.e, external silhouettes, to infer 3D information (Williams, 1991, 1998), sometimes adding internal lines to refine this approximation (Johnston, 2002). The "creation" phase of the Teddy system (Igarashi et al., 1999) could be classified in this category. Systems that use other image information are either associating depth values with luminance (Williams, 1990), or manipulating depth through shading (van Overveld, 1996). It is difficult to say what is the most effective solution: even if it is true that shading is a way we perceive relative depth in the real world, it is also true that most sculptors think rather in terms of depth (positions and volumes) than in terms of shading.

Like Maya's Artisan (Alias|wavefront, 2002b), our system is interactive: the user alternatively sculpts and "steps back" to examine the result by changing the viewpoint. Our system also outputs full 3D geometry, but it is not limited to modifying a given reference surface with a fixed resolution: the user creates pieces of geometry by drawing, without being constrained by the underlying polygonal surface representation. Just as 3D paint (Williams, 1990), our system maps image luminance to depth, but in an indirect way: the user draws according to sculptor shading conventions, and a 2.5D surface is inferred using both stroke geometry and drawing image information. We expect that these key differences will make our system a more effective modeling tool for the initial stage of design.

4.3 From 2D to 2.5D

From a simple 2D drawing made with a computer input device, and without the help of any information other than user strokes, we infer a corresponding 2.5D polygonal surface (a relief). This surface is obtained in a two-step process: first, strokes are digitized into points, and a non-convex hull of the union of the strokes point sets is determined; second, a height field is inferred using strokes and hull information, and approximated with a polygonal surface.

User drawn strokes are captured using a computer mouse, or using a computer tablet with a stylus. The first device records only a two-dimensional position, while the second measures also pressure on the stylus tip and two-dimensional tilt of the stylus body, thus giving on the whole five-dimensional information about user gesture. To limit the digitization rate, we impose an arbitrary minimum distance between two positions of the device. In practice, this threshold is equivalent to the size of a few screen pixels. We call drawing a collection of such strokes, each stroke being discretized into n-dimensional points, $n \in \{2, 5\}$.

4.3.1 Finding a Non-Convex Hull

Finding the *convex* hull of a 2D point set is a classic problem in computational geometry, and the subject of a large number of papers. Still, in many cases, the determination of a *non-convex* hull would be very useful to obtain something like the "shape" described by the point set. Since this is not a well-defined concept, there are many possible solutions to this problem. We will discuss two previously proposed algorithms.

Edelsbrunner and Mücke (1994) have defined the α-shape of a dense unorganized set of data points as the frontier between a reachable and an unreachable region of space. The definition of this frontier depends on the size parameter α (related to the size of the carving spoon in the famous "vanilla ice cream with chocolate chips" analogy). The α-shape is thus equal to the point set for $\alpha \to 0$, and to the convex hull of the set for $\alpha \to \infty$. More recently, Amenta et al. (2001) proposed the "power crust" algorithm for shape reconstruction, based on the medial axis transform. It performs well even in places where point sampling is not sufficiently dense, and it guarantees the "watertight" property of the output mesh.

Although these algorithms give good results, they are not suited to our particular problem of determining a non-convex hull from a drawing, and this, for two reasons. First, shape reconstruction makes the hypothesis that data points are sampled from the boundary of an object. This is not true in our case since points are sampled on strokes, and these strokes can describe either geometry, texture, or shading information. This is particularly obvious in the case of the power crust algorithm: the medial axis transform of our drawing is meaningless because it can be easily perturbed by any stroke that does not lie on shape boundary. Second, and this is a consequence of the first observation, we do not meet point sampling density requirements to produce geometrically and topologically correct approximation to the true shape. Since a drawing is

fundamentally ambiguous,[4] strokes describe very loosely the true surface of an object, sometimes leaving wide open regions between them. Under such circumstances, how does one correctly deal with this erratic sampling using the definition of the parameter α, or the computation of the "local feature size" function?

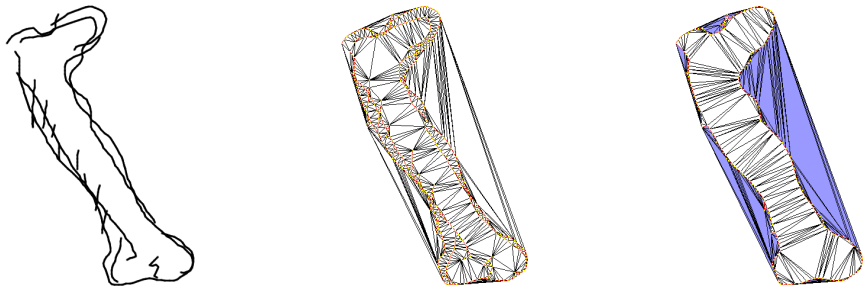
Definition of the Non-convex Hull of a Drawing

To construct a non-convex hull from a set of 2D points, we have been inspired by the method of Watson (1997): starting from an initial Delaunay triangulation of the points, it applies a "short-leg" criterion to determine which edges to remove, guaranteeing a non-convex hull with a maximal sum of inverse lengths of contiguous edges. Nevertheless, Watson's definition of his criterion ("any edge whose end points are connected by a contiguous sequence of shorter edges" is deleted) does not seem practical, and he has not published a description of an algorithm implementing it. But this method better suits our needs since it does not make the hypothesis of boundary point sampling, and gives an output even for poorly sampled data sets. Furthermore, since our points belong to ordered subsets (strokes) that give explicitly the connectivity relationships between points, we do have more information than a rough unorganized point set. Thus, we will take advantage of this topological criterion, in addition to Watson's geometrical criterion, in our definition of the non-convex hull of a drawing.

Using the Computational Geometry Algorithms Library (CGAL, 2002), a constrained Delaunay triangulation is easily obtained from the strokes, considered as sets of vertices connected by constraints (see Fig. 4.10a and Fig. 4.10b). Triangles of this triangulation fulfill a weaker constrained empty circle property. The CGAL data structure also introduces new vertices and subconstraints at input constraint intersection points. Then, the non-convex hull algorithm processes the previous triangulation by examining successively each non-constrained convex hull edge and inquiring if it satisfies a simpler version of Watson's criterion, based on the lengths of the two edges of the underlying face. If the convex hull edge does not satisfy the criterion, its underlying face is tagged as "outside", and the two aforementioned edges are then examined recursively, in a depth first manner (see Algorithm 4.1). As a result of the process, triangulation faces are classified in two categories: either inside or outside the object, and the non-convex hull is the chain of edges belonging to one inside face and one outside face. Finally, in a postprocessing step, non-convex hull edges that do not belong to the original strokes are set as new constraints; vertices that do not belong to the non-convex hull are removed (see Fig. 4.10c).

Our algorithm produces only non-convex hulls with spherical topology, and this is an important limitation compared to α-shape (Edelsbrunner and Mücke, 1994) and power crust (Amenta et al., 2001) algorithms. However, this limitation was also present in "Teddy" (Igarashi et al., 1999), which trivially defined the non-convex hull of the

[4] As opposed to sculpting, which defines presence or absence of matter in every point in space, drawing can play with the in-between: the white of the paper can as well signify filled, or empty.



(a) (b) (c)

Figure 4.10: Finding a non-convex hull. In (a), the original drawing. In (b), the constrained Delaunay triangulation of the strokes (constrained edges are colored in red). In (c), the constrained Delaunay triangulation after non-convex hull processing (faces tagged as outside are colored in blue, non-convex hull edges are colored in red).

Input: A constrained Delaunay triangulation
Output: A constrained Delaunay triangulation with tagged faces
F N C H (*Triangulation t*)
foreach *ConvexHullEdge e*
  if *e* **not** constrained
    Stack s
    s.P (*e*)
    while *s* **not** empty
      Edge $e_1 \leftarrow$ *s*.T ()
      Initialize two other edges e_2 and e_3 of the underlying face f_1
      Initialize the underlying faces f_2 and f_3 of edges e_2 and e_3
      Initialize edges square lengths l_1, l_2, l_3
      if $l_2 < l_1$ **and** $l_3 < l_1$
        if not ((e_2 constrained **and** f_2 outside)
          or (e_3 constrained **and** f_3 outside))
          Tag underlying face f_1 as outside
          s.P ()
          if e_2 **not** constrained
            s.P (e_2)
          if e_3 **not** constrained
            s.P (e_2)
        else
          s.P ()
      else
        s.P ()
return *t*

Algorithm 4.1: Non-convex hull algorithm, inspired from Watson (1997).

drawing by allowing the user to draw only one closed stroke. Thus, to define non-spherical topology, we allow the user to indicate hole marks, as is commonly done in comic book production (see Fig. 4.11a and Fig. 4.11b). These marks are taken into account before the non-convex hull postprocessing step: first, CGAL is used to determine triangulation faces intersected by each segment of a mark stroke; second, the marked faces are tagged as "outside", and their edges are used as starting edges for a tagging algorithm, very similar to the one used for the definition of the non-convex hull (see Algorithm 4.1 and results on Fig. 4.11c and Fig. 4.11d).

(a) (b) (c) (d)

Figure 4.11: Hole marks to define non-spherical topology. In (a) and (b), hole marks in comic book production: artwork from *Stone* #3, Avalon Studios. In (a), pencils by Whilce Portacio. In (b), inks by Gerry Alanguilan. Note the small crosses drawn by the penciler to give information on the positions of holes to the inker. In (c) and (d), hole marks in our system (same color conventions as in Fig. 4.10). Mark strokes are currently selected by pressing a modifier key on the keyboard. Future extensions could include a stroke recognition algorithm that would automatically detect small scale cross patterns.

4.3.2 Inferring a Height Field

In the previous section, we obtained an approximation of the shape of the drawing by determining a non-convex hull of it. Since our goal is three-dimensional modeling, we need now to "inflate" the shape into the third dimension. This inflation process has been the subject of many papers (see Section 4.2) and can be classified as either object-based (Igarashi et al., 1999) or image-based (Williams, 1991, 1998). The former approach considers only geometrical and topological informations, ignoring the other half of the drawing data, constituted by textural information. On the contrary, the latter approach takes into account all these elements, and therefore allows more subtle results.

An image-based inflation algorithm must meet two requirements for a good "blobbing" effect: first, it must be independent of image size, i.e., the sampling of the drawing should not influence the inflation; second, it must be dependent on the two-dimensional feature size, i.e., it should confirm the minimal hypothesis that large features have large inflations. Our algorithm for inferring a height field from a drawing

satisfies these properties. It is composed of three steps, taking into account successively drawing geometry and topology, drawing texture, and then combining all this information to produce the final result.

In the first step, a drawing mask is obtained by rendering only triangles inside the non-convex hull of the triangulation obtained previously (see Fig. 4.12a). To analyze this shape, the Euclidean distance transform of the mask is performed and gives the distance field $d(x, y)$ (see Fig. 4.12b). This distance operator produces much smoother results than an erosion morphological image processing operator. To give the shape a bulgy appearance, we use the idea of Oh et al. (2001), and map the distance field to a unit sphere height field $z(x, y)$, i.e.,

$$z(x, y) = \sqrt{1 - \left(1 - \frac{d(x, y)}{d_{max}}\right)^2}$$

where d_{max} is the maximum value of the scalar field d (see Fig. 4.12c). However smooth our height field is, small discontinuities remain around local maxima (corresponding to skeleton axes of the medial axis transform). To get rid of these, the sphere-based height field is filtered using a two-dimensional Gaussian low pass filter, whose support size s and standard deviation σ are proportional to d_{max}, i.e., $s = \frac{1}{2} d_{max}$ and $\sigma = \frac{1}{2} s$ (see Fig. 4.12d). The idea of adapting the filter to the characteristic scale of the mask was inspired by the "pyramidal filtering" scheme of Williams (1991). It ensures the feature size dependence of our algorithm.

In the second step, the drawing image is rendered with full stroke information, resulting in a gray level image if a tablet is used (see Fig. 4.12e). This image is filtered using the previously defined filter, taking advantage of previous geometrical and topological information (see Fig. 4.12f). In the third step, the filtered height field f is used as a matte for the filtered drawing image to give the final height field (see Fig. 4.12g). Thus, influence of the drawing texture is maximum only for the maximum values of the height field, allowing texture-based shape information to modulate geometry and topology-based information but not to distort it.

Three aspects of the inflation process have to be stressed. First, since each low pass filtering dissipates image "energy" (by lowering the maximum grayscale pixel value), the resulting height field is not normalized, even if the source images are. We decided to keep it that way. Second, the value of d_{max} gives us a distance (in pixel units) that can be used in a heuristic to estimate the mapping of the height field to a depth field (in the next section). Third, computation time is dependent of both image size and filter size. We optimize the former by processing only the image of the bounding box of the drawing.

The source code of the height field algorithm is given in MATLAB script language (see Algorithm 4.2). In order to define the key functions used, we give a short description of them below, modified from the MATLAB software documentation (The MathWorks, 2002). `bwdist(bw)` computes the Euclidean distance transform of the binary image `bw`. For each pixel in `bw`, the distance transform assigns

a number that is the distance between that pixel and the nearest nonzero pixel of `bw`. `fspecial('gaussian', hsize, sigma)` returns a rotationally symmetric Gaussian low pass filter of size `hsize` with positive standard deviation `sigma`. If `hsize` is a scalar, the filter is a square matrix. `imfilter(a, h, 'replicate')` filters the multidimensional array `a` with the multidimensional filter `h`. With the `'replicate'` option enabled, input array values outside the bounds of the array are assumed to equal the nearest array border value.

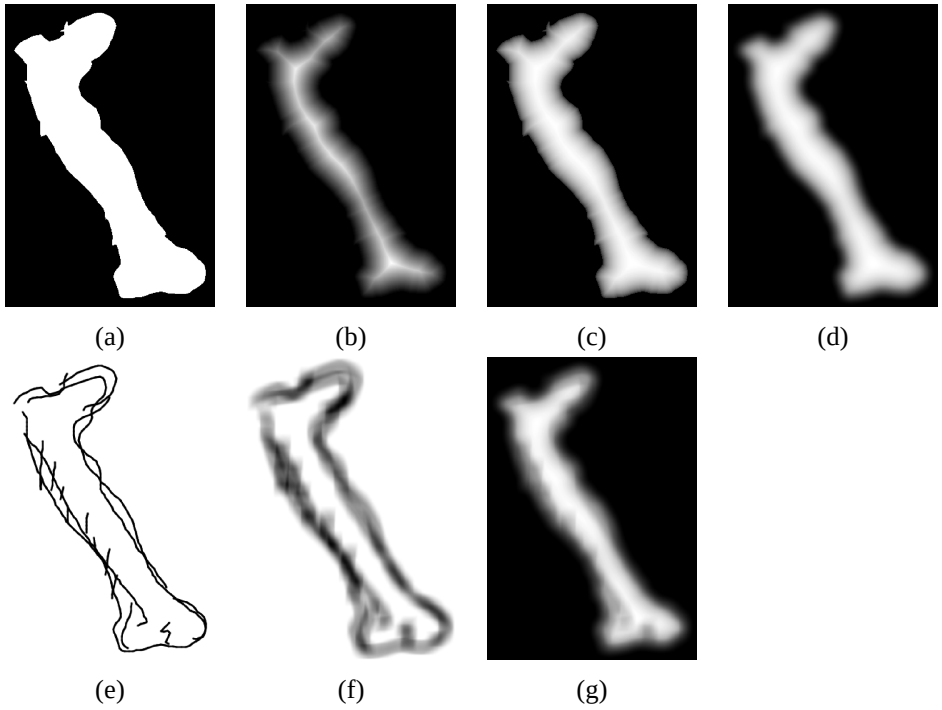


Figure 4.12: Inferring a height field. This is done in three steps. First, from the drawing mask (a), one obtains its Euclidean distance transform (b), which is mapped to a unit sphere height field (c), and then adaptively low pass filtered (d). Second, the drawing image (e) is filtered (f) with the same filter that was used in (d). Third, the previous height field (d) is used as a matte of the filtered image (f) to give the final height field (g).

Height Field Polygonal Approximation

Finally, using an algorithm inspired from Garland and Heckbert (1995), a polygonal surface approximating the previous height field is computed, minimizing both the error and the number of triangles. Starting from the previous constrained Delaunay triangulation (see Section 4.3.2) as an initial approximation, the algorithm, at each iteration, finds the input point with the highest error in the current approximation and insert it as a new vertex in the triangulation (see Algorithm 4.3 and results on Fig. 4.13). The error metric used is the L_∞ error, also called the maximum error. For two vectors of

```
% input:  two image files (drawing image and drawing mask)
% output:  a two-dimensional array of real values (height field)
image = im2double(imread("image.png"));
mask = im2double(imread("mask.png"));
% Euclidean distance transform
dist = bwdist(~mask);
dmax = max(dist(:));
% mapping to unit sphere height field
sphere = sqrt(1 - (1 - dist/dmax).^2);
hsize = round(dmax/2);
sigma = hsize/2;
% two-dimensional Gaussian filter
h = fspecial('gaussian', hsize, sigma);
spheref = imfilter(sphere, h, 'replicate');
imagef = imfilter(image, h, 'replicate');
field = imagef.*spheref;
```

Algorithm 4.2: MATLAB code for computing height field. Most of the functions come from the Image Processing Toolbox (The MathWorks, 2002).

size n, **u** and **v**, representing respectively the discrete height field and its polygonal approximation, it is defined as $\|\mathbf{u} - \mathbf{v}\|_\infty = \max_{i=1}^{n} |u_i - v_i|$.

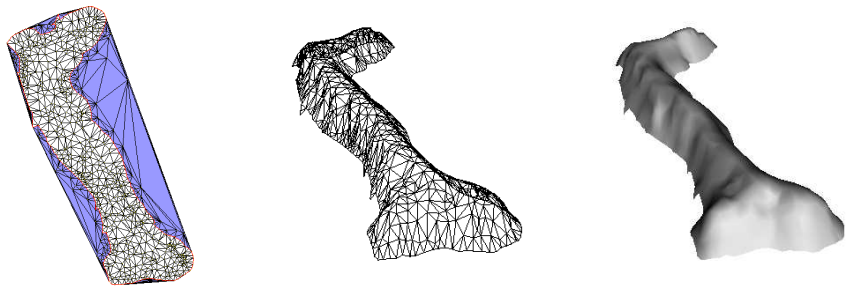
This greedy insertion algorithm has been accelerated using commodity graphics hardware and the OpenGL API. We decided to speed up the search for the maximum error point per face using the following approach. First, the approximated height field is simply obtained by rendering the triangulation with a grayscale value at each vertex corresponding to its height (unfortunately, the grayscale dynamic range is currently limited to 8 bits only on the average graphics board). Second, in order to assign a triangulation face to each pixel, an identification (ID) buffer is rendered with a unique color identifier for each triangulation face. Third, the maximum error point per face is obtained by evaluating the error for each pixel belonging to the face. To prevent selecting a pixel previously selected or selecting pixels belonging to face edges, we mask them during the ID buffer phase using a null color identifier.

4.4 Conclusion and Future Work

We have presented a system that allows relief modeling, taking plain strokes as input and producing a three-dimensional manifold polyhedral surface as output. We proceed in two steps, from 2D discontinuous strokes information, to 2.5D continuous height field information. First, we infer the shape described by the drawing by determining the non-convex hull of the strokes set. Second, we infer the height field using both

Input: A constrained Delaunay triangulation and a height field
Output: A constrained Delaunay triangulation with height value at each vertex, approximating the height field
A H F (*Triangulation t, HeightField h*)
ErrorOrderedSet s
foreach *TriangulationFace f*
Candidate c ← *f*.M E P (*h*)
s.I (*c*)
f.S C A (*c*)
nPointsInserted ← 0
while not G M (*s*.H E (), *nPointsInserted*)
t.G C F (*s*.H E P ())
foreach *ConflictFace f*
s.E (*f*.G C A ())
t.I (*s*.H E P ())
foreach *NewFace f*
Candidate c ← *f*.M E P (*h*)
s.I (*c*)
f.S C A (*c*)
nPointsInserted ← *nPointsInserted* + 1
return *t*

Algorithm 4.3: Fast polygonal approximation of height field, inspired from Garland and Heckbert (1995).



(a) (b) (c)

Figure 4.13: Height field polygonal approximation. In (a), the resulting approximation, obtained with the Algorithm 4.3. In (b), the three-dimensional mesh obtained after unprojecting each vertex of (a) in world space, on an arbitrary depth plane. In (c), the Gouraud shading rendering of (b).

geometry and topology information given by the non-convex hull, and drawn image information.

The interface of our system is still unfinished, e.g., the eraser is missing, but was found easy to master by first-time users. However, we noticed that it is necessary to provide an explicit control by the user of the relationship between gray level value and relative depth offset. Furthermore, implementation of the second part of the three-dimensional modeling system we described in the introductory overview is underway. The results currently obtained with the relief modeling system lead us to expect exciting outcomes when multiple relief surfaces will be modeled and combined from different viewpoints.

Chapter 5

Conclusion and Future Work

Looking back at the contributions presented in this thesis, some of them more than two years old, we have mixed feelings of achievement and dissatisfaction. We will talk about them, trying to be as objective in our judgments as a father can be on his own children when he sees them leaving the family home definitively. We will also present some of our would-be contributions, still written on our urgent accomplishments checklist. As a conclusion of the conclusion, we will give our views on the future of pedagogical applications of computer tools, or in other words what could possibly be the classroom of tomorrow.

Our model of anisotropic elastic material has proven very cost-effective, implementation time and computation time being low, and also very intuitive to use, obtained behaviors being always close to what we expected. However, this model seems mainly adapted to locally orthotropic material. We are afraid that it will be considered obsolete in a short time by the soft tissue modeling community, even though the most physically accurate models seldom meet the intuitive system image requirement, so important in situations where the user has no background at all in physics (medical student, artist, etc.).

Our 3D drawing system has been significantly successful as a proof-of-concept of an alternative 3D stroke representation, useful when true geometry can be traded against modeling speed and simplicity, as in initial phase of design. Nevertheless, the two obvious flaws of our system, that is limited expressiveness of 3D strokes (only planar silhouettes, with rough inference for local surface), and awkward user interface (stroke modes, frequent explicit depth settings), makes it much more tedious to use than 2D drawing – but not more than the average three-dimensional modeling software! In starting this project, we had such high expectations (invent a 3D equivalent of drawing) that we were disappointed by the result, obtained after a series of trials and errors with different approaches. But we are convinced that this concept is not a dead end and deserves other attempts.

In theory, our brand new modeling by drawing system is very promising. In practice, since we do not have, at the time this sentence is written, a complete working implementation, we cannot have many opinions on it. We expect to have a demonstra-

ble system for this thesis' defense.

As a first future work, the two aspects of our thesis, animation and modeling, could be integrated in a system of animated schemas, that would be very helpful in the teaching of dynamical phenomena such as the cardiac beat. This system would close the loop between the other systems we have proposed: a teacher could first draw a 3D illustration, then specify parameters by drawing, and finally animate it, thus creating an animated functional schema that can be observed from several viewpoints. Our second future work is more involved. In order to evaluate pedagogical applications of these techniques in medical school curriculum, and in particular for anatomy courses, we would like to test our software tools in a real-life setting with a group of volunteer students. This testing would have to be performed according to rigorous ergonomics methodologies, as we advocated in Section 1.2.

Finally, we wonder about the possible future of these technologies in the classroom. Before risking this perilous exercise, examining the present situation will give us a good start. Today, the computer has entered the classroom. Long ago, the old blackboard, and dusty, inexpensive chalk, were replaced by the new whiteboard, and the solvent-free, expensive marker. Cleanliness replaced expressiveness (try to obtain with a marker the subtle stroke effects you can achieve with chalk). But the whiteboard triumph was short. Now, the poor whiteboard is often used as a projection screen that reflects digital slides projected from a computer. Thus, in a way, students are now subject to conditions quite equivalent to those of a movie theater: a bright screen displaying nice pictures, a voice commenting what is on the screen, darkness for a good contrast, and, unfortunately, either passivity (watching the show), or discouragement (too much information, too fast, therefore impossible to take notes). Indirectly, this also has consequences on teachers: the pedagogical drawing know-how slowly disappears with the retiring faculty members.

Reintroducing writing and drawing as fundamental learning tools is one solution for reversing the trend. In the classroom of tomorrow, the motto could be "drawing as a front-end to everything" (Gross and Do, 1996), either as traditional paper and pencil, as computer-assisted two-dimensional input, or even as something we cannot imagine yet. There are so many possibilities. Drawing: not only because it is one of the few thought supporting activities;[1] not only because it is one of the few elementary interfaces;[2] but also simply because it is part of our human nature.[3]

[1] See, for example, Verstijnen et al. (1996), Suwa and Tversky (1996), Edmonds and Moran (1997).

[2] "Paper and pencil is a direct manipulation interface *par excellence* — you draw what you want, where you want it, and how you want it to look. Structured mouse-menu interactions force designers into premature commitment, demand inappropriate precision, and are tedious to use compared with pencil and paper." (Gross and Do, 1996).

[3] In the Chauvet–Pont-d'Arc cave (Ardèche, France), charcoal drawings have been dated between 30,340 and 32,410 years BP.

Appendix A

What is Ergonomics?

This is the full transcription of the text presented on the web site of the International Ergonomics Association (IEA).

"In August 2000, the IEA Council adopted an official definition of ergonomics as shown below:

The Discipline of Ergonomics

Ergonomics (or human factors) is the scientific discipline concerned with the understanding of interactions among humans and other elements of a system, and the profession that applies theory, principles, data and methods to design in order to optimize human well-being and overall system performance.

Ergonomists contribute to the design and evaluation of tasks, jobs, products, environments and systems in order to make them compatible with the needs, abilities and limitations of people.

Domains of Specialization

Derived from the Greek $\epsilon\rho\gamma o\nu$ (work) and $\nu o\mu o\varsigma$ (laws) to denote the science of work, ergonomics is a systems-oriented discipline which now extends across all aspects of human activity. Practicing ergonomists must have a broad understanding of the full scope of the discipline. That is, ergonomics promotes a holistic approach in which considerations of physical, cognitive, social, organizational, environmental and other relevant factors are taken into account. Ergonomists often work in particular economic sectors or application domains. Application domains are not mutually exclusive and they evolve constantly; new ones are created and old ones take on new perspectives.

There exist domains of specialization within the discipline, which represent deeper competencies in specific human attributes or characteristics of human interaction. Domains of specialization within the discipline of ergonomics are broadly the following:

Physical ergonomics is concerned with human anatomical, anthropometric, physiological and biomechanical characteristics as they relate to physical activity. (Relevant topics include working postures, materials handling, repetitive movements, work related musculoskeletal disorders, workplace layout, safety and health.)

Cognitive ergonomics is concerned with mental processes, such as perception, memory, reasoning, and motor response, as they affect interactions among humans and other elements of a system. (Relevant topics include mental workload, decision-making, skilled performance, human-computer interaction, human reliability, work stress and training as these may relate to human-system design.)

Organizational ergonomics is concerned with the optimization of sociotechnical systems, including their organizational structures, policies, and processes. (Relevant topics include communication, crew resource management, work design, design of working times, teamwork, participatory design, community ergonomics, cooperative work, new work paradigms, virtual organizations, telework, and quality management.)"

Appendix B

Bézier Curves and Surfaces

We present below a reminder of the mathematics of Bézier curves and surfaces. For an in-depth treatment of the subject, we refer the reader to the latest edition of the classic book by Farin (2001). The NURBS curves and surfaces could be an alternative representation: the interested reader will find sufficient details in another book by Farin (1999).

The Bernstein polynomials $B_{i,n}$ of degree n (order $n+1$), are given by

$$B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i} \quad \text{with} \quad \binom{n}{i} = \frac{n!}{(n-i)!\, i!} \qquad i \in \{0, \ldots, n\}$$

where $\binom{n}{i}$ is a binomial coefficient.

The Bézier curve Q_n of degree n, of a set of $n+1$ control points $V_0, V_1, \ldots, V_n$, is written

$$\mathbf{Q}_n(u) = \sum_{i=0}^{n} B_{i,n}(u)\ \mathbf{V}_i \qquad u \in [0,1]$$

and its kth derivative $\frac{d^k Q_n}{du^k}$ can be written

$$\frac{d^k\, \mathbf{Q}_n(u)}{du^k} = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} B_{i,n-k}(u)\ \Delta^k \mathbf{V}_i \quad \text{with} \quad \Delta^k \mathbf{V}_i = \Delta^{k-1}\mathbf{V}_{i+1} - \Delta^{k-1}\mathbf{V}_i$$

thus, for a cubic Bézier curve ($n = 3$), we have the following equations:

$$\begin{aligned}
\mathbf{Q}_3(u) &= (1-u)^3\,\mathbf{V}_0 \ +\ 3u\,(1-u)^2\,\mathbf{V}_1 \ +\ 3\,(1-u)\,u^2\,\mathbf{V}_2 \ +\ u^3\,\mathbf{V}_3 \\
\dot{\mathbf{Q}}_3(u) &= 3\left[(1-u)^2\,(\mathbf{V}_1 - \mathbf{V}_0) \ +\ 2u\,(1-u)\,(\mathbf{V}_2 - \mathbf{V}_1) \ +\ u^2\,(\mathbf{V}_3 - \mathbf{V}_2)\right] \\
\ddot{\mathbf{Q}}_3(u) &= 6\left[(1-u)\,(\mathbf{V}_2 - 2\mathbf{V}_1 + \mathbf{V}_0) \ +\ u\,(\mathbf{V}_3 - 2\mathbf{V}_2 + \mathbf{V}_1)\right]
\end{aligned}$$

where $\dot{Q}_3$ and $\ddot{Q}_3$ are first and second derivatives of Q_3 w.r.t. u. The cubic Bézier curve has at most one inflection point I, it can be easily determined with the value of the parameter u on that point, i.e.,

$$\mathbf{I} = \mathbf{Q}_3(u_I) \quad \Longleftrightarrow \quad \ddot{\mathbf{Q}}_3(u_I) = \mathbf{O} \quad \Longleftrightarrow \quad [u_I \; u_I]^T = \frac{-\mathbf{V}_2 + 2\mathbf{V}_1 - \mathbf{V}_0}{\mathbf{V}_3 - 3\mathbf{V}_2 + 3\mathbf{V}_1 - \mathbf{V}_0}$$

where O is the origin of the two-dimensional Cartesian orthonormed coordinate system. If I exists, the last equation is verified for all coordinates of the control points V_i.

A piecewise cubic Bézier curve S_3 is defined as a sequence of n cubic Bézier curves: each cubic curve (with the exception of the first and the last of the sequence), shares its first (resp. last) control point with the previous (resp. succeeding) cubic curve. The parameterization of the piecewise cubic curve is given by

$$\mathbf{S}_3(t) = \mathbf{Q}_3^i(u) \quad \text{with} \quad u = \frac{t - t_i}{t_{i+1} - t_i} \qquad u \in [0,1] \quad \text{and} \quad i \in \{0, \ldots, n-1\}$$

where Q_3^i is the ith cubic Bézier curve of the sequence, and $t_0, t_1, \ldots, t_n$ is called the "knots" sequence.

As a generalization of the Bézier curve equation to two parameters, the Bézier surface $Q_{n,m}$ of degrees n and m of a set of $(n+1)\times(m+1)$ control points $V_{00}, V_{01}, \ldots, V_{nm}$, is written

$$\mathbf{Q}_{n,m}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) \; B_{j,m}(v) \; \mathbf{V}_{ij}$$

Bibliography

Adobe Systems (2002). Adobe Photoshop 7.0. The professional image-editing standard software.
http://www.adobe.com (cited on p. 12)

Akeo, M., Hashimoto, H., Kobayashi, T. and Shibusawa, T. (1994). Computer graphics system for reproducing three-dimensional shape from idea sketch, *Computer Graphics Forum* 13(3): 477–488. (cited on p. 61, 62, 69, 123)

Alias|wavefront (2002a). Fake or Foto? A visual Turing's test: Is the picture a photograph of something real, or an image generated by computer software?
http://www.fakeorfoto.com (cited on p. 10)

Alias|wavefront (2002b). Maya 2.5 Artisan. Suite of integrated pressure-sensitive brush tools: sculpting; attribute, selection, and script painting; 3D paint.
http://www.aliaswavefront.com (cited on p. 92, 93, 94, 95, 124)

Alias|wavefront (2002c). Maya 2.5 Paint Effects. Paint technology for creating natural detail on 2D canvas, in true 3D space, and directly onto textures.
http://www.aliaswavefront.com (cited on p. 59, 60, 94, 123)

Amenta, N., Choi, S. and Kolluri, R. K. (2001). The powercrust, *Proceedings of the Sixth Symposium on Solid Modeling and Application*, ACM Press, New York, pp. 249–260. (cited on p. 96, 97)

Barzel, R. (1992). *Physically-Based Modeling for Computer Graphics: A Structured Approach*, Academic Press, London. (cited on p. 32)

Beall, J., Doppelt, A. and Hughes, J. F. (1997). Developing an interactive illustration: Using java and the web to make it all worthwhile, *in* R. Earnshaw and J. Vince (eds), *The Internet in 3D: Information, Images, and Interaction*, Academic Press, London, pp. 55–63. (cited on p. 49)

Blinn, J. F. (1990). Jim Blinn's corner: The ultimate design tool, *IEEE Computer Graphics and Applications* 10(6): 90–92. (cited on p. 14)

Boden, M. A. (1992). *The Creative Mind: Myths & Mechanisms*, Basic Books, New York. (cited on p. 16)

Bolas, M. T. (1994). Human factors in the design of an immersive display, *IEEE Computer Graphics and Applications* 14(1): 55–59. (cited on p. 57)

Bourguignon, D. and Cani, M.-P. (2000). Controlling anisotropy in mass-spring systems, *Proceedings of the Eleventh Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 113–123. (cited on p. 20)

Bourguignon, D., Cani, M.-P. and Drettakis, G. (2001). Drawing for illustration and annotation in 3D, *Computer Graphics Forum* 20(3): 114–122. (cited on p. 53)

Boux de Casson, F. (2000). *Simulation dynamique de corps biologiques et changements de topologie interactifs*, PhD thesis, Université de Savoie, France. (cited on p. 34, 45, 46)

Branco, V., Costa, A. and Ferreira, F. N. (1994). Sketching 3D models with 2D interaction devices, *Computer Graphics Forum* 13(3): 489–502. (cited on p. 62, 69)

Bro-Nielsen, M. and Cotin, S. (1996). Real-time volumetric deformable models for surgery simulation using finite elements and condensation, *Computer Graphics Forum* 15(3): 57–66. (cited on p. 26, 28)

Bronshtein, I. N. and Semendyayev, K. A. (1998). *Handbook of Mathematics*, Springer-Verlag, Berlin, p. 554. (cited on p. 71)

Butterworth, J., Davidson, A., Hench, S. and Olano, M. T. (1992). 3DM: A three dimensional modeler using a head-mounted display, *Proceedings of the Second ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 135–138. (cited on p. 55)

Buxton, B. (1997). Artists and the art of the luthier, *Computer Graphics* 31(1): 10–11. (cited on p. 14)

CGAL (2002). Computational Geometry Algorithms Library, version 2.4. Useful, reliable geometric algorithms in a C++ library.
http://www.cgal.org (cited on p. 97, 99)

Chadwick, J. E., Haumann, D. R. and Parent, R. E. (1989). Layered construction for deformable animated characters, *Computer Graphics* 23(3): 243–252. (cited on p. 20, 30, 31, 38)

Chen, D. T. and Zeltzer, D. (1992). Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method, *Computer Graphics* 26(2): 89–98. (cited on p. 25)

Chen, Y., Zhu, Q. and Kaufman, A. (1998). Physically-based animation of volumetric objects., *Proceedings of IEEE Computer Animation 98*, IEEE Computer Society, Los Alamitos, California, pp. 154–160. (cited on p. 38, 43)

Chilvers, I., Osborne, H. and Farr, D. (eds) (1997). *The Oxford Dictionary of Art*, second edn, Oxford University Press, Oxford. (cited on p. 86)

Cignoni, P., Montani, C. and Scopigno, R. (1997). Computer-assisted generation of bas- and high-reliefs, *Journal of Graphics Tools* 2(3): 15–28. (cited on p. 88)

Cohen, J. M., Hughes, J. F. and Zeleznik, R. C. (2000). Harold: A world made of drawings, *Proceedings of the First International Symposium on Non Photorealistic Animation and Rendering*, ACM Press, New York, pp. 83–90. (cited on p. 66, 68, 69, 123)

Cohen, J. M., Markosian, L., Zeleznik, R. C., Hughes, J. F. and Barzel, R. (1999). An interface for sketching 3D curves, *Proceedings of the Fifth ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 17–22. (cited on p. 58, 59, 68)

Cotin, S., Delingette, H. and Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation, *IEEE Transactions on Visualization and Computer Graphics* 5(1): 62–73. (cited on p. 19, 27)

da Vinci, L. (1956). *Treatise on Painting*, McMahon, Princeton, New Jersey. A. Philip (ed.). (cited on p. 85, 86)

Daily, J. and Kiss, K. (1995). 3D painting: Paradigms for painting in a new dimension, *Proceedings of ACM CHI 95*, Vol. 2 of *Short Papers*, pp. 296–297. (cited on p. 95)

Daniels, E. (1999). Deep canvas in Disney's Tarzan, *Proceedings of ACM SIGGRAPH 99: Conference abstracts and applications*, ACM Press, New York, p. 200. (cited on p. 58, 60, 69, 123)

Debunne, G. (2000). *Animation multirésolution d'objets déformables en temps-réel, Application à la simulation chirurgicale*, PhD thesis, Institut National Polytechnique de Grenoble, France. (cited on p. 21, 28, 29, 34, 47, 49)

Debunne, G., Desbrun, M., Barr, A. and Cani, M.-P. (1999). Interactive multiresolution animation of deformable models, *Proceedings of the Tenth Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 133–144. (cited on p. 19)

Debunne, G., Desbrun, M., Cani, M.-P. and Barr, A. H. (2001). Dynamic real-time deformations using space & time adaptive sampling, *Proceedings of ACM SIGGRAPH 2001*, ACM Press, New York, pp. 31–36. (cited on p. 19, 27, 29, 47)

Deering, M. F. (1995). HoloSketch: A virtual reality sketching/animation tool, *ACM Transactions on Computer-Human Interaction* 2(3): 220–238. (cited on p. 55, 57)

Desbrun, M. and Gascuel, M.-P. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies, *Proceedings of the Seventh Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 61–76. (cited on p. 32)

Desbrun, M., Schröder, P. and Barr, A. (1999). Interactive animation of structured deformable objects, *Proceedings of Graphics Interface 99*, Canadian Human-Computer Communications Society, Toronto, pp. 1–8. (cited on p. 47)

Deussen, O., Kobbelt, L. and Tucke, P. (1995). Using simulated annealing to obtain good nodal approximations of deformable objects, *Proceedings of the Sixth Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 30–43. (cited on p. 34, 47)

Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes, *ACM Transactions on Graphics* 13(1): 43–72. (cited on p. 96, 97)

Edmonds, E. A. and Moran, T. P. (1997). Interactive systems for supporting the emergence of concepts and ideas, *Proceedings of ACM CHI 97*, Vol. 2 of *Workshops*, p. 233. (cited on p. 106)

Eggli, L., Brüderlin, B. D. and Elber, G. (1995). Sketching as a solid modeling tool, *Proceedings of the Third ACM Symposium on Solid Modeling and Applications*, ACM Press, New York, pp. 313–322. (cited on p. 62, 64, 69, 123)

Eggli, L., Hsu, C., Brüderlin, B. D. and Elber, G. (1997). Inferring 3D models from freehand sketches and constraints, *Computer-aided Design* 29(2): 101–112. (cited on p. 62)

Eysenck, H. J. (1994). The measurement of creativity, *in* M. A. Boden (ed.), *Dimensions of Creativity*, MIT Press, Cambridge, Massachusetts. (cited on p. 16)

Farin, G. E. (1999). *NURBS: From Projective Geometry to Practical Use*, second edn, A. K. Peters, Wellesley, Massachusetts. (cited on p. 109)

Farin, G. E. (2001). *Curves and Surfaces for CAGD: A Practical Guide*, fifth edn, Academic Press, New York. (cited on p. 109)

Fekete, J.-D. and Plaisant, C. (1999). Excentric labeling: Dynamic neighborhood labeling for data visualization, *Proceedings of ACM CHI 99*, Vol. 1 of *Navigation and Visualization*, pp. 512–519. (cited on p. 81)

Felger, W. (1995). Head-coupled display system – research issues on health aspects, *Proceedings of the Sixth International Conference on Human-Computer Interaction*, Vol. Ergonomics and Health Aspects of Work with Computers, pp. 593–598. (cited on p. 57)

Freud, S. (1964). Leonardo da Vinci and a memory of his childhood, *in* J. Strachey (ed.), *Five Lectures on Psycho-Analysis, Leonardo da Vinci and Other Works (1910)*, second edn, Vol. 11 of *The Standard Edition of the Complete Psychological Works of Sigmund Freud*, The Hogarth Press and the Institute of Psycho-Analysis, London, pp. 63–137. (cited on p. 85)

Fung, Y. C. (1965). *Foundations of Solid Mechanics*, Prentice Hall, Englewood Cliffs, New Jersey. (cited on p. 21)

Garland, M. and Heckbert, P. S. (1995). Fast polygonal approximation of terrains and height fields, *Technical Report CMU-CS-95-181*, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania. (cited on p. 101, 103)

Gibson, S. and Mirtich, B. (1997). A survey of deformable modeling in computer graphics, *Technical Report TR-97-19*, Mitsubishi Electric Research Laboratories, Cambridge, Massachusetts. (cited on p. 21)

Goel, V. (1995). *Sketches of Thought*, MIT Press, Cambridge, Massachusetts. (cited on p. 15)

Gourret, J.-P., Thalmann, N. M. and Thalmann, D. (1989). Simulation of object and human skin deformations in a grasping task, *Computer Graphics*, Vol. 23, pp. 21–30. (cited on p. 25)

Gross, M. D. and Do, E. Y.-L. (1996). Ambiguous intentions: A paper-like interface for creative design, *Proceedings of the Ninth ACM Symposium on User Interface Software and Technology*, pp. 183–192. (cited on p. 15, 106)

Guckenberger, D. and Stanney, K. M. (1995). Poor man's virtual reality, *Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting*, Vol. 2 of *Demonstrations*, p. 1063. (cited on p. 57)

Han, S. and Medioni, G. (1997). 3D Sketch: Modeling by digitizing with a smart 3D pen, *Proceedings of The Fifth ACM International Conference on Multimedia*, ACM Press, New York, pp. 41–50. (cited on p. 55, 57)

Hanrahan, P. and Haeberli, P. (1990). Direct WYSIWYG painting and texturing on 3D shapes, *Computer Graphics* 24(4): 215–223. (cited on p. 95)

Hapeman, E. R., Zeuch, W. R., Crandall, J. A., Carioti, S. M., Barclay, S. D. and Underkoffler, C. (2001). Telecom Glossary 2000 – American National Standard T1.523-2001.
http://www.atis.org/tg2k/ (cited on p. 9, 89)

Hertzmann, A. and Zorin, D. (2000). Illustrating smooth surfaces, *Proceedings of ACM SIGGRAPH 2000*, ACM Press, New York, pp. 517–526. (cited on p. 75)

Hill, R. (2002). Froguts. The first online virtual frog dissection.
http://www.froguts.com (cited on p. 16)

Hollister, S. J. (2002). BME/ME 456 Biomechanics course notes. University of Michigan, College of Engineering.
http://www.engin.umich.edu/class/bme456/ (cited on p. 21)

Hunter, P. J. (1995). Myocardial constitutive laws for continuum mechanics models of the heart, *Advances in Experimental Medicine and Biology* 382: 303–318. (cited on p. 19)

Hutchinson, D., Preston, M. and Hewitt, T. (1996). Adaptive refinement for mass-spring simulation, *Proceedings of the Seventh Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 31–45. (cited on p. 34)

Igarashi, T., Matsuoka, S. and Tanaka, H. (1999). Teddy: A sketching interface for 3D freeform design, *Proceedings of ACM SIGGRAPH 99*, Addison-Wesley, Boston, Massachusetts, pp. 409–416. (cited on p. 65, 67, 69, 88, 95, 97, 99, 123)

Illich, I. (1981). *Shadow Work*, Marion Boyars, Boston, Massachusetts, chapter Research by People, pp. 86–87. (cited on p. 3)

International Ergonomics Association (2000). What is ergonomics? *http://www.iea.cc/ergonomics/* (cited on p. 13, 15)

James, D. L. and Pai, D. K. (1999). ArtDefo: Accurate real time deformable objects, *Proceedings of ACM SIGGRAPH 99*, Addison-Wesley, Boston, Massachusetts, pp. 65–72. (cited on p. 26, 28)

Johnston, S. F. (2002). Lumo: Illumination for cel animation, *Proceedings of the Second International Symposium on Non-photorealistic Animation and Rendering*, ACM Press, New York, pp. 45–52. (cited on p. 92, 93, 95, 124)

Keefe, D. F., Feliz, D. A., Moscovich, T., Laidlaw, D. H. and LaViola Jr., J. J. (2001). CavePainting: A fully immersive 3D artistic medium and interactive experience, *Proceedings of the Sixth ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 85–93. (cited on p. 55, 56, 57)

Koehoorn, M., Demers, P. A., Hertzman, C., Village, J. and Kennedy, S. M. (2001). Work organization and musculoskeletal injuries among a cohort of health care workers, *Technical Report 126*, Institute for Work & Health, Toronto. *http://www.iwh.on.ca* (cited on p. 14)

@Last Software (2002). SketchUp 2.0. Intuitive and accessible 3D design tool. *http://www.sketchup.com* (cited on p. 15)

Lee, Y., Terzopoulos, D. and Waters, K. (1995). Realistic face modeling for animation, *Proceedings of ACM SIGGRAPH 95*, Addison-Wesley, Boston, Massachusetts, pp. 55–62. (cited on p. 30, 31, 33, 39)

Lipson, H. and Shpitalni, M. (1996). Optimization-based reconstruction of a 3D object from a single freehand line drawing, *Computer-aided Design* 28(8): 651–663. (cited on p. 63, 64, 69, 123)

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics* 21(4): 163–169. (cited on p. 82)

Louchet, J., Provot, X. and Crochemore, D. (1995). Evolutionary identification of cloth animation models, *Proceedings of the Sixth Eurographics Workshop on Animation and Simulation*, Springer-Verlag, Berlin, pp. 44–54. (cited on p. 33, 47)

Luciani, A., Jimenez, S., Florens, J. L., Cadoz, C. and Raoult, O. (1991). Computational physics: A modeler-simulator for animated physical objects, *Proceedings of Eurographics 91*, North-Holland, Amsterdam, pp. 425–436. (cited on p. 20)

Luskin, J. (1993). Man vs. mouse, *Proceedings of ACM SIGGRAPH 93*, ACM Press, New York, p. 401. (cited on p. 13)

Markosian, L., Kowalski, M. A., Trychin, S. J., Bourdev, L. D., Goldstein, D. and Hughes, J. F. (1997). Real-time nonphotorealistic rendering, *Proceedings of ACM SIGGRAPH 97*, Addison-Wesley, Boston, Massachusetts, pp. 415–420. (cited on p. 75)

Marks, J., Cohen, M., Ngo, J. T., Shieber, S. and Snyder, J. (1994). Optimization – an emerging tool in computer graphics, *Proceedings of ACM SIGGRAPH 94*, ACM Press, New York, pp. 483–484. (cited on p. 7)

Meseure, P. and Chaillou, C. (2000). A deformable body model for surgical simulation, *The Journal of Visualization and Computer Animation* 11(4): 197–208. (cited on p. 33)

Miller, G. and Pearce, A. (1989). Globular dynamics: A connected particle system for animating viscous fluids, *Computers and Graphics* 13(3): 305–309. (cited on p. 32)

Miller, G. S. P. (1988). The motion dynamics of snakes and worms, *Computer Graphics* 22(4): 169–178. (cited on p. 20, 30, 32, 33)

Ng-Thow-Hing, V. and Fiume, E. (1997). Interactive display and animation of B-spline solids as muscle shape primitives, *Proceedings of the Eighth Eurographics Workshop on Computer Animation and Simulation*, Springer-Verlag, Berlin. (cited on p. 31, 32)

Norman, D. A. (1988). *The Psychology of Everyday Things*, Basic Books, New York. (cited on p. 12, 13, 15, 17)

Norman, D. A. (1993). *Things that Make Us Smart: Defending Human Attributes in the Age of the Machine*, Perseus Books, Cambridge, Massachusetts. (cited on p. 14)

O'Brien, J. F. and Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture, *Proceedings of ACM SIGGRAPH 99*, ACM Press, New York, pp. 137–146. (cited on p. 26, 28)

of Saint Victor, H. (1966). *Epitome Dindimi in philosophiam, in* R. Baron (ed.), *Opera Propaedeutica, Practica Geometriae, De grammatica, Epitome Dindimi in philosophiam*, Vol. 20, University of Notre Dame Publications in Mediaeval Studies, Notre Dame, Indiana, pp. 167–207. (cited on p. 3)

Oh, B. M., Chen, M., Dorsey, J. and Durand, F. (2001). Image-based modeling and photo editing, *Proceedings of ACM SIGGRAPH 2001*, ACM Press, New York, pp. 433–442. (cited on p. 100)

Ohayon, J., Bourdarias, C., Gerbi, S. and Oddou, C. (2001). A finite element method for the active cardiac muscle, *Proceedings of the Fifth International Symposium on Computer Methods in Biomechanics and Biomedical Engineering*. (cited on p. 21, 46)

Pascarelli, E. F. (1994). *Repetitive Strain Injury: A Computer User's Guide*, John Wiley, New York. (cited on p. 14)

Pausch, R. and Crea, T. (1992). A literature survey for virtual environments: Military flight simulator visual systems and simulator sickness, *Technical Report CS-92-25*, Department of Computer Science, University of Virginia. (cited on p. 57)

Pentland, A. and Williams, J. (1989). Good vibrations: Modal dynamics for graphics and animation, *Computer Graphics* 23(3): 215–222. (cited on p. 27, 28)

Picinbono, G., Delingette, H. and Ayache, N. (2000). Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model, *Proceedings of MICCAI 2000*, Springer-Verlag, Berlin, pp. 643–652. (cited on p. 27, 28, 29)

Picinbono, G., Delingette, H. and Ayache, N. (2001). Non-linear and anisotropic elastic soft tissue models for medical simulation, *Proceedings of IEEE ICRA 2001*. (cited on p. 27, 28)

Pixologic (2002). ZBrush 1.5. 2D and 3D painting, texturing and sculpting tool. *http://www.pixologic.com* (cited on p. 93, 94, 95, 124)

Platt, J. C. and Barr, A. H. (1988). Constraint methods for flexible models, *Computer Graphics* 22(4): 279–288. (cited on p. 41)

Preim, B., Ritter, A. and Strothotte, T. (1996). Illustrating anatomic models — A semi-interactive approach, *Lecture Notes in Computer Science* 1131: 23–32. (cited on p. 81)

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). Integration of ODE: Second-order conservative equations, *Numerical Recipes in C*, second edn, Cambridge University Press, Cambridge, pp. 732–734. (cited on p. 42)

Promayon, E., Baconnier, P. and Puech, C. (1996). Physically-based deformations constrained in displacements and volume, *Proceedings of Eurographics 96*, Blackwell Publishers, Oxford, pp. 155–164. (cited on p. 41)

Provot, X. (1995). Deformation constraints in a mass-spring model to describe rigid cloth behavior, *Proceedings of Graphics Interface 95*, Canadian Human-Computer Communications Society, Toronto, pp. 147–154. (cited on p. 20, 30, 33)

Pugh, D. (1992). Designing solid objects using interactive sketch interpretation, *Proceedings of the Second ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 117–126. (cited on p. 61, 69, 123)

Putz-Anderson, V., Bernard, B. P., Burt, S. E., Cole, L. L., Fairfield-Estill, C., Fine, L. J., Grant, K. A., Gjessing, C., Jenkins, L., Jr., J. J. H., Nelson, N., Pfirman, D., Roberts, R., Stetson, D., Haring-Sweeney, M. and Tanaka, S. (1997). Musculoskeletal disorders and workplace factors: A critical review of epidemiologic evidence for work-related musculoskeletal disorders of the neck, upper extremity, and low back, *Technical Report 97-141*, U.S. Department of Health and Human Services – National Institute for Occupational Safety and Health. B. P. Bernard (ed.). *http://www.cdc.gov/niosh/ergosci1.html* (cited on p. 14)

Raskar, R. and Cohen, M. (1999). Image precision silhouette edges, *Proceedings of the Fifth ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 135–140. (cited on p. 75)

Sachs, E., Roberts, A. and Stoops, D. (1991). 3-Draw: A tool for designing 3D shapes, *IEEE Computer Graphics and Applications* 11(6): 18–26. (cited on p. 54, 56)

Sagar, M. A., Bullivant, D., Mallinson, G. D., Hunter, P. J. and Hunter, I. W. (1994). A virtual environment and model of the eye for surgical simulation, *Proceedings of ACM SIGGRAPH 94*, ACM Press, New York, pp. 205–213. (cited on p. 26)

Sakaguchi, H. and Sakakibara, M. (2001). Final Fantasy: The Spirits Within. USA-Japan production, color, 106 min. (cited on p. 10)

Schkolne, S., Pruett, M. and Schroder, P. (2001). Surface drawing: Creating organic 3D shapes with the hand and tangible tools, *Proceedings of ACM CHI 2001*, Tangible Interfaces, ACM Press, New York, pp. 261–268. (cited on p. 57)

Schneider, P. J. (1990a). An algorithm for automatically fitting digitized curves, *in* A. S. Glassner (ed.), *Graphics Gems*, Academic Press, New York, pp. 612–626. (cited on p. 70)

Schneider, P. J. (1990b). Solving the nearest-point-on-curve problem, *in* A. S. Glassner (ed.), *Graphics Gems*, Academic Press, New York, pp. 607–611. (cited on p. 71)

Simulog (2002). TetMesh-GHS3D. Tetrahedrical mesh generator and optimizer. *http://www.simulog.fr* (cited on p. 32, 36)

Stanney, K. M. and Kennedy, R. S. (1997). The psychometrics of cybersickness, *Communications of the ACM* 40(8): 66–68. (cited on p. 57)

Strothotte, T., Preim, B., Raab, A., Schumann, J. and Forsey, D. R. (1994). How to render frames and influence people, *Computer Graphics Forum* 13(3): 455–466. (cited on p. 69)

SUMMIT (2002a). Simbryo. Stanford University. *http://simbryo.stanford.edu* (cited on p. 16)

SUMMIT (2002b). Virtual creatures. Stanford University. *http://k-2.stanford.edu/creatures/* (cited on p. 16)

Sun microsystems (2002). Ahead of the pack: Ivan Sutherland. *http://www.sun.com/960710/feature3/ivan.html* (cited on p. 10, 53)

Sutherland, I. E. (1963). Sketchpad: A man-machine graphical communication system, *Proceedings AFIPS Spring Joint Computer Conference*, Vol. 23, American Federation of Information Processing Societies Press, pp. 329–346. (cited on p. 53, 54, 59)

Sutherland, I. E. (1968). A head-mounted three-dimensional display, *Proceedings AFIPS Fall Joint Computer Conference*, Vol. 33, Thompson Books, pp. 757–764. (cited on p. 53, 54)

Suwa, M. and Tversky, B. (1996). What architects see in their sketches: Implications for design tools, *Proceedings of ACM CHI 96*, Vol. 2 of *Short papers*, ACM Press, New York, pp. 191–192. (cited on p. 106)

Szeliski, R. and Tonnesen, D. (1992). Surface modeling with oriented particle systems, *Computer Graphics* 26(2): 185–194. (cited on p. 32)

Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K. (1987). Elastically deformable models, *Computer Graphics* 21(4): 205–214. (cited on p. 27)

Terzopoulos, D., Platt, J. and Fleischer, K. (1989). Heating and melting deformable models (from goop to glop), *Proceedings of Graphics Interface 89*, Canadian Human-Computer Communications Society, Toronto, pp. 219–226. (cited on p. 31)

The MathWorks (2002). MATLAB 6.5 and Image Processing Toolbox 3.2. Flexible environment for technical computing and toolbox with functions for enhancing and analyzing digital images. *http://www.mathworks.com* (cited on p. 100, 102)

Tolba, O., Dorsey, J. and McMillan, L. (1999). Sketching with projective 2D strokes, *Proceedings of the Twelfth ACM Symposium on User Interface Software and Technology*, ACM Press, New York, pp. 149–158. (cited on p. 58, 59, 69)

Tolba, O., Dorsey, J. and McMillan, L. (2001). A projective drawing system, *Proceedings of the Sixth ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 25–34. (cited on p. 58)

Usson, Y., Parazza, F., Jouk, P. S. and Michalowicz, G. (1994). Method for the study of the three-dimensional orientation of the nuclei of myocardial cells in fetal human heart by means of confocal scanning laser microscopy, *Journal of Microscopy* 174(2): 101–110. (cited on p. 19)

van Gelder, A. (1998). Approximate simulation of elastic membranes by triangulated spring meshes, *Journal of Graphics Tools* 3(2): 21–41. (cited on p. 34)

van Overveld, C. W. A. M. (1996). Painting gradients: Free-form surface design using shading patterns, *Proceedings of Graphics Interface 96*, Canadian Human-Computer Communications Society, Toronto, pp. 151–158. (cited on p. 90, 91, 95, 124)

van Overveld, C. W. A. M. and Wyvill, B. (1997). Polygon inflation for animated models: A method for the extrusion of arbitrary polygon meshes, *The Journal of Visualization and Computer Animation* 8(1): 3–16. (cited on p. 88)

Verstijnen, I. M., Stuyver, R., Hennessey, J. M., van Leeuwen, C. C. and Hamel, R. (1996). Considerations for electronic idea-creation tools, *Proceedings of ACM CHI 96*, Vol. 2 of *Short papers*, ACM Press, New York, pp. 197–198. (cited on p. 106)

Walsh, R. J., Raskin, S., Sherif, M. F. and Jurjus, A. R. (2002). NetAnatomy. Web site to teach human anatomy to students of the health professions. *http://www.netanatomy.com* (cited on p. 16)

Wang, W. and Grinstein, G. G. (1993). A survey of 3D solid reconstruction from 2D projection line drawings, *Computer Graphics Forum* 12(2): 137–158. (cited on p. 61)

Watson, D. F. (1997). Finding non-convex hulls, *Proceedings of Geodynamics and Ore Deposits Conference*, Australian Geodynamics Cooperative Research Centre, Ballarat, Victoria, Australia, p. 101. *http://www.agcrc.csiro.au/publications/conferences/Ballarat97/posters/watson.html* (cited on p. 97, 98)

Williams, L. (1990). 3D paint, *Proceedings of the First ACM Symposium on Interactive 3D Graphics*, ACM Press, New York, pp. 225–234. (cited on p. 86, 88, 89, 90, 91, 95, 124)

Williams, L. (1991). Shading in two dimensions, *Proceedings of Graphics Interface 91*, Canadian Human-Computer Communications Society, Toronto, pp. 143–151. (cited on p. 89, 90, 92, 95, 99, 100, 124)

Williams, L. (1998). Image jets, level sets, & silhouettes – visible means of support. *http://graphics.stanford.edu/workshops/ibr98/Talks/Lance/silhouettes.html* (cited on p. 91, 92, 95, 99, 124)

Williams, T., Kelley, C., Lang, R., Kotz, D., Campbell, J., Elber, G., Woo, A. and many others (2001). gnuplot 3.7. Command-driven interactive function plotting program. *http://www.gnuplot.info* (cited on p. 46)

Witkin, A. (1999a). Constrained dynamics, *Proceedings of ACM SIGGRAPH 99: Course Notes 36 – Physically Based Modeling*, ACM Press, New York. (cited on p. 41)

Witkin, A. (1999b). Particle system dynamics, *Proceedings of ACM SIGGRAPH 99: Course Notes 36 – Physically Based Modeling*, ACM Press, New York. (cited on p. 36, 42)

Witkin, A., Fleischer, K. and Barr, A. (1987). Energy constraints on parameterized models, *Computer Graphics* 21(4): 225–232. (cited on p. 41)

Witkin, A. and Welch, W. (1990). Fast animation and control of nonrigid structures, *Computer Graphics* 24(4): 243–252. (cited on p. 41)

Zeleznik, R. C., Herndon, K. P. and Hughes, J. F. (1996). SKETCH: An interface for sketching 3D scenes, *Proceedings of ACM SIGGRAPH 96*, Addison-Wesley, Boston, Massachusetts, pp. 163–170. (cited on p. 15, 63, 65, 69, 123)

Zhuang, Y. and Canny, J. (1999). Real-time and physically realistic simulation of global deformation, *Proceedings of ACM SIGGRAPH 99: Conference abstracts and applications*, ACM Press, New York, pp. 270–270. (cited on p. 26)

List of Figures

Contents

Animation interactive et modélisation par le dessin
Applications pédagogiques en médecine

La compréhension et la mémorisation de données visuelles sont une part importante de l'apprentissage des étudiants en médecine. Cependant, la nature tridimensionnelle et dynamique du corps humain pose de nombreux problèmes. Leur solution nécessite de véritables outils informatiques interactifs pour permettre aux étudiants de créer et de manipuler des données complexes. Nous proposons dans ce but plusieurs approches.

Tout d'abord, nous nous sommes intéressés à l'animation par modèles physiques de matériaux élastiques anisotropes. Son utilisation pendant un cours d'anatomie physiologique du myocarde offre la possibilité aux étudiants de construire des échantillons de tissu musculaire cardiaque. Pour atteindre cet objectif, notre modèle présente deux caractéristiques importantes : la première est un faible coût en temps de calcul afin atteindre un affichage interactif ; la seconde est une apparence intuitive qui facilite son contrôle par l'utilisateur.

Ensuite, nous nous sommes intéressés à l'interaction en trois dimensions à l'aide d'interfaces bidimensionnelles, en vue de l'annotation de modèles existants, ou de la création de nouveaux modèles. Cette approche tire parti du fait que le dessin est encore considéré comme une importante méthode d'apprentissage par certains professeurs français d'anatomie. Notre système de dessin 3D possède une représentation des traits de l'utilisateur qui permet l'affichage d'un même dessin sous plusieurs points de vue. Cette représentation est d'ailleurs compatible avec celle de surfaces polygonales existantes, qui peuvent ainsi être annotées. De manière complètement différente, notre outil de modélisation par le dessin utilise conjointement les informations provenant de la géométrie des traits et de l'analyse de l'image produite, afin de créer des modèles en trois dimensions sans passer par une spécification explicite de la profondeur.

Mots-clefs : informatique graphique, dessin, animation par modèles physiques, modélisation, interface.

Interactive Animation and Modeling by Drawing
Pedagogical Applications in Medicine

Medicine is a discipline where visualization is an essential component of learning. However, the three-dimensional, dynamic structure of the human body poses difficult teaching challenges. There is a need for truly interactive computer tools that will enable students to create and manipulate computer models, not just watch them. We propose different approaches with that goal in mind.

We were first interested in interactive physically-based animation of anisotropic elastic materials. One possible application scenario is an anatomy course on heart physiology where students can build interactive samples of cardiac muscular tissue. To achieve this, our model exhibits two key features. The first one is a low computational cost that results in high frame rates; the second one is an intuitive *system image* that ensures easy control by the user.

Next, we were interested in interaction in three dimensions using two-dimensional input, either for annotating existing models, or for creating new models; taking advantage of the fact that drawing practice is still considered a fundamental learning method by some anatomy teachers in the French medical school curriculum. Our 3D drawing system has a stroke representation that enables drawing redisplay when the viewpoint changes. Moreover, this representation can be mixed freely with existing polygonal surfaces for annotation purposes. In contrast, our modeling by drawing tool uses information from both stroke geometry and the drawn image, to allow three-dimensional modeling without explicit depth specification.

Keywords: computer graphics, drawing, physically based animation, modeling, interface.

Spécialité modèles et instruments en médecine et biologie
Projet iMAGIS, laboratoire GRAVIR, INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot Saint Martin, France