Sketching and annotation for the procedural modelling of complex phenomena

Jamie Wither

Institut National Polytechnique de Grenoble

Sketching and annotation for the procedural modelling of complex phenomena

Jamie WITHER

Thèse présentée pour l'obtention du titre de Docteur de l'INP Grenoble
Spécialité Informatique
Arrêté ministériel du 5 juillet 1984 et du 30 mars 1992
Préparée au sein du Laboratoire Jean Kuntzmann. UMR 5224.
Projet de recherche EVASION.
Soutenue le 24 Novembre 2008

Composition du jury :

Loïc	BARTHE	Examinateur
Laurent	GRISONI	Rapporteur
John	HUGHES	Rapporteur
Michiel	VAN DE PANNE	Rapporteur
Marie-Paule	CANI	Directrice de thèse
James	CROWLEY	President du jury

TABLE OF CONTENTS

Introduction

Sketch - A rough drawing or delineation of something, giving the outlines or prominent features without the detail, esp. one intended to serve as the basis of a more finished picture, or to be used in its composition; a rough draught or design. - *Oxford English Dictionary*

“A picture is worth a thousand words” - *Proverb*

What do we mean by sketching?

Drawings pre-date writing[Clo00, Kra88]. Everyone can draw to some degree, though most abandon the practice before adulthood. A drawing no matter how crude can communicate more efficiently than words, and is independent of language. Drawing is the basic skill underpinning most visual arts, whether painting, sculpture, architecture, fashion design, product design or computer generated imagery (CGI).

A sketch is the precursor to a drawing, usually a rapidly created rough version, used to study and understand the thing being drawn, in order to better represent it. With real media, whether the final output is a drawing, painting or sculpture, many sketches may be made before the final composition is attempted. With virtual media too pencil and paper sketches are made before the artist turns to a specialised modelling package to begin creating a model.

Until recently computers weren’t powerful enough to harness the use of drawing as a means of input, specifically: they couldn’t interpret a drawing in a useful fashion. An artist could certainly use a tool like photoshop or illustrator to create an image. But the image would then have no further part in the creation process, except to serve as study and inspiration. The problem of interpreting a drawing in general is still too hard for a computer to solve, especially as there are often many interpretations of a drawing (Figure 1). Therefore even when a sketch is used at the design stage, everything still has to be modelled using general 3D modelling software.

In the film and games industries the time between commencing and delivering a finished product is critical. Many productions require a considerable amount of artist-generated natural content like trees and plants (Fig-

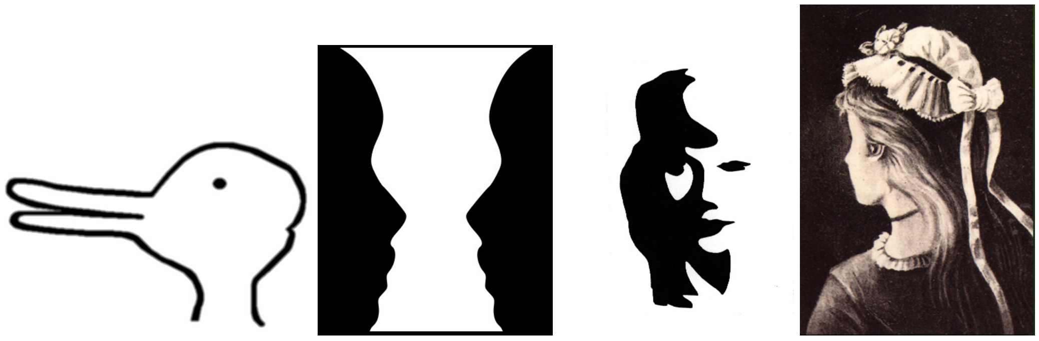


Figure 1: Reversible images provide evidence that we construct the world we perceive. Here the examples are duck/rabbit, faces/vase, sax player/womans face, old lady/young maid.

ure 2), terrain, characters and clothing, but modelling such things requires expertise and time. Systems which reduce the time or experience required are a boon. Sketching systems can address this need.

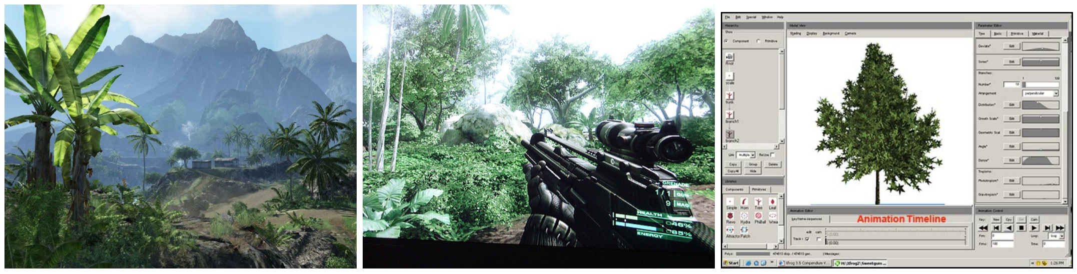


Figure 2: **(left, middle)** In-game images from Crysis [Cry07]. Note the number and variety of trees and plants and the interesting terrain. Modelling these elements typically takes hours and one of the goals of this work is to reduce the time and effort required of the designer through the use of sketch-based techniques. **(right)** A standard tree modelling interface [Gre08].

This thesis explores the use of sketching and annotation for modelling specific complex phenomena, where some prior knowledge can be used to interpret a sketch. The methodology we introduce will be illustrated by the design of sketch-based systems for modelling clothing, hair, clouds and trees. We say “sketching” rather than “drawing” because we treat the user input as rough intentions of form and placement - not always faithfully reproduced but close enough to satisfy the user’s intentions. A sketch is produced quickly, and we want our modelling processes to respond quickly to the user’s input so the process flows smoothly, reducing the time taken between initial ideas and final models.

Automatic shape modelling from a sketch shares many of the problems encountered in the field of computer vision. The human visual system is extraordinarily good at rapidly recognising objects and motion in a scene, under a wide variety of lighting and obscuring conditions. It is so good, in fact, that we take this ability for granted. Many years of study have gone into attempting to enable computers to replicate this feat, and yet we have barely begun to understand the processes supporting our visual system. A detailed discussion of this topic is beyond the scope of this thesis, but we should at least place our theoretical approach in a human visual perception context. The dominant theoretical approach in in contemporary vision research is Constructivism [Yan01], which formed from the ideas of Hermann von Helmholtz (1821–1894). Constructivism holds that we *construct* an internal representation of a scene based on the sensory information from the eyes *and* experience - we have to learn our ability to perceive through experience. Thus our perception of a scene can be viewed as coming from two sources: “bottom-up” information supplied by the eyes, and “top-down” information supplied by memory and prior learning. Visual illusions such as reversible images (Figure 1) provide evidence to support this case. In viewing the young girl/old maid image your visual system gathers enough "bottom-up" local shape cues to make a "top-down" guess as to the nature of the object. Once a likely candidate is chosen your interpretation of the shape cues in the image is re-enforced by this decision. But if you had started your viewing at some other position you may well have chosen the opposite interpretation of the image.

Our approach in modelling complex shapes through sketching is based on this top-down integration of prior knowledge in interpreting shape. We concentrate on using prior knowledge of what is to be modelled in order to extract parameters from sketched features, which can then be used to drive procedural or physical models which generate a (potentially detailed) shape in 3D. The advantage of this approach is that the user need have no knowledge of the underlying modelling techniques. The only requirement is the ability to draw and an understanding of the constraints on what can be drawn.

The artist's approach - from global to local

In addition to a perceptual "top-down" approach, we also take as inspiration the traditional global-to-local approaches used by artists. The global features are sketched in first and proportions and composition are decided upon before local detail is added. Figure 3 illustrates the evolution of a drawing of a tree in stages, Figure 4 illustrates a similar process for drawing a rock; note that the same large scale outlines can have many different interpretations depending on the details added at smaller scales.

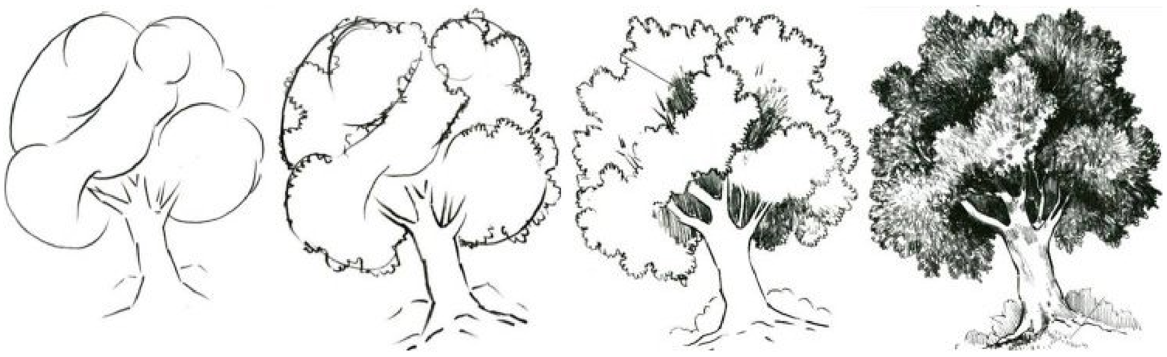


Figure 3: An Illustration of the global to local approach used by artists (from [Pow98]).

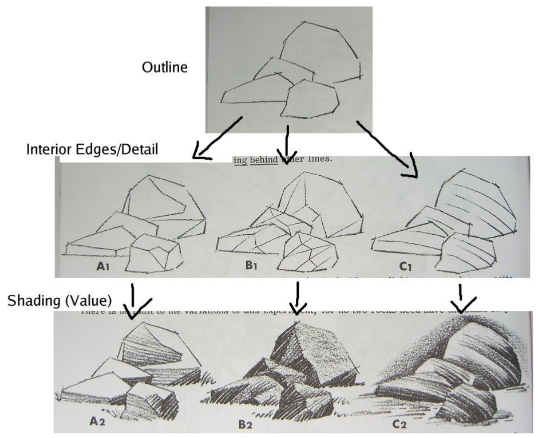


Figure 4: The global to local stages in drawing a rock (from [Ham72]). Notice how the same initial sketch can be refined in different ways at the local scale to get different final shapes.

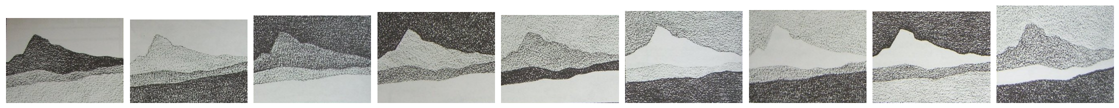


Figure 5: Shading can strongly affect our perception of a scene, but it is also ambiguous. Would two people interpret each of these shaded landscapes in the same way? (from [Ham72]).

It can be seen that at the final stage the artist may use shading effects such as hatching or stippling to add fine detail. These shading effects can certainly have a strong effect on shape interpretation (for example the final stages of the rock construction in Figure 4), however in this thesis we don't attempt to interpret shading effects. This is for a few reasons. First, because many of the shading processes (especially hatching and stippling) are time consuming, these techniques belong more to the final detail stages of production, rather than the earlier exploratory design stages. Second, shading effects are highly dependent on the lighting conditions of a scene, rather than the being easily predictable from the nature of the shape in question. Finally, not everyone is able to use such techniques effectively, and concentrating on their use would exclude potential users of any resulting systems. Certainly some shading techniques can be done quickly and could enhance a sketch based modelling system, but consider Figure 5. Here the shading is the only thing changing, and it allows many different interpretations of the scene presented. Overcoming such ambiguity is a difficult problem worthy of future research. Much useful research has been done in approaching the problem of shading from the opposite direction (non-photo realistic rendering), so once the shape has been determined via a sketch-based modelling approach, detailed shading can be added automatically. For example Figure 6 illustrates automatic determination of hatching lines for shading [HZ00].

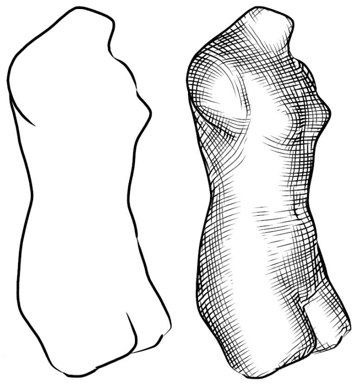


Figure 6: The automatic shading (hatching) technique from [HZ00]. Note that without the shading, occluding contours alone do not give enough information to determine the whole shape.

Contributions

We present a methodology for sketch-based modelling of complex phenomena given various levels of prior knowledge on the phenomena being modelled. This prior knowledge can be as simple as rule-of-thumb constraints or as complex as a full procedural model which describes the phenomenon. This use of prior knowledge simplifies the modelling process and reduces the need to specify desired shapes from multiple viewpoints, often allowing the construction of 3D shapes using only 2D input from a fixed viewpoint. This methodology is demonstrated through a number of example implementations. In Part II via the example cases of modelling garments, folds and hairstyles from fixed viewpoints by annotating a supporting surface. As the chapter progresses, so does the level of prior knowledge involved. Starting with simple rule-of-thumb constraints such as ‘the fit of the garment from the front viewpoint is indicative of the fit everywhere’, adding more advanced constraints such as ‘the garment surface should unfold flat on a plane (be *developable*)’ and ending with a full physically based model for a hairstyle parameterised from a fixed viewpoint sketch. In Part III we address the case of sketch-based modelling performed without an initial supporting surface, at multiple scales and optionally from multiple viewpoints. Not having an initial support surface leads us to determine the structure of the shape from user-drawn silhouettes, an approach we believe is stronger than asking the user to pre-draw the supporting structure. First, in the case of cumulous clouds, we describe a system which allows a user to rapidly define the rough shape of a set of clouds, and fine scale detail is added automatically via a procedural method. We then address the case of modelling trees at multiple scales through a seamless interface - combining the simplicity of an artist driven global-to-local workflow with the power of procedurally generated similar details. The user may add detail at any level of scale within the tree using the same interface metaphor.

Organisation of thesis

In Part I we discuss the relevant human perceptual concerns in designing sketch-based interfaces for modelling and we examine the State of the Art in such interfaces. We review some common basic techniques, take a brief look at sketch-based interfaces for modelling general shapes and then a more detailed look at sketch-based interfaces for modelling complex shapes where prior knowledge can be utilised. In Part II we demonstrate our methodology as applied to sketch-based modelling of clothing, folds and hairstyles. In Part III we demonstrate our methodology applied to multi-scale phenomena such as clouds and trees. In the final chapter we discuss our conclusions and future directions for research.

Résumé en Français

Sketch - Un croquis ou une esquisse de quelque chose, indiquant ses grandes lignes ou ses caractéristiques sans le détail; en particulier il peut être destiné à servir de base à une image plus finalisée ou à être utilisé dans sa composition; une ébauche ou un plan. - Oxford English Dictionary

≪Une image vaut mieux que mille mots≫ - *Proverbe*

Qu'entendons nous par "sketching"?

Les dessins datent d'avant l'écriture[Clo00, Kra88]. Tout le monde sait dessiner dans une certaine mesure, même si la plupart abandonnent la pratique avant l'âge adulte. Le dessin, même rudimentaire, peut véhiculer une information plus efficacement que des mots, et ce indépendamment de la langue pratiquée. Le dessin est une compétence de base qui sous-tend la plupart des arts visuels, que ce soit la peinture, la sculpture, l'architecture, le design de mode, la conception de produits ou la réalisation d'images générées par ordinateur (CGI).

Le croquis ("sketch" en anglais) précède le dessin. Il en est généralement une version grossière, rapidement exécutée, utilisée pour étudier et comprendre le modèle en cours de dessin afin de mieux le représenter. Avec des supports réels, où le résultat final est un dessin, une peinture ou une sculpture, de nombreux croquis peuvent être faits avant de tenter la composition finale. Avec les supports virtuels aussi des croquis au crayon sur papier sont faits avant que l'artiste ne se tourne vers un logiciel de modélisation spécialisé pour commencer à créer le modèle.

Jusqu'à récemment, les ordinateurs n'étaient pas assez puissants pour utiliser le dessin comme interface d'entrée; en particulier, ils ne pouvaient pas interpréter le dessin de façon pertinente. Un artiste aurait pu bien sûr utiliser un outil comme Photoshop ou Illustrator pour créer une image. Mais l'image n'aurait alors pas été utile au processus de création, sauf à servir de support d'étude et d'inspiration. Le problème de l'interprétation d'un dessin en général est toujours trop difficile à résoudre pour un ordinateur, d'autant plus qu'il existe souvent de nombreuses interprétations à un dessin. Par conséquent, même lorsqu'un croquis est utilisé lors de la phase de conception, tout doit encore être modélisés en utilisant un logiciel de modélisation 3D généraliste.

Dans les industries du cinéma et de jeux, le temps entre le commencement et la livraison du produit final est critique. Beaucoup de productions exigent de l'artiste qu'il génère une somme considérable d'éléments naturels comme des arbres, des plantes (Figure 2), des terrains, des personnages et des vêtements. Leur modélisation requiert une expertise et du temps. Les systèmes qui permettent de réduire le temps ou l'expérience requises sont bienvenus. Les systèmes de sketching peuvent répondre à ce besoin.

Cette thèse explore l'utilisation des croquis et annotations pour la modélisation de phénomènes spécifiques et complexes, où une certaine connaissance peut être utilisée pour interpréter le dessin. La méthodologie que nous introduisons sera illustrée par la conception de systèmes basés sur le sketching pour la modélisation de vêtements, de cheveux, de nuages et d'arbres. Nous utilisons ≪sketching≫ plutôt que de "dessin" parce que nous traitons l'entrée de l'utilisateur comme une intention brut de forme et de position - pas toujours fidèlement reproduit, mais suffisamment proche pour satisfaire les intentions de l'utilisateur. Un croquis est réalisé rapidement, et nous voulons que notre processus de modélisation réagisse rapidement à l'utilisateur de sorte que le processus se déroule de façon fluide, réduisant ainsi le délai entre l'idée initiale et le modèle final.

La modélisation automatique de formes à partir d'un croquis partage de nombreux problèmes rencontrés dans le domaine de la vision par ordinateur. Le système visuel humain est très bon pour la reconnaissance rapide des objets et de leur mouvement dans une scène, et ce dans un large éventail de conditions d'éclairage et de visibilité. Il est en fait si bon que nous considérons cette capacité comme allant de soi. De nombreuses études ont, depuis longtemps, été menées dans le but de permettre aux ordinateurs de reproduire cette capacité, et pourtant nous avons à peine commencé à comprendre les mécanismes d'appui de notre système visuel. Une discussion détaillée de cette question dépasse la portée de cette thèse, mais nous devons au moins placer notre approche théorique dans le contexte de perception visuelle humaine. L'approche théorique dominante dans la recherche en vision contemporaine est le constructivisme[Yan01] qui s'est formé à partir des idées de Hermann von Helmholtz (1821-1894). Le constructivisme considère que nous construisons une représentation interne d'une scène sur la base des informations sensorielles venant des yeux et de l'expérience - nous devons apprendre notre capacité à percevoir par l'expérience. Ainsi, notre perception d'une scène peut être considérée comme provenant de deux sources: les informations "remontantes" fournies par les yeux, et les informations "descendantes" fournies par la mémoire et l'apprentissage préalable. Les illusions visuelles telles que des images réversibles (Figure 1) en fournissent une preuve. En regardant l'image jeune fille / veille femme, votre système visuel rassemble assez d'indices "remontants" sur la forme locale pour ensuite faire un "descente" et deviner la nature de l'objet. Une fois un candidat choisi, votre interprétation des formes présentes dans l'image est renforcé par cette décision. Mais si vous aviez commencé votre observation à un autre endroit vous auriez pu choisir une autre interprétation de l'image. Notre approche de la modélisation de formes complexes par le biais du sketching est basée sur cette approche "descendante" d'intégration des connaissances dans l'interprétation de la forme. Nous nous concentrons sur l'utilisation de connaissance préalable de ce qui doit être modélisé dans le but d'en extraire des paramètres caractéristiques de tracé qui peuvent ensuite être utilisés pour guider les modèles procéduraux ou physiques qui génèrent une forme (possiblement détaillée) en 3D. L'avantage de cette approche est que l'utilisateur n'a pas besoin de connaissances sur les techniques de modélisation. La seule condition est sa capacité à dessiner et à comprendre les contraintes relatives à ce qui peut être dessiné.

La démarche de l'artiste - du global au local

En plus d'une approche perceptuelle ”descendante”, nous prenons également comme source d'inspiration les approches traditionnelles global-vers-local utilisées par les artistes. Les caractéristiques globales sont esquissées d'abord et les proportions et la composition sont décidés avant que les détails ne soient ajoutés. Figure 3 illustre l'évolution d'un dessin d'un arbre par étapes, Figure 4 illustre un processus similaire pour le dessin d'un rocher; notez que les mêmes lignes à grande échelle peuvent avoir de nombreuses interprétations différentes en fonction des détails ajoutées à plus petite échelle.

On peut constater que, lors de la phase finale, l'artiste peut utiliser des effets tels que le hachurage ou le pointillage pour ajouter les détails fins. Ces effets d'ombrage peuvent aussi avoir un effet important sur l'interprétation de la forme (par exemple, la phase finale de construction du rocher Figure 4). Cependant, dans cette thèse, nous n'essayerons pas d'interpréter les effets d'ombrage pour les raisons suivantes. Tout d'abord, bon nombre des effets d'ombrage (en particulier le hachurage et le pointillage) prennent du temps; ces techniques appartiennent plus à la phase finale de la production qu'aux phases précédentes de conception exploratoire. Ensuite, les effets d'ombrage sont très dépendants des conditions d'éclairage d'une scène plutôt que d'être facilement prédictible à partir de la nature de la forme en question. Enfin, tout le monde n'est pas en mesure d'utiliser ces techniques de manière efficace, et se concentrer sur leur utilisation conduirait à exclure des utilisateurs potentiels de tout système en résultant. Certes, certaines techniques d'ombrage peuvent être exécutées rapidement et pourrait améliorer un système de modélisation basé sur le sketching, mais envisagez Figure 5. Ici l'ombre est la seule chose qui change, et elle induit de nombreuses interprétations différentes de la scène présentée. Surmonter cette ambiguïté est un problème difficile. Beaucoup de recherches ont été menées sur le problème de la synthèse simplifiée de l'ombrage (comme le rendu non photo-réaliste), donc une fois que la forme déterminée via un système de modélisation par sketching, le détail des ombres peut être ajouté automatiquement. Par exemple Figure 6 illustre la génération automatique de hachurage pour l'ombrage [HZ00].

Contributions

Nous présentons une méthodologie pour la modélisation de phénomènes complexes basée sur le sketching et sur divers niveaux de connaissances préalables des phénomènes modélisés. Cette connaissance préalable peut être aussi simple que des contraintes empiriques ou aussi complexe qu'un modèle procédural décrivant le phénomène. Cette utilisation de connaissance simplifie le processus de modélisation, elle réduit la nécessité de recourir à plusieurs dessins de point de vues différents pour préciser les formes souhaitées, et souvent elle permet la construction de formes 3D en utilisant uniquement un dessin 2D relatif à un point de vue fixe. Cette méthode est illustrée par un certain nombre d'exemples de mise en œuvre. Par exemple Part II illustre les cas de la modélisation de vêtements, de plis et de coiffures à partir d'un point de vue fixe et d'annotations. De chapitre en chapitre, le niveau de connaissance préalable requis progresse. Nous partons de simples règles empiriques telles que ≪l'ajustement d'un vêtement vu de face est indicatif de l'ajustement dans les autres vues≫, nous en ajoutons de plus avancées telles que ≪la surface du vêtement doit se dérouler à plat sur un plan (ie., développable)≫, et nous terminons par un modèle physique complet pour créer une coiffure à partir d'un croquis vu d'un point de vue fixe. Dans Part III nous abordons le cas du sketching effectué sans surface d'appui initiale, à des échelles multiples, et éventuellement à partir de plusieurs points de vue. L'absence de surface d'appui initiale nous amène à déterminer la structure de la forme à partir des silhouettes tracées par l'utilisateur: une approche qui, nous le croyons, est plus intéressante que de demander à l'utilisateur de prédessiner la structure d'appui. Tout d'abord, dans le cas de nuages de type cumulus, nous décrivons un système qui permet à un utilisateur de définir rapidement la forme approximative d'un ensemble de nuages; les détails sont alors ajoutés automatiquement à une échelle plus fine par le biais d'une méthode procédurale. Nous avons ensuite abordé le cas de la modélisation des arbres à diverses échelles par le biais d'une interface intuitive combinant la simplicité de la méthode de travail global-vers-local de l'artiste avec la puissance de la génération procédurale pour créer du détail par similarité. L'utilisateur peut ajouter des détails dans l'arbre à tous les niveaux d'échelle en utilisant une même interface.

Organisation de la thèse

Dans Part I nous examinons les caractéristiques de la perception humaine pertinentes dans la conception d'une interface de sketching pour la modélisation, et nous présentons l'état de l'art relatif à de telles interfaces. Nous passons en revue quelques techniques de base, brossons un bref aperçu des interfaces de sketching pour la modélisation de formes générales, et examinerons plus en détails les interfaces de sketching pour la modélisation de formes complexes où la connaissance à priori peut être utilisée. Dans Part II nous montrons l'applicabilité de notre méthodologie à la modélisation par sketching de vêtements, de plis et de coiffures. Dans Part III nous démontrons son intérêt dans le cadre de la modélisation d'éléments naturels multi-échelles tels que les nuages et les arbres. Dans le dernier chapitre, nous présentons nos conclusions et les orientations futures de recherche.

Part I

Background

CHAPTER 1

Perception and Depiction

Have you ever stood in front of a drawing or painting in a gallery and discussed it with a friend? You may have said something like “The trees are wonderful autumn colours,” or “The people look frantic”. But there are no trees or people in front of you, only marks on a 2D surface, and yet your companion nods and agrees — they understand completely what you are referring to. The fact that you both perceive the same phenomena in the work of art (unless perhaps you are viewing a Picasso) is a testament to the skill of the artist in depicting that scene, and this example highlights the fact that in thinking about and dicussing an image we will readily substitute *the representation of the thing* in question for *the thing itself*. This tendency means care must be taken in the use of language when discussing the two separate processes of perception and depiction. For example the phrase ‘the edge of that tabletop,’ when used to describe an actual table, is referring to a depth discontinuity between a surface of the table and whatever forms the background to the view of the tabletop from the viewing position. The same phrase when describing a painting of a tabletop is likely referring to a gradient of paint colour on a 2D canvas, or a straight 2D line in a pencil drawing.

The author Willats [Wil97] understood this point and in his book ‘Art and Representation’ he defines a vocabulary for the techniques people use to depict the scenes that they perceive. This vocabulary was later refined by Durand for use by the computer graphics community, and we summarise their work in section 1.2. Durand described the challenge of interpreting a painting or drawing as ‘the inverse of an inverse problem’ (see Figure 1.1). Before we can depict a scene, we must first perceive it, and so we begin our discussion with a brief look at perception and some of the rules used in shape construction by the human visual system.

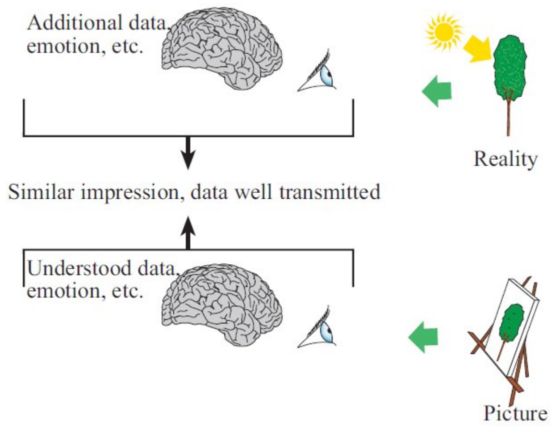

Additional data, emotion, etc.
Reality
Similar impression, data well transmitted
Understood data, emotion, etc.
Picture

Figure 1.1: Durand [Dur02] illustrates that depiction is the inverse of an inverse problem.

1.1 Perception

The study of human perception has a direct bearing on sketch-based interfaces. When someone draws a sketch of an object, he is interpreting that object as he draws. Naturally their perception of the object influences the result. In fact studies show [Wil97] that people tend to draw from the *idea* of the object that they have, rather than the appearance of the actual object in front of them. It is this instinct to draw from an internal representation (an implication of the theory of Constructivism mentioned in the introduction) that skilled artists have to overcome through training — to sketch what is actually in front of them, rather than their idea of it. This effect is demonstrated in upside-down drawing exercises [Edw01]. In these exercises the student is asked to reproduce an existing sketch, first by copying it rightside up, and then by copying while viewing upside down. The upside down rendering is usually more true to the original, because rotating the image impairs our normal shape cognition processes. This influencing of the result stems from the 'top-down' part of our perceptual construction, our prior knowledge and experience inform our perception of the shape. It is this prior knowledge and experience of shape that we exploit in our methodology for designing sketch-based interfaces for modelling complex phenomena. If we can map the salient features of a 2D sketch of a complex object (often found in silhouette lines) to operations that construct a 3D model of that object, then we have defined an interface that will seem natural to a user.

Further evidence that we construct our perception of an object can be seen by studying the way children draw. When young children are asked to draw a 3D object, their depiction bears little relation to the image of the object. Their drawings reveal that they actually map attributes of the 3D scene to attributes on the 2D depiction (Figure 1.2 (lower right)). They have constructed an internal representation of the object, and it is this representation they are trying to depict through their drawing. As they age their drawing skills increase and they learn the 'tricks' required to render a more faithful representation. These 'tricks' can be utilised in interpreting a drawn image. For example a powerful shape cue is occlusion. Where two lines meet in a 'T'-junction, we assume the horizontal line forming the top of the 'T' represents a boundary of a surface which occludes the surface boundary represented by the line which is the body of the 'T'. Hoffman [Hof98] collects and discusses many of these 'rules' in his book. These shape cue 'rules' are the basis of the 'bottom-up' part of our perceptual construction. The eye uses the rules to make inferences about local surface structure and features, and from a collection of these local details we infer a probable global shape. This inference is then used as feedback to inform the search for more local features providing evidence to support the inference. These 'bottom-up' rules are used in general shape modelling where only a few assumptions on the nature of what is to be modelled are made. They can be used in interpreting simple sketches (for example the work of Williams *et al.* [WH96] in interpreting occluding contours and an implementation based on these ideas by Karpenko *et al.* [KH06] (see subsection 2.2.2)). However these results must be used with care. A user may easily draw inconsistent cues, either through lack of skill or lack of knowledge of these techniques (consider the impossible drawing in Figure 1.2), so they cannot be followed blindly. These rules can only be used as indicators of the most likely user intentions.

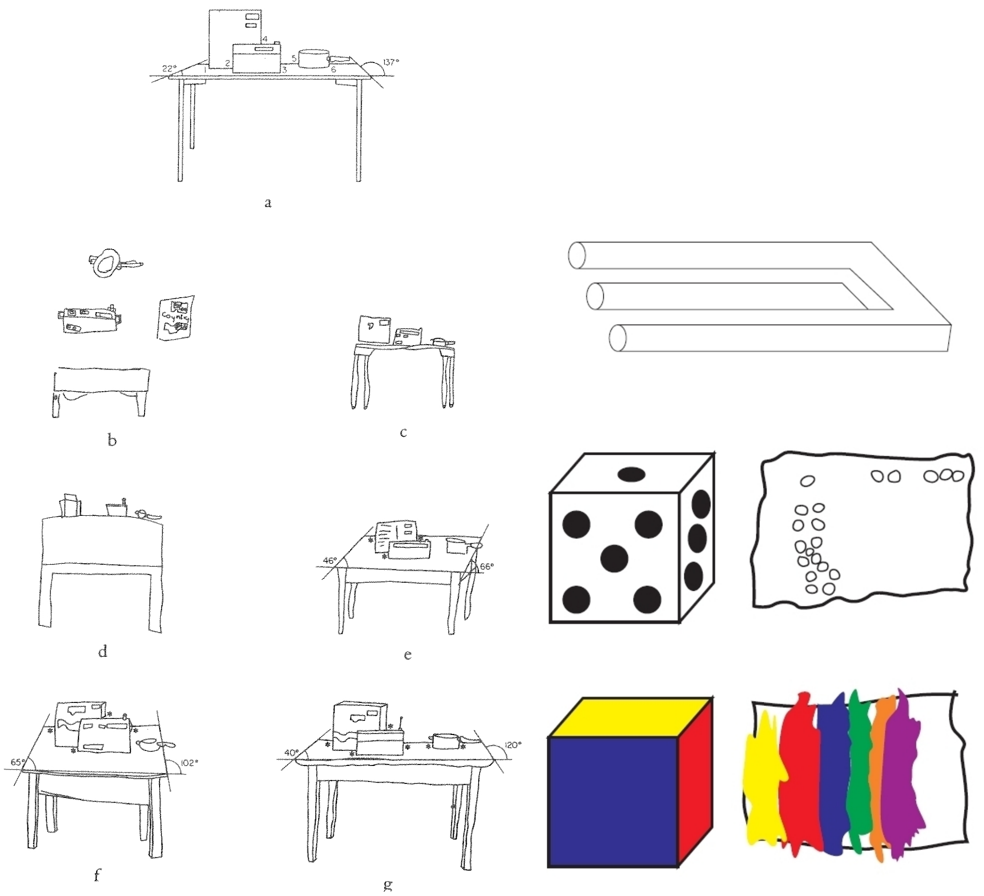


Figure 1.2: **(left)** Children's drawing ability progressing with age. **(a)** A table with objects on for the children to copy. **(b)** to **(g)** as age increases so does sophistication of the depiction. Notice the earliest examples look nothing like the image we see, they are representations of the *idea* of the objects [Wil97]. **(lower right)** The different ways a child draws a die (from [Dur02], from an original in [Wil97]). **(upper right)** An impossible drawing, created by drawing local shape cues in an inconsistent way.

1.1.1 Common perceptual 'rules'

Here we briefly summarise some of the 'bottom-up' shape construction rules described in 'Visual Intelligence', a book by Hoffman [Hof98]. These rules can be considered as 'rules of thumb' that the human perceptual system uses when constructing shape from images. They are useful in designing sketching systems that model general shapes (described briefly in section 2.2). We don't detail the justification for these rules as our focus is not on general shape modelling. Suffice to say they are the result of many careful perceptual studies and are summarised excellently in [Hof98].

The rule of generic views

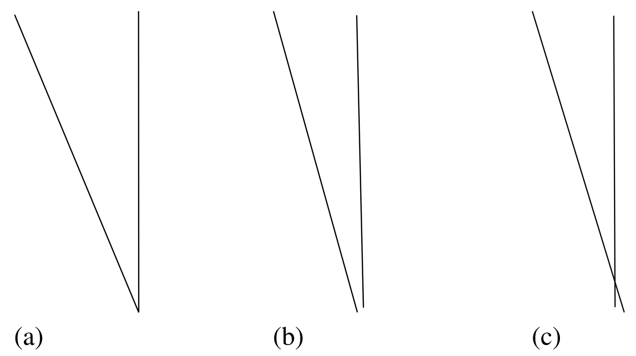


Figure 1.3: The rule of generic views: The natural 3D interpretation of the image **(a)** is two lines meeting at a point. If in fact the initial image represents two disconnected lines, then they would look like **(b)** or **(c)** from a different but nearby viewpoint. The chances of just happening to view a pair of disconnected lines from the one viewpoint which makes them appear like a 'V' are very slim. Hence the 3D interpretation **(a)** is the most probable, and the one that our perceptual system makes.

Consider Figure 1.3a, if you give this image a 3D interpretation you are likely think of two lines meeting at a point, like a 2D letter 'V' in 3D space. It's unlikely that you interpreted the image as two separate lines that don't meet at the same point, but viewed from an angle where they just happen to appear to meet. Such a configuration would look like Figure 1.3b or Figure 1.3c when viewed from a slightly different angle from Figure 1.3a. This is an example of the *rule of generic views*, which holds that the visual system constructs only those interpretations for which the image is a stable (i.e. generic) view. By generic view we mean a view which doesn't change in any major way when you shift your viewpoint slightly. If you consider the sphere of possible viewing directions around a pair of disconnected lines, there are only two directions from which they would appear to be joined at one end, but countless other views where they would not appear joined. In the general case you are very unlikely to be viewing from these two 'accidental' viewpoints - it's far more probable that if you are seeing Figure 1.3a from an arbitrary viewpoint and that they are indeed two lines joined in space.

The rule of proximity

Figure 1.4 shows a 'Necker cube' with bubbles added. The Necker cube can be interpreted in one of two ways, with the lower left vertex existing in a plane parallel to the view plane being either in front of or behind the plane (also view parallel) that the upper right vertex belongs two. This is interesting in itself, as it shows we are constructing depth from a flat image. But the addition of the bubbles illustrates another perceptual rule. The perceived depth of the bubbles changes according to the current interpretation of the cube. We assign the same depth to each bubble as we do to the cube edge nearest it.

Some rules for curved surfaces

The rule of generic views also helps us derive rules which can be used for perceiving curved surfaces. For example where possible we interpret a curve that is smooth in an image as being smooth in 3D; if we did not, then a small change in viewpoint would destroy the smooth appearance. This is illustrated on the right of Figure 1.4. We also interpret the curves in the image as being the 'rim' of a shape (also known as occluding contours). The T-Junctions, of which there are two in the figure, help us construct relative depth. Where possible, we interpret the cap of the T-junction as occluding the stem. We also consider the curvature of the

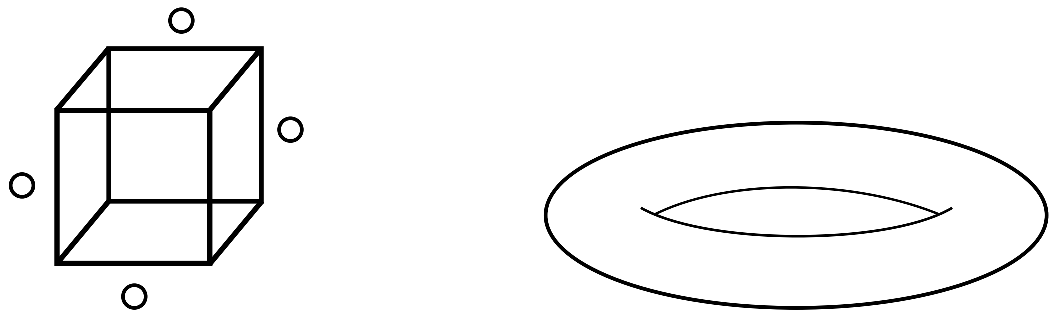


Figure 1.4: **(left)** The rule of proximity. You assign a depth to the bubbles based on the nearest edge in your depth construction of the cube. **(right)** The rules of curved surfaces, you easily perceive a 3D doughnut shape in this image.

lines, convex points on a 2D boundary are interpreted as points on a convex surface in 3D, points on a concave boundary are interpreted as points existing on a saddle surface in 3D.

In particular smooth inflation systems such as [IMT99] utilise these rules, and more complex shape cues such as T-junctions are used in [KH06]. These rules can also be used in 'top-down' perceptual sketching applications (for example T-Junctions can indicate relative depth order), but often they are not required, as the prior knowledge of the object being modelled means that sketched lines already have a preset definite meaning.

1.2 Depiction

When discussing sketches one requires a vocabulary. Willats' book [Wil97] (summarised in the Appendix) defines a system for describing the representational systems used in pictures. This system was adapted for the field of computer graphics by Durand [Dur02] and it is reviewed briefly here. Willats describes pictures in terms of *drawing*, and *denotation* systems, which are basically systems for deciding which drawing primitives to use and where to put them in the picture. Durand maps these to the more precise (for computer graphics purposes) concepts of *spatial* and *primitive* systems, and extends the framework with the concepts of *attribute* and *mark* systems. Together these systems can be thought of in terms of a 3D to 2D picture production pipeline of four stages: spatial mapping, primitive choosing, assigning attributes of these primitives, and mark implementation. This pipeline is a simplification but is useful in defining a common vocabulary for talking about computer image rendering. Sketching interfaces could also benefit from this vocabulary by considering the same pipeline in reverse. What sort of marks can the user make? What attributes can we extract? What does the picture (screen) space primitive represent in object space? Where in object space does it map to and how do we infer the missing 3D shape information?

1.2.1 Spatial system (Drawing system)

The spatial system maps 3D spatial properties to 2D spatial properties. Two examples are orthographic or perspective projections, and these are the ones commonly used in sketch-based modelling, but other systems such as non-linear perspective are possible (Figure 1.5).

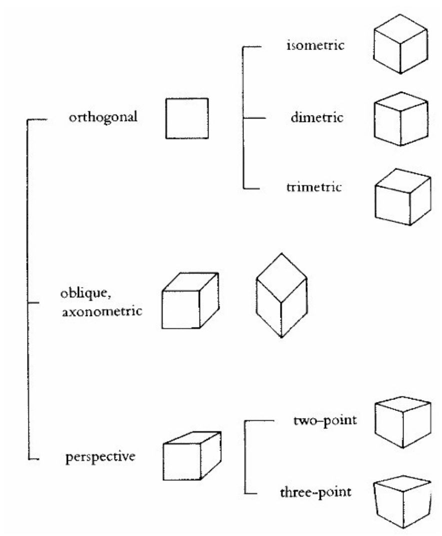


Figure 1.5: Projection systems (from [Wil97]).

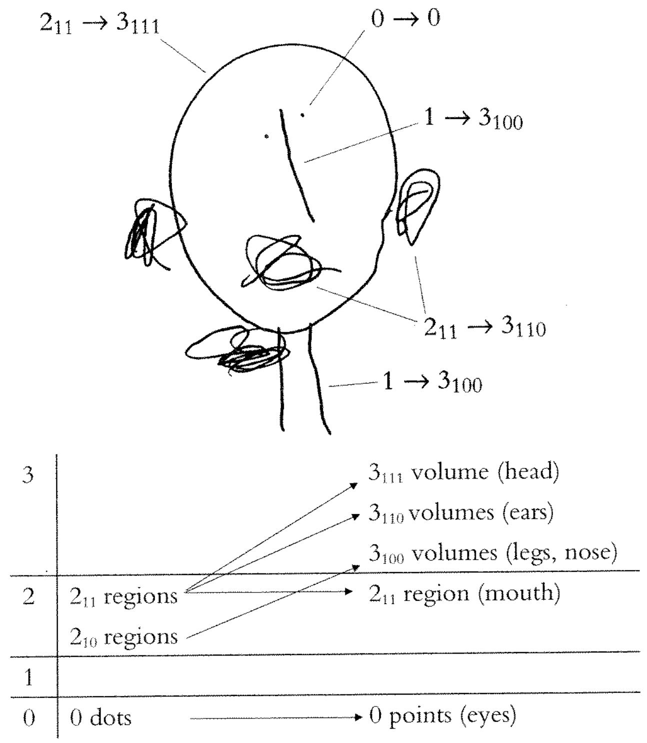


Figure 1.6: Analysis of a drawing of a man by a five year old child, using Willats formal descriptive system (from [Wil97]).

1.2.2 Primitive system (Denotation system)

The primitive system maps object (scene) space primitives such as points, lines, surfaces or volumes into picture space primitives like points, lines and regions. For example a 1D line in picture space could represent the silhouette of a 3D object (for example the sketched outlines shown in Figure 2.4), or it could stand for the object itself (for example the 3D branches of a tree are sketched as 1D lines in Figure 2.16). In sketching systems the reverse of this mapping (from picture space to object space) is of central importance and the taxonomy Willats suggests for describing these mappings is a useful way to make explicit the functionality of a sketch-based modelling system. In Willats' system each picture primitive and each scene primitive are assigned a dimensional index, and subscripts indicating the dimensions into which the primitive can potentially be extended. For example: A point or a T-Junction in a picture, or a point of occlusion or a corner in a scene have a dimensional index of '0'. Lines in pictures or edges in scenes which can only be extended in one direction have a dimensional index of '1'. Regions and surfaces have dimensional index of '2', and subscripts of '0' or '1' for each dimension. A subscript of '1' indicates a significant extension of a primitive in the given dimension, which a subscript of '0' indicates an insignificant extension in this dimension. Scene volumes are represented by dimensional index 3 and 3 subscripts. The categories are shown in Figure 1.7 and the system is illustrated by the child's drawing in Figure 1.6. Here single lines in the picture space are used to represent long thin volumes in scene space (legs, nose). This can be described compactly as $1- > 3_{100}$, and this technique is used commonly in sketch-based modelling - for example when sketching branches or hair.

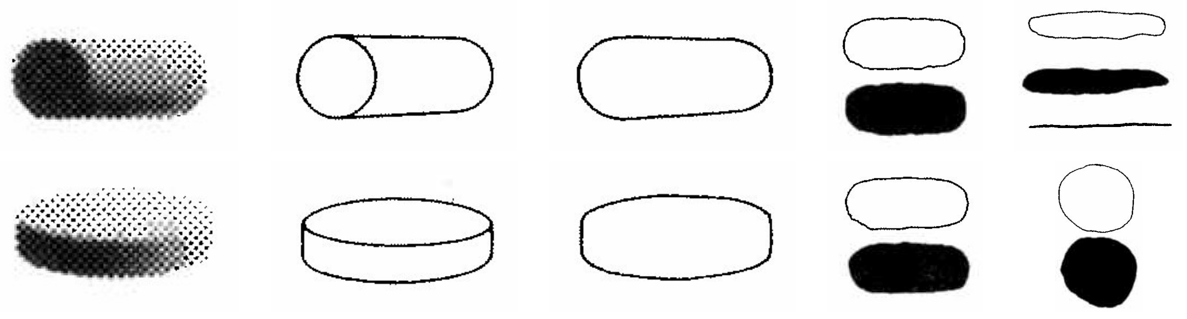


Figure 1.7: Types of denotation systems, illustrated on a cylinder (stick) and a disc. **(a)** Point primitives denote intercepts of small bundles of light rays. **(b)** Lines denote exterior occluding contours and interior edges (here geometric ridges, explained later). **(c)** Lines denote occluding contours only (drawn carefully they can convey useful shape information). **(d)** Regions denote regions in the frontal plane; the outlines are too rough to convey detailed shape information, but they provide information about the extendedness of the shape. **(e)** Regions denote volumes; the extendedness of the regions carries information about the extendedness of the object in 3D space. Lines can sometimes be viewed as very narrow regions (adapted from [Wil97]).

Single lines can have many interpretations, and in realistic sketches many types of lines are used. Willats [Wil97] mentions outlines, edges and occluding contours, but recent studies [CGL$^+$08] on what people actually draw distinguish between many more types of lines. Different researchers may use slightly different names for the same type of lines. Here we summarise the most common:

Internal/External silhouettes, contours, occluding contours

A *silhouette* can be thought of as a line separating an object from its background (the figure from its ground). In non-photorealistic rendering techniques this line can be found on a mesh where the local surface normal switches from front facing to back facing (it is thus view dependent). In fact this definition provides more than just the silhouette, as portions of the object within the silhouette also meet this definition. Such lines are *occluding contours* (sometimes refered to as just contours) or *internal silhouettes*.

Creases, geometric ridges and valleys

Creases are lines drawn where there is a sharp discontinuity in the normal to the object surface (the surface bends sharply). *Geometric ridges and valleys* are smoothed creases, i.e. the normal is varying rapidly in one

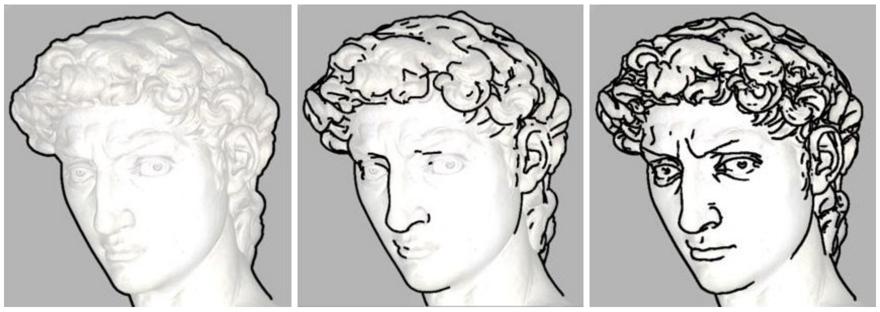


Figure 1.8: The lines people draw. Indicated lines are in black. **(left)** External silhouette. **(middle)** All occluding contours (External + Internal silhouettes). **(right)** Occluding contours and suggestive contours. From [RCDF08].

direction rather than being discontinuous (Figure 1.9).

Suggestive contours and apparent ridges

Suggestive contours and apparent ridges are view dependent extensions of the previous two categories. A suggestive contour (Figure 1.8) is a line which would become an occluding contour in a nearby viewpoint; they extend from the end of contours. Apparent ridges (Figure 1.9) take foreshortening into account when considering surface curvature. They also join smoothly to occluding contours in the image and they approach standard geometric ridges and valleys when viewed head on.

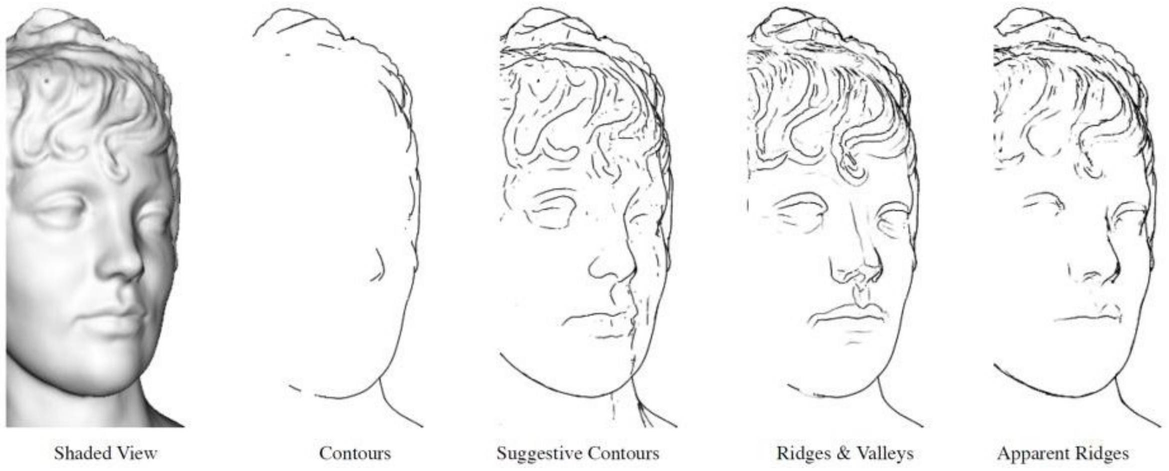


Figure 1.9: Geometric ridges and valleys and apparent ridges compared. From [RCDF08].

Illumination dependent lines

A scene can only be viewed if there is a source of light. Different lighting conditions cast different shadows and cause various forms of highlights. People may draw lines where there are strong illumination discontinuities in a scene.

Conceptual lines

The previously defined lines were all lines that could be perceived from a view of a scene, and are used when drawing in a realistic style. However when communicating ideas people often draw lines that don't correspond

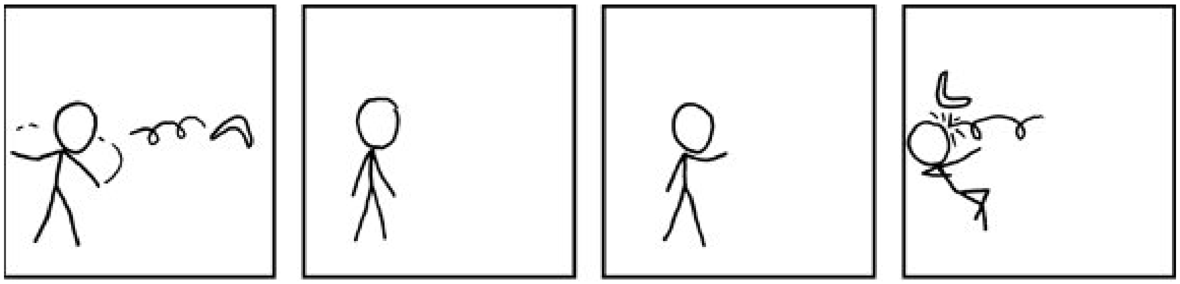


Figure 1.10: Conceptual lines in comics. The motion of the arm and the boomerang, plus the impact are depicted using lines (from [Mun08]).

to any visual feature of a scene. In the comic in Figure 1.10 for example there are lines representing the motion of the arm, the motion of the boomerang, and the impact of the boomerang, none of which are visible in a real scene. Lines such are these can be useful for specifying extra information in a sketching system (for example for specifying motion paths in the work of Thorne *et al.* [TBvdP04]).

1.2.3 Attribute system

The attribute system records visual properties of picture primitives such as colour, thickness, texture, transparency, etc. In a traditional realistic rendering pipeline these attributes are the result of physically based lighting calculations. In a non-photo realistic (NPR) rendering pipeline they might be assigned by the artist or controlled via level-of-detail calculations (for example a larger distance from the viewer in 3D means a greater level of transparancy). Often in a sketching system some of the attributes of a line are pre-selected (e.g. colour) and may denote a predetermined interpretation of the line (for example when sketching a tree, brown lines are branches, green lines are leaf silhouettes. Note that in this example the pre-selection also sets the primitive interpretation of the mark). Properties such as thickness can be pre-selected or vary according to the input device available (for example pen pressure can map directly to thickness, pen angle can determine transparency).

1.2.4 Mark system

In the forward pipeline, the mark system deals with how the primitives will be implemented at their spatial location, according to their attributes. The mark system describes the physical strokes made by a brush or pencil in a traditional depiction. In a reverse pipeline (sketch-based interface) the mark is again pre-determined. Whatever the 2D appearance of the mark it always has an underlying 1D line representation when a traditional pen based interface is used. One could imagine that future input devices will relax this constraint, for example input devices which track the position of a finger tip cater for much more ambiguity in input than the fine point of a digital pen, and could allow for different mark shapes.

1.3 Summary

This chapter outlined some of the perceptual rules we use when trying to determine 3D shape from a 2D image and also examined the process of depiction, in which we try to represent 3D shape through a 2D drawing. In particular we outlined the denotation system of Willats [Wil97] which supplies a vocabulary for talking about the intended meaning of sketched primitives. We will apply this system of analysis in the next chapter, which examines the state of the art in sketch-based modelling interfaces.

CHAPTER 2

State of the Art

Automatic interpretation of drawings to produce 3D models can be divided into two camps, those that take the finished drawing and try to interpret the whole, and those that incrementally interpret what the user draws in an interactive system (an approach known as using a *sketch-based interface* (SBI) for *modelling* (SBIM)) . The problems of the first camp are akin to computer vision problems and are very hard to solve. Often to reduce the complexity of the problem, the domain of what can be in the drawing is restricted, as in the automatic interpretation of architectural blue-prints. We do not address the problems of this domain, but refer the reader to [CPCN05, FHS07].

This thesis focuses on the second camp (SBIM), interpreting the user's input a few strokes at a time in order to model some shape. This interactive approach has the advantage of knowing in what order and what manner each stroke was drawn, giving extra information not available in problems from the first camp. This extra information reduces the scope of the problem, as interpreting the specific meaning of a stroke becomes easier when the current context is known.

The first recognisable sketching system of this type was SKETCHPAD [Sut63]. Using the newly invented light-pen and a CRT display, Ivan Sutherland's thesis detailed a system for interacting with a computer via a drawing interface, it was the inspiration for the many graphical user interfaces which followed. Later SKETCH [ZHH96] introduced the idea of computer aided design (CAD) operations specified through sketched gestures, and was restricted to modelling scenes composed of unions and differences of primitives (using an underlying constructive solid geometry (CSG) model). More freeform shape design was made possible with the introduction of the Teddy [IMT99] system, which allowed the modelling of 'softer', more rounded shapes through the inflation of sketched contours and a set of gestural modelling operations. The underlying shape representation in this case was a boundary (mesh), but as computing power has increased, more complex underlying models could be created through sketching. In particular a large body of worksubsection 2.2.3 deals with the construction of implicit surfaces specified through sketch-based interfaces.

The systems described so far allowed the modelling of very general shapes. Their power came from the initial simplifying assumption of a global shape property (either angular CAD shapes or blobby smooth shapes), but this assumption also limited the range of things which could be modelled. For example designing complex organic shapes like a flower or tree is almost impossible with the angular shape assumption, and perhaps possible but overly time consuming with a system like Teddy. Sketching specific types of complex models which have repetitive details could benefit from a more specific approach, utilising prior knowledge of the nature of the shape and where possible allowing automatic generation of some aspects. For example, in the case of a tree or plant the user doesn't want to model and place every leaf or organ — he would prefer to give an example of leaf shape and placement and have many more similar leaves automatically placed. The same argument applies to hair (see Part II). This approach of combining procedural techniques with the freedom of sketching is where this thesis makes its contributions.

Sketch-based interfaces are not limited to modelling only shape. They are also applicable to a much wider range of problems such as specifying motion [TBvdP04] or visualising mathematical problems [LZ06]. Some systems give the appearance of generating a model, but are in fact general visualisation tools which don't produce a true 3D representation [BCD01]. In this chapter we discuss only sketch-based interfaces for modelling, where the model can be expressed in a reusable form such as an oriented boundary representation (mesh).

Underlying all these interfaces, there are a number of supporting concerns specific to sketching such as stroke processing, stroke edition and gesture detection and disambiguation. The field of SBIM overlaps the field of *Human Computer Interaction* in the area of interface design, for example in the use of elements like suggestion engines (used in [OOI05]). We restrict the focus to the interpretation of the user's drawn strokes rather than advanced interface components such as this.

This chapter first summarises some of the basic sketch processing techniques used in SBIM. Then the main works in general shape modelling are described briefly (as they are related to, but are not the focus of this thesis). The chapter ends with a summary in greater depth of the area of complex specific shape modelling (the focus of this thesis).

2.1 Sketch processing tools

Before processing 2D strokes into 3D and interpreting them, we first need to capture them in an appropriate form. If the user is sketching an object silhouette, will he draw just one strong line or many smaller lines? For simplicity the interface can constrain the user, but the easiest to learn interfaces will be those that allow the user to work in his own way. In this section we briefly examine the research into sketch processing techniques that will support any advanced sketch processing interface. An excellent SIGGRAPH course [JJL07] is available which covers some of the issues involved.

The most fundamental issue is stroke sampling and representation. Any user input captured via electronic devices will be noisy, so some smoothing of the input may be required. A simple, regularly sampled polyline is the most basic of representations and is often sufficient. However it is not the most compact of representations. For example, if the user draws a straight line, regular sampling of this line could produce 20 sampled points, where just 4 or 5 may be sufficient. Adaptive sampling of the polyline is an improvement, where detailed sections of the stroke are assigned more samples than slowly changing sections. Polyline simplification methods are useful in removing redundent stroke samples and are a simple form of adaptive sampling. Horst and Beichl [HB97] present a simple method for simplifying curves known as the chord and arc length (CAL) method. This method starts from an arbitrary point on the curve and proceeds to move around the curve, measuring both the arc length (S) and the chord length (C) for each successive point. When the value of $\frac{1}{2}\sqrt{S^2 - C^2}$ exceeds a deviation threshold parameter, the previous point is denoted a dominant point. Dominant points are then used to form an approximation of the original curve.

A more sophisticated stroke representation is to represent the stroke in a continuous manner by fitting a mathematical curve to the input. There is a large body of research in curve representations, each having particular properties which may recommend them for different purposes. For general sketching implementations Bezier curves, or B-Splines are simple to fit to a stroke [Sch90] and offer a compact, continuous stroke representation which can be cleanly rendered.

There are more sophisticated stroke representations available, such as wavelet analysis [FS94], this representation allows low frequency edits to be made without changing the high frequency appearance.

Once the stroke is captured certain apparently simple judgements about its nature may be required. For example in the clothing design interface (see Part II) we needed to determine the sharp corners of a stroke. The simple approach we used didn't always match the user's intention; a more sophisticated method such as [WEH08] would be an improvement to the system. This method creates a regular resampling of the stroke and then looks for variations in the length of a "straw" (window defined by offsets from the current point index) while advancing along the path of the stroke. Sharp variations are indications of a corner.

If the artist doesn't draw exactly the stroke he intends on the first try (which is often the case) then how should he improve the first attempt? Overtracing the stroke without erasing the previous one is one approach. This can lead to many overtraced lines that need to be interpreted as one line. Sezgin and Davis [SD04] provide a method to do this which treats the oversketched input lines as a point cloud which is first thinned and then approximated in terms of primitives like lines and arcs. Another way an artist may improve a stroke is to redraw a section of the stroke [Bau94, Ale05]. Instead of having to draw one continuous line, an artist may wish to use many smaller strokes to represent a feature which is continuous in the scene. In this case the strokes must be combined in an intelligent way. Pusch *et al.* [PSNW07] present just such a method which sub-divides space to the level of a single stroke, and then reconnects these spaces using principle component analysis and input point ordering information to create a single B-Spline curve, which can be noisy. A multi-resolution technique is then used to fit a smooth B-Spline curve.

When using a pen based interface it is desirable to keep the focus on the pen and not have to reach for the keyboard to change modes. Gesture based operations are one way to reduce keyboard dependency, but distinguishing between strokes used in gestures and strokes intended for modelling is not easy. Also distinguishing between gestures can be difficult, and so the whole set of gestures available must be designed with care. A brief overview of the many available techniques can be found in [WWL07].

2.1.1 Shape skeletons

Determining the geometric skeleton of a shape is a useful technique for analysing a stroke used as an object boundary. There are a number of ways to obtain a shape skeleton, either through image based erosion techniques [BPCB08] or (the approach we use) geometric analysis of the boundary [Pra97]. The Medial Axis Transform [Blu67] is a popular skeleton representation. The transform in 2D is formed by set the of the centres of all circles of maximal radius within the shape. That is the set of points for which the nearest two or more points are on the boundary. A similar transform is the Chordal Axis Transform (CAT) [Pra97] which is used by [IMT99] and other sketching systems as a basis for their inflation techniques. We also use the chordal axis transform to determine shape information (Chapters 6 & 7). We outline the basic technique of [Pra97] here.

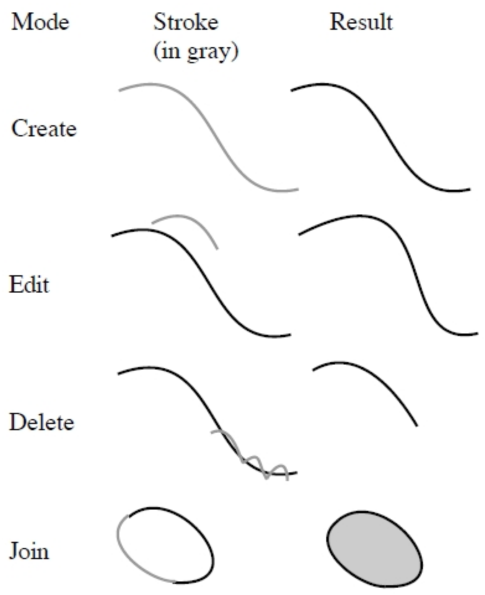


Figure 2.1: Stroke modification techniques from [Bau94].

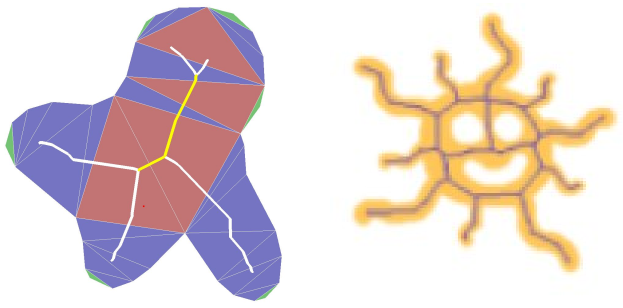


Figure 2.2: **(left)** Shape skeleton represented by a Chordal Axis Transform (CAT) formed through constrained Delaunay triangulation. Terminal triangles in green. Sleeve triangles in blue. Junction triangles in red. Terminal segments in the transform are thick white lines, segments between junction triangles are thick yellow lines. **(right)** Shape skeleton represented by Medial Axis Transform (MAT) formed through iterative erosion (from [BPCB08]).

Given a polygon, we wish to determine the chordal axis transform. We take the segments of the polygon and form a *constrained delaunay triangulation (CDT)*[1] using the polygon segments as constrained edges of the triangulation. The internal triangles can now be classified as one of three types:

Terminal Triangle A triangle which shares two edges with the shape boundary.

Sleeve Triangle A triangle which shares one edge with the shape boundary.

Junction Triangle A triangle which doesn't share an edge with the shape boundary.

The CAT is formed by connecting the circumcentres of adjacent classified triangles. Terminal branches will consist of a Terminal triangle followed by zero or more Sleeve triangles, and will terminate with either another

[1]A delaunay triangulation is a triangulation of a set of points such that no point within the set falls within the circumcircle of any triangle. A constrained delaunay triangulation adds the constraint that certain edges defined between points within the set must also be edges in the triangulation.

Terminal triangle (this is a degenerate case where the skeleton is a single curve) or a Junction triangle. The CAT can be seen as a directed graph with no cycles.

Like many skeletonisation techniques the CAT may suffer from unwanted artifacts in the skeleton, where small discontinuities in the outline have created skeletal branches which distract from the global nature of the shape. Initial smoothing of the outline can reduce such artifacts, and in addition [Pra97] outlines a metric for detecting and suppressing such unwanted features.

2.2 Sketch-based modelling of general shapes

2.2.1 Angular shape modelling

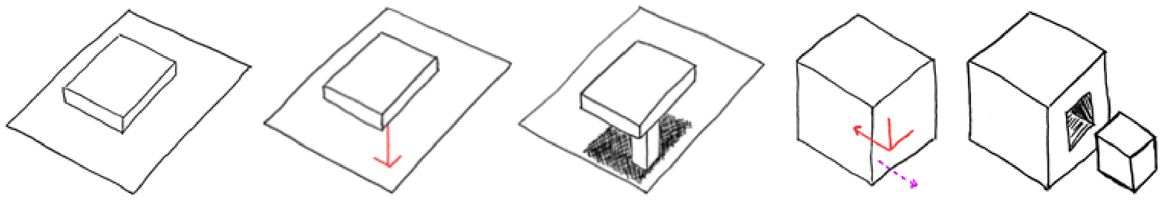


Figure 2.3: The key operations and a resulting scene from SKETCH [ZHH96]

SKETCH [ZHH96] was the first system to use sketching ideas in a computer modelling package. Based on a reduced set of constructive solid geometry (CSG) primitives commonly used in CAD packages, the idea behind SKETCH was to keep the interface lightweight and intuitive. Operations were gestural and based on observations from perceptual research, such as using T-Junctions as an indication of object occlusion (as mentioned in section 1.1). Rendering was non-photo realistic and "sketchy" to help convey the imprecise nature of the ideas. The authors pointed out a fundamental problem with gestural interfaces — that too many gestures can be hard to learn, thus defeating the objective of a simple to learn interface. Also as the number of gestures increases, the design of each gesture becomes more important, as ambiguity between gestures increases the chance of the wrong operation being selected. Nevertheless SKETCH was an exciting demonstration of what is possible, and led the way to more modern CAD modelling systems such as Google's SketchUp [Goo08].

2.2.2 Smooth/Organic shape modelling

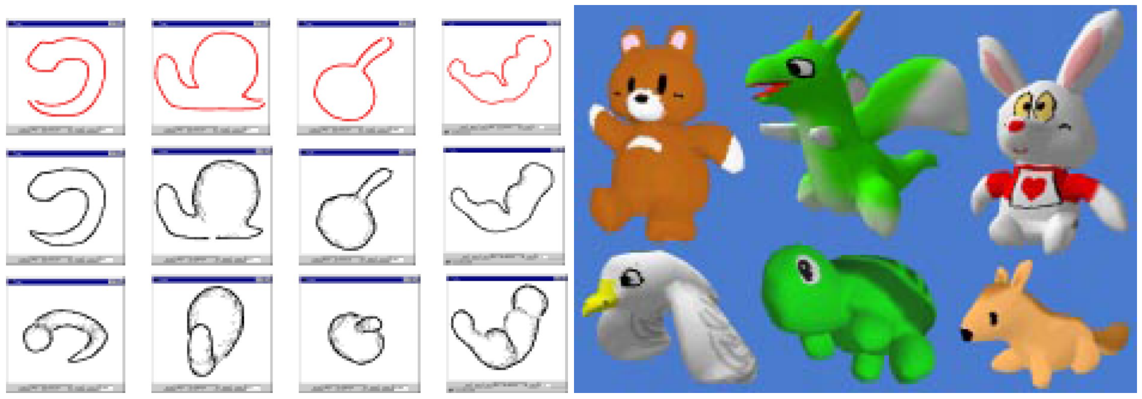


Figure 2.4: Teddy[IMT99], a sketch-based smooth shape modelling system. **(left)** Some creation strokes and the resulting shape. **(right)** Some final results after a separate texture painting stage.

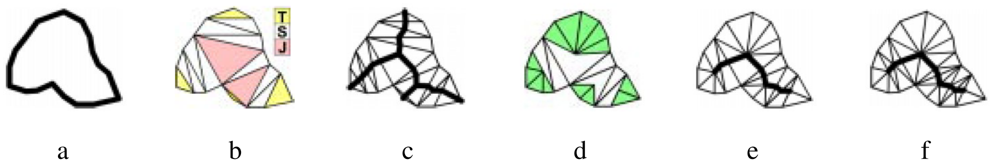


Figure 2.5: The skeletonisation and retriangulation steps of the Teddy system [IMT99]. **(a)** Initial silhouette, **(b)** Constrained delaunay triangulation (terminal triangles yellow, sleeve triangles white, junction triangles red), **(c)** The skeleton formed by chordal axis transform, **(d)** Triangles close to terminal triangles converted to triangle fans, **(e)** The spine of the shape, **(f)** The retriangulation, incorporating the spine of the shape (Images from [IMT99]).

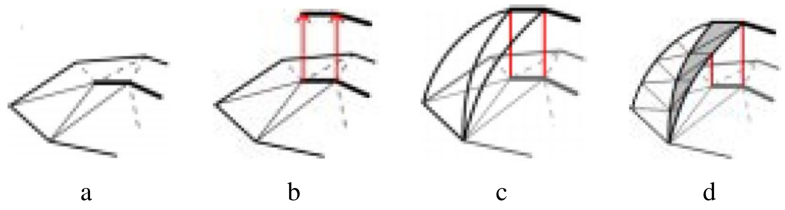


Figure 2.6: The inflation steps from the Teddy system[IMT99]. **(a)** Initial spine, **(b)** Extruded spine, **(c)** Creation of arcs, **(d)** Triangulation of arcs (Images from [IMT99]).

Polyhedral modelling, although very useful in manufacturing, isn't suitable for designing organic ("softer"/"smoother") shapes. These are the sort of shapes our perceptual system excels at inferring from sketched drawings (section 1.1). The first sketch-based interface to address this issue was Teddy [IMT99]. The approach used was similar to SKETCH, in that sketched gestures mapped directly to operations on the model. However the underlying represention was a smooth 3D mesh, inferred from 2D outlines using an novel skeletal inflation approach. First a 2D shape skeleton was determined from the Chordal Axis Transform (CAT) using the method of [Pra97] (explained in section 2.1). This skeleton is considered the spine of the shape. Triangles along the terminal branches of the CAT are converted into triangle fans (Figure 2.5). The initial constrained delaunay triangulation is then re-triangulated to incorporate the spine edges, and then the spine edges are extruded perpendicular to the triangulation plane. The height of extrusion along the spine varies in proportion to the distance of the original 2D spine point from the 2D shape boundary. These extruded spine points are then connected back to the boundary via arcs which are then also triangulated, resulting in a 3D mesh surface (Figure 2.6). Teddy had some weaknesses such as low quality triangulations and mesh artifacts, which were addressed in later work [IH06].

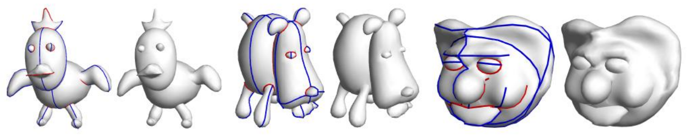


Figure 2.7: Three results from the FiberMesh [NISA07] system.

Teddy succeeded in modelling organic shapes, as intended. But it wasn't aimed at supporting angular shapes more common to the polyhedral primitives of the CAD world. The FiberMesh [NISA07] system (Figure 2.7) addresses this by offering a sketch-based mechanism for allowing sharp or smooth mesh editing operations. An initial curve is inflated as in Teddy. Then subsequent curves can be drawn onto the shape and used as handles to deform it. Curves may be one of two types: A smoothness constraint is enforced across smooth curves, while only C^0 continuity is enforced across sharp curves. Thus the shape is defined entirely by a set of curves.

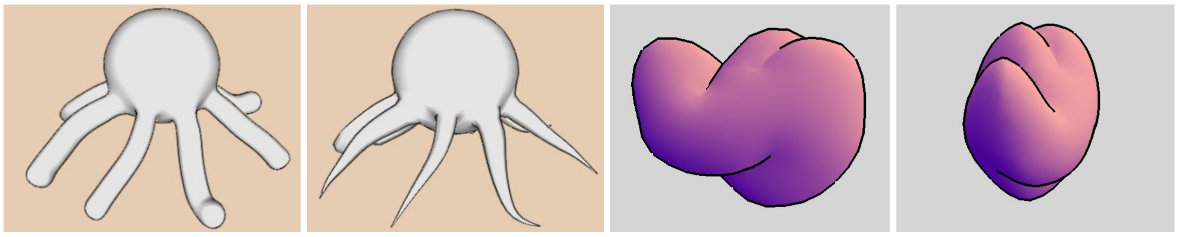


Figure 2.8: **(left)** Smooth and sharp models from the Free-form sketch [WM07] system. **(right)** T-Junctions and internal silhouette information interpreted for smooth shapes in SmoothSketch [KH06].

Free-form sketch [WM07] is another system which addresses the sketch-based modelling of angular and smooth shapes, but via a different approach based on a multiresolution shape representation called a layered mesh, similar to subdivision surfaces, but with shape control at any level of the hierarchy (Figure 2.8).

Few systems so far have attempted to incorporate the more subtle visual shape cues given by features like T-Junctions or interior silhouettes (described in section 1.1). SmoothSketch [KH06] is one such system which infers the hidden contour information implied by a T-Junction and uses it to reconstruct a smooth 3D shape (see Figure 2.8).

2.2.3 Advanced surface representation and volume modelling

Using meshes as the underlying data structure has the advantage of easy visualisation and compatability with many other modelling packages. However mesh modification is a difficult problem. Blending artifacts and self intersections must be handled and the mesh quality can suffer during a long editing session (as shown in the original Teddy system).

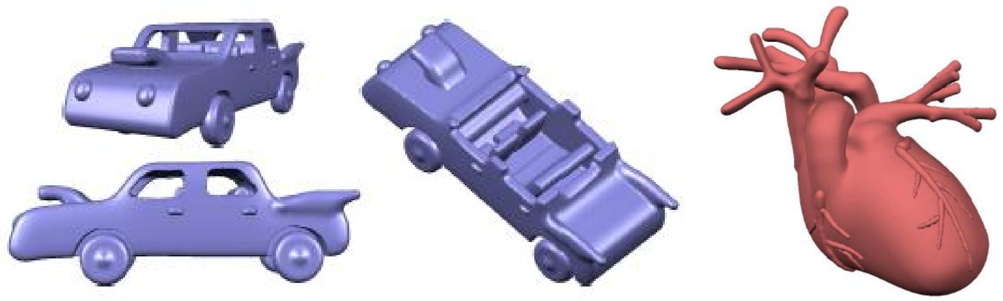


Figure 2.9: Results from the ShapeShop [SWSJ05] system.

A large body of work addresses these issues by using more sophisticated mathematical surface representations. Among these are convolution surfaces [BS91, ABCG05, BPCB08], variational implicit surfaces [KHR02, DAJ03], implicit functions [AGB04], volume data [ONNI06] and hierarchical implicit volume models (BlobTrees) [WGG99]. These more abstract representations provide more natural frameworks for blending, topological changes and deformations, but operations can be much more expensive to compute and the representation must be converted to a mesh for visualisation which is also expensive. The ShapeShop system [SWSJ05] is a well balanced example of such a system (Figure 2.9). Using optimisations such as a hierarchical spatial caching system and volume primitives that can be combined using CSG operations, it provides a flexible yet interactive sketch-based modelling system for general shapes.

2.3 Complex, specific shape modelling using prior knowledge

The previous section demonstrated the power of sketch-based interfaces for quickly modelling general shapes. However using such a system when modelling a specific shape such as a tree or a hairstyle (where many elements are similar, follow general form or placement rules, and are potentially hierarchical) would be unnecessarily time consuming. They would also require the artist to have strong expertise in the complex 3D shape being modelled. In these cases prior knowledge of the nature of the shape can be incorporated into the sketch modelling system. This makes modelling much faster, and can ensure the shapes created look plausible, even if the artist is inexperienced.

In this section we will describe a number of works that use this approach to model a variety of shapes, but differ in the nature of the underlying representation and the generality of their interfaces. The most significant difference is the level of abstraction used to represent the prior knowledge of the shape. The most basic approach is to model shape geometry directly: the shape primitives available in the system are chosen to best represent the form being modelled (for example cylinders for branches or polylines for hair) and the final result is a static geometric model which may accurately match the user's input and can be rendered but not easily animated or re-purposed. The prior knowledge of the nature of the underlying shape is lost once the modelling process is completed.

A more sophisticated approach is to use a procedural model to generate the final shape. Here instead of modelling geometry directly, the drawn input should be interpreted in terms of higher level parameters used to control the underlying procedural model. In these cases the prior knowledge of the underlying shape is represented compactly in the model, and if parameterised correctly the model should automatically create geometry which matches the user's input strokes. This approach has the advantage of retaining the prior knowledge of the nature of the shape beyond the initial modelling stage so that it can be reused later (perhaps for animation). However the variety of shapes that can be expressed is constrained by the underlying model, and the user may wish to sketch some shape which is unobtainable.

Another important aspect of these systems is the "generality" of the interface. A way to gauge this is to ask the question 'how far is this interface from the experience of sketching the whole shape with pencil and paper?'. This is not to say that interfaces shouldn't differ from the traditional sketching experience - of course digital interfaces offer many unique advantages that should be explored - however the more different an interface is from this (familiar too most people) pencil and paper metaphor, the harder it will be to learn. Additionally if the interface becomes too specific to the underlying model and data-structures then it may not generalise to sketching other types of shapes. For example the flower sketching system of Ijiri [IOOI05] (see subsection 2.3.4) offers multiple different windows for sketching the different parts of a flower. This offers very precise control over every aspect of the shape, but uses different controls in each window and requires the user to learn the whole system. This approach is similar to that used for traditional CAD tools, where every function has its own widget that must be learned. This is the opposite of the approach (inspired from traditional sketching) taken by Anastacio [ASSJ06] in modelling plants (see subsection 2.3.4). Here the use of construction lines guides the plant formation in a global to local fashion and the user sketches much as an artist studying the plant would. The system does however require a specific interface for sketching the leaf shapes.

2.3.1 Categorising prior work

Before detailing the previous work in the area, let us discuss some criteria for analysing them. Defining a clean categorisation is not easy. Some works focus solely on deriving from visual input (the sketched lines), some allow extra haptic input such as pressure and tilt, some focus on animation, or mix sketching with other input metaphors (painting for example). In each target shape area I will consider three criteria and try to place the work relative to the others, in order to better compare them. The three criteria will be:

- Amount of prior knowledge incorporated. Not something precisely measured, but some systems incorporate more assumptions or "rules" than others. Systems are at their best when they incorporate the right balance of prior knowledge, enough to aid rapid design while not limiting expressiveness.
- Expressiveness. This can be considered as 'What percentage of all possible (and plausible) shapes are achievable with this system?'. For example if one hair sketching system allows only straight hair, but another allows straight and curly, then the second could be considered more expressive.
- User effort required, or alternatively, the amount of input required from a user. For example, requiring the user to sketch every hair on a head or every leaf on a tree requires a lot of user effort.

These criteria come with some strong caveats. Certainly expressiveness is subjective. For example a system which utilises sketching for specifying shape and animation for some phenomena may allow much more expressiveness in animation control than in shape control. We will concentrate on expressiveness for shape control. The relative value for any of these criteria should not be considered in terms of 'good' or 'bad'. For example more user effort required is not necessarily a bad thing if it enables a much larger range of expressiveness. Less prior knowledge can be a good thing when it avoids over-constraining a solution and thus increases the expressiveness of a solution (of course perfect prior knowledge of a shape implies no user input is required at all, and defeats the purpose of a creative interface). These criteria have a degree of correlation which cannot be avoided, but they provide an opportunity to look for patterns or groupings which may illuminate future work. In any sketch-based modelling system there are three key issues to address:

1. What does the sketched picture primitive represent in the scene?
2. How is this representation given a 3D position in the scene?
3. How is this representation given a 3D shape?

To address the first issue I will make specific observations on the type and meaning of the lines it is possible to sketch in each system, using the definitions from [Wil97], outlined in section 1.2. It is often difficult to determine from a sketching paper what is sketched and how it is interpreted. I hope that by introducing this more rigorous language to the discussion of prior work it will be possible to more easily see the similarities and differences and again perhaps suggest new directions for research.

The second issue (determining 3D position in the scene) is commonly solved via one of the following approaches (in order of sophistication):

- Projection onto an arbitrary plane, previously positioned by the user. This is the most simple approach, the shape remains planar but is given a position in the scene. User effort is required in positioning the plane.
- Projection onto a plane parallel to the view plane, with the plane depth assigned by the user. The shape remains planar. A little less user effort is required in positioning the plane.

- Projection onto a plane positioned relative to the view and a supporting object. For example with a tree or plant, the initial stroke is assumed to start on a supporting ground plane, and the drawing plane is automatically positioned parallel to the view plane. Thus the depth and orientation in 3D is completely automatic, but the shape remains planar in the scene. Subsequent strokes are positioned on planes parallel to the view plane with a depth determined from the nearest previously defined structure (in our example a branch or stem).

- Projection onto a non planar supporting surface, local shape is due to the intersection of the projected stroke and the supporting surface. This is used in the flower modelling system of [IOI06a] (see subsection 2.3.4): a curved surface is derived from the skeletal stem stroke, and then the external silhouettes of the organ shape from the top view are projected down onto the curved surface. Complications arise such as how to deal with multiple intersections.

- Projection onto a plane positioned relative to the view and a supporting object, but with local shape modifications based on prior knowledge. The projection of the 3D shape matches what was drawn in 2D. This is the approach used to form spiral shaped branches in the tree sketching system of [IOOI05] (and is used in [OOI05, IOI06b], see subsection 2.3.4); the prior knowledge exploited is that often a branch or stem will have constant curvature in 3D, which can be derived from the curvature in 2D and maintains the projection of the branch to the viewplane.

In all these approaches the projection of the final 3D feature represented by the 2D sketched line onto the viewplane corresponds exactly to what was drawn.

The third issue (determining 3D shape) can be solved via one of these approaches:

- From one viewpoint:

  - The screen space primitive represents the whole of the shape in the 3D scene shape. For example a line in screen space becomes a 3_{100} thin hair, or branch in the scene. Thickness is pre-set, or derived from a supporting model (for example the varying radii of branches using the pipe model, depends on the depth of the branch in the tree hierarchy).
  - The screen space primitive is a line and it represents a curve in 3D. Then the user can specify more information from the same viewpoint by sketching the shadow that would be cast by the 3D curve onto a ground plane. This method was introduced in [ZHH96] and explored in more detail in [CMZ$^+$99]. However as noted in [CMZ$^+$99] determining the correct shadow to sketch for the desired shape can be difficult even for trained artists.
  - The screen space primitive represents the external silhouette of the shape in the 3D scene. Prior knowledge of the shape represented by the silhouette is used to generate a surface or volume in 3D. For example [IOOI05] allows this as an alternative way to draw branches. [ZS07] allows this as a way to draw a buttress modification to a trunk. [TCH04] uses this to define clothing.

- From multiple viewpoints:

  - Screen space primitives from orthogonal viewpoints are used to specify or deform the shape. [IOOI05] uses a top view of a petal's external silhouette to define an initially flat petal, deforming cross-sectional strokes from orthogonal viewpoints then refine the shape. Similarly [ASSJ06] uses three orthogonal views to define an initial leaf shape.

– If the screen space primitive is a line and it represents a curve in 3D then it can be redrawn from a second viewpoint while maintaining its projection to the first viewpoint using the epipolar techique of [KHR04].

There now follows a detailed description of the recent work using sketching and prior knowledge for complex, specific shape modelling.

2.3.2 Clothing

Clothing design is a skilled profession which requires an understanding of tailoring. Virtual clothing of characters traditionally emulates the real process (using a system such as [Aut08b]). 2D cloth panels are designed, placed around a 3D body model and "stitched" together before running a physical simulation to find the rest position of the clothing on the model. This whole process is time consuming and requires expertise in fashion design. Sketch-based systems can expedite the process.

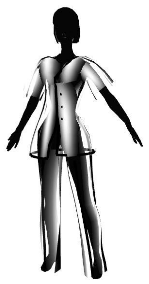


Figure 2.10: A sketched garment design which can be visualised from multiple viewpoints (from [BCD01]), however no clothing model is produced.

For example the work of Bourguignon *et al.* [BCD01] can be used to sketch out ideas for a garment from multiple viewpoints. But the system does not reconstruct a virtual garment; it only enables the 3D visualisation of ideas (see Figure 2.10). This is possible because the drawn lines (1D external silhouettes) are interpreted locally as small curved surfaces (3_{110}), disconnected from anything else in the scene.

The related problem of placing previously designed virtual garments on a 3D body using a 2D interface was addressed by Igarashi and Hughes [IH02]. This system (see Figure 2.12) requires as input a body model and a set of 2D clothing patterns. The user then sketches marks on the patterns and corresponding marks on the body. The system then determines the placement of the garment on the body while respecting the cloth constraints and the sketched user input. Once positioned the user can make adjustments by dragging the clothing along the body surface, using virtual pushpins to constrain motion if required.

[TCH04] presented the first sketch-based system for virtual clothing design (Figure 2.11). This system enabled the production of a 3D garment mesh using only a sketched 2D garment silhouette from a fixed viewpoint. In Willats taxonomy, the picture primitives sketched are 1D external silhouettes which represent the boundaries of a 3_{110} garment in the scene. The garment is automatically positioned by utilising prior knowledge (the body model and assumptions on garment fit). However the produced garments did not look realistic, as they consisted of smooth surfaces which did not exhibit folds. Also the garment surfaces were not piecewise developable, which means they could not be unfolded onto a plane and used as patterns for making the garment in reality or creating undistorted texture maps. The two chapters of Part II detail my contributions to solving these problems.

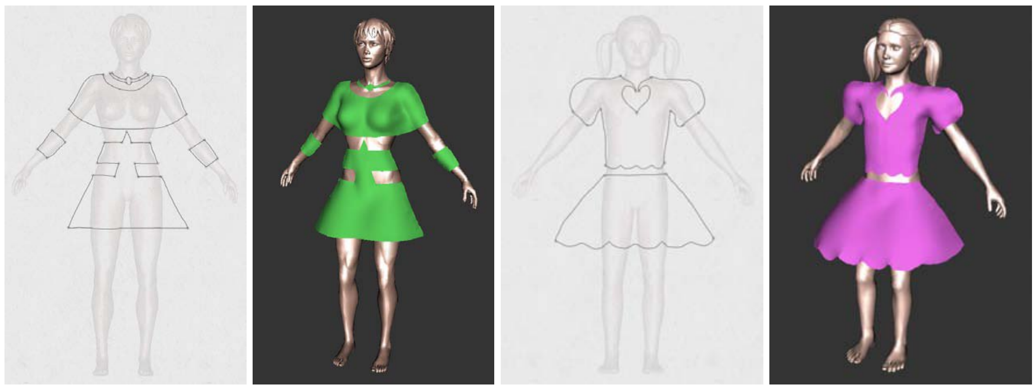


Figure 2.11: A sketch-based system for clothing design [TCH04]. However the garments do not exhibit folds.

As there are so few prior works in the area of sketching clothing I will omit a relative comparison based on the criteria previously defined.

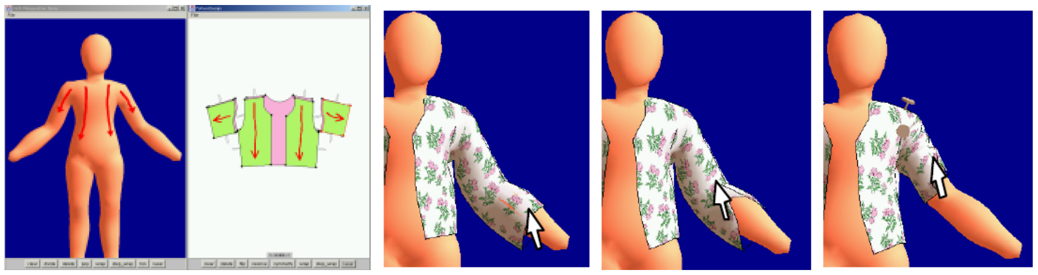


Figure 2.12: The interface for placement and then dragging of pre-designed clothing (from [IH02]).

2.3.3 Hair

Realistic virtual hairstyles are expensive to design due to the large number of hair strands on a head. Standard modelling techniques typically allow the user to define and shape a few hundred guide strands around the head. They then generate many individual hair strands through interpolation or wisp based models. The underlying guide strand models are either geometric or physically based (Ward *et al.* [WBK$^+$07] present a good survey).

A number of geometric interactive modelling systems were proposed for hair design [CSDI99, KN00, XY01]. Most of them provide the user with precise control of the length, position and curliness of hair, but require a large number of successive manipulations, from the delimitation of the scalp to the positioning and shaping of guide strands. This can lead to several hours for the creation of a single head of hair. This process can be made easier using multi-resolution editing [KN02]. Hair is shaped from coarse to fine using a hierarchy of hair wisps, and by copy-pasting some sub-wisp's style to others wisps. Skilled users can achieve visually realistic results but still requiring more than an hour for each head of hair.

An alternative to manually creating hair geometry is to reconstruct it from images [PBS04, WOQS05], an approach which achieves unsurpassed realism. However, using it to generate the desired hair for a CG character requires having a real model with exactly the required style, in addition to the possible problems of adjusting the captured hair to a head of different morphology. In this work, we borrow the idea of re-using some data (such as hair color and examples of strand shapes) from a real photograph, but just use it as a guide and generate hair from scratch onto the desired 3D head.

Physically-based hair styling [AUK92, HMT00, Yu01, CK05] was introduced as a good alternative to purely geometric hair modelling, since it reduces the amount of work required by the user while enhancing realism. Bertails *et al.* [BAQ+05] presented a validated static model for hair and used it to generate a number of natural hair styles through the tuning of a few mechanical parameters. Such parameters can be tedious to set up by hand when all the strands do not have the same length, curliness or stiffness. To ease the generation of various hair styles, the authors demonstrated the ability of their model to reproduce a classical hair styling process, from growing hair to wetting, cutting and drying it. Ward and Lin [WGL07] took the same idea even further. They plugged a haptic interface into their physically-based engine for hair [WGL04], enabling the user to feel hair while combing and styling it.

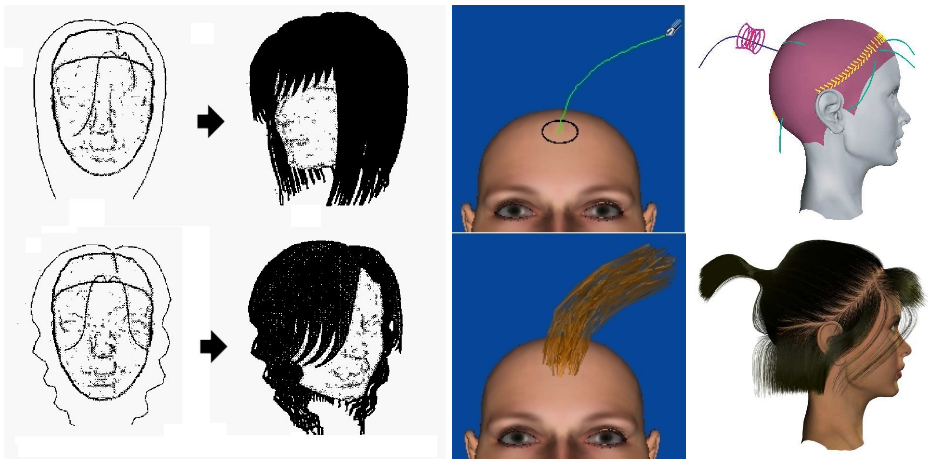


Figure 2.13: Three geometric hair models with sketching interfaces. The cartoon style hair of [MIAI05], the hair dressing sketching system of [Mal05] and the vector field editing technique of [FWTQ07].

Specific hair sketching approaches have been used previously for modelling geometric hairstyles. Mao et al. [MKKI02] present a system for creating geometric models of cartoon character hairstyles. The user first sketches the scalp boundary through a series of strokes that are drawn directly onto a 3D model of the head from user-chosen viewpoints. He then draws one stroke specifying the silhouette of the hairstyle viewed face on. This stroke is used to infer the hair volume and global shape using the assumptions that hairs are planar, that all hairs are similar in shape to the silhouette stroke and that they deform smoothly in relation to the head model shape. Front and back regions are treated as sweep surfaces using the hair stroke as the profile. In [MIAI05] (Figure 2.13) this work is extended to allow greater local control. Global shape is now controlled by specifying four silhouette strokes and the hair partition line can be drawn directly instead of assuming a central parting. Hair clusters are created which conform to this global shape. Individual clusters can then be reshaped by sketching strokes onto either a front or side view of the head if desired.

Malik [Mal05] presents a comprehensive suite of sketching tools for creating and editing virtual hairstyles (Figure 2.13). The approach emulates the real-world hairstyling process by providing operations such as cutting, combing, curling, frizzing and twisting - controlled by gestures sketched onto arbitrary views of a 3D head model. A tablet is required; as pen pressure is used to determine the scope of many operations. All guide strands must be specified by the user, though newly drawn guide strands inherit shape and positional information from existing nearby guide strands.

Fu et al. [FWTQ07] demonstrate a sketch-based system for creating hairstyles shaped by vector fields (Figure 2.13). The user sketches different types of 3D 'style curves' which are interpreted as boundary constraints in a 3D vector field, formulated as the solution of a linear system which can be solved quickly. The style curves are sketched onto the scalp or a supporting surface which is derived from an existing stroke and the viewplane.

The results are good for straight hair styles, but due to the global nature of the approach local details such as curls would require a high resolution vector field and additional style primitives.

Aras et al. [ABaO08] recently developed a physically based hair design and animation system controlled via a sketch-based interface. Some hair parameters are mapped to stylus input status such as pressure and tilt, while curliness is derived from sketched gestures. The focus in this paper is on animating hair. Using sketched key frames and decomposing guide strands into outline and detail components enables them to animate the hair in realtime.

Discussion

All of these works take a picture primitive of a 1D line, and interpret it as a 3_{100} scene primitive, a thin hair. Figure 2.14 shows a comparison based on our criteria. [MKKI02] uses basic prior knowledge (planar hair, restrictive propagation rules) which restricts the variety of styles that can be modelled (and thus the expressiveness). [MIAI05] adds more prior knowledge (fringe strokes) and more expressiveness (individual cluster and partition control), in both cases little user input is required for a result - though more user input is possible with [MIAI05] thus allowing greater expressiveness. [FWTQ07] is further to the right and higher because more prior knowledge is added (constraints for ponytails, more general interpolation of hair and positioning of partings) leading to greater expressiveness. [Mal05] and [ABaO08] are more expressive still as they allow fuzzy hair as well as straight and guide strands can be placed at will, but at the higher cost of requiring a lot more user input (hence larger dots).

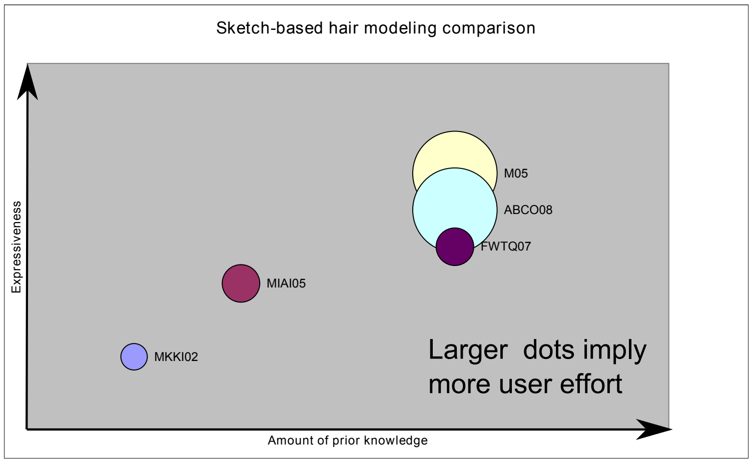


Figure 2.14: Comparing hair sketching systems according to the categorisation discussed in subsection 2.3.1. Amount of prior knowledge incorporated increasing on x-axis. Expressiveness increasing on y-axis. Larger dots indicate greater amounts of user input required.

In all of these previous works the ultimate goal is to create a geometric model for a head of hair which in some way corresponds to the set of sketched inputs. Our work [WBC07] has a different goal. The contribution of our work is to create a set of parameters which control a physical model, which would in turn generate a geometric hair model corresponding to the set of sketched inputs. A correctly parameterised physical model has a number of benefits over a pure geometric representation (it can be used for animation, it could be transferred to a different head model) and places our work further to the right and higher still based on this set of criteria as explained in section 5.4.

2.3.4 Plants

Trees, plants and flowers exist in most natural environments and are thus important in many applications (film, games, military simulations, virtual worlds).

Trees

A real tree may consist of hundreds of thousands of leaves, thousands of twigs and hundreds of branches. Thus modelling a realistic tree without some form of automatic generation is tedious work. The most famous procedural method of generating trees are L-Systems [Lin68, MP96], though other simpler systems exist [WP95, Kru99]. L-Systems define local growth rules, which are recursively applied. Thus global shape emerges from local properties. The main problem when modelling with these systems is that the overall shape of the tree is hard to control, there are a large number of parameters that the user must first understand and even if the user is an expert in the underlying model he may not be able to create a tree that matches one in his imagination. Some work has been done on generating trees from photographs [SRDT01, TZW$^+$07, NFD07], however these systems would require a real tree to exist and to grow in sufficient isolation for the multiple images to be taken. Thus when an artist has a specific tree in mind a sketched based system for designing trees would be useful.

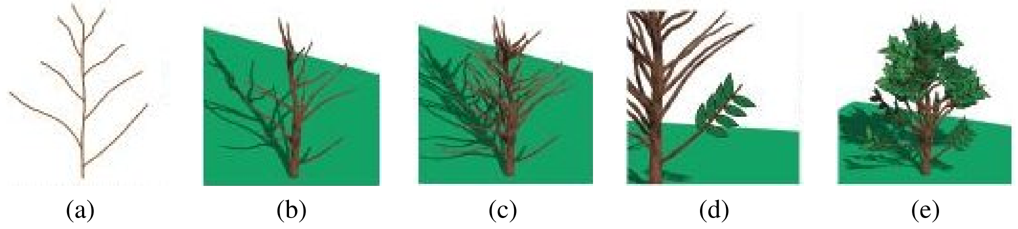


Figure 2.15: The basic tree sketching process from Okabe *et al.* [OOI05]. **(a)** The 2D sketch, **(b)** 3D construction, **(c)** Multiplication, **(d)** Leaf arrangement, **(e)** Propagation.

Okabe *et al.* [OOI05] presented the first sketch-based system for modelling trees (Figure 2.15 and Figure 2.16). The user draws a 2D sketch of the trunk and the hierarchy of branches in the tree. The trunk and branches may be sketched as 1D line picture primitives representing 3_{100} scene primitives (like for all hair sketching systems), or they may be sketched as 2_{11} regions representing 3_{100} scene primitives, where the variation in region width corresponds to a variation in 3D branch cross section. The tree is then automatically embedded in 3D while respecting the user constraints represented by the projection to the screen of the drawn branches. Example based branch multiplication, leaf arrangement and propagation operations are defined. The underlying model is a geometric hierarchy of branches, with each child branch being defined in terms of the parent using the L-System 'turtle' approach. The heart of the system is the 2D to 3D algorithm, which has three main assumptions:

- Assume that user draws only branches growing sideways - This means that roughly only half the 3D tree is supplied by the drawing. The other half is generated by duplicating at 90°the result of placing the first half of the tree in three dimensions.
- Assume that the branches have constant curvature in 3D space. This means that sine wave like branches in 2D are assumed to follow a spiral path in 3D. This technique was first introduced by Ijiri *et al.* [IOOI05].
- Assume that branches spread apart in space as much as possible, child branches are projected onto a 2D distance field at the base of the parent branch, and when new branches are created, they are added as far

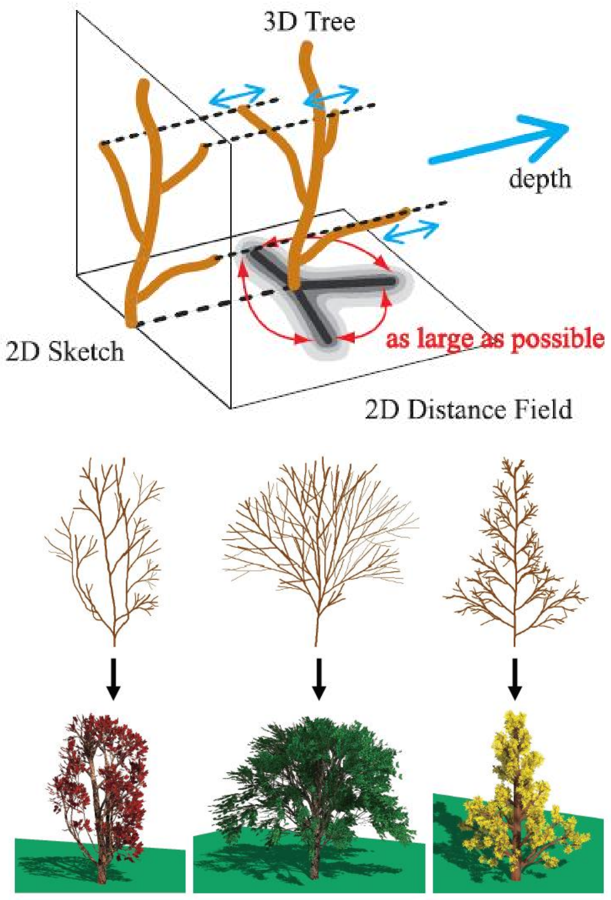


Figure 2.16: The depth computation (spread branches as much as possible) and some results from Okabe *et al.* [OOI05].

from existing branches as possible. To prevent branches from extending too far towards or away from the viewpoint, the search is limited to a scaled 3D convex hull of the tree drawing, generated by sweeping a circle along the 2D convex hull.

The system allows an artist to create a 3D tree in a matter of minutes, which compares well against traditional methods. (They state that a tree which took 30 minutes to model in the Xfrog system took only 10 minutes in their sketch-based system). However the system has limitations: the assumption that branches spread evenly in space doesn't account for naturally occuring growth patterns (phyllotaxy) and fine control over the shape of sub-crowns isn't possible without drawing many branches at each level in the hierarchy. We address these limitations with our multi-scale sketched based tree sketching system (chapter 7).

Zakaria and Shukri [ZS07] detail an alternative 'sketch and spray' tree creation system (Figure 2.17), which combines sketching and painting metaphors. The creation process is split into tree and leaf editing modules. Like [OOI05] it focuses on drawing branches directly ($1 \rightarrow 3_{100}$) but leaf placement and density are controlled by an 'airbrush'-like operation where particles are sprayed onto the branches of the tree and 3D leaf positions and small connecting twigs and branches are inferred. A simple branch deformation stroke is proposed (an example of a conceptual line (section 1.2.2) as it doesn't correspond to a visible line in the scene but rather to an imagined force operating on the branch), where branches (which initially lie in the plane parallel to the defining viewplane) can be deformed, with the extent and direction of the deformation being directly related to the length and direction of the deformation stroke. Branch copy operations are defined, but no phyllotaxy information is incorporated so modelling realistic trees depends on the knowledge of the artist. The tree trunk shape can be deformed through the use of a ‘buttress’ stroke (redefining a $1D$ occluding contour to add a 3_{110} mostly flat deformation to the existing trunk), this is effectively a redefinition of the trunk silhouette from an arbitrary viewpoint which doesn’t appear to offer control over the extent of the deformation in the direction parallel to the view direction.

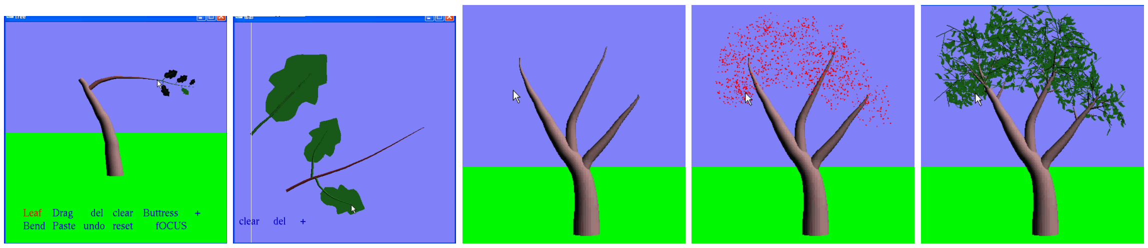


Figure 2.17: The tree and leaf editor windows and the leaf spraying process from [ZS07].

.

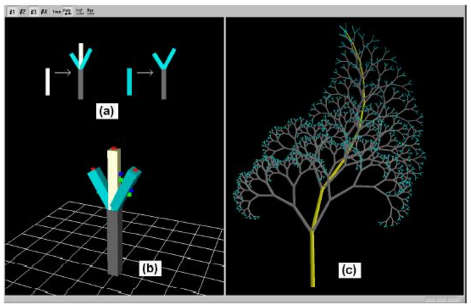


Figure 2.18: The single stroke tree L-System interface from Ijiri *et al.* [IOI06b].

Ijiri et al. [IOI06b] presented a first step towards addressing the controllability of L-Systems for generating trees conforming to a single stroke representing the main trunk $(1 \rightarrow 3_{100})$ (Figure 2.18). As the stroke is drawn it is segmented, with the most recent segment being used as the generating rule of the L-system. The length of the stroke directly relates to the depth of recursion of the system. In this way a whole tree can be quickly defined. Although the shape of the trunk is defined by the user, the shape of the tree crown is not - as it is an emerging property of the system (as with all L-systems). The user can manually adjust some parameters controlling the L-system to change the tree appearance, but architecture changes are limited and crown shapes are not easily defined.

General Plants

Anastacio et al. [ASSJ06] presented a system for sketch-based modelling of plants (Figure 2.19). This system was inspired by the way artists commonly approach plant sketching: construction lines are drawn first which define the layout and scale of the plant. These lines are the *stem* $(1 \rightarrow 3_{100})$, the *boundary* (1D external silhouettes which are interpretated as a 3_{111} scene volume which contains the whole plant) and *inclination* lines, which are conceptual lines that correspond to a segmentation of the plant structure into layers. Finally the plant organs are defined by drawing three lines from three orthogonal viewpoints. The 1D external silhouette is specified from a top view and two 1D cross-sectional profiles (conceptual lines rather than lines that would be visible) are sketched from the side and front views. Together these lines are interpreted as a 3_{110} organ (leaf) surface in 3D (an approach similar to that used for petals on flowers in [IOOI05] described in section 2.3.4).

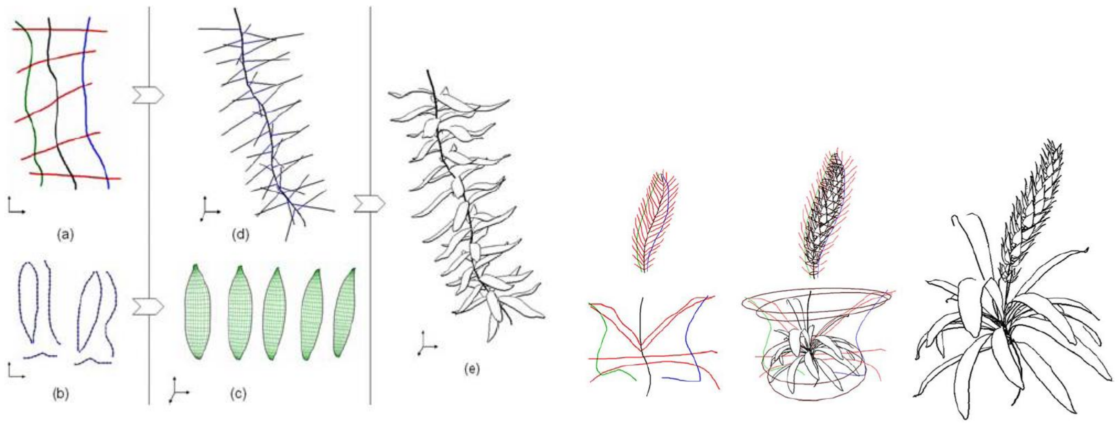


Figure 2.19: Sketching general plants (from [ASSJ06]). **(left)** Process overiew. The user sketches stem, boundary and inclination lines, leaf silhouettes and orthogonal cross-sections. **(right)** Two results combined.

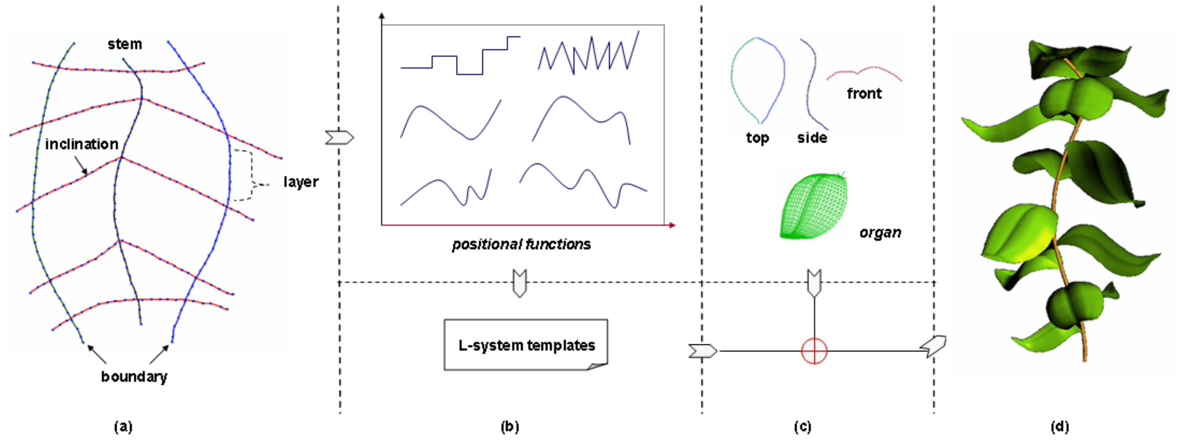


Figure 2.20: Sketching general plants, Anastacio *et al.* extend their earlier system to use L-Systems as the underlying representation (from [APS08]).

A geometric model is then assembled to fit the construction lines. Some parameters like phyllotaxy cannot be sketched and have to be specified via manually set parameters. This work was later extended [APS08], adding L-System templates as the underlying representation (Figure 2.20). The approach taken here (inspired by the way artists construct a plant drawing) is an example of the global-to-local approach this thesis advocates in Part III. Here artists (and thus the system of Anastacio *et al.*) use the prior knowledge that plants consist of many similar repeating features whose layout can be parameterised by silhouettes and internal planes. The silhouette controls variation in size of the repeating features, the internal planes control the frequency and inclination angle of the repeating features. In effect two scales of detail are being specified by the artist: the global structure (silhouette and internal planes) specified from a fixed viewpoint and the local detail (leaf shapes) specified in a separate stage from three orthogonal viewpoints.

Flowers

Ijiri *et al.* [IOOI05] present a system for modelling three dimensional flowers by tailoring a sketch-based interface to the botanical concepts of floral diagrams and inflorescences (arrangements of multiple flowers). Through the separation of structural and geometrical editing they achieve a flexible modelling system. With many specially tailored interfaces the process is quite different from the artist's approach to flower sketching

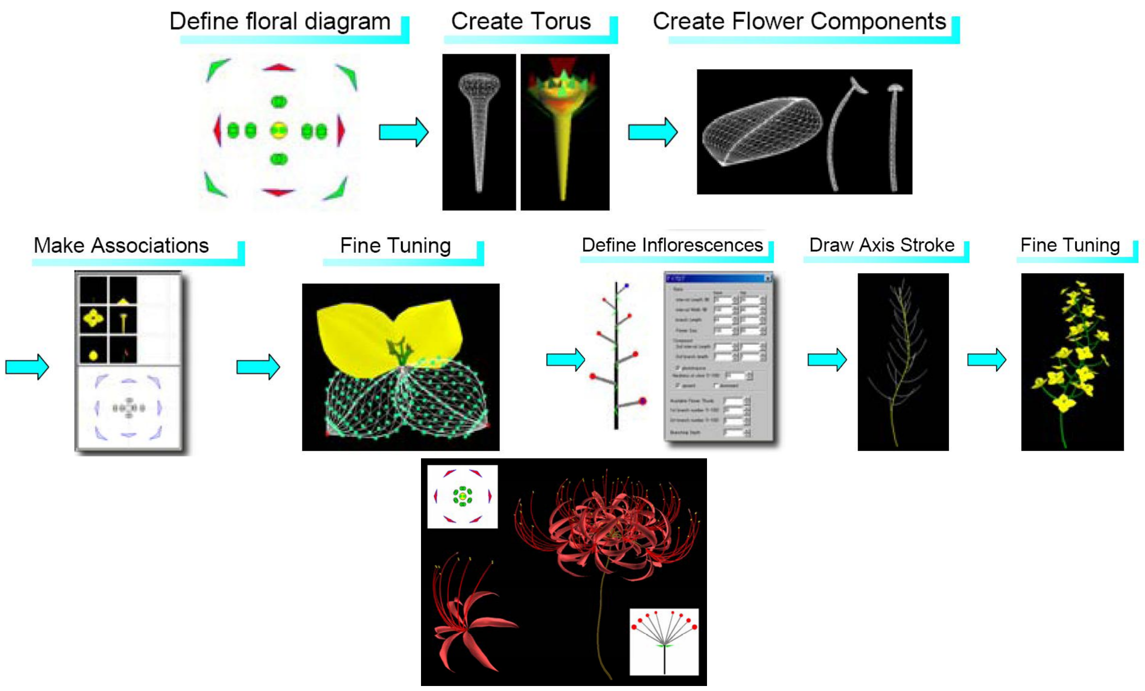


Figure 2.21: The local-to-global flower modelling interface steps and a result from Ijiri *et al.* [IOOI05].

and much closer to the way a botanist would study the many aspects of a flower. However they demonstrate that with a little training users can design interesting flower models in half an hour. The sketch-based operations are limited to the geometrical editors which use prior knowledge on the shape of specific flower features such as the floral receptical (surface of revolution of a 1D external silhouette), pistil (an inflation algorithm similar to [IMT99] applied to a 1D external silhouette), stamen (circle swept along a sketched axis), petals and sepals (defined by a 1D external silhouette from a top view and two cross sectional profiles which deform the shape). The stem is sketched as a line but placed in 3D using the assumption that the curvature is constant in 3D space along the length of the stroke, so that a wavy stroke like a sine wave becomes a spiral in 3D.

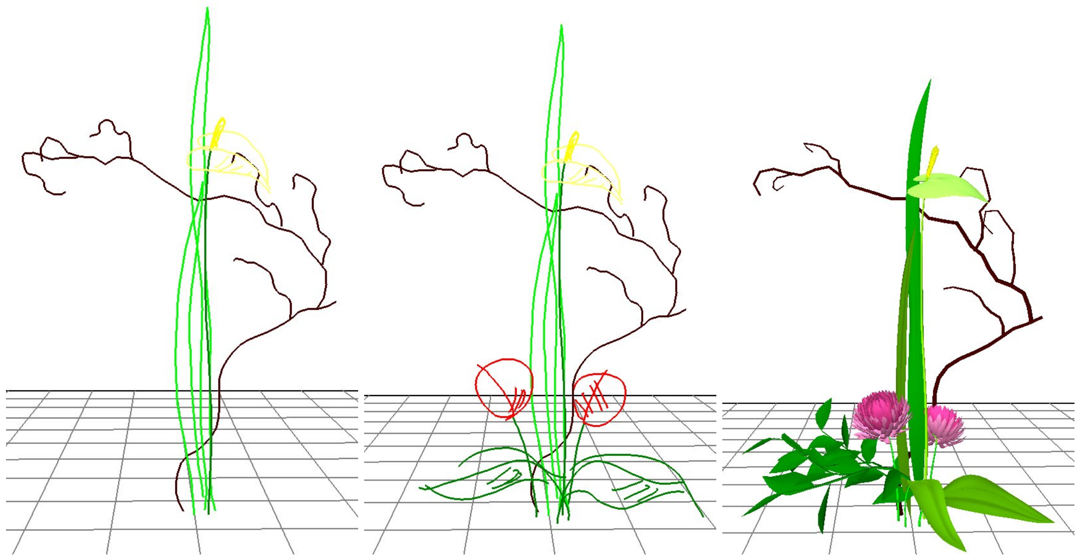


Figure 2.22: The global-to-local flower modelling interface steps and a result from Ijiri *et al.* [IOI06a].

This system produces realistic looking flowers but the approach is local-to-global: the user must specify many details before the final flower is produced. Ijiri *et al.* [IOI06a] later enhanced this system by providing an interface with a global-to-local approach. The user defines a hierarchy of 2D billboards by sketching skeletal strokes (Figure 2.22). Each billboard is placed in 3D by using a plane parallel to the viewport and attached to the nearest existing stroke in the scene. Plant details can be sketched onto each billboard so the ‘look’ of the scene can be quickly explored in rough outlines. Once the rough look is set, the user selects each billboard in turn and uses the relevant specific interface from [IOOI05] to create a detailed element, which is then placed in the scene at the position of the billboard. Only the skeletal strokes and the billboard position are used as input to the interface; the other details must be added as before. However this approach greatly enhances the expressiveness of the previous system, as it allows the whole scene to evaluated roughly before detailed work begins.

Discussion

A comparison for plants is harder than for hair. Trees, flowers and general plants vary widely in scale, for a flower detailed organ shapes and arrangements are more important then general supporting architectures, as they strongly affect the overall appearance, for a tree the architecture is more important than the organ appearance, as individual leaves are rarely seen close up and the tree branches are what define the overall shape. The approaches taken in modelling plants and flowers would be suitable for designing the organs on trees, if a user wanted this level of control over all scales of a tree. All the work described except [IOOI05] follows a global-to-local approach to specifying shape (and in fact [IOOI05] was later extended to support such an approach). This supports the idea that a global-to-local approach is most appropriate in tools designed to support an artist.

Consider Figure 2.23. In this chart we place [ZS07] furthest to the left. It uses some simple prior knowledge on tree structure (branches form a hierarchy and so can be used as supporting surfaces for new branches), but branches have to be oriented manually by moving the viewpoint (they are created as planar branches parallel to the specifying viewplane); there is no automatic orientation as used by [OOI05]. Branches can be deformed away from planar through bend strokes, thus making [ZS07] more expressive than [OOI05], but at the cost of requiring more user effort (larger dot). The two plant modelling systems by Anastacio *et al.* [ASSJ06, APS08] contain a similar amount of prior knowledge to [OOI05]. Phyllotaxy is supported in all three of these systems. However [OOI05] is judged more expressive (at the cost of more user input) as individual branches can be placed exactly by the artist, while exact organ positioning in the Anastacio systems is not directly user specified. The dots for Anastacio’s two works both occupy exactly the same position and size, because in both the user input and possible final models are almost exactly the same; the difference is the underlying representation. Although [IOI06b] is based on an underlying L-System (which in itself contains a large amount of prior knowledge of plant structure), the approach taken doesn’t extract enough parameters from the sketch to capitalise on the expressiveness of the underlying representation. It’s true that little user effort is required to derive a model, but most results will be very similar in architecture and appearance, hence its position on the chart. The flower modelling works of Ijiri *et al.* appear high and to the right of the chart, as they incorporate lots of prior knowledge on the structure of flowers and can produce a large variety of results. [IOI06a] is more expressive than [IOOI05], at the cost of more user effort.

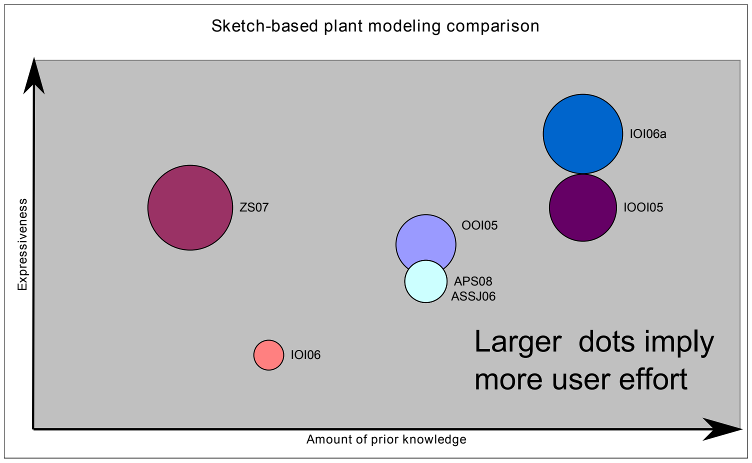


Figure 2.23: Comparing tree sketching systems. Amount of prior knowledge incorporated increasing on x-axis. Expressiveness increasing on y-axis. Larger dots indicate greater amounts of user input required.

2.3.5 Terrain

Terrain is traditionally modelled as a height field and edited from a bird's eye view (directly from above) using an intensity image and a traditional painting package, or a painting package adapted to modelling heightfields specifically [Gra08, Pla08]. However landscapes are nearly always experienced by the viewer from ground level, and so a more natural approach would be to specify the contours of the terrain from this viewpoint. Two previous works have made steps towards such a terrain editing system. The Harold system by Cohen *et al.* [CHZ00] defined a number of sketch-based operations aimed at constructing a simple 'sketchy' virtual world from within that world. The terrain height field could be modified by sketching a silhouette contour starting and ending on the ground, and the ground would be raised to meet that contour. However the profile shape of the contours could not be specified and the results were limited to simple rolling hills along linear paths.

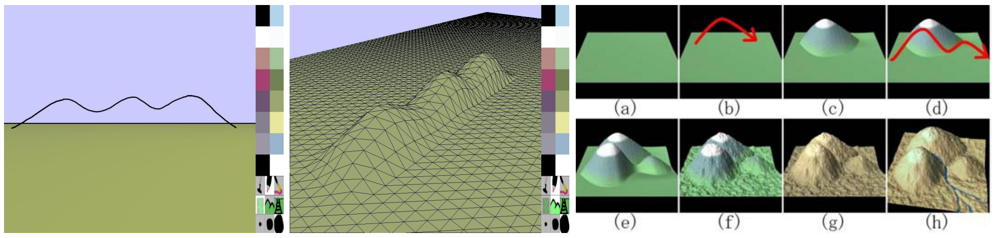


Figure 2.24: **(left, middle)** A ground stroke and the resulting terrain deformation from the Harold system [CHZ00]. **(right)** The terrain sketching system by Watanabe *et al.* [WI04].

Watanabe *et al.* [WI04] suggested an improvement to this system, where the profile shape was determined from the contour shape, a result more in keeping with a user's perceptual expectation. However the terrain paths were still linear and terrain had to be specified from the back to the front.

In chapter 8 we outline our on-going work addressing terrain sketching. We introduce a new method of generating terrain heightfields from only 3D polylines representing terrain silhouettes as viewed from a position within the landscape, and a method of specifying these lines from such positions.

Part II

Annotation of 3D models: Applications to clothing and hair

Introduction

This part presents applications of sketch-based modelling when the nature of the object to be modelled is well known, and can be defined in relation to a supporting surface (modelling a garment on a body or a hairstyle on a head). This can be viewed as annotating the supporting surface. Knowing the nature of the model the user wants to create enables us to use the prior knowledge we have of the object to infer the third dimension from 2D. This reduces the need for specifying the desired shape from several different viewpoints. In some cases, the technique can even be seen as designing a procedural model, and measuring its shape parameters on the user's sketch. 3D is then easily inferred, but the quality of the reconstruction depends on how well the sketch fits the potential outputs of the procedural model.

We illustrate the strength of these dedicated sketch-based interfaces by detailing the specific examples of designing clothing and hair for a virtual character. These examples, for which several different sketch-based reconstruction techniques are presented, will help us characterize the prior knowledge that can be exploited when reconstructing a complex model from a sketch, from basic rules of thumb to more intricate geometric or physically-based properties. This will provide the basis for a general methodology for sketch-based interfaces for complex models.

Let us first emphasize the usefulness of sketch-based modelling for the specific applications described in this chapter. Modelling a garment or a hairstyle for a given character is tedious using a standard modelling system. Usually it is done in one of two ways:

- Geometrically: asking a computer artist to design the garment or hairstyle shape geometrically, by manually modelling the shape of the garment mesh, including the folds that will make it look natural, or creating and shaping the hundreds of generalized cylinders representing the hair wisps of the character (a long process even with the multiresolution editing and style copy/paste techniques of [KN02]). In these cases, the user gets no help from the system (the level of realism will only depend on his or her skill); animating this garment or hair will be difficult since they are not the rest position of a physically-based model; finally, the same process will need to be started from scratch if another piece of clothing or another hairstyle needs to be modelled.
- Using physically-based modelling, which guarantees some degree of realism and eases subsequent animation: for garments, systems such as Maya nCloth [Aut08b] are based on the fact that a garment is a set of flat patterns sewn together, which fold due to gravity and due to collisions with the character's body. In this case the user requires some skill in tailoring in order to design and position the patterns, before a physically-based simulation is applied to compute the garment's shape; similarly for hair, using a physically-based model is possible [BAQ$^+$05] but then the designer requires hair-dressing skills since the hair will need to be wetted, cut, and shaped before obtaining the desired hairstyle.

Whichever method is used, computer artists typically spend hours designing a garment or a hairstyle. In contrast, the sketch-based interfaces presented below enable the creation of a variety of clothing and hairstyles in minutes, using intuitive sketching and annotation techniques which leverage the existing sketching skills of the artist.

The following three chapters present different solutions to sketch-based clothing and hairstyling, classified according to the nature of the prior knowledge they rely on: chapter 3 presents a simple method for generating a plausible 3D garment from silhouettes and fold lines sketched over a front (and optionally back) view of a mannequin[2]. The method for inferring 3D then simply expresses our basic understanding when we see such a

[2]This work was a collaborative effort which resulted in a journal paper [TWB$^+$06]

sketch. chapter 4 compares two solutions[3] for incorporating some prior geometric knowledge, namely using the fact that a garment is a piecewise developable surface, made by assembling a set of 2D patterns; the associated folds can then be generated either procedurally or using physically-based simulation. chapter 5 illustrates the case when a full procedural model of the object in question is available, here a static physically-based model for hair. The sketching interface[4] can then be seen as a way to offer quick and intuitive control over the parameters that indirectly shape the model. Finally, chapter 5.4 summarizes and discusses the general methodology used in these systems, namely combining procedural modelling with sketch-based interfaces to quickly design complex models.

Résumé en Français

Cette partie présente les applications de modélisation basée sur le sketching dans le cas où la nature de l'objet à modéliser est connue et peut être définie par rapport à une surface d'appui (modélisation d'un vêtement sur un corps ou un style de coiffure sur une tête). Ceci peut être interprété comme le fait d'annoter la surface d'appui. La connaissance de la nature du modèle que l'utilisateur veut créer nous permet d'utiliser l'état de connaissance que nous avons de l'objet afin d'en déduire la troisième dimension à partir de la 2D. Cela permet l'extraction de nombreuses informations à partir d'un seul croquis, ce qui réduit la nécessité de spécifier la forme désirée à partir de différents points de vue. Dans certains cas, la technique peut même être considérée comme la conception d'un modèle procédural où la mesure des paramètres se fait sur le croquis dessiné par l'utilisateur. La 3D est alors facile à déduire, mais la qualité de la reconstruction dépend de la manière dont le croquis s'identifie au modèle procédural.

Nous illustrons le potentiel du sketching à travers les exemples d'une interface de conception de vêtements et de cheveux pour un personnage virtuel. Ces exemples, pour lesquels plusieurs techniques à base de sketching sont présentées, nous permettront de caractériser les connaissances préalables qui peuvent être exploitée lors de la reconstruction d'un modèle complexe à partir d'un croquis, ainsi que les principes de base de règles géométriques plus complexes ou de propriétés physiques. Cela fournira la base d'une méthodologie générale pour le design d'interface de sketching pour les modèles complexes.

Permettez moi d'abord de souligner l'utilité de la modélisation par sketching pour les applications décrites dans ce chapitre. La modélisation d'un vêtement ou une coiffure pour un personnage donné est fastidieuse si on utilise un système de modélisation standard. Généralement, elle est faite de deux façons:

- Géométriquement: on demande à un graphiste de concevoir la forme géométrique du vêtement ou de la coiffure, ce qui peut impliquer de modéliser manuellement le mesh du vêtement, y compris les plis pour le rendre plus naturelles, ou la création et la mise en forme de centaines de cylindres généralisés représentant les mèches de cheveux du personnage (un long processus, même avec les techniques d'édition multi-style et de copier/coller de [KN02]). Dans ces cas, l'utilisateur ne reçoit aucune aide du système (le niveau de réalisme ne dépend que de ses compétences), l'animation de ce vêtement ou des cheveux sera difficile car ils ne sont pas dans la position de repos du modèle physique sous-jacent. Enfin, le même processus devra être repris de zéro si un autre vêtement ou un autre style de coiffure doit être modélisé.

- Utilisation la modélisation basée sur un modèle physique, ce qui garantit un certain degré de réalisme et facilite ultérieurement l'animation: pour les vêtements, des systèmes tels que Maya nCloth [Aut08b]

[3]These collaborations resulted in two conference papers [DJW$^+$06, RSW$^+$07].

[4]I was the primary author of this work which was published at the Shape modelling International conference [WBC07].

sont basés sur le fait que le vêtement est un ensemble de pièces de tissu planes cousues entre elles, qui vont se déformer et se plisser sous l'action de la gravité et des collisions avec le corps du personnage. Dans ce cas, cela demande une certaine habileté de l'utilisateur pour la conception et le positionnement de la structure physique avant que la simulation ne soit appliquée pour calculer la forme du vêtement. De même pour les cheveux, l'utilisation de modèles physiques est possible [BAQ$^+$05], mais ensuite des compétences en coiffure sont exigées du concepteur puisque les cheveux doivent être mouillés et coupés avant d'obtenir la forme de coiffure désirée. Quelle que soit la méthode utilisée, les graphistes passent en général des heures pour modéliser un vêtement ou une coiffure. En revanche, les interfaces de sketching présentées ci-dessous permettent la création d'une variété de vêtements et de coiffures en quelques minutes, et ce intuitivement en utilisant des techniques de dessin et d'annotation qui augmentent les compétences en dessin du graphiste.

Les trois chapitres suivants présentent des solutions différentes pour le sketching de vêtements et de coiffure, classés en fonction de la nature des connaissances sur lesquelles elles reposent: chapter 3 présente une méthode simple pour générer des vêtements plausibles en 3D à partir de l'esquisse de silhouettes et de plis vus de face (et éventuellement de dos) sur un mannequin. La méthode de déduction de la 3D exprime alors simplement notre compréhension immédiate du dessin lorsque l'on voit un tel croquis. chapter 4 compare deux solutions pour l'intégration des connaissances géométriques préalables, à savoir l'utilisation du fait que le vêtement est une surface développable par morceaux, et est obtenu par l'assemblage d'un ensemble de patrons 2D. Les plis peuvent être générés soit physiquement, soit en utilisant une simulation procédurale. chapter 5 illustre la cas où un modèle procédural est disponible; ici, un modèle physique pour les cheveux. L'interface de sketching peut alors être considérée comme un moyen rapide d'offrir un contrôle intuitif des paramètres qui induisent indirectement la forme du modèle. Enfin, chapter 5.4 résume et analyse la méthode utilisée dans ces systèmes, à savoir la combinaison de la modélisation procédurale avec le sketching pour concevoir rapidement des modèles complexes.

CHAPTER 3

Sketching in distance fields: Application to garment design

Clothes are as varied as the people who wear them. From a plain tee-shirt to an intricate ball gown, we would like a simple to use system that lets us model as wide a variety of clothing as possible, ideally a system which is close to the design process an artist would follow using paper and a pencil. Figure 3.1a is an image of a fashion designer's concept sketch. Our interface [TWB$^+$06] pictured in Figure 3.1b closely matches this drawing process by allowing the artist to sketch the outlines and folds of a garment from a front (and optionally rear) view of a character model. The resulting inferred garment is shown in Figure 3.1c. In this section we will discuss the prior knowledge incorporated in the system, describe the interface and then detail the method and implementation.

3.1 Expressing prior knowledge

The key to developing a simple yet expressive sketch-based interface is to carefully reduce the complexity of the problem by first asking: 'What prior knowledge of the problem domain do I possess?' and then following with 'How can I exploit this knowledge when designing the system?'. Our approach is based on the simple observation that garments are designed relative to an underlying body. Designers often annotate a 2D view of a mannequin (Figure 3.1) and so we based our interface on the process of annotating a body model. We made the following simple observations:

- The fit of the garment (tight/loose) from the front view is indicative of the fit from a side view.

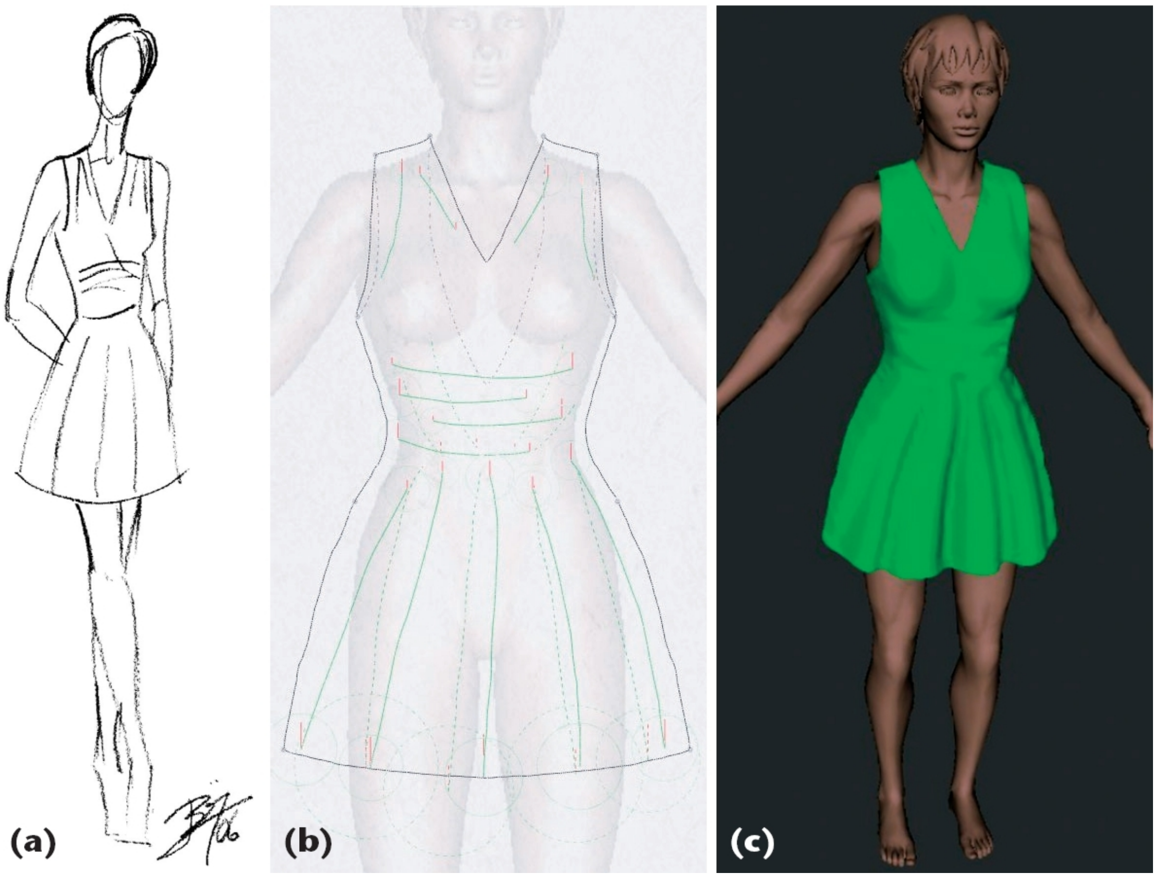


Figure 3.1: **(a)** A designers concept sketch. **(b)** The sketching interface. **(c)** Resulting garment.

- Garments consist of layers of cloth, and usually the cloth doesn't overlap itself within a layer.

Treating these observations as assumptions about the types of garments we can model enables us to reconstruct a whole garment from a single frontal sketch, at the price of only slightly reducing the variety of clothing that can be modelled. In particular the first assumption is the key to solving the main problem in any sketch-based interface: how to assign a third dimension (depth) to two dimensional points along the sketched contours? Our assumption effectively states that once you know the offset of the garment from the body in the frontal plane, then you have the required offset from a side view - which is enough information to place the garment in 3D. We can pre-calculate the offset from the body at any point in space using a distance field, and then use this distance field to rapidly construct garments according to the current sketch. The second assumption gives us a hint about the type of data structures we can employ. If we assume a layer of cloth cannot overlap itself, then we can represent each layer using a height field.

3.2 The sketch-based interface

To keep the experience close to that of using paper and a pencil we wish the interface to be as unobtrusive as possible. This means minimising the use of buttons, modifier keys and input modes. Our system uses only one mouse button (corresponding to the pen down event when using a tablet), two pen modes (drawing garment contours (the default mode) or drawing garment folds), and an optional time-saving vertical symmetry mode. Functionality while drawing is offered through gesture interpretation, with the intention of keeping the user focused on the design task.

Typical garment design session

We now describe a typical user session in order to illustrate the whole process. Our hypothetical designer, Lucy will sketch a skirt on a female model.

Contour mode

Lucy first draws a line across the waist (Figure 3.2a), indicating the top of the skirt, and then a line down the side, indicating the silhouette of the skirt, then a line across the bottom in a vee-shape indicating that she wants the front of the skirt to dip down, and finally the last side, forming a closed 2D boundary. A simple corner-detection process is applied to break the sketch into parts; one extra corner is detected by accident and Lucy can delete it with a deletion gesture. She may also add new breakpoints if required by drawing a small stroke crossing an existing contour. Breakpoints play an important role in the 3D positioning process (detailed later), since they determine the global 3D position of the garment with respect to the body. The two lines on the sides are classified as silhouettes, the others are classified as border lines.

Now Lucy asks to see the garment inferred by the system by pressing a button. A garment surface matching the drawn constraints and adapted to the shape of the underlying model appears almost instantly (Figure 3.2c).

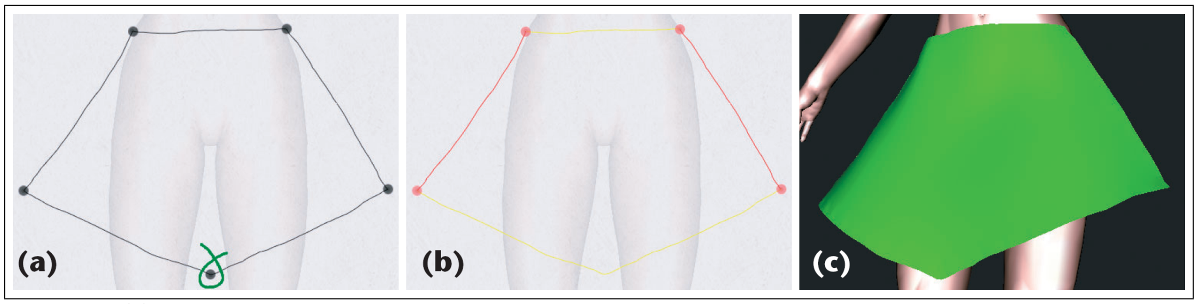


Figure 3.2: **(a)** Lucy has drawn a few lines to indicate the shape of the skirt in *contour mode*; the corner-detector has detected a breakpoint that she does not want. Lucy makes a deletion gesture (a curve in the shape of an α enclosing the mistaken point) to delete it. **(b)** The breakpoint is deleted, and the lines have been classified: the *silhouettes* are in red and the *borders* in yellow. **(c)** The surface inferred by the system once Lucy requests a reconstruction.

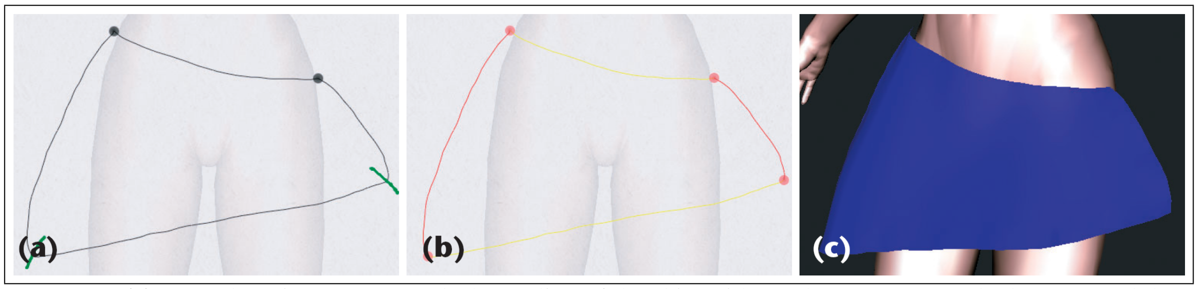


Figure 3.3: **(a)** Lucy drew in *contour mode* the outline of the skirt without sharp corners at the bottom, and the corner-detector failed to put breakpoints there, she therefore gestures (overdrawn in green here) to indicate the need for new breakpoints, in the form of short strokes that cross the contour. **(b)** The new breakpoints have been inserted. **(c)** The reconstructed skirt.

Front/back modes

By default, the user's strokes affect both the front and back parts of the garment. Usually, most of the lines are shared by the two views. This is always the case for *silhouettes*, which by definition join the front and back parts, and it is true for the *borders* in many cases. It is, however, possible to edit front and back borders independently by toggling to the appropriate mode (with the constraint that the contour remains closed), as shown in Figure 3.4. To avoid confusion borders belonging to the current view are rendered with a continuous stroke whereas those belonging to the opposite viewpoint appear dashed.

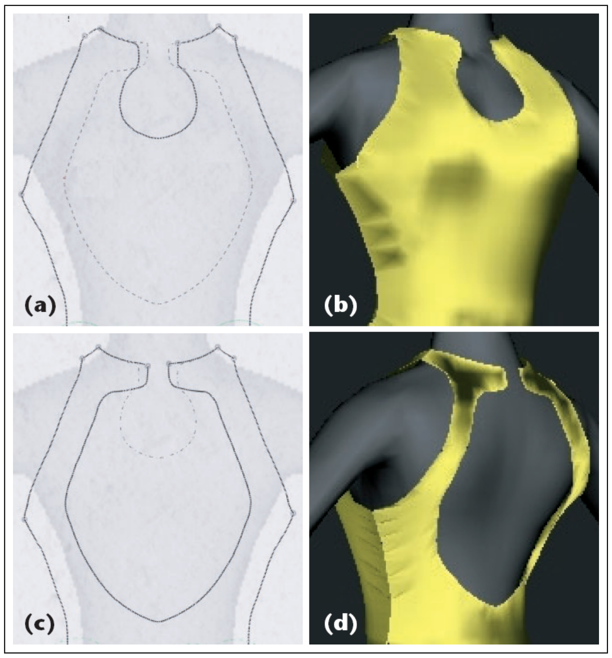


Figure 3.4: Front **(a,b)** and back **(c,d)** of the garment. The borders of the opposite view are shown as dashed line.

Vertical symmetry

It is common for garments to exhibit vertical (i.e. left-right) symmetry. The system offers a *mirror mode* where only half the canvas is active: the other half automatically reproduces mirrored versions of the strokes.

Gestural interface components

The user's marks are interpreted as gestures; in *contour mode* the default stroke interpretation is to construct silhouette and border line segments. Other gestures add breakpoints for the classification process, delete breakpoints, delete a segment or an entire chain of segments, and clear all segments, as shown in Figure 3.5.

The breakpoint-deletion gesture is similar to the standard proof-reader's deletion-mark; the other deletion gestures require multiple intersections with existing strokes to prevent accidental deletions.

3.3 Construction of the garment surface in 3D

Given a set of closed 2D garment boundaries, the technique used to generate a 3D surface consists of three main steps (described for one layer of the garment):

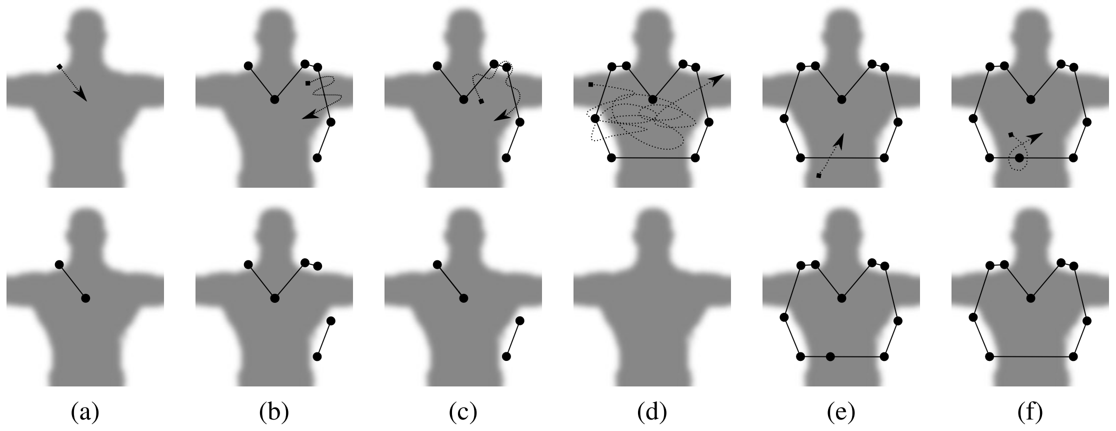


Figure 3.5: The gestures in *contour mode*. **(Top row)** newly drawn strokes as dotted lines with an arrow. **(Bottom row)** result of stroke operation. Black dots are breakpoints in the boundary. **(a)** adding a segment, **(b)** deleting a segment (intersecting scribble gesture), **(c)** deleting several segments, **(d)** clearing all segments (there must be many self-intersections), **(e)** adding a breakpoint, **(f)** deleting a breakpoint.

First, we segment the garment boundary by classifying sections of the boundary as being one of two types:

- *Silhouette* sections exist in the same plane as the body model.
- *Border* sections cross the projection of the body model.

Initially these sections are delimited by automatically detecting points of high curvature along the boundary (these points are denoted breakpoints), the user can add or delete them as required.

Second, these boundary sections are placed in 3D by assigning depth information based on our understanding of garments. As silhouette sections exist in the same plane as the body they are assigned a depth of zero along their interior (z=0). Border sections must be assigned a depth that varies smoothly and in relation to the body. This is achieved by calculating the distance (d) of each breakpoint from the body and then interpolating this distance along the border between breakpoints. A depth that maintains this distance from the body at each point is then assigned along the interior of the section.

Third and finally, this depth information is propagated from the boundary to the interior of the garment by a diffusion process. We now have all the information required to generate a garment surface.

Many of these steps are accelerated by the use of a distance field, pre-calculated from the model. We now explain the distance field, and then each step is described in more detail.

Distance field

To accelerate the algorithm we precompute a distance field (using the octree-based algorithm of [JS01]) when the model is first loaded. This field is a regular 3D grid which stores the closest distance to the model at each grid point (Figure 3.6b). Distances from non-grid points can be calculated using tri-linear interpolation. The distance field is signed so that points inside the model have negative distances.

The system uses the distance field each time it needs to find the z-coordinate to assign to a point $p(x_0, y_0)$ to position it at a given distance from the model. This is accomplished by stepping along the ray $R(z) = (x_0, y_0, z)$ and stopping when the required distance value is reached.

Mesh quality and computation time depends on the distance field resolution, which is user configurable. For our examples we use a 128^3 grid.

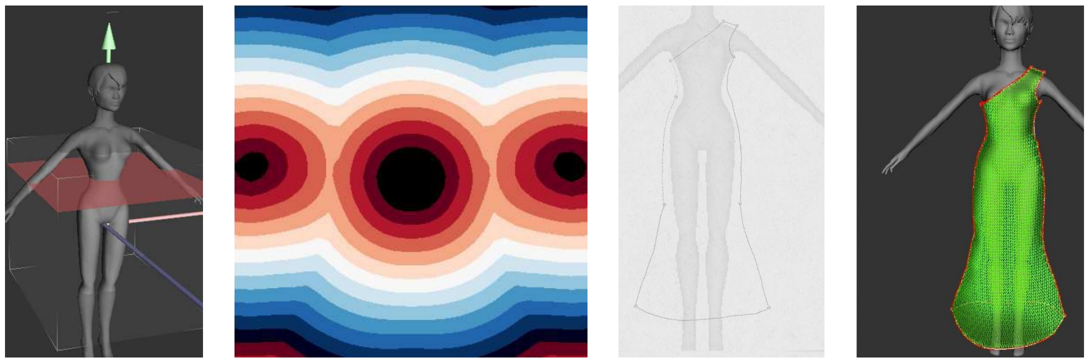

(a) (b) (c) (d)

Figure 3.6: **(a)** A 2D slice through a model. **(b)** The corresponding isocontours of the 2D slice of the 3D distance field. **(c)** A garment sketch. **(d)** The corresponding garment surface calculated using the distance field.

Converting the 2D contours into 3D

Once the boundary contour is complete the contour segments are classified as either *border lines* or *silhouettes* depending on whether the segments projection crosses the models projection in the xy-plane (border line) or not (silhouette). This is done efficiently using a projection mask of the body (body mask) stored in a buffer.

To position the silhouette lines in 3D we simply set the depth (z) to zero, as these lines exist in the same plane as the body, and act as seams joining the back and front layers of the garment. We then set the d-values for interior points of the silhouette to those stored in the distance field.

Having established the values of z and d along silhouette edges, we need to extend this assignment to the border lines. We do this in the simplest possible way: we interpolate d linearly along each border line, and then at each interior point search the distance field for a point with a z value which is that distance (d) from the model.

All points along the contours are now in 3D, and have an associated distance to the model (d-value).

Surface generation from 3D contours

Just as with the contour lines, our main clue for inferring the 3D position of the interior of the garment is the interpolation of distances to the body. The process consists of propagating distance values from the boundary within the garment. We generate a 2D buffer sized to the bounding box of the sketch. Each pixel within the buffer is classified as *in*, *out* or *border* based on its position with respect to the boundary. The border pixels are then assigned the distance values taken from the boundary. We want to minimize the distance variation inside the garment so that it fits as tightly as possible given the border constraints. We pose the problem as a Laplace equation with Dirichlet boundary conditions. Let Ω be the set of inside and boundary pixels, with the boundary $\delta\Omega$. We already know $f_d^*|_{\delta\Omega}$, the pre-determined distance values on the boundary, and want to find an interpolant f_d without extrema over Ω. This interpolant satisfies the following Laplace equation:

$$\Delta f_d = 0 \; over \; \Omega, \; with \; f_d|_{\delta\Omega} = f_d^*|_{\delta\Omega} \tag{3.1}$$

Equation 3.1 can be solved simply by iterating convolutions with a 3x3 neighbor averaging mask over Ω. We then convert the 2D grid to 3D by using the distance field to compute the z-values corresponding to the distances obtained with Equation 3.1.

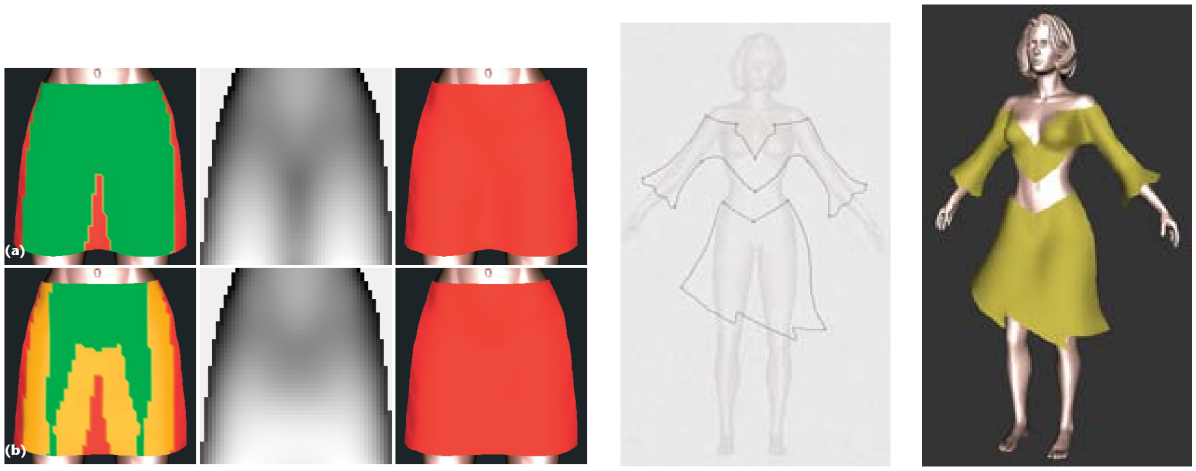


Figure 3.7: **(left)** Mimicking cloth tension. **(a)** Surface reconstruction without accounting for tension. **(b)** Surface reconstruction that takes tension into account. In the left images, the part of the surface over the body mask is shown in green. At the bottom left, the body mask is eroded and the system uses Bézier curves to infer the z-values between the legs. The middle images show the resulting z-buffers. The images on the right show the reconstructed surfaces. **(right)** A smooth garment surface without folds.

Mimicking cloth tension

A garment should not fit too tightly in the region between two limbs because the cloth has a tension of its own. In these cases we correct the garment surface by smoothly interpolating the limbs largest z-values, through a process of eroding the body mask and then using Bezier curves to interpolate the z-values (Figure 3.7).

3.4 Drawing folds

The garment models generated in section 3.3 will appear artifically smooth, unlike real garments which exhibit folds under the effect of gravity. Folds may be added automatically via physical simulation or a procedural method (discussed in section 4.5), but an artist may wish to control precisely where folds appear. This motivates the addition of a sketch-based method for specifying such folds described next. The fold strokes can be seen as being an annotation of another model, the newly generated garment surface model.

Folding mode

Once satisfied with the global shape of the skirt, Lucy decides to add a few folds to obtain a more physically plausible 3D surface (see Figure 3.8). To do this she simply switches to *folding mode* and draws strokes that mark the presence of either *ridges* or *valleys*, and can specify the width and amplitude of these folds in an intuitive way (see Figure 3.9). Fold strokes can be deleted using the same deletion gestures as used for other strokes.

The folds are expressed as deformations to the underlying garment surface. The depth magnitude of the deformation is at a maximum along the fold stroke, and decreases away from the stroke. The deformation's magnitude corresponds to a 2D Gaussian convolved along the stroke path. The support and the amplitude of the gaussian at each end of the stroke are inferred from the 'U' shaped gestures (Figure 3.9) and linearly interpolated along the length of the stroke. The deformations are applied to the garment surface depth map before the final mesh is re-created.

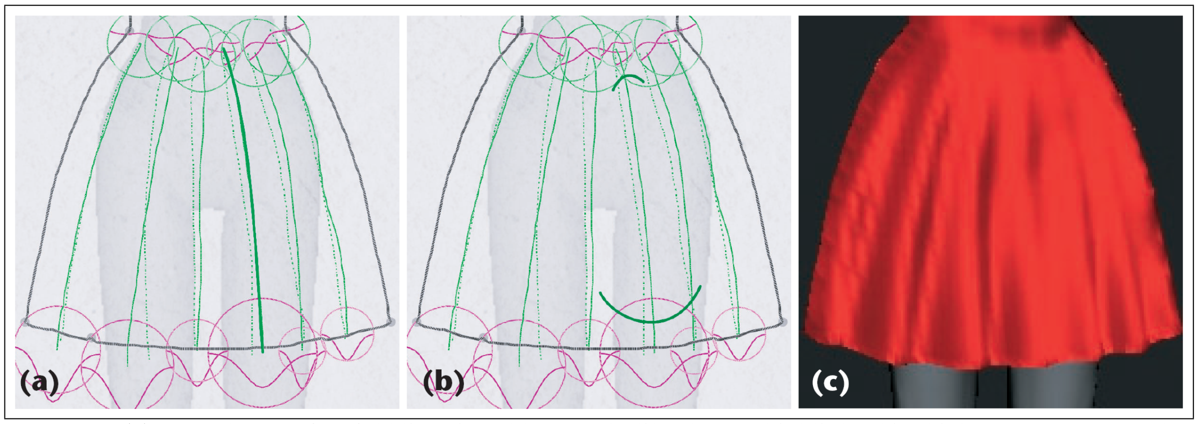


Figure 3.8: **(a)** Lucy draws a few fold lines in *folding mode*, like the one highlighted in thick green, corresponding to *ridges* or *valleys* on the surface of the garment. **(b)** She may draw a "u-shaped" gesture crossing a fold line near either end. The width of the U determines the width of the fold; the depth determines the depth of the fold. The orientation of the U determines whether it is a ridge or a valley fold. These are indicated for the user at all times by a pink circled Gaussian profile at each end of the fold-line, indicating both the width and the depth of the fold. **(c)** The system adds folds to the skirt.

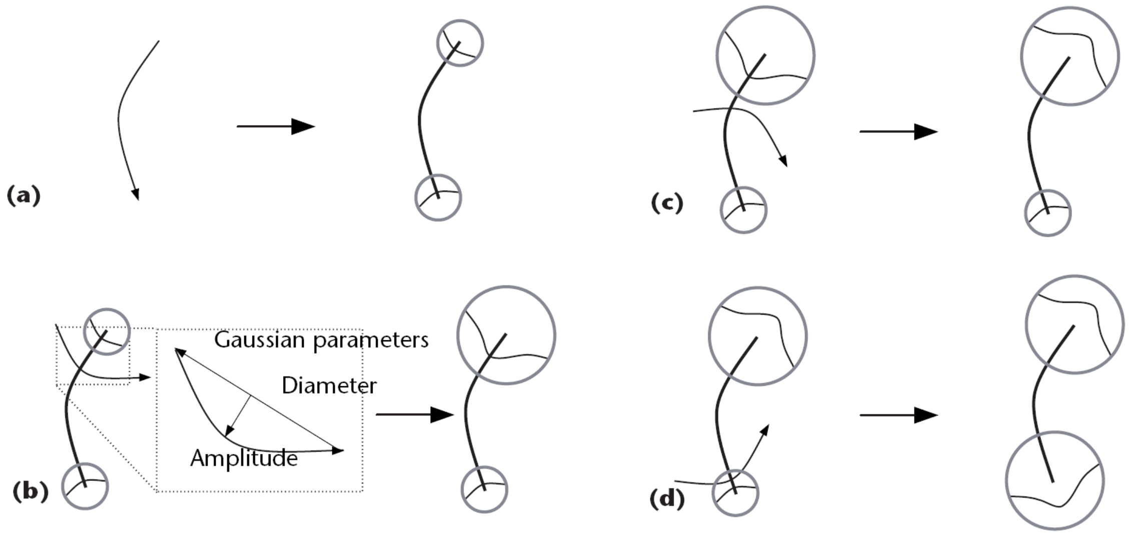


Figure 3.9: The gestures in *folding mode*; new strokes are drawn with arrows. **(a)** adding a fold, by default a *valley*; **(b)** modifying the profile of the fold at one extremity (the closest to the intersection). The shape of the stroke defines both the amplitude and the width of the fold. A stroke that is convex with respect to the end point of the fold results in a *valley*; **(c)** conversely, a concave stroke results in a *ridge*; **(d)** changing the other extremity, making the fold a pure *ridge*.

Although expressive and quick to use, this system relies on the designers skill in reaching some degree of realism: nothing is done to ensure that the garment surface is piecewise developable (that it can be unfolded onto a plane without distortion). It is common knowledge that clothing is made of flat pieces of cloth sewn together. The next chapter discusses ways to add this prior knowledge to the garment creation process, making plausible garment design accessible to non-specialist users.

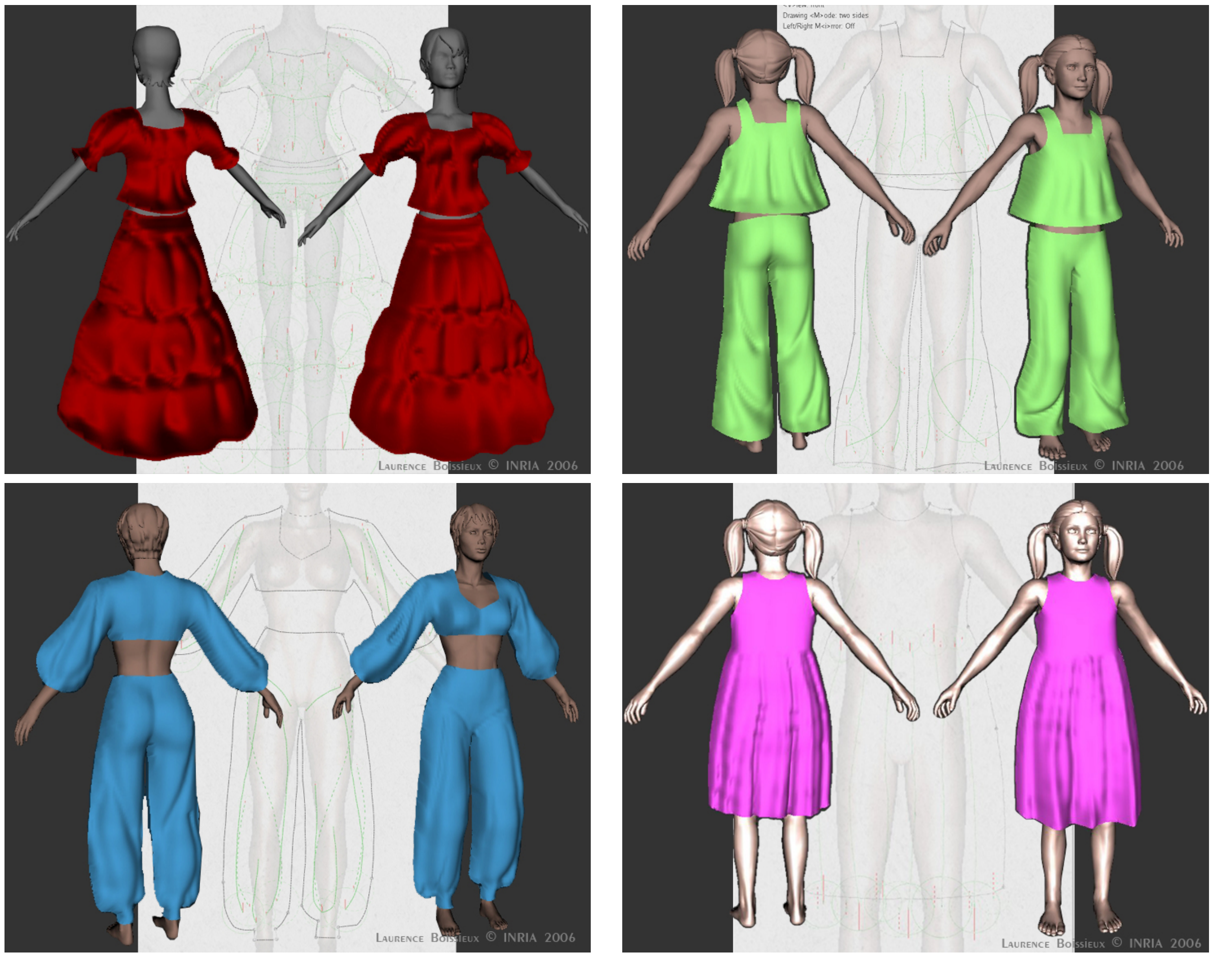

Drawing <M>ode: two sides
Left/Right M<i>rror: Off
Laurence Boissieux © INRIA 2006
Laurence Boissieux © INRIA 2006
Laurence Boissieux © INRIA 2006
Laurence Boissieux © INRIA 2006

Figure 3.10: Final results including sketched folds, each created by fashion designer Laurence Boissieux in less than five minutes.

CHAPTER 4

Incorporating geometric properties: sketch-based modelling of developable surfaces

The previous section outlined a system for quickly producing a visually plausible, virtual garment, which could exhibit folds in the form of deformations drawn by the artist. Although the garment mesh may look plausible, its geometric properties differ from a real garment in an important way - the garment cannot be unfolded flat onto a plane without distortion (surfaces which have this property are known as *developable* surfaces). As real garments are made from panels of flat cloth sewn together this geometric disparity has some implications for the virtual garment:

- Texture maps used for the garment will appear distorted.
- The behaviour of the garment under a physically based simulation would appear strange. For example folds may not fall as expected.
- It is not possible to produce a real version of the virtual garment using real cloth.

These drawbacks motivate an extension of the previous approach. In this section we present our work [DJW$^+$06, RSW$^+$07], which address these drawbacks by extending the approach in the previous section so that the produced garment model consists of a set of developable surfaces. Two different approaches are used. The first is to incrementally alter the existing garment mesh until it closely approximates a piecewise developable surface. The second is to generate a developable surface directly from sketched 3D contours. Both approaches have wider applicability than just clothing, for example they could be used in architectural modelling and engineering, where developability is an important surface property. Once the garment is finished, folds can be added in a post-process, as explained in section 4.5.

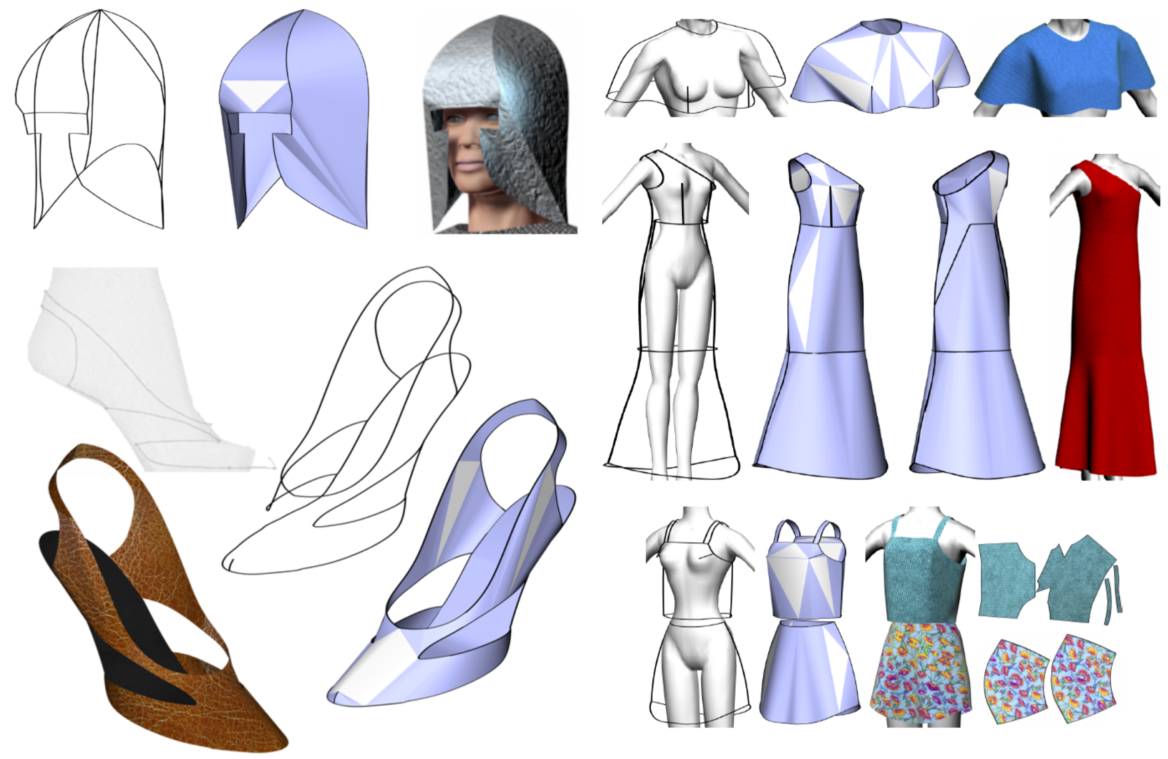


Figure 4.1: Developable helmet, shoe and garments generated from their sketched contours.

4.1 Expressing prior knowledge

The prior knowledge in this case is that clothing is assembled from panels which have the specific geometric property of being developable. An important property of a developable surface is that its normal map on the unit sphere is one dimensional, thus the normal map forms a network of curves (Figure 4.2). We wish the generated garment surface to adhere to this additional constraint, while still respecting the user supplied boundary lines.

In order to decompose the garment into panels we need some additional input from the user, namely the location of seams and darts. Seams are the boundaries between the panels of cloth which assemble to form the garment. Darts are lines on the garment where different parts of the same panel were stitched together, effectively removing a section of material in order to improve the fit of the garment.

4.2 Sketching seams and darts

Seams and darts are a natural extension of the existing drawing method and don't require any additional modes. In the previous section breakpoints along the boundary of a garment panel would always have two associated boundary polylines (one incoming and one outgoing, we call this an order two breakpoint). Now we allow a breakpoint to have one, two or three associated polylines (order one, two or three). A breakpoint with only one

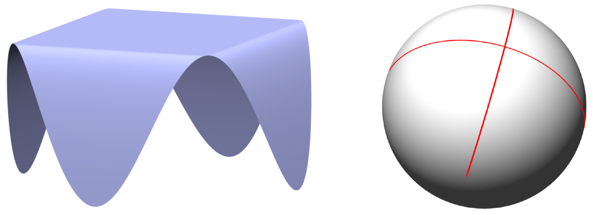


Figure 4.2: (left) A developable surface and **(right)** its normal map.

associated polyline can be considered to be the termination of a dart line. That dart line would have to begin at an order three breakpoint, somewhere on the panel boundary. Of course darts are only valid on the interior of a garment panel boundary. If a dart is extended to rejoin the panel boundary (so that it begins and ends with order three breakpoints) then that panel is split into two separate sub-panels, and the dart has become a seam (Figure 4.3).

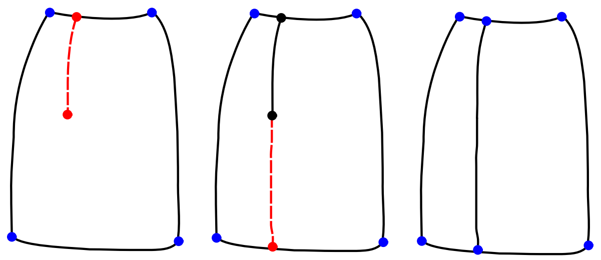


Figure 4.3: **(left)** The user extends an existing skirt outline with a dart. **(middle)** The dart is extended to meet the outline again, **(right)** forming two panels joined by a seam.

We now outline the two methods of producing a set of developable panels which respect the 3D boundaries, seams and darts inferred from the users sketch.

4.3 Creating a developable surface via approximation

This method generates an initial surface using the same distance field method outlined in chapter 3. This surface consists of a set of panels connected at the seams. The approach is to then incrementally modify each panel to bring it closer to being an ideal developable surface, without deviating too far from the input surface. We use an approach inspired by moving least squares approximation [Lev98]. The algorithm follows a two-step procedure for each step of the iteration:

- For each triangle on the surface we find the best-fitting developable surface and move the triangle onto that surface. This breaks the triangle connectivity.
- We then reconnect the triangles, while trying to preserve the new triangle normals and positions.

Each pass of the algorithm further improves the developability of the approximation, at the price of deviating further from the original surface. With each pass the normal map of the garment onto the unit spheres moves closer to the ideal case of a network of curves (Figure 4.4). Once the desired level of developability is reached, the surface panels are unfolded onto the plane to produce the garment patterns (Figure 4.5). When unfolding we use an extension of angle based flattening (ABF++ [SLMB05]) which minimizes shearing.

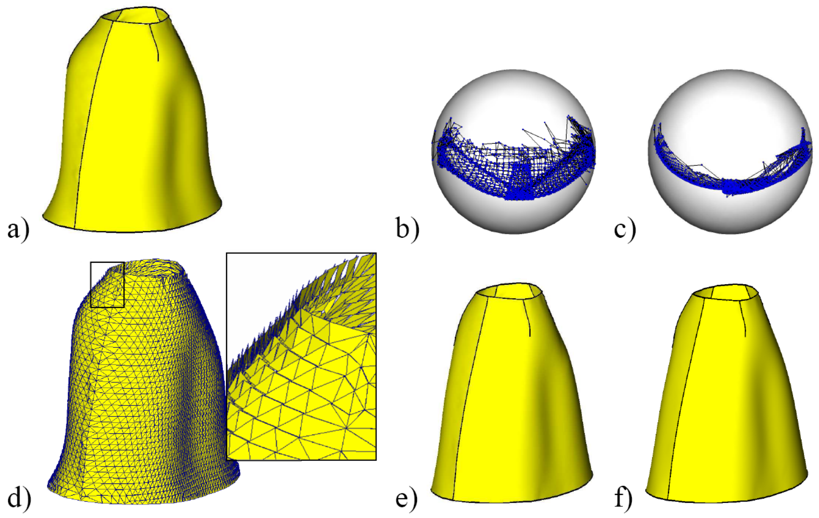


Figure 4.4: Developable approximation stages: **(a)** input; **(b)** normal map of the front panel; **(c)** normal map after transformation; **(d)** mesh triangles after transformation; **(e)** glued mesh after one iteration; **(f)** mesh after three iterations. Between one and three iterations the distortion decreases.

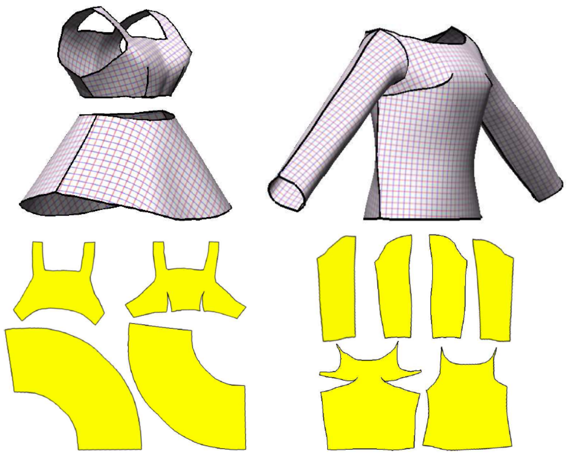


Figure 4.5: Resulting texture mapped developable surfaces and corresponding patterns.

4.4 Creating a developable surface directly from the 3D boundary lines

The approximation method in the previous section produces reasonable results which suffice for non-distorted texture mapping, but the result will rarely be analytically developable, which is a problem when the results are to be used in manufacturing. A more elegant solution than generating a non-developable surface and then approximating it would be to generate the developable surface directly. Our system [RSW^{+}07] models general developable surfaces using the 3D boundaries directly and so requires less user input and less user expertise than most existing techniques. The same input technique is used (annotating an existing model), except that to increase expressiveness, we added the ability for the user to smooth and also deform existing 3D contours if desired by redrawing them from orthogonal viewpoints.

Our method of generating a developable surface from a 3D boundary assumes the input boundary is a piecewise smooth curve. This curve is sampled to produce a polyline. A *boundary triangulation* (a manifold triangulation with no interior vertices) is then generated, using this polyline as its boundary. By construction, any boundary triangulation is developable as the triangles can be unfolded onto the plane with no distortion. We require additionally that the majority of the boundary triangulations interior edges should be locally convex, to ensure a close approximation to a smooth developable surface (see [RSW$^+$07] for details). This condition forms the basis of the method: since most edges of a desirable triangulation must be locally convex, a natural place to identify developable regions interpolating a boundary polyline is the *convex hull* of the boundary, where every edge is locally convex.

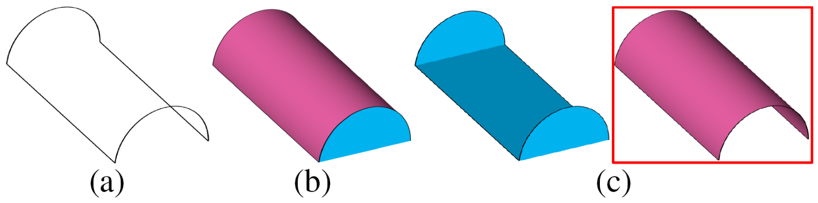


Figure 4.6: Envelope triangulations for a polyline that lies on its convex hull: **(a)** polyline; **(b)** convex hull with envelopes; **(c)** the two envelope triangulations, the framed (right) one is the one selected by the algorithm.

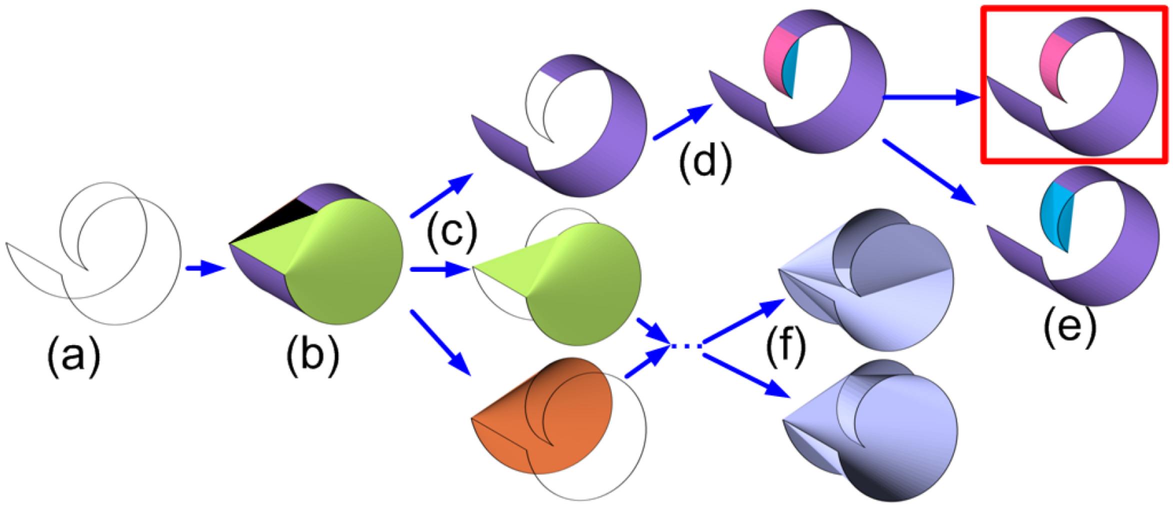


Figure 4.7: Extracting a locally convex triangulation: **(a)** boundary; **(b)** convex hull with extracted charts (interior triangle shown in black) **(c)** individual charts and remaining subloops after subtraction; **(d)** recursing on the subloop formed by removing the purple chart; **(e)** resulting triangulations (the framed triangulation is the one returned by the algorithm); **(f)** two of the triangulations created with different chart choices.

Hence the method proceeds recursively, by taking the convex hull of the polyline, dividing regions where the polyline lies on the convex hull into two *envelope triangulations* (Figure 4.6a,b,c). When the polyline does not lie on the convex hull, the hull is subdivided into charts (sets of hull triangles having certain properties) and the algorithm proceeds recursively on these charts (Figure 4.7). This leads to a number of possible valid surfaces from which the desired result must be selected. The search is guided using a branch and bound algorithm, automatically testing envelope triangulations for desirability using metrics such as smoothness and surface fairness. It can also be guided manually by the user who can choose between the visual representations of the results at each stage in the recursion. Results such as the shoe in Figure 4.1 demonstrate the modelling complexity achievable via this approach.

4.5 Automatic generation of folds

Once again we would like to make the virtual clothing look realistic, which means adding folds. However using the sketching method of forming folds described in section 3.4 would destroy the developable property of the garment surfaces. Fortunately as we now have the 2D patterns for the garments we can use automatic methods to generate folds. We briefly outline two approaches (used in our works [RSW$^+$07] and [DJW$^+$06] respectively): time-consuming but realistic physically based simulation, and a quicker but more limited procedural simulation of folds.

Physically based simulation of folds

The dress in Figure 4.1 is the result of taking the developable garment generated directly from the boundary curves and using a physical cloth simulation [Aut08a] to generate folds. Physical cloth simulations are usually based on modelling the garment as a grid of small masses interconnected by springs. Body collisions are taken into account as correcting forces acting on the cloth. The simulations can be time consuming, but with some expertise in setting the physical parameters very realistic results are possible (such as the red dress in Figure 4.1).

Procedural generation of folds

The garments in Figure 4.8 were generated using our procedural method [DJW$^+$06]. We use the prior knowledge that when cloth is wrapped around a cylindrical object (such as a torso or an arm) and compressed or twisted, it exhibits characteristic buckling patterns, such as diamond shaped folds under compression. We automatically reproduce these patterns by fitting a *buckling mesh* to the 2D garment patterns. This mesh embeds the diamond patterns and the diagonal folds formed by twisting. The buckling mesh is placed in 3D using the correspondance between the 3D garment model and its 2D patterns. When the 3D bucking mesh is deformed via compression or twisting it is constrained to buckle along its major directions, thus forming the desired folding patterns (Figure 4.8). This efficient method can be implemented in realtime, and produces realistic looking folds, although it is limited to pre-determined types of fold.

4.6 Positioning our work

In Figure 4.9 we compare our garment sketching work [TWB$^+$06, DJW$^+$06, RSW$^+$07] with each other and [TCH04] with respect to the categorisation discussed in subsection 2.3.1. [TWB$^+$06] improves on [TCH04] by incorporating the ability for the artist to draw folds directly onto the garment, and to modify the garment from a rear view. The ability to directly express our prior knowledge of folds greatly improves the expressiveness for a moderate amount of extra user input. Hence [TWB$^+$06] is placed higher, further to the right and larger than [TCH04]. [DJW$^+$06, RSW$^+$07] both incorporate the prior knowledge that the garments should be developable, an advanced geometrical constraint, hence placing them even further to the right. The user may also sketch darts and sub-panel panel seams, making them marginally more expressive than [TCH04], but not as expressive as [TWB$^+$06], as we judge the artists ability to draw folds is more expressive than the ability to add them procedurally or via simulation. [RSW$^+$07] is more expressive than [DJW$^+$06] as it also allows resketching curves from orthogonal viewpoints, at the cost of slightly more user effort.

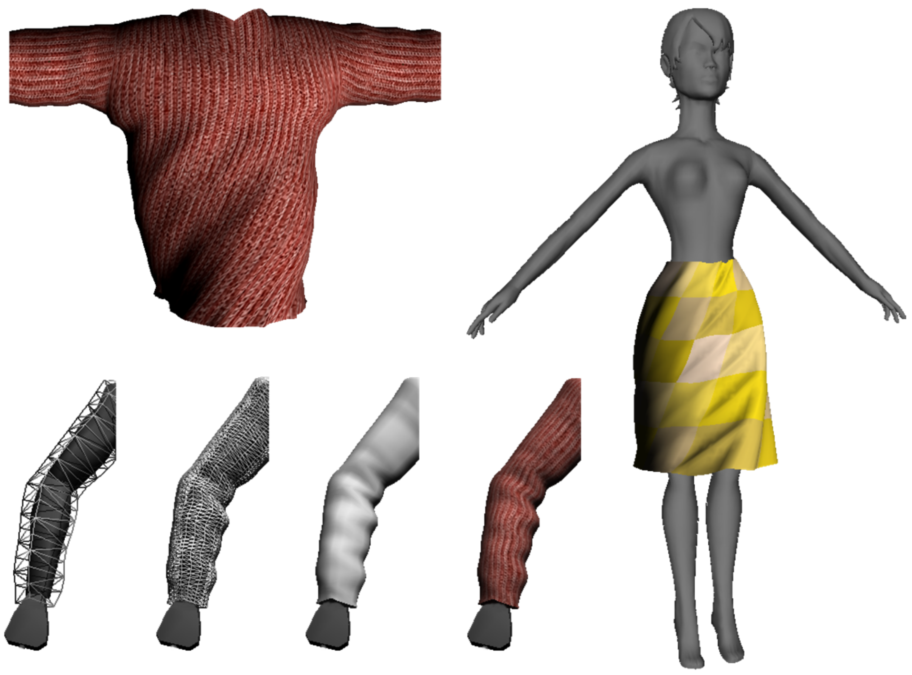


Figure 4.8: Procedural folding of developable garments [DJW+06]. The buckling control mesh is shown around the arm (bottom left).

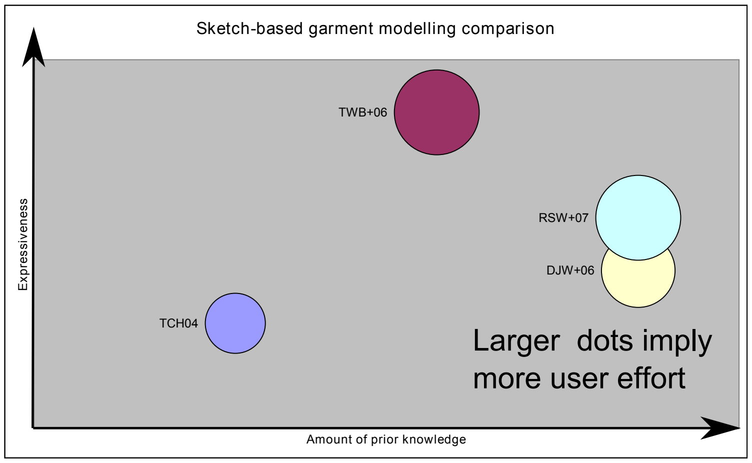


Figure 4.9: Comparing our garment sketching systems [TWB+06, DJW+06, RSW+07] with each other and [TCH04], according to the categorisation discussed in subsection 2.3.1. Amount of prior knowledge incorporated increasing on x-axis. Expressiveness increasing on y-axis. Larger dots indicate greater amounts of user input required.

CHAPTER 5

Sketch-based interface for a physically-based system: Hairstyle design from a sketch

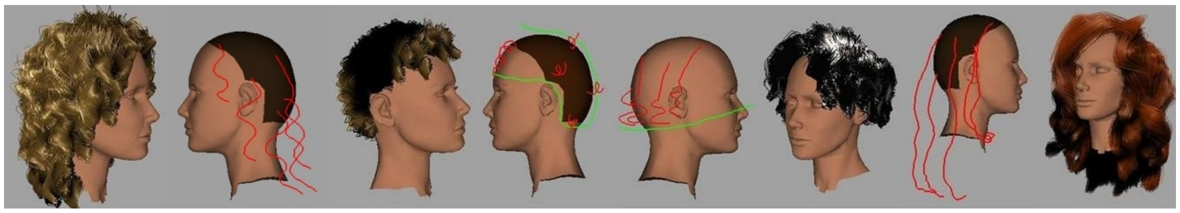


Figure 5.1: Some results from our hair sketching system. In each case the input sketch lines and the resulting hairstyle are shown.

In the previous sections we used our prior knowledge of the object being modelled to simplify the problem and guide the design of the interface. In this section some of the prior knowledge is already expressed concisely for us in the form of a physically based model for a strand of hair. The problem becomes extracting the parameters required to drive this model, and then to generalise to a whole head of hair.

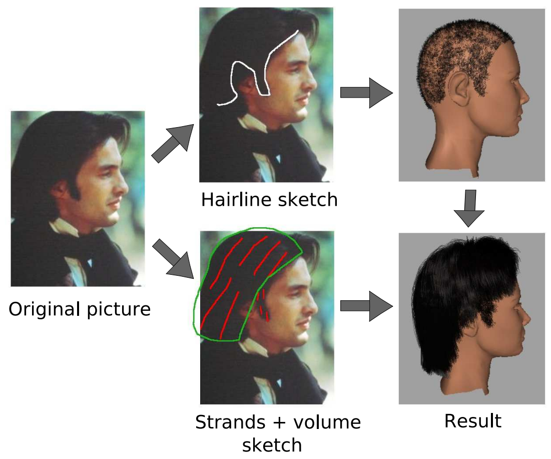


Figure 5.2: An overview of the whole process. The user draws a scalp line, some example hair strands and an optional volume and cut stroke.

5.1 Expressing prior knowledge

The work of [BAQ$^+$05, BAC$^+$06] presents a physically based method for modelling a strand of human hair (called Super-Helices), a method of modelling a full head of hair using this model and a hair-styling methodology based on a simulated hair dressing process (wetting, cutting and drying). The model determines the shape of a hair by incorporating a number of geometrical shape constraints on a series of elastic rods. A simple way to visualise the model is to consider a strand of hair as consisting of a series of helical shapes. The model requires a number of parameters: length, natural curvature, ellipticity, stiffness, mass per unit volume, root position and root orientation. The model can incorporate collisions and the effect of gravity due to being expressed in terms of an energy minimisation problem. This combination of ideas produces convincing 3D virtual hairstyles, but the styling process is time consuming (at least 30 minutes per hairstyle - as each wisp of hair requires many parameters to be set by hand) and requires some hairdressing expertise from the user.

Our aim is to avoid the hair-dressing process using a sketch-based interface. Our problem becomes deriving the parameters needed to drive this physically-based model from a simple sketch. To do this we use the prior knowledge that:

- Hairs clump together in wisps, which can be modelled using a *guide strand*.
- A guide strand can be modelled as a series of helical shapes.
- Neighbouring wisps tend to exhibit similar properties.
- Hairstyles are often symmetrical, and a side view displays most of the variety.

The first observation was used in [BAQ$^+$05]. The idea of an individual hair being modelled as a series of helical shapes leads to the approach of extracting helical parameters from a sketch of an example guide strand from a side view. The observation that neighbouring wisps are similar leads to the use of interpolation between the parameters controlling each guide strand, so that the properties of the wisps can gradually vary across the head, and don't need to be explicitly specified for all wisps. Finally because hairstyles are often symmetrical (when viewed from the front), and because a side view captures most of the detail of the style, we choose to allow the user to sketch from a side view only. This limits the complexity of the interface (we could allow sketching from any viewpoint, but this requires the user to control the camera position and to sketch different projections of a desired individual hair shape) while still allowing the user to draw strokes in the important regions of the head (fringe, side and back).

We now describe the interface and then detail the method and implementation.

5.2 The sketch-based interface

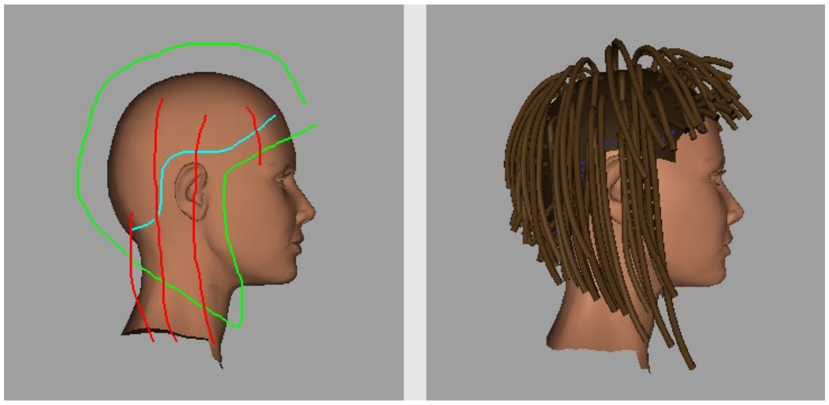


Figure 5.3: The two-view interface presented to the user, only guide strands are rendered during sketching. A high quality hair render is produced as a separate pass.

Our work [WBC07] represents the first sketch-based interface for physically based hair styling. The user is presented with two views of a model head (Figure 5.3). The left view is the sketch input area which consists of a (zoomable) side projection of the head. The right view is the result area within which the camera can be moved freely. An image may be loaded as a background to be used as a guide for oversketching (Figure 5.4). The colour used to render the hair can be selected from the pixels in the photograph. The user progresses through three simple modelling stages:

Hairline

The user first defines the scalp area by drawing one stroke delimiting the scalp extent on one side of the head. The other side is deduced via symmetry. The user can redraw this stroke until happy with the resulting scalp shape, at which point the shape is fixed for the remainder of the modelling process. The scalp is initially covered with short, straight hair.

Example strands

The user may then draw example hair strands starting anywhere within the newly defined scalp area. These examples can be redrawn or deleted. Each time an example is drawn a similar physically modelled strand is immediately created or updated in the corresponding location on the head in the result viewport. At any point the user may calculate and render a full head of hair based on the example strands.

Volume and cut

Finally the user may draw a volume stroke, which is used to both alter the global volume of the style and to cut the hair if desired. The volume stroke mode is selected by a keypress in our implementation - but a simple heuristic based on position and length would easily distinguish it from an example strand stroke, should an automatic interpretation be desired.

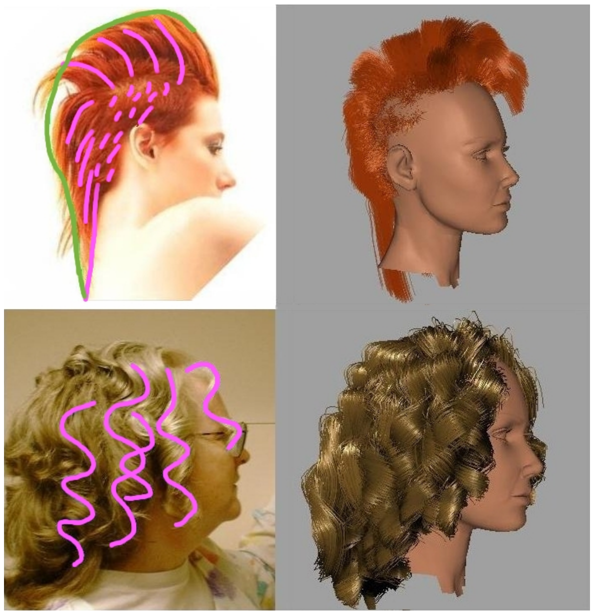


Figure 5.4: Styles resulting from the annotation of photographs (Flaming hair photo ©DH Kong [Kon05]).

5.3 Shaping the hair in 3D

In this section we briefly describe the underlying physical hair model, and then detail the process of extracting from a 2D sketch the parameters required to parameterise this model in order to construct a full head of hair in 3D.

5.3.1 Determining helical parameters

We make the assumption that when a user draws an example hair strand, he is drawing the side view projection of a 3D strand of hair hanging from a head under the influence of gravity. We assume that the user will draw the stroke along a roughly straight axis. We also assume that the user would like a 3D model of a hair strand which has a 2D projection which is *similar*, but not necessarily *exactly* the same as the strand that he drew. Under these assumptions the user is providing *examples* of the sort of hair strands he would like to see in the final model. The lack of exact matching is appropriate as our system will need to generate a large number of plausible 3D strands from only a few sketched ones - no one strand is noticable and should not need to match exactly any example.

The strand parameters (Figure 5.5) required by the physical model of [BAQ$^+$05] are length l, natural curvature κ_0, ellipticity e, stiffness k and mass per unit volume (usually we fix this last parameter to a reasonable value measured from real hair). The ellipticity stands for the shape of the strand's cross-section, which affects the distribution of curls along a strand hanging under gravity: elliptical cross-sections (none zero ellipticity) produce curls evenly distributed along the strand (such as in African hair); circular cross-sections (zero ellipticity) produce strands which tend to be straight at the top and curly at the bottom. The last parameter, stiffness, controls how strongly the hair fiber tends to recover its natural curliness, and thus how much curliness balances

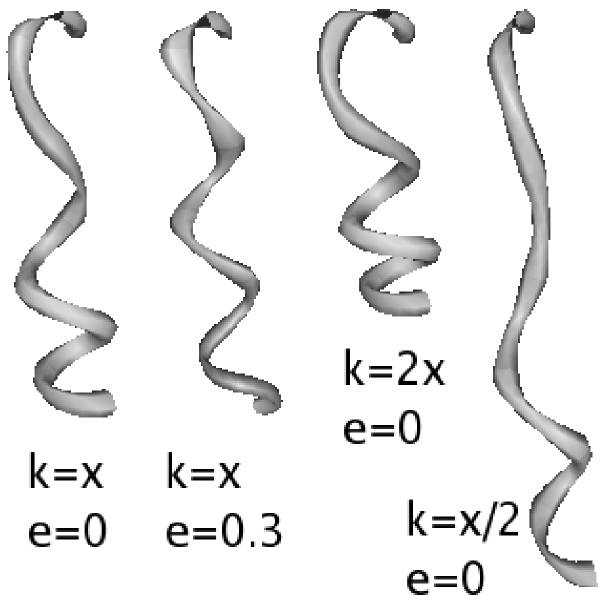


Figure 5.5: The physically-based hair strand is controlled by four main parameters: length, natural curliness, ellipticity and stiffness. Several strands hanging under the effect of gravity are depicted. All have the same length and natural curliness, but different values for the ellipticity e and the stiffness k. $x = 1.6$GPa.

the effect of gravity. Although this parameter value can be measured on natural hair, curls can be made stiffer using styling products, so we need to infer stiffness from the sketch to allow for this effect.

As shown in [BAQ$^+$05] and depicted in Figure 5.5, hair strands under gravity tend to take helical shapes at rest, where the helical parameters vary along the strand. This is the key observation we use to infer parameters from a sketched 2D example of a hair strand. Our basic approach is to divide the stroke into segments and model each segment as the projection of a half helix. We can make measurements on these small segments and use the equations describing a helix to infer the length and curvature required by the strand model.

The equation for a circular helix can be expressed parametrically in Cartesian coordinates as:

$$x = r\cos(t) \qquad y = r\sin(t) \qquad z = ct$$

for $t \in [0, 2\pi)$, where r is the radius of the helix and $2\pi c$ is the vertical distance between consecutive loops.

The arc length of the helix is given by $s = (\sqrt{r^2 + c^2})t$. Thus the arc length of one complete turn of the helix is given by s when $t = 2\pi$, and a half turn is given by $t = \pi$. Therefore the physical length of the half helical segment can be estimated if we measure r and c.

We determine the central axis of the drawn strand using principal component analysis, and then determine the zero-crossing, maxima and minima points along the stroke with respect to this axis (Figure 5.6c). These points delimit the half helical segments shown by axis aligned bounding boxes in the figure, and we measure radius (r) and c from these boxes. As the arc length of a helix is given by $s = (\sqrt{r^2 + c^2})t$, we can determine the 3D length of a segment by letting $t = \pi$ and using the r measured from the segment. Summing the arc length from all segments gives us an estimate for the length of the hair strand in 3D. We estimate the natural curvature κ_0 of the hairstrand as being $1/max(r)$ measured from the set of all segments.

We estimate ellipticity from the distribution of maxima and minima along the primary axis of drawn strand. If they are positioned towards the end of the drawn strand then we assume the curls form at the end of the

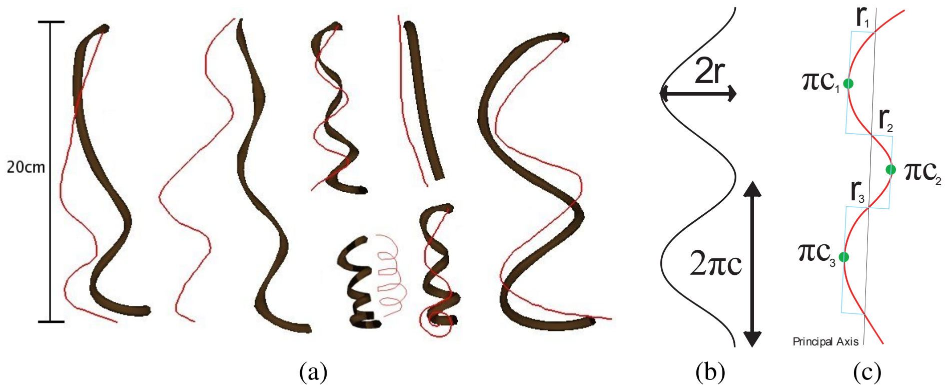


Figure 5.6: **(a)** Single strands inferred from sketches (thin lines): the sketch is interpreted as an example of the desired shape, our system generates similar, physically-based 3D strands (thick lines). **(b)** The orthogonal projection of a circular helix, and the measurements which can be made on it. **(c)** Inferring the length of a stroke using half helical segments.

strand which implies a low ellipticity. If they are more evenly distributed then we assume evenly distributed curls which implies a high ellipticity. Technically this is acheived by looking at the displacement along the primary axis of the median point taken from the set of extrema points. We measure the displacement of this point from the halfway point along the axis and use this value to interpolate between realistic ellipticity values. The range of eccentricies observed for real hair is 0.0 (European) to around 0.2 (African), so we chose to use a linear scale for ellipticity clamped in a similar range [0,0.3] as the displacement of the median inflection points varies from 50% to 70% of the strand extent.

To determine the stiffness and mass of the strand we solve an optimisation problem. The span of a strand of hair is the distance between the root and the tip. Given the previously determined length and curvature and fixing a reasonable mass, we allow the stiffness to vary and try to minimise the difference between the span of the drawn example and the span of the 3D model. This allows the user to set much higher values of stiffness than would be found in natural hair (but could be caused if hair spray was used, for example). Figure 5.6a shows the drawn examples and the resulting 3D models.

Should the stiffness determination routine not find a good span match - then we assume the user has drawn an unrealistically large radius of curvature. Natural hair would be too heavy to hold the curls drawn for any reasonable value of stiffness. In this case we use a similar fitting routine but this time alter the mass per unit volume to make the hair lighter and more able to hold the curl (The curl on the right of Figure 5.6a was found this way).

As the radius of a helix decreases, the projected image of the helix approaches the appearance of a straight line. Also - if the user draws a very straight stroke then many inflection points will be detected due to freehand sketching as the stroke will closely follow the major axis. These inflection points will have very little variation along the secondary axis and thus create narrow segments which imply high curvature. To discriminate between these cases and thus correctly detect straight strokes we apply the following tests.

1. Sum of squared errors (SSE). If the SSE of the stroke samples with respect to the secondary axis is below a small threshold - we assume the line is close to parallel to the axis and so this is a straight line. Set the curvature to zero.

2. Ratio of mean segment length to segment width. We measure the mean segment length and widths. If the ratio of length to width is above a threshold (we use 20) then we assume we have detected few narrow segments which implies a straight hair strand and we set the curvature to zero.

5.3.2 Generalising to a full head of hair

At least 30 wisps are required to generate a realistic looking head of hair, however we don't want the user to have to draw an example strand for every wisp. Instead we extrapolate the information from the few examples the user does provide. To do this, we need two things:

- a scalp area covered with wisp guide strands
- a way of parameterising all the wisp guide strands given just a few examples.

Defining the scalp

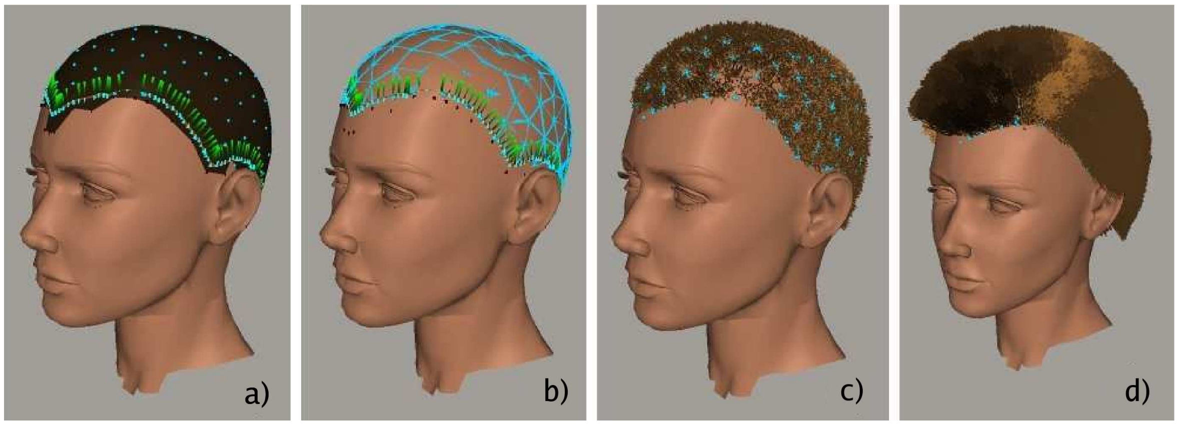


Figure 5.7: Generation of the hair scalp from a sketched hairline. (a) Generation of the scalp triangles from the hairline. (b) Triangular tessellation of the scalp area. (c) Clamping hair on the scalp. (d) Growing hair.

In contrast with [BAQ$^+$05], where the scalp region was pre-defined as a set of triangles belonging to the head mesh, our sketching system provides the user with an easy way to shape the scalp by sketching an arbitrary delimiting curve onto the head mesh. This approach is similar to [MKKI02] except we draw the curve from a fixed side view.

The 3D scalp and guide strand positions are inferred using the following steps (Figure 5.7):

1. The stroke is first projected along the view direction onto the 3D head model. We duplicate the stroke using symmetry to get a closed 3D curve which is the boundary of the scalp.

2. The direction of sketching (which orients the curve) is used to determine the interior of the scalp region.

3. A closed ring of triangles intersecting the curve are identified, and the property of belonging to the scalp is propagated inwards from these triangles over the head model.

4. Similar to [BAQ$^+$05] a particle repulsion method is used to position $N+2$ particles over the scalp, where N is the desired number of guide strands. The difference here is that we respect the sketched scalp curve - each time a particle crosses the scalp contour it is projected back inside the scalp area.

5. Finally the particles are tessellated using delaunay tetrahedralisation, from which we keep the surface triangles to use as the supporting base for each wisp guide strand - which is placed at the triangle centre with a root orientation co-inciding with the triangle normal.

Interpolating wisp guide strand parameters over the scalp

We don't want to constrain the user more than necessary and so the minimum input requirement is just one example strand. We set no maximum constraint. Each example strand is required to start in the valid scalp zone and consist of a continuous stroke. An example strand can be replaced by redrawing the strand starting close to the root, or deleted by double clicking on its root. The user chooses when to update the result view using a keypress.

We use the following scheme to extrapolate from the input: When only one example is provided, we use the same parameters everywhere. If two or more examples are provided we use an interpolation scheme. Consider a vertical plane bisecting the head model and passing through the nose. All wisp root positions are projected onto this plane (each wisp giving a point P_n).

If two examples are provided then a line is formed on this plane between the two projected example root positions. For each wisp the closest point on this line to P_n is found and used to linearly interpolate between the parameters of the two example strands.

If three examples are provided then a delaunay triangulation of the projected example root positions is generated. Barycentric co-ordinates are then used to interpolate between example strand parameters. P_n outside the triangulation are projected to the nearest point on the triangulation.

A more complex interpolation scheme such as projecting onto a sphere and using spherical barycentric co-ordinates could be envisioned. However this simpler scheme provides visually convincing results, as illustrated in Figure 5.10.

5.3.3 Setting the volume, adjusting the cut

The volume of the overall hairstyle is a useful global parameter which should be simple to specify. The global shape of the hair is also often determined by the length of the 'cut'. To allow control over both these aspects with one simple stroke we introduce the idea of a 'volume stroke' (Figure 5.9). With this stroke the user roughly indicates the outer boundary for the silhouette of the hairstyle. The part of the stroke above the scalp is used to determine the volume of the hairstyle, the part of the stroke below the scalp is used to trim hairs which intersect the stroke.

The hair volume is controlled using the multiple layer hulls method of [LK01]. In this method hair strands with roots higher up the head are tested for collisions against larger offsets of the head model (Figure 5.8). The volume is set via a hair volume scaling factor. We determine this factor by calculating the distance from the head model (using a precalculated distance field) of each point of the volume stroke above the lowest point of the scalp model. The maximum of these offsets is used to determine a suitable hair volume scaling factor.

To determine which hairs to cut we project the sections of the volume stroke with normals pointing roughly upwards onto the modelled wisps. Any hairs which intersect are cut back to their length at this intersection point. We also provide separate individual cut stroke functionality if required.

For long hair styles without a short fringe we need a method to ensure hair does not fall in front of the face. To address this we add an optional spherical force field tool in front of the face that the user can see, move and resize. If enabled this spherical force field is accounted for in the body collisions routine. Ideally this spherical tool should be inferred from the sketch.

Such force fields could be used to create a 'comb' tool similar to Malik [Mal05], allowing the user to influence the position of individual guide strands in the resulting model.

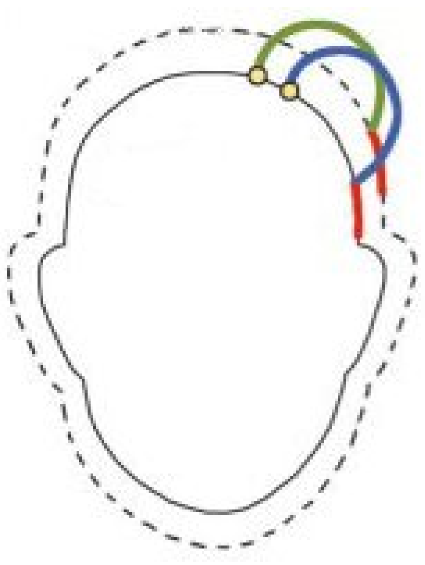


Figure 5.8: Hair volume is simulated using the multiple layer hulls method of [LK01]. Hairs with roots in higher positions collide with larger offsets of the head model (figure adapted from [LK01]).

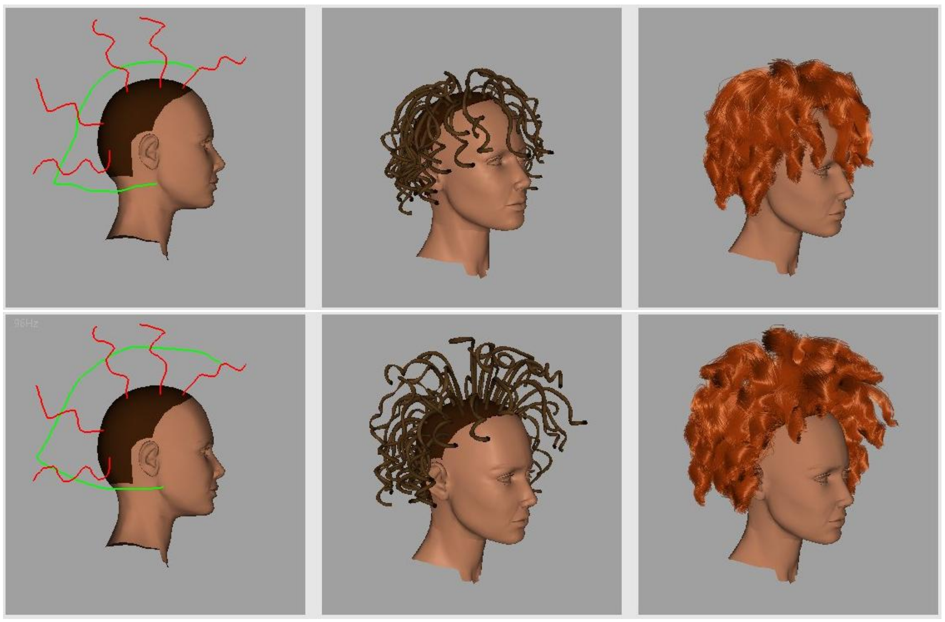


Figure 5.9: The effect of increasing the size of the volume stroke. Note the hairs at the back of the head have been cut where they extend below the stroke.

5.4 Positioning our work

In subsection 2.3.3 we presented a chart which categorised the prior work with respect to the categorisation discussed in subsection 2.3.1. Figure 5.11 is the same chart with a data point added representing our evaluation of our hair sketching system [WBC07]. Our system is the first hair sketching system to incorporate a realistic, physically-based hair model. This model compactly represents a large amount of prior knowledge on the structure of hair, and so we place our system further along the x-axis than the prior work. One could argue that because we don't currently support constraints such as ponytails (as [Mal05, FWTQ07] do) then perhaps we should be closer to the prior work along this axis. However we consider easily parameterising this

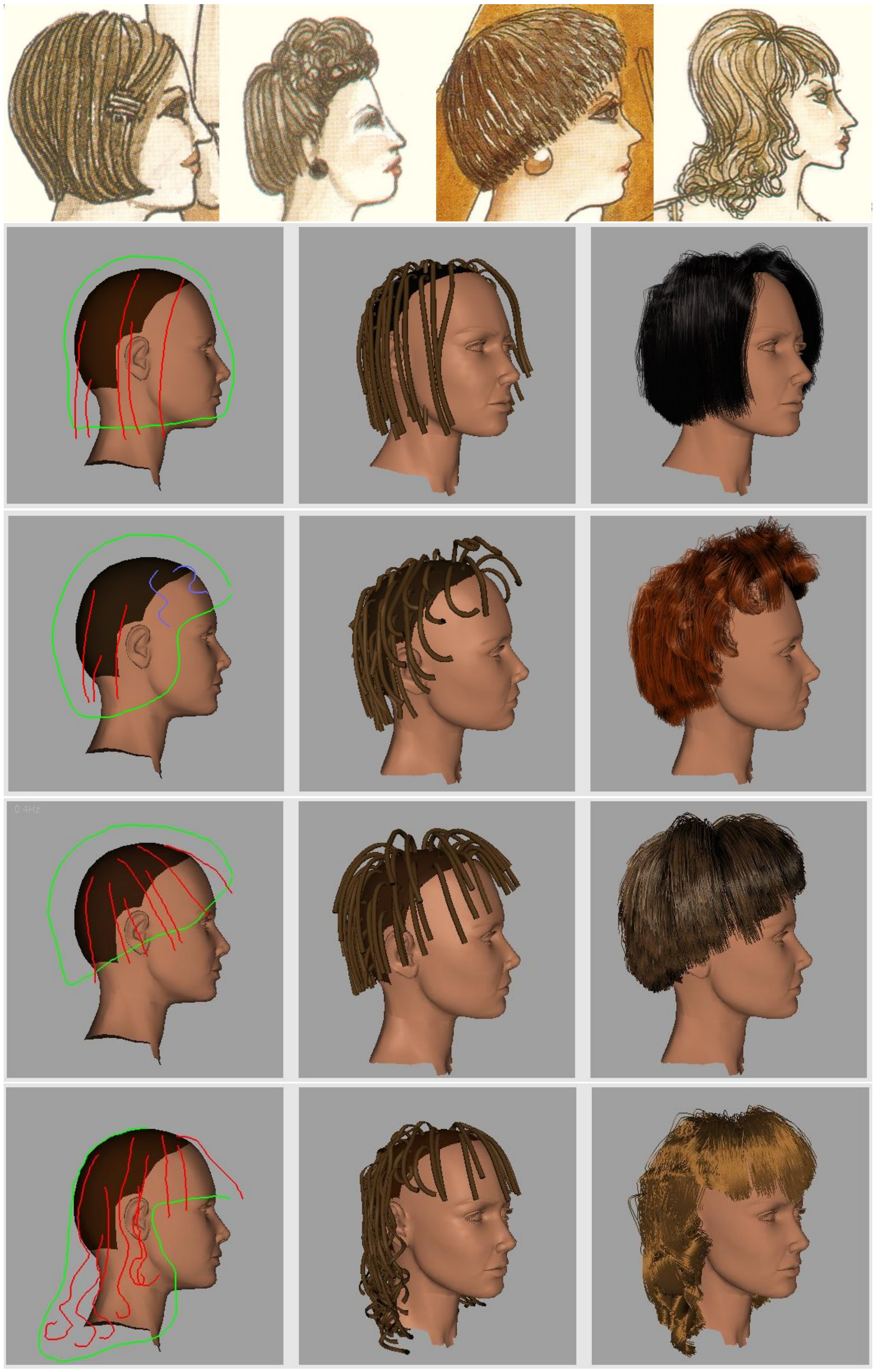


Figure 5.10: Styles from 1927, 1945, 1965 and 1971. Inspired by drawings from [Pea03] **(top)**.

fundamentally flexible represention of the hair is of more importance, and that such constraints to the system can be added as future work. We consider our system to be marginally more expressive than the prior work, as both straight and curly hairs are easily created using the same simple sketching operations. Finally due to the small number of example hairs required and the interpolation scheme, the amount of user input required is minimal - hence the small size of the data point.

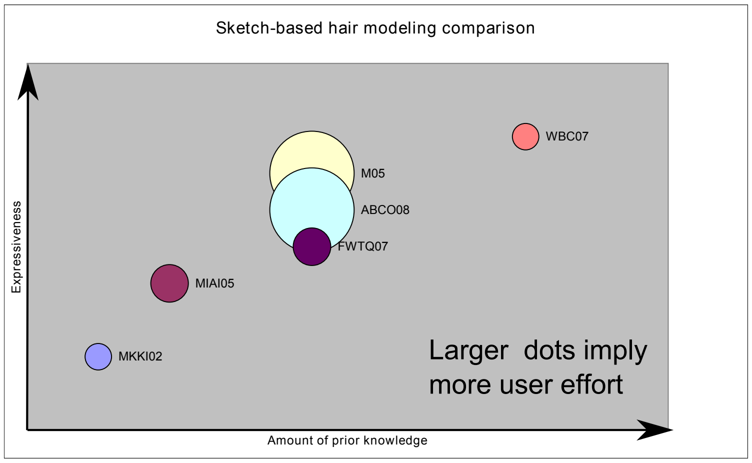


Figure 5.11: Comparing our hair sketching system [WBC07] with others, according to the categorisation discussed in subsection 2.3.1. Amount of prior knowledge incorporated increasing on x-axis. Expressiveness increasing on y-axis. Larger dots indicate greater amounts of user input required.

Summary of Part II

Modelling realistic virtual characters is a challenge, and standard interfaces for designing 3D clothes and hair can be improved. The systems we presented enable the generation of realistic virtual garments and hairstyles fitting a given character in a just few minutes, from a single intuitive sketch, similar to those used in fashion or hairstyle design. This is an important contribution to Computer Graphics. The systems we presented for garments and hair both use annotation of views of a mannequin model. The user quickly sketches the silhouette of garment over a front or back view of the mannequin. Details such as folds can be sketched by the user or generated procedurally using prior knowledge. The system results in both the 3D shape of the garment and the 2D patterns that can be used to sew it together, or be input to a physically-based system for subsequent animation. In the case of hair modelling the user sketches a scalp contour and some example hair strands (varying from straight to curly), from which the parameters of the physically-based hair model are inferred; he may simply add a volume stroke to further specify the hair cut and volume. The resulting 3D head of hair is then ready to be animated.

Amount of prior knowledge The systems described in this part illustrate the fact that sketch-based modelling can effectively be used in very complex cases, granted that the right amount of prior knowledge is incorporated (similar to a human recognizing a garment or hairstyle from a sketch and inferring the full 3D shape). All these systems differ in the amount of prior knowledge they use. In the examples we discussed, several levels of knowledge were incorporated, from rules of thumb, to mathematical properties of surfaces (the piecewise developability of garments) and physical properties (expressed through a static strand model for hair). The associated sketching systems range between two extremes: starting from the way people would sketch the element in real life, and trying to incorporate just the necessary amount of knowledge to adequately infer 3D; or instead designing an intuitive interface for a standard generic, procedural model. Note that the latter requires some kind of inverse engineering to compute the parameters that indirectly control the shape of the model. In the system we presented for hair the sketch is seen as a rough example, and is not required to exactly match the results.

In all the methods presented, sketching on a known 3D surface (the body or head) was a great help. Therefore the method we presented can be seen as an annotation of a 3D model rather than a pure sketch-based system. The underlying 3D model provides information about the structure of the shape to be constructed, and helps with the relative positioning of elements. However often there is no definite supporting structure available before sketching starts. We discuss such cases in the next part.

Résumé en Français

La modélisation réaliste de personnages virtuels est difficile, et les interfaces standards pour la conception 3D de vêtements et de cheveux peuvent être améliorées. Les systèmes que nous avons présentés permettent la production de vêtements virtuels réalistes et de coiffures appliquées sur un personnage donné en quelques minutes, intuitivement à partir d'un seul croquis, semblable à ceux utilisés dans le design de mode ou de coiffure. Ceci est une contribution importante au domaine de l'informatique graphique. Les systèmes que nous avons présentés pour les vêtements et les cheveux utilisent tous les deux l'annotation de vues 2D d'un mannequin. L'utilisateur esquisse rapidement la silhouette du vêtement sur la vue de face ou de dos du mannequin. Les détails tels que les plis peuvent être dessinés par l'utilisateur ou générés en utilisant un modèle procédural issu de connaissances préalables. Le système produit à la fois la forme 3D du vêtement et les patrons 2D qui peuvent être cousus ensemble ou contribuer à définir un système physique pour l'animation. Dans le cas de la modélisation de cheveux, l'utilisateur dessine le contour du cuir chevelu et quelques mèches de cheveux d'exemple (droites ou frisés) à partir desquelles les paramètres du modèle physique sont déduits. Il peut ensuite ajouter des traits de volume pour préciser d'avantage la coupe de cheveux et son volume. La chevelure est alors prête à être animée.

Quantité de connaissances préalables Les systèmes décrits dans cette partie illustrent le fait que la modélisation par sketching peut effectivement être utilisée dans des cas très complexes, à condition qu'une quantité suffisante de connaissances préalables soient intégrée au système (semblable à celle nécessaire à un humain pour reconnaître un vêtement ou une coiffure et d'en déduire la forme en 3D à partir d'un croquis 2D). Tous ces systèmes sont très différents en ce qui concerne la quantité de connaissances préalables qu'ils utilisent. Dans les exemples présentés, plusieurs niveaux de connaissances ont été intégrés: les paramètres empiriques, les propriétés mathématiques de surfaces (la développabilité par morceaux des vêtements) et les propriétés physiques (exprimées par le biais d'un modèle statique de mèche pour les cheveux). Les systèmes de sketching varient entre deux extrêmes: partant de la façon dont les gens dessineraient naturellement les objets en 2D et essayer d'y intégrer la quantité minimale de connaissances nécessaires pour en inférer un modèle 3D; ou à l'autre extrême, la conception d'une interface intuitive pour définir un modèle procédural générique. Notez que ce dernier requiert une certaine forme de reverse engineering pour calculer les paramètres qui contrôlent indirectement la forme du modèle. Dans le système que nous avons présenté pour les cheveux, le croquis est considérée comme un exemple auquel le résultat final n'est pas tenu de correspondre exactement.

Dans toutes les méthodes présentées, le support d'une surface 3D connues (le corps ou la tête) a été d'une grande aide. Par conséquent, ces méthodes peuvent être considérées comme des systèmes d'annotation de modèle 3D plutôt qu'un système de sketching classique. Les modèles 3D fournissent des informations sur la structure de la forme qui doit être reconstruite, et contribue à positionner les éléments relativement les uns aux autres. Mais souvent, il n'existe pas de structure d'appui disponible avant que le sketching commence. Nous discutons de ces cas dans la partie suivante.

Part III

Structure from silhouettes: Applications to clouds, trees and terrain

Introduction

In the previous part we considered the case of sketching as a form of annotation to an already existing 3D model - which helped us infer 3D positions for the things represented by the sketch. In this part we consider the case where there is no pre-existing structure to annotate. For some phenomena we have prior knowledge of the nature of the shape being modelled, but the shape of specific examples can vary widely and must be inferred from the sketch. This is the case for clouds and trees. To enable the modelling of such shapes we could require the user to first sketch explicitly a supporting structure, but of course this structure would have to be in 3D and would be complex and time consuming to create. Instead, in this part we claim that in many cases, the supporting structure can be created rapidly and incrementally by inferring it from sketched silhouettes using some form of skeletonisation technique (subsection 2.1.1) combined with some prior knowledge on the way the underlying shape extends into 3D.

In this part we present two sketch-based systems for modelling complex natural phenomena that demonstrate the utility of such an approach. In the case of both clouds[1] and trees[2] we use the sketch to parameterise underlying procedural models. In each case we ask the user to incrementally supply silhouette information and we infer structure given that silhouette and the prior knowledge we have on the object being modelled. The inferred structure becomes the supporting surface for further operations. In both approaches we try to address the multi-scale nature of the phenomena. For clouds, the user supplies the coarse scale shape of the cloud and fine-scale detail is added using a procedural method. For trees the user may supply detail at any scale within the tree, from the main branches down to the leaves. In contrast to the previous chapter where the user viewpoint was deliberately fixed, in this part the user may edit from arbitrary viewpoints if desired.

We end the part with a brief chapter outlining some work in progress using a similar approach for modelling terrain from sketched silhouettes.

Résumé en Français

Dans la partie précédente, nous avons étudié le cas du sketching comme une forme d'annotation d'un modèle 3D déjà existant - ce qui nous a permis de déduire les positions 3D des objets dessinés.

Dans cette partie, nous considérons le cas où il n'y a pas de structure pré-existante à annoter. Pour certains phénomènes, nous avons une connaissance préalable de la nature de la forme modélisée, mais la forme des objets spécifiques peut varier considérablement et doit être déduite du croquis. C'est le cas des nuages et des arbres. Pour permettre la modélisation de ces formes, on pourrait demander à l'utilisateur de faire un premier croquis explicite d'une structure d'appui, mais bien sûr, cette structure devrait être en 3D, serait complexe, et demanderait beaucoup de temps pour être réalisée. Au lieu de cela, nous montrons dans cette partie que dans de nombreux cas, cette structure d'appui peut être créée rapidement et progressivement par déductions à partir de dessins de silhouettes, en utilisant une technique de ”squelettisation” (subsection 2.1.1) combinée à une certaine connaissance sur la façon dont la forme sous-jacente se prolonge en 3D.

Dans cette partie, nous présentons deux systèmes à base de sketching pour la modélisation de phénomènes naturels complexes, qui démontrent l'utilité d'une telle approche. Dans le cas de clouds et trees nous utilisons le sketching pour paramétrer les modèles procéduraux sous-jacents. Dans chacun de ces cas, nous demandons à l'utilisateur de détailler de plus en plus la silhouette et nous en déduisons la structure en fonction de la

[1]This collaborative work of which I was first author was published at the Sketch-Based Interfaces and Modelling workshop [WBC08].

[2]This collaborative work [WBCG08] has been submitted to the Eurographics 2009 conference. The authors are myself, Fred Boudon, Marie-Paule Cani and Christophe Godin.

silhouette et des connaissances que nous avons sur l'objet à modéliser. La structure déduite devient la surface d'appui pour les opérations suivantes. Dans les deux approches, nous essayons d'intégrer la nature multi-échelle des phénomènes. Pour les nuages, l'utilisateur fournit la forme de nuages à grande échelle, et le détail à petite échelle est ajouté en utilisant une méthode procédurale. Pour les arbres, l'utilisateur peut fournir le détail à toutes les échelles à l'intérieur de l'arbre, des branches principales aux feuilles. Contrairement au chapitre précédent où le point de vue été délibérément fixé, dans cette partie l'utilisateur peut dessiner à partir de n'importe quel point de vue s'il le souhaite.

Nous concluons cette partie avec un bref chapitre sur certains travaux en cours qui utilisent une approche similaire pour la modélisation de terrain à partir de sketching de silhouettes.

CHAPTER 6

Rapid sketch-based modelling of clouds

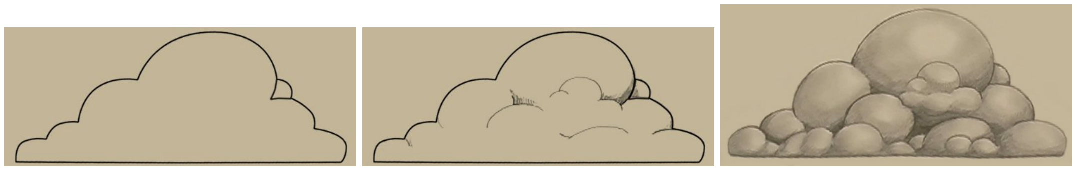


Figure 6.1: An artist [Coa08] approaches cloud drawing in a global to local fashion, using silhouettes.

6.1 Introduction and previous work

Modelling virtual clouds is a difficult process. Cloud formation follows the laws of fluid mechanics and thermodynamics. It is a chaotic process which depends strongly on initial conditions. Solutions based on the direct simulation of these laws [HBSL03] or even simpler cellular automata [DKY$^+$00] are difficult to control and thus hard for computer artists to influence in order to attain a desired shape. Recent methods have been proposed to control fluid simulations [TMPS03, MTPS04] but they still require the user to supply keyframes. Moreover, these methods only simulate fluid dynamics but not thermodynamics which play an important role in cloud dynamics.

An alternative is to use procedural methods. These methods allow users to control the general shape of the clouds through large-scale tools. Stratiform clouds are easily modelled by horizontal layers of varying thickness [Gar85, BNL06], while cumuliform clouds are generally modelled as sets of ellipsoids [Gar85, Ney97, TB02, SSEH03, BN04]. Because of the puffy nature of convective clouds this ellipsoid set representation is well suited and has been used at several scale levels [Ney97, BN04] to mimic the fractal aspect of clouds.

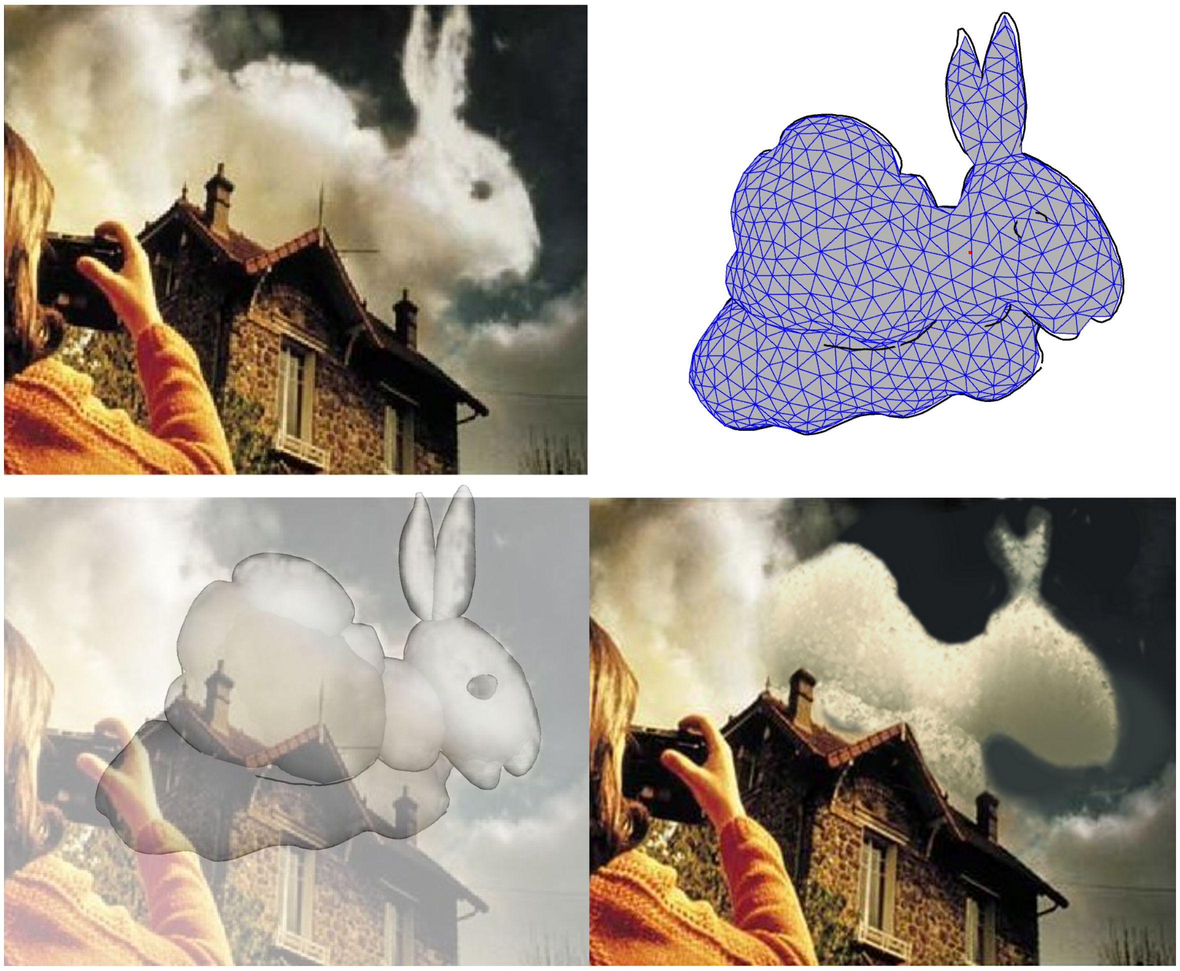


Figure 6.2: Top left is an image from the film Amélie, we use it as a guide to create the mesh at the top right and the rendered result at the bottom right. The whole process took 3 minutes.

To add details to these large-scale models (or make up for the coarse resolution of simulations) it is common to add high-frequency procedural noise [Gar85, SSEH03]. The resulting shape is then passed to a renderer, either in the form of volume densities [SSEH03] or a mesh [Gar85, TB02, BNM$^+$08] (for a survey of cloud rendering techniques see [SSEH03, BNM$^+$08]). Implicit surfaces are sometimes used to convert the set of ellipsoids into a manifold mesh [SSEH03, BN04].

However, controlling the overall shape which emerges from these procedural methods is not trivial. Either not enough control is given to the user or he/she has to go through a long session of 3D placement of ellipsoids. Easy and intuitive control is important in applications such as video games, 3D feature films and special effects (see Figure 6.2) where the artist may wish to achieve a very specific cloud shape.

Our aim is to allow rapid modelling of cloud shapes via a simple to use sketch-based interface. We target fluffy, cumulous type clouds in particular since they carry the most visibly appealing features (Figure 6.4). Our method allows modelling of cloud like meshes in a few seconds using a sketch-based metaphor. These meshes can then be rendered by a real-time technique such as [BNM$^+$08]. Such meshes could also be used directly as animation keyframes in recent fluid control methods [MTPS04]. Using meshes allows us to simplify the modelling process and later control the rendering through further sketch-based annotation.

In contrast to the modelling systems presented in Part II, for clouds we have no pre-existing structure to

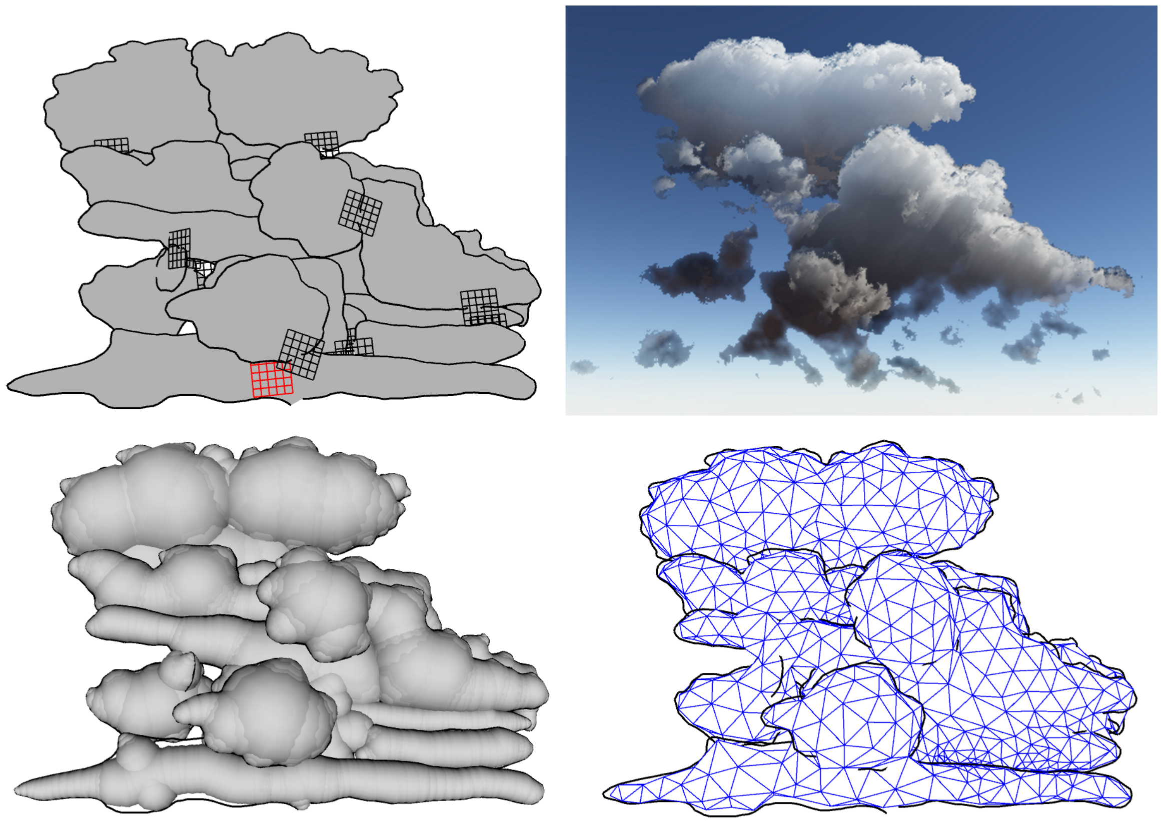


Figure 6.3: A natural cloud formation modelled in three minutes and result rendered using the method of [BNM$^+$08].

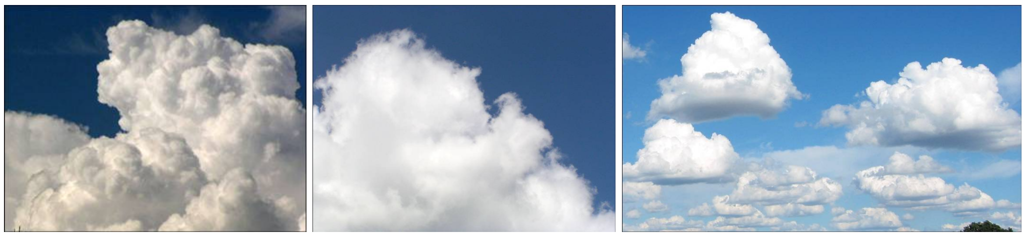


Figure 6.4: Some natural features of cumulous clouds. Lobes, wisps and the flat bottom.

annotate. Instead we define the structure incrementally, by inferring it from sketched silhouettes (which can be drawn from arbitrary viewpoints). Each new silhouette is defined in relation to previous ones and coarse, large scale structure emerges. Cumulous clouds consist of many fine details at smaller scales which would be time consuming to draw explicitly. For these details a procedural noise approach is more appropriate. For these reasons we use the real-time cloud rendering system of [BNM$^+$08], which adds procedural detail to a coarse cloud mesh, as an underlying model. Our solution is a first example of inferring large scale structure from silhouettes and using a procedural method to provide fine details.

Our system exploits prior knowledge of cloud shape [Gar85, Ney97], telling us that cumulous clouds can be approximated by a union of spherical primitives. Thus we propose techniques to infer spherical primitives from sketched 2D outlines, and to automatically generate 3D surface detail while retaining the 2D outline. This makes our method easier to understand and faster to learn than a general sketching system (such as those discussed in section 2.2) or a traditional modelling package.

To our knowledge there has been no previous work combining sketch-based modelling and rendering of

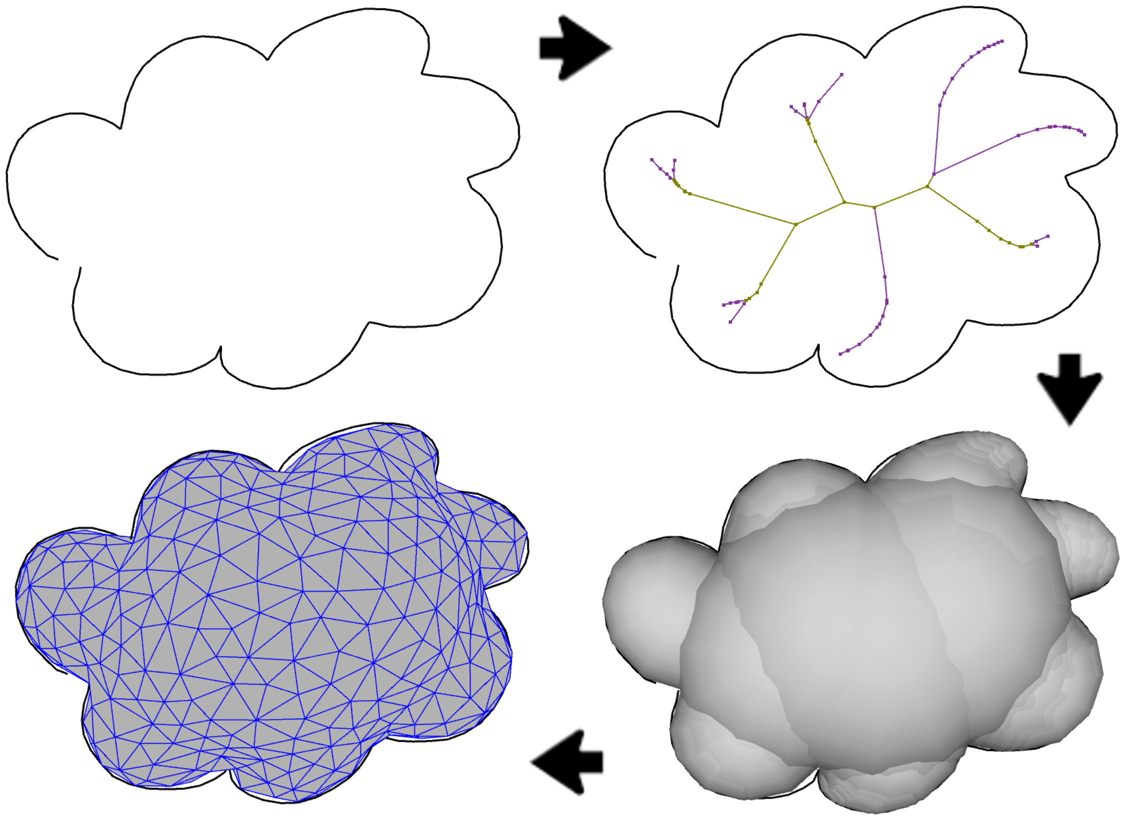


Figure 6.5: The user draws a silhouette. We infer a skeleton, spheres along that skeleton and finally create a mesh.

clouds, and so we don't provide a comparison chart as with other chapters. A comprehensive (but not sketch-based) cloud modelling system was presented in [SSEH03]. It used a traditional modelling environment consisting of resizable implicit ellipsoid primitives and a hierarchy of user modifiable parameters. Our approach is not as comprehensive as this and does not address cloud animation. It is rather a complementary interface which could be integrated into such a system to speed up the shape modelling.

6.2 User scenario

The user draws a stroke representing the whole or a section of a cloud silhouette (Figure 6.8). The user may alter or extend his stroke (using the technique of [Ale05]) and when he's happy with it he presses 'space' to interpret the stroke. A cloud volume (a union of spheres) matching the silhouette is then inferred. Many silhouettes can be drawn, with overlapping volumes being attached to the underlying ones. The camera can be positioned anywhere and repositioned at will.

Each interpreted stroke is stored on a separate drawing plane in the scene (Figure 6.6). The spheres which represent the volume of the cloud are generated along the skeleton (chordal axis transform) of the closed stroke (See subsection 6.3.1).

Each drawing plane can be selected and then moved, rotated, duplicated, inflated/deflated or deleted. The user can begin a new section of cloud on a new drawing plane by drawing a new outline. If the new outline overlaps existing spheres, then the new drawing plane is defined parallel to the viewplane and passes through

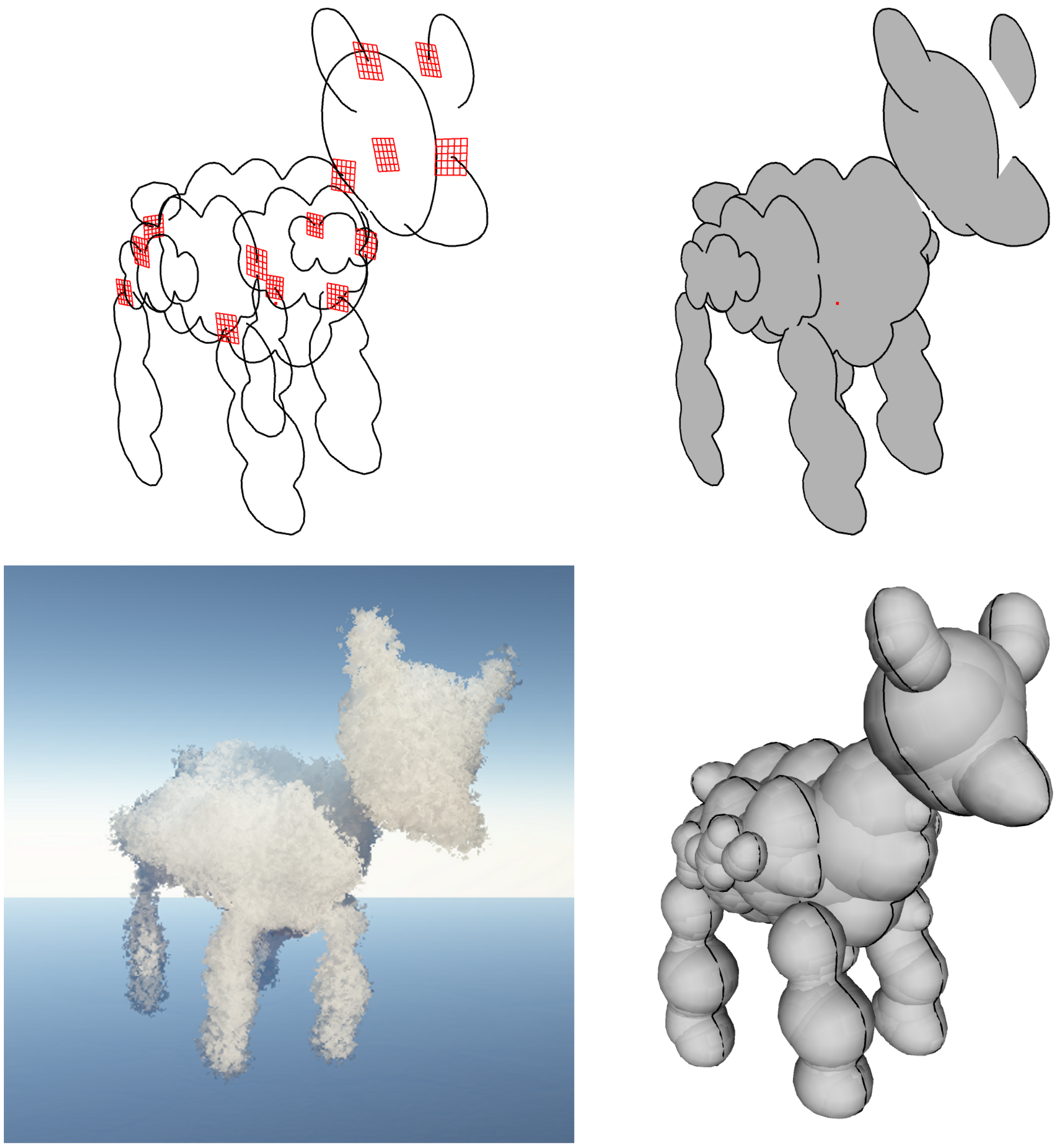


Figure 6.6: This sheep like cloud took two minutes to model. Each red grid in the top left image represents a drawing plane containing a silhouette.

the initial point of the silhouette projected onto the existing surface.

The user can thus quickly build up the layers within a cumulous cloud (without manually defining or moving drawing planes) by simply sketching each layer from the back to the front. To aid this process an imported image may be used as a guide (Figure 6.2).

As the spheres generated in a given drawing plane follow a planar skeleton the resulting cloud section could be too flat. The user may of course add more volume through further sketching (Figure 6.7), but we also propose a novel method for generating a plausible 3D cloud from only the 2D silhouette information (Figure 6.10, see subsection 6.3.2).

Once the user is happy with the cloud volume a surface mesh is produced by interpreting the spheres as point based implicit primitives and blending them (see subsection 6.3.3). The resulting surface is used for

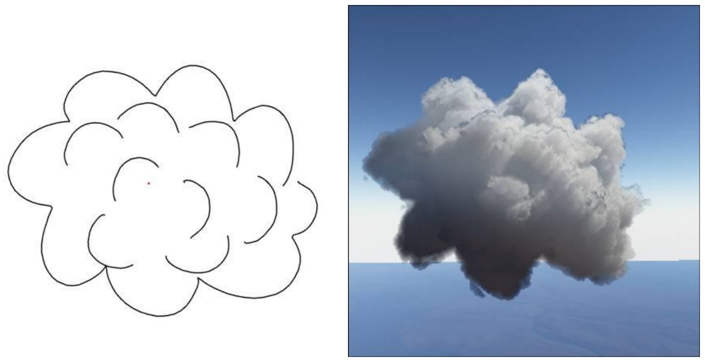


Figure 6.7: One way to form 3D clouds from 2D sketches is to quickly build them from back to front by using layers of strokes. Each subsequent stroke depth is inferred by projecting it onto the current cloud mesh.

rendering the cloud.

A common feature of cumulous clouds is the flat bottom. The user may quickly add this feature by drawing a cut stroke which will ensure no part of the cloud mesh extends beyond the stroke (Figure 6.13, see subsection 6.3.3).

6.3 Cloud shapes from sketching

6.3.1 Skeleton from 2D silhouette

Although we do not need to use complex implicit surfaces generated by skeletal structures, computing a geometric skeleton from the silhouette will help in reconstructing the silhouette using spherical primitives. To generate a shape skeleton we first close the outline stroke by repeating the initial point. The skeleton is then generated from the outline using a chordal axis transform (CAT) derived from a constrained delaunay triangulation using the technique of [Pra97] (described in subsection 2.1.1). We use an area metric to prune insignificant branches in skeleton (if triangles associated with terminal branches contribute less than 2% of the total outline polygon area we prune them). While pruning we maintain a mapping of outline edge segments to skeleton segments (this technique is explained in subsection 7.3.1 as we also use it for modelling trees). This mapping is required for later silhouette definition (see subsection 6.3.2). The skeleton is a network of polylines, and for

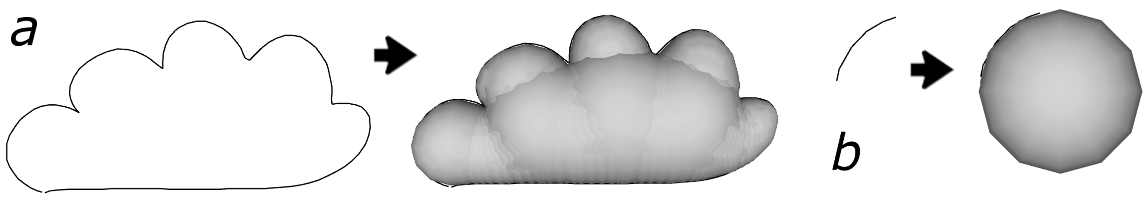


Figure 6.8: (a) closed stroke and interpretation. (b) open stroke and interpretation.

each vertex in the polylines the radius of the maximal ball (touching the outline) is stored. These points and the associated radii are used to define the spheres in the cloud volume.

To reduce the skeleton resolution we can simplify the outline stroke before generating the skeleton. We use the technique of Horst and Beichl[HB97] (described in section 2.1) and offer the user a single parameter to control the degree of simplification (using a slider widget).

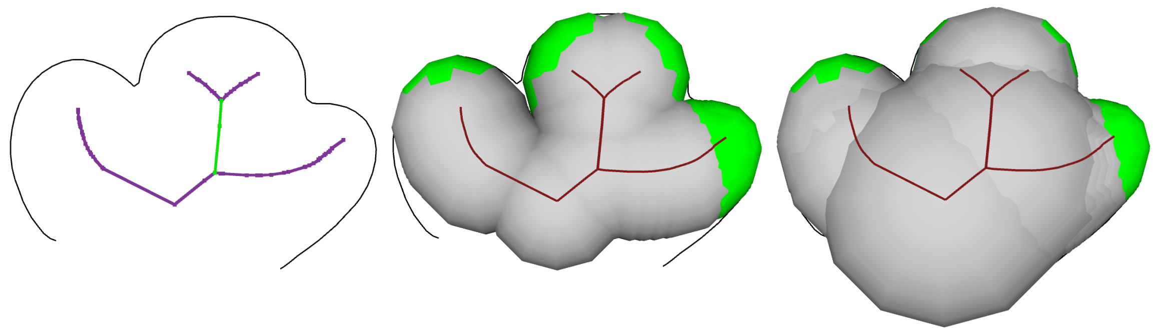


Figure 6.9: **(left)** Skeleton segmentation. Terminal branches in purple and body branches in green. **(middle)** Sphere inflation minimum and **right)** maximum. Spheres with radii below r_f are shown in green (see subsection 6.3.1).

We segment the skeleton into terminal branches (sequences of skeletal segments which are connected at one end only) and body branches (connected at both ends). We observe that terminal branches appear to point towards curvature maxima in the outline, and thus can be thought of as defining shape features in the cloud outline.

When drawing large outlines the central (body) spheres of the skeleton are necessarily large as they must touch the outline. However the user may wish a flatter shape. We allow the user the option of shrinking the largest spheres, without affecting the smaller spheres (which contribute to the finer details of the cloud). To provide this we offer an inflation/deflation slider control to the user, described below.

We define the minimal feature radius r_f. This radius will represent the smallest size any sphere can be shrunk to, and should be representative of the smaller spheres at the end of the skeletal branches in order to preserve fidelity to the drawn outline in these areas. We determine r_f as the maximum radius from the set of the spheres contributing to the last 50% of all terminal branch segments in the skeleton (Figure 6.9). 50% was chosen through experiment with typical outlines as retaining the spheres contributing to the smaller outline features while excluding the larger body spheres. The user may interactively specify an interpolation parameter $t \in [0,1]$ using a slider. We adjust the size of all spheres with radii above r_f using t, the minimum radius is limited to r_f and the maximum radius is the radius of the maximal ball for that point.

One disadvantage of this approach is that the fidelity to the drawn outline is reduced for spheres above the r_f radius. This could be addressed in future work by allowing ellipsoid primitives which could then be shrunk along the direction perpendicular to the drawing plane. Also defining a minimal feature radius for each skeletal branch would improve the result for outlines consisting of a wide variety of feature sizes. Another disadvantage is that the body of the cloud could get disconnected into several disjoint components if the skeleton isn't sampled densely enough. Automatic resampling could be implemented to avoid this.

6.3.2 Extending a flat cloud to a 3D structure

If a cloud is constructed from just one outline, it will be unrealistically flat (with a skeleton lying in its original plane) and unlike a real cloud. We present a technique which automatically generates a plausible 3D structure in case the user doesn't wish to build it up manually in layers.

To do this we consider the terminal branches of the skeleton as defining volume features that should be duplicated over the cloud shape. We segment the set of body branches in the skeleton into 'trunks' (wherever three body branches meet) and duplicate terminal ('feature') branches and reposition and rotate them around the trunks in the manner of branches around a tree, scaled to the surface of revolution of each trunk silhouette

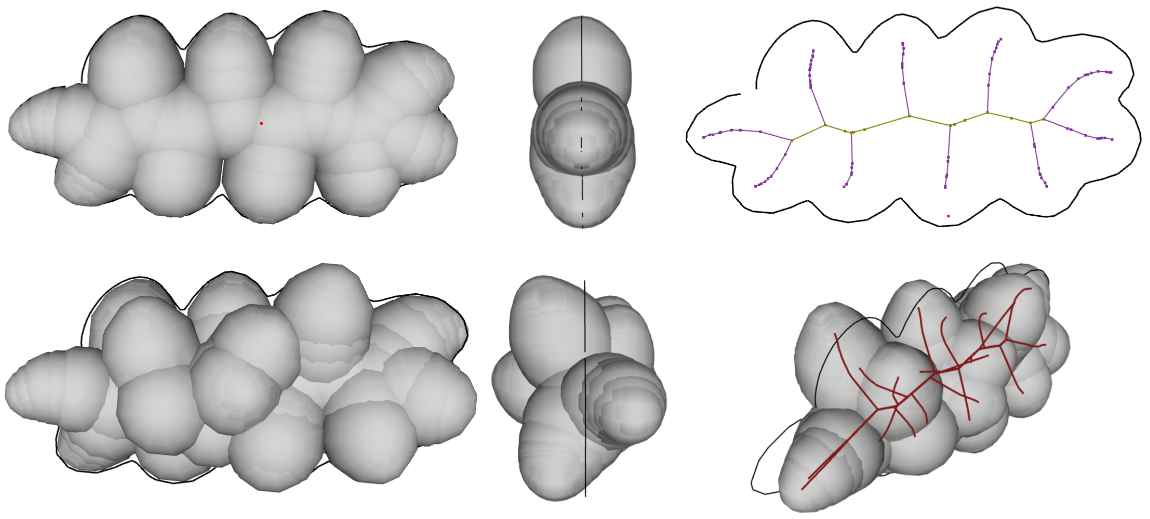


Figure 6.10: The 2D skeleton **(top row)** is automatically placed in 3D **(bottom row)**, generating new, similar detail while preserving the existing silhouette (See subsection 6.3.2).

(constructed using the edge segment to skeleton segment mapping constructed in subsection 6.3.1). The existing branches are allowed a small 'jitter' rotation from their initial plane, but their projection to the viewplane is maintained and the new branches are placed so as to try and maximise the distance between branches on the surface of the 'trunk'. This ensures that the generated 3D cloud is entirely within the sketched silhouette. This approach is similar to that used for constructing trees from sketches (explained in more detail in section 7.5).

6.3.3 Mesh generation

Once the user has finished creating and editing his drawing planes (thus having modelled a union of spheres) a surface representation is generated by considering each sphere as a point based implicit primitive. To blend these primitives we represent them using the blending function of [Wyv92] (Figure 6.11) which is cubic in r^2. r is the distance from the centre of the sphere and R is the radius of the sphere.

$$F(r) = -\frac{4}{9}\frac{r^6}{R^6} + \frac{17}{9}\frac{r^4}{R^4} - \frac{22}{9}\frac{r^2}{R^2} + 1,$$

Figure 6.11: Implicit blending function[Wyv92].

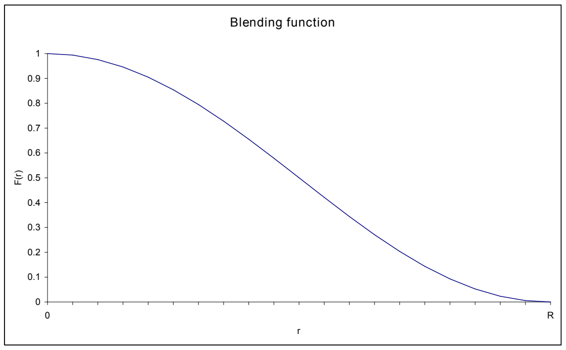


Figure 6.12: A plot of the blending function.

We convert this implicit surface to a manifold mesh using the CGAL library [CGA]. This flexible surface meshing approach [BO05] is based on restricted delaunay triangulation of the zero level set of the implicit surface. It offers a number of mesh quality parameters the user may alter. For a typical cloud this takes around 10 seconds (Table 6.1). Note that faster polygonisation methods exist (e.g. [SWG05]) but as we need only a

Table 6.1: Result timings. The mesh parameters are a distance bound and radius bound, see [CGA].

Name	modelling (min)	Mesh gen. (sec)	Spheres	Facets	Param.
Sheep	2	18	896	2838	0.1, 0.3
Rabbit	2	14	486	1682	0.05, 0.15
Ship	1	2	398	852	0.1, 0.3
Figure 6.3	3	26	1866	2248	0.1, 0.3

simple mesh with no special surface properties using a library was appropriate.

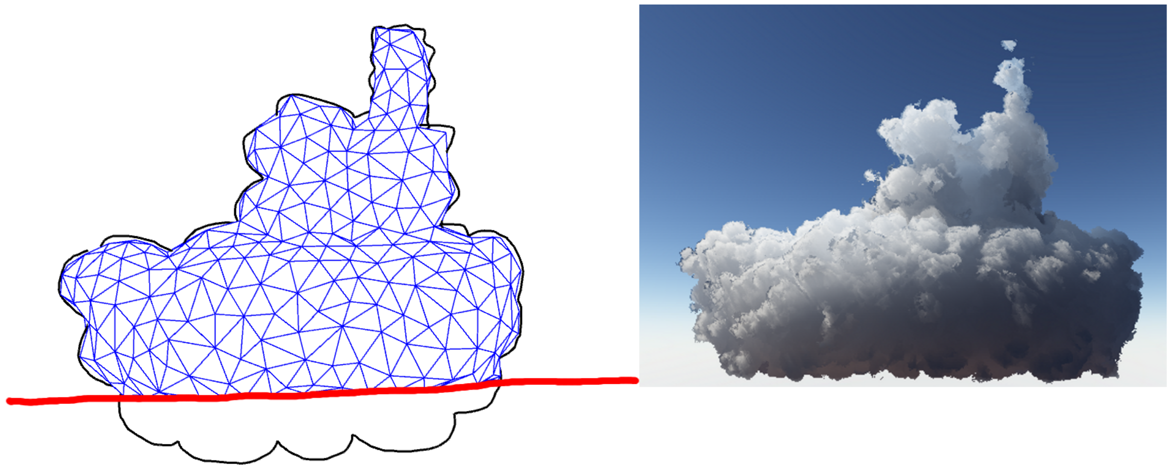


Figure 6.13: **(left)** The mesh can be clipped by cut strokes which suppress the implicit surface below the line (see subsection 6.3.3). **(right)** Our rendering of this cloud using the method of [BNM+08].

Finally we observe that most cumulous clouds have a flat bottom. We enforce this by allowing the user to specify a large 'clipping' sphere using a stroke. This sphere acts as a large negative primitive in the implicit function which ensures the implicit surface generated is flat along the bottom.

6.4 Discussion and future work

We have presented a system for rapid modelling of clouds using a sketch-based interface. Quality meshes can be created and rendered within a minute. We proposed a method for automatically generating 3D surface structure from 2D silhouettes. Our method is coupled with a procedural model to set fine details such as wisps.

Our test system was an Intel Xeon Quad Core 2.8GHz, 2GB RAM, with NVidia Quadro FX 1400. For performance times for models created by the authors see Table 6.1.

The modelling times are typically less than two minutes even for fairly complex looking clouds. The correspondance between the final rendered clouds and the input strokes is good, as can be seen from the resulting images.

Our prototype implementation isn't optimized, and mesh generation times could be improved by removing redundant spheres from the skeleton definition. We already offer outline simplification as a method for reducing the number of spheres required, but additionally we could pre-cull many spheres which don't contribute significantly to the surface definition. A pre-calculated spatial query method (such as a 3D voxel grid) could be used to accelerate density queries used to form the blended implicit surface.

We invited a computer artist with experience in modelling clouds to use the system. Previously she had modelled 3D clouds using traditional modelling packages by manually creating, deforming and positioning spheres within the scene. The work was tedious and required hours of effort. With our system she created in a few minutes clouds that had previously taken her a few hours.

Although in theory clouds could be modelled using a standard sketch-based modelling system for free form shapes, our method saves the user time by relying on specific prior knowledge. The use of spherical primitives and the automatic extension to a 3D structure makes the shape cumulous cloud-like. The flat bottom and the ability to shrink internal primitives while preserving the size of those on the outline helps in constructing clouds in layers.

The method of generating a 3D cloud structure (subsection 6.3.2) works well for simple cloud outlines, but could be improved for more complex shapes. We would like to develop more sophisticated ways of segmenting the skeleton into trees and their child feature branches, and thus extend the volume new cloud detail may occupy, while still being constrained to the user's input. The procedural cloud structure approach of [BN04] is another way to generate 3D structure and may be well suited for animation, however it doesn't offer the user detailed control over the shape, as only the highest level of the shape hierarchy is user defined. Branches offer natural handles for the user to move surface features around. Our method could be used to generate the level zero structure for [BN04] and interesting future work would be to constrain such an approach to the user supplied silhouette.

We would like to enhance our system by using more prior knowledge of cloud structure. In particular rendering parameters such as noise frequency and density could be set through the use of sketched annotation. An interesting experiment would be for the user to annotate parts of the clouds with sketched wavy lines (like sine waves). The spacing of maxima along the sine wave would indicate frequency, and the amplitude could be mapped to density. This would be useful in addressing problems like the smaller than expected rabbit ears in Figure 6.2, this is because the rendering system we use [BNM$^+$08] interprets thin sections of the mesh as having low density.

Currently our implicit reconstruction only supports spherical primitives. Adding support for more general primitives such as ellipsoids would be a natural enhancement.

We have not addressed cloud animation, it would be interesting future work to consider how different sketches could be used as keyframes in an animation. Two approaches come to mind. The simplest approach would be to use the generated mesh directly as a keyframe for a technique such as [MTPS04]. More complex would be to find sensible mappings between skeletons in different keyframes and then animating the resulting sets of spheres.

Clouds were a good case study in the use of the structure from silhouettes methodology. However the derived geometric skeletons all exist in different drawing planes and are independent of each other. An interesting extension would be investigating ways to link the individual skeletons to give one complete 3D skeleton (this would also have implications for animation). The fine scale detail of the clouds was controlled via a separate procedural method. In the next chapter we re-apply the structure from silhouettes idea to the case of modelling trees in an integrated, multi-scale manner, so that silhouettes are organised in a hierarchy and both large and small scale details emerge from the same methodology.

CHAPTER 7

Seamless multi-scale sketch-based design of trees

As shown in chapter 2, modelling natural elements such as trees in a plausible way, while offering simple and rapid user control is challenging. This section presents our work to address this challenge using the structure from silhouettes methodology, applied at multiple scales: the user draws the overall shape of a tree, from which the internal structure is automatically inferred. Then the user progressively zooms in and adds silhouette detail until reaching the leaf scale. While he zooms out, the branching structure inferred from the multiple silhouettes

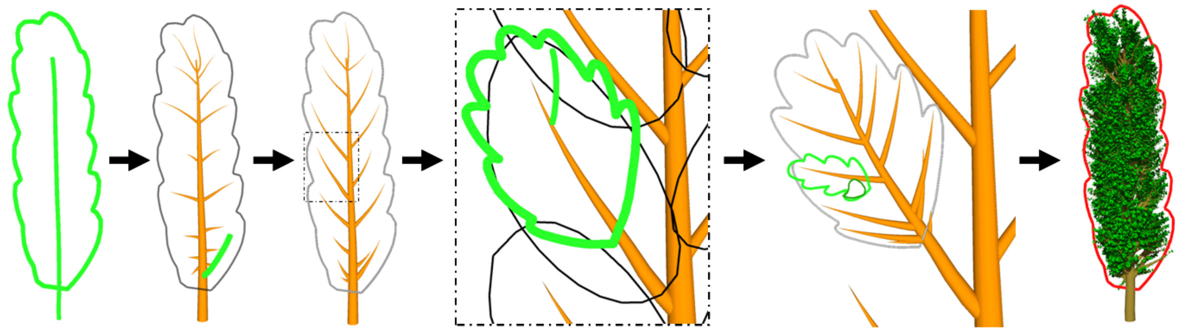


Figure 7.1: Creation of a 3D poplar tree in less than two minutes, using seamless sketching at different zoom factors: the user first draws the trunk and coarse silhouette to control the overall tree shape. Branches and construction lines for smaller-scale silhouettes are inferred from this sketch. After re-drawing one branch to make the orientation of all branches more vertical, the user progressively zooms in to refine one of the sub-silhouettes and finally draws a leaf. When he zooms back, styles are copied and 3D is inferred, resulting in a full 3D tree. The final tree can still be edited by over-sketching elements from a different viewpoint.

is positioned in 3D in a plausible way, while its style is duplicated to the neighboring branching systems. The full tree is thus designed and generated in a few seconds, although the user can still edit it or add specific details through 2D annotation from different viewpoints and zoom factors. The provided combination of user control and prior knowledge of botany, used to infer all the unknown parameters, makes the system particularly useful for quickly generating plausible trees which fit the requirements of an art director.

7.1 Knowledge of tree structure

Trees exhibit many different forms and there are many different detailed models approximating the growth of a specific species (especially for useful commercial trees such as apples[SGG^{+}07]). However these complex models are akin to physical simulation - so they are time consuming to execute and apply to only one specific type of tree. There are many different tree architectures (Figure 7.2 shows a study of just tropical tree architectures) and there is no general rule which can be used to model all of them, thus a sketching approach is a powerful way for the user to directly model the architecture he desires. However as we wish to infer tree structure from silhouettes, how do we ensure the inferred structure is close to the user's intention? Because there are infinitely many possible tree forms which could give rise to a given silhouette - we try to infer the most simple given some very general observations on tree structure, and then allow the user to rapidly alter the result if desired.

Along with [PMKL01] we note that the lateral branching angle of branches along the trunk of a tree is an important factor in the overall shape. A common pattern is for the lower branches to spread horizontally and for the angles to decrease as the branches progress higher in the tree.

Leonardo Da Vinci wrote a number of detailed observations on the structure of trees to aid in representing them through drawing. For example after examining the radii of the trunk, branches and twigs of a tree he observed that at each level the total cross-sectional area of each type would be the same [DV70]. This was later formalised as the pipe model. This model has been shown to be not quite accurate [MSA03] but it is close enough for the purposes of visual representation. We use the pipe model to automatically set branch radii, but the user can over-sketch a conceptual stroke representing a new radius for an individual branch if desired. Da Vinci also made observations about the patterns of branch and leaf growth. We now know that plants axes are built by small embryogenic tissues at the tip of branches, called meristems. During plant development, meristems create lateral organs, leaves, lateral meristems or flowers, that are arranged in well organized patterns along the axes. This organ patterning is called *phyllotaxy*. Botanists have identified various types of phyllotaxy: spiral, decussate, alternate-decussate, whorled, etc. Different phyllotaxies can be exhibited at different scales within a plant. Depending on the species, all the axes of a plant may have the same phyllotaxy or show different organ organization according to the axis status in the plant.

To explain these patterns, one of the most widely accepted theories relies on the assumption that young lateral organs created by meristems generate an inhibitory field in their neighborhood that prevents any new organ appearing in a region around the inhibitory organ. The size of the region depends on the intensity of the inhibitory field and has been shown to control the type of observed phyllotactic pattern [DC96, SKP06]. For young plants, the phyllotactic organization of lateral organs can be readily observed, but for trees the original organization is often blurred by the death of lateral organs due to internal competition for resources (for example water and light).

To model the apparent phyllotaxy of plants, we designed a mixed model in which we combine a inhibitory-field approach with a global optimization process. Each branch generates a inhibitory field in its neighborhood

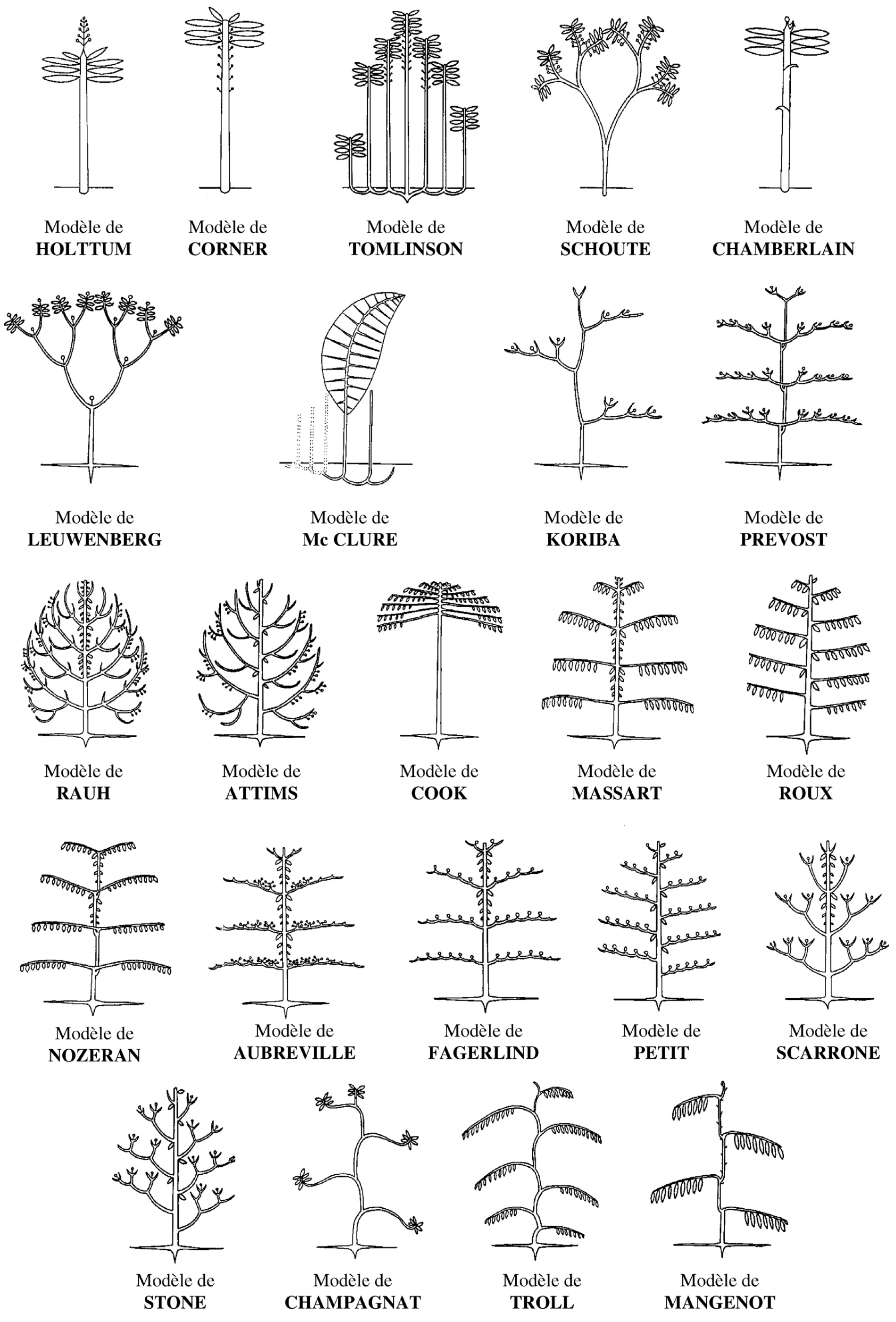


Figure 7.2: A survey of different tropical tree architectures from [HO70].

which deters other branches from being placed nearby. Then, given a particular distribution of branches and constraints on the positions of these branches, we find a distribution that minimizes the energy of branch distribution in their own global inhibitory field. This is explained in section 7.5.

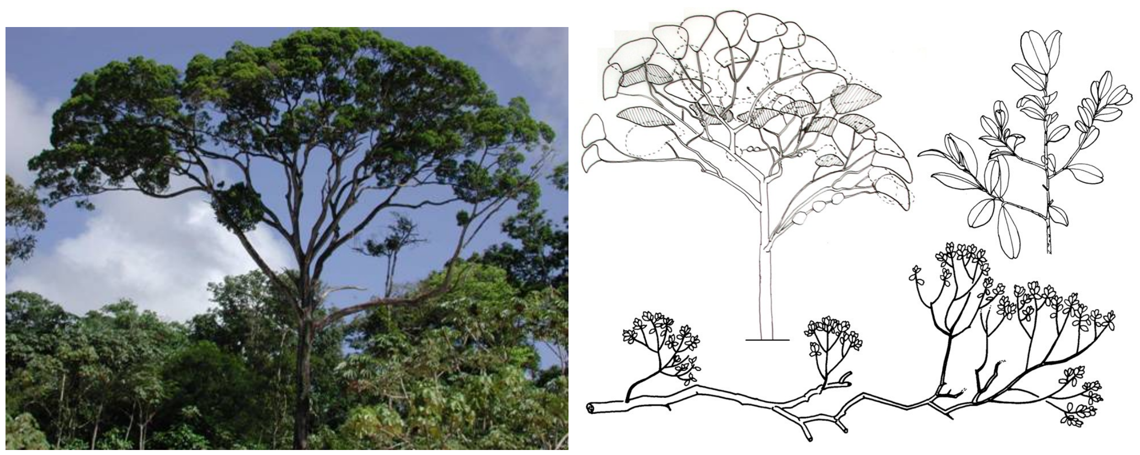


Figure 7.3: **(left)** A photo of a real tree. **(right)** An illustration of how a botanist sketches this tree. Courtesy of Yves Caraglio.

7.2 General Methodology

Our approach builds on the drawing methodologies of both botanists and artists.

Botanists create 2D drawings of trees using pen and paper (Figure 7.3). These drawings aim to capture the essential architectural features of trees and can be used to study the way trees grow. They are time consuming to produce. A botanist will only draw detail in certain sections of the tree, using ‘zoom boxes’ to indicate where the detail should go. If a 3D model is needed an expert in a procedural 3D tree modelling system will study the 2D drawing and try to discern the parameters required to create a faithful 3D version.

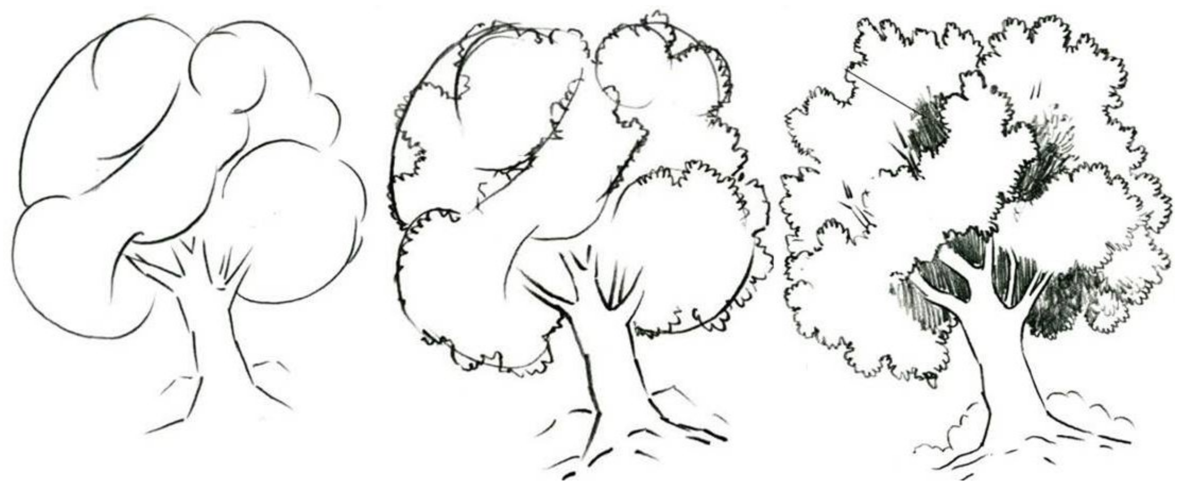


Figure 7.4: Illustration of how artists work. From [Pow98].

Traditional artists create drawings incrementally (see Figure 7.4), placing early construction lines used to guide later more precise refinements. Detail is added after the general outline, and fine detail is often represented only locally. Unfortunately, computer artists cannot use this methodology for creating tree models in standard modelling systems. [PMKL01] presented a system which brought procedural tree generation using such a methodology a step closer - by allowing graphical functions to be manipulated, but it did not support a simple to use sketching interface and still required a good understanding of the underlying model.

Our work addresses the needs of both disciplines by enabling rapid design of 3D trees at multiple scales using a seamless interactive sketching system. Rapid because repeated detail needs to be drawn only once and their 'style' can then be copied elsewhere. Seamless in the sense that the same interface is presented at all times: the user sketches all elements from arbitrary 3D viewpoints within a landscape, from drawing a simple tree outline far away or a detailed leaf shape up close. The system can be used by both novice users and experts, the latter being able to add all the necessary details to express their observations on a specific tree.

A main contribution of our system is the application of inferring structure from silhouettes, using more prior knowledge than for the case of clouds (chapter 6) and applying the approach at multiple scales. When compared to the previous sketch-based tree modelling approaches (where branching structure had to be explicitly drawn) we believe that this methodology both improves shape control and makes the design of complex trees much faster. The user is only required to sketch crown silhouettes at different scales, although he can optionally edit the shape of the inferred branches through over-sketching. The user starts at the largest level of scale - the trunk and the crown silhouette of the whole tree (level zero (L0)). Child branches (L1) are automatically inferred from the silhouette shape using botanical knowledge of trees and guidelines for sub-silhouettes (L1) are generated. After possibly making alterations, the user selects an area to zoom into and progressively draws finer silhouettes until the level of leaves. In the simplest user scenario, once the finest level of detail is reached, the last branch is placed in 3D using one of our botanical-based distribution laws and the style is copied to the siblings at that level; when the user zooms out, 3D distributions and styles are copied until a full 3D tree has been created. Note that advanced users tend to choose different 3D distribution laws at the different scales, to better represent real tree species. The generated tree can be modified further once it has been embedded in 3D, as the camera is free to move at any stage and all operations work from any viewpoint.

We don't want the user to have to indictate whether he is drawing a branch or a silhouette, so we automatically detect the type of stroke drawn using a simple heuristic. We observe that a branch tip will usually end far from the branch root, while a silhouette must be an almost closed stroke enclosing a significant area of the screen. If we measure the span (root to tip distance) and arc length (actual distance travelled by the cursor) of a stroke, then for a branch stroke the ratio of span over arc length will be close to 1, while for a silhouette stroke the ratio will be much lower. Through experiment we find a threshold of 0.5 is excellent at distinguishing between the two stroke types.

The tree is stored as a hierarchy of branching systems with associated 2D drawing planes (Figure 7.5). Each branching system stores two strokes, respectively representing the trunk axis and the silhouette of the system. The strokes are expressed in local coordinate frames, with the origin coinciding with the branch root (insertion point), and the y-axis oriented along the branch span (the line from branch root to tip). The position of each branch along the parent branch is stored using normalised arc length (t). The orientation is stored as elevation angle and azimuth angle relative to the normal vector on the parent branch at the insertion point. This gives a complete tree in 3D with each individual branch being planar. This planar restriction allows the tree to be easily assembled from billboards at render time, thus providing a natural way to rapidly visualize scenes with many drawn trees.

Several challenging problems need to be solved to implement the methodology we just presented: a method for inferring realistic tree structures from user-sketched silhouettes needs to be defined. We build such a structure based on botanical knowledge of trees, as presented in section 7.3. Our method for refining, editing, and copying the structure style between branches is given in section 7.4. Placing the 2D structure in 3D and generating extra branches is discussed in section 7.5.

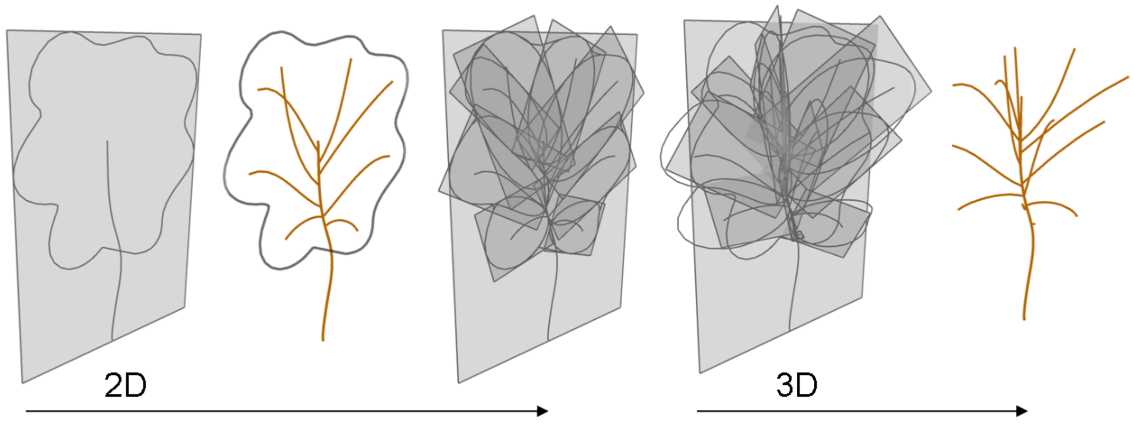


Figure 7.5: The core data structure of our editable tree is a hierarchy of branches with associated drawing planes, each placed relative to its parent. Example of 2D and 3D versions of the hierarchy.

7.3 Inferring 2D structure from a silhouette

Given a silhouette of a tree crown, there are an infinite number of 2D branch and leaf arrangements that could provide that silhouette. We aim to provide one ‘reasonable’ solution from among the alternatives, where reasonable means a solution that attempts to meet the expectation of the user and that bears some resemblance to common tree architectures. Of course there are an infinite number of ‘reasonable’ solutions, and the user expectation may have been for a different solution to the one chosen. Thus the chosen arrangement must be easy for the user to alter quickly, without much effort.

7.3.1 Silhouette segmentation and major branches

Our first assumption about the major branches producing the silhouette of a tree is that they start from a common trunk (a monopodial structure) and they extend into the ‘bumps’ observed on the silhouette. These bumps can be seen as indication of the crowns of the associated sub-branching systems. These assumptions confirmed by conversation with a botanist (section 6.4) match the natural segmentation humans perceive when regarding tree silhouettes, thus helping to meet the user expectation.

As in the case of clouds described in chapter 6 our method for segmenting the bumps (groups of silhouette edges associated with local maxima of curvature), and for finding the direction of major branches is based on the *Chordal Axis Transform* (CAT) [Pra97] (described in subsection 2.1.1).

We generate a constrained Delaunay triangulation using the silhouette edges as constraints (see Figure 7.6(a). We classify the resulting interior triangles as *terminal* triangles, *sleeve* triangles and *junction* triangles. We then connect the circumcentres of adjacent triangles to form the CAT skeleton (Figure 7.6(a)).

We select the terminal branches of this graph, i.e. those associated with a terminal triangle followed by a sequence of sleeve triangles. The associated bump shape on the silhouette is defined by the silhouette edges that belong to these terminal and sleeve triangles. Figure 7.6(b) depicts the segmented silhouette bumps in different colors and indicates the circumcentres of terminal triangles.

As in [Pra97], the CAT may need to be simplified before use, since small artifacts in the drawn silhouette could generate insignificantly small terminal branches. However care must be taken as we wish to maintain a correspondance between drawn silhouette edges and the shape skeleton, in order to later segment the silhouette. This is more important for the case of trees than for clouds, as for clouds we determine only coarse structure -

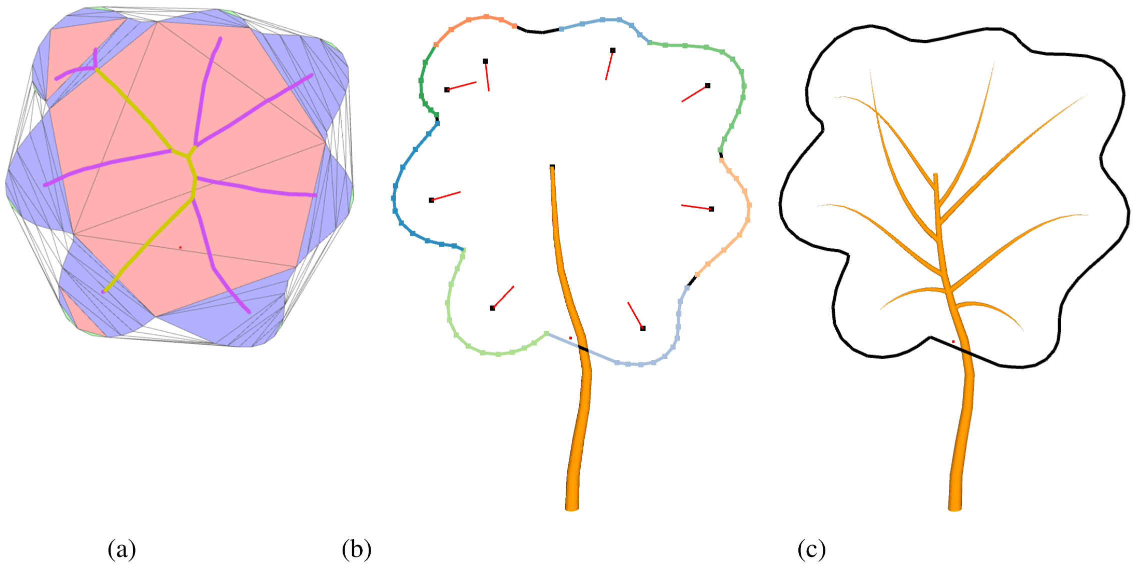


Figure 7.6: **(a)** Chordal Axis Transform (CAT) computed from a Delaunay triangulation. **(b)** Information derived from the CAT: Black points are circumcentres of terminal triangles, used as branch end-points. Red tick marks indicate the tangent direction along each terminal branch. Colored groups of silhouette edges represent the individual bump shapes. **(c)** The inferred branches.

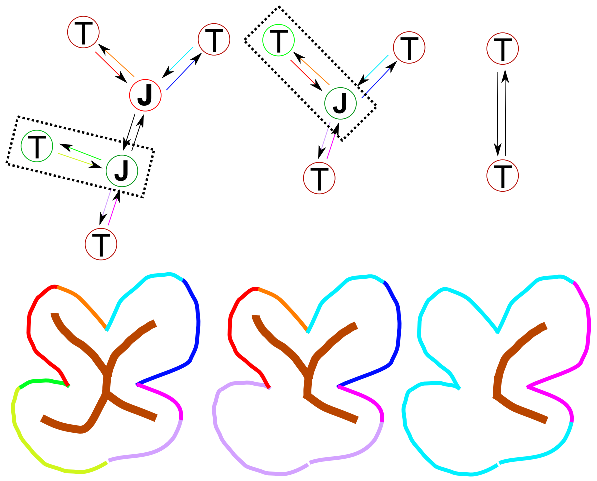


Figure 7.7: Skeleton simplification. The branch sizes here are much larger than the threshold (0.2% of total triangle area) but it serves to illustrate the process. **(a)** Two half-edges and two nodes are merged with the adjacent half-edges. **(b)** again two-half edges are merged, leaving **(c)** the degenerate case where no further simplification is possible.

but for trees the terminal triangles define the end point of branches - which at the largest scales are distinctive shape features. Hence we now describe the method of maintaining a correspondance between silhouette edges and the underlying skeleton during pruning. We discard a terminal branch b if the combined area of its triangles is less than 0.2% of the total area enclosed by the whole silhouette. To avoid losing silhouette information, we assign the silhouette segments belonging to b to the branches adjacent to the discarded branch, maintaining their connectivity (and thus ordering). To facilite this process we represent the CAT skeleton using a convenient abstraction: a half-edge data structure. The CAT skeleton can be represented by a directed, non-cyclic graph. The nodes of the graph are denoted as either junction nodes or terminal nodes. The edges of the graph represent branches in the skeleton. Each branch of the graph may be represented by two half-edges with opposing directions. Each half-edge can be classified as being one of four types depending on the type of the node at either end (TJ, JT, JJ or TT). In this way the whole graph can be traversed by following half-edges (we choose a counter clockwise ordering for half-edges). Each half-edge has zero or more segments from the silhouette associated with it (also stored in counter clockwise order). This way, when a terminal branch (a JT half-edge followed by a TJ half-edge) is discarded, the associated silhouette segments are concatenated with the set of silhouette segments from the prior half-edge (if this half-edge is JT) or the next half-edge (if this half-edge is TJ), the associated node is deleted and the adjacent half-edge types are changed accordingly. Using this scheme, all silhouette edge segments are retained (Figure 7.7).

7.3.2 From skeleton to tree branches

It is tempting to use the CAT skeleton directly as the geometry of the tree branches, however the CAT branches do not look like those of a real tree (see Figure 7.6(a)). Similar observations were made in previous work: [SRDT01] attempted to use the 3D MAT from the 3D visual hull of a tree directly as the complete tree skeleton (see Figure 7.8). The results differed significantly from realistic trees so other approaches, such as using particle flows [NFD07] or self-similarity [TZW$^+$07] were introduced. Our approach is different: following [PMKL01], we think that the gradient of lateral branching angle (elevation angle) along the trunk is very important to the designer, since it strongly influences the general posture of the structure. We therefore propose to combine the user supplied shape information, extracted from the CAT, with botanical insights on the variation of this branching angle. This method, described next, leads to the generation of simple, natural looking branching structures which the user can easily alter if desired.

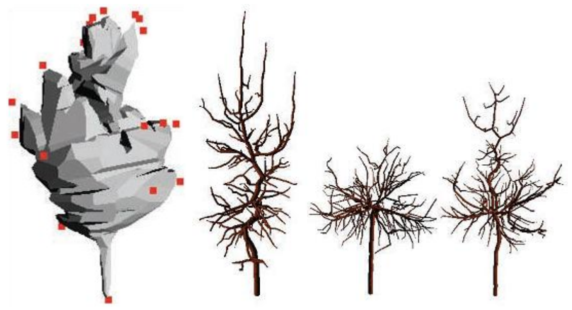


Figure 7.8: **(left)** In [SRDT01] the 3D visual hull of a real tree is constructed from multiple photographs, then **(right)** the 3D Medial Axis Transform is used to generate the main tree architecture. The authors comment that using the MAT directly doesn't give very plausible results. This motivates our approach of using botanical information about branching shapes (incorporated in the BAF) in addition to the Chordal Axis Transform to determine branch end points.

The gradual variation in lateral branching angle (elevation angle) of child branches along a parent trunk is modelled in our system by a function we name the *branching angle function* (BAF). The BAF maps positions $t \in [0,1]$ along the parent trunk to angles between the tangent of the parent trunk and the tangent of a child branch at their junction point. The default values of the BAF are chosen to match the common botanical observation that branches at the bottom of a trunk tend to grow horizontally (plagiotropy), but become more vertical towards the top of the tree (orthotropy). Therefore, the BAF is defined by pre-set maximal and minimal angle values at the bottom and top of the trunk, linear interpolation being used in-between (the BAF is depicted using the black segments along the trunk on Figure 7.9). When the user re-draws a branch (see section 7.4), this may either modify one of the existing values or add another sample point to the function.

The major branches, defined by their end-point E and the associated tangent to the CAT branch (see Figure 7.6(b)) have already been extracted from the sketched silhouette. The goal is now to find a starting point B on the trunk for these branches, the direction of the branch at this starting point being given by the BAF. Once this is done, a spline curve will be a good guess for the shape of the branch in-between. In our implementation, we use 3rd order Hermite curves since they are defined from their endpoints and the associated tangent vectors. If desired, the user will be able to redraw a branch using a free form curve later on.

We use a simple, incremental method for finding the best B for a given E: using a regular sampling along the trunk, we approximately compute the length of each Hermite curve joining B_i to E with the tangent vectors mentioned above (the length of these vectors being set to half the distance between B_i and E). Then we simply select the shortest of these branches. For a long trunk with about 40 sample-points, generating all branches shapes only takes a fraction of a second. Examples of branches resulting from different tangents at endpoints are depicted on Figure 7.9 and a full result is shown on Figure 7.6(c).

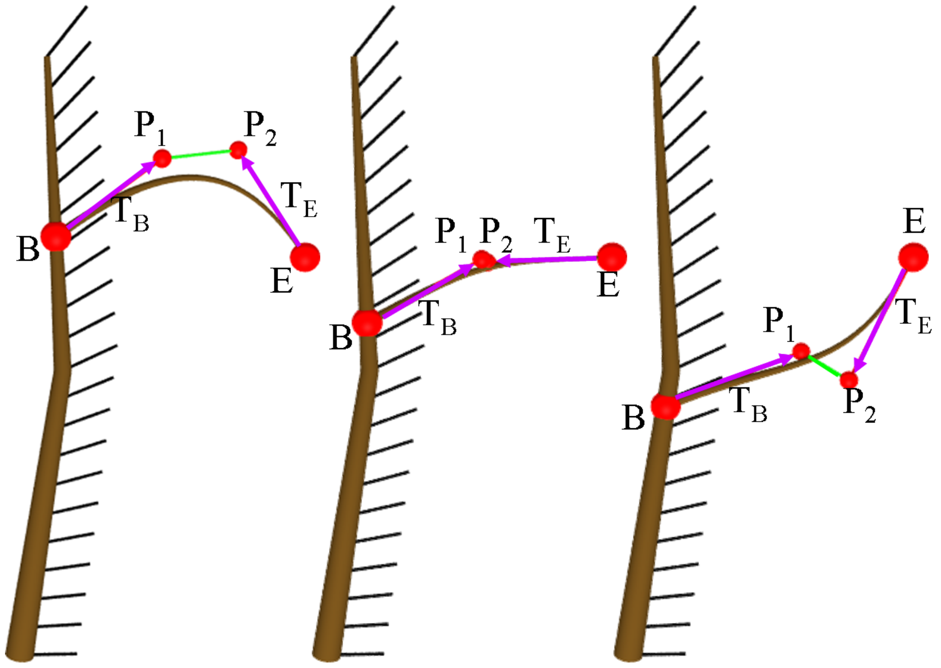


Figure 7.9: Three examples of automatic branch shape determination starting from the same endpoint E but with different end tangents T_E. The Branching Angle Function (BAF) is shown using black vectors along the trunk.

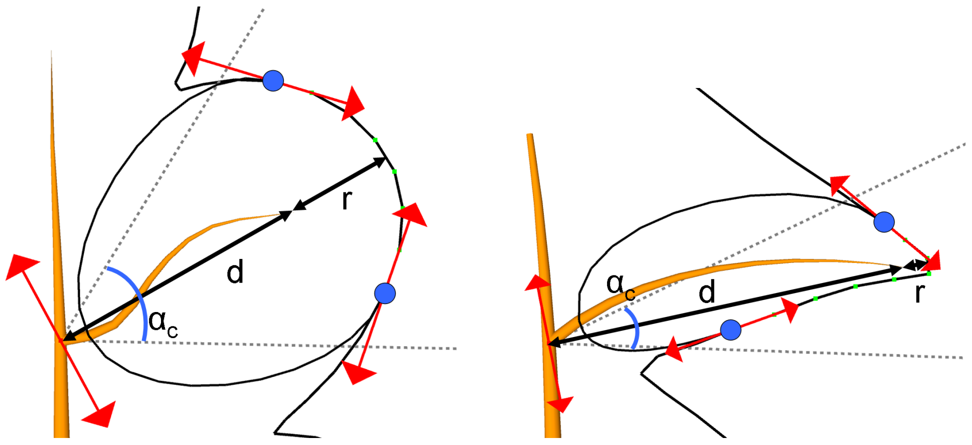
7.3.3 Inferring sub-silhouette shapes

As mentioned earlier, the bumps we detected on the sketched silhouette indicate the crowns of the sub-branching systems associated with the major branches. Rather than leaving the user to redraw these sub-silhouettes from scratch while zooming in, we propose a method to extract a simple version of them from the original sketch. This has several benefits:

- The inferred sub-silhouettes are used as the area in which to generate more branches if the coverage of the region inside the tree is not good
- They serve as construction lines, guiding the user as he refines the sketch
- They constrain the size of new branches generated during style copying from one branch to another.

An inferred sub-silhouette should share some of its edges with a bump on the parent silhouette, and then deviate smoothly to connect near to the base of the associated branch. Although using our segmentation of the tree silhouette is natural, we found out that in practice, the lateral extent of the detected bumps would often produce overly large regions (see for instance the blue bump on Figure 7.6(b)). Thus we select only those edges of the segmented bump that lie inside a cone defined around the segment $[B,E]$ of the associated branch. As shown on Figure 7.10, the angle of this cone should depend, for a given branch length, on the radius of curvature at the end point of the branch: large, smooth features on the silhouette should generate large sub-crowns, while sharp features should generate narrow ones. More precisely, we use the ratio r/d between the radius at the extremity and the length of the branch to select an adequate value for the cone angle, using linear interpolation between two pre-set extreme angle values (we currently use 20° and 110°, values that we observed from real trees).

Once the part of the silhouette bump inside the cone has been selected, we complete it using two Hermite curves joining it to the root of the new branch, choosing the tangents so that the shape is G^1 continuous. See Figure 7.10.



(a) (b)

Figure 7.10: Silhouette features with **(a)** large and **(b)** small radii of curvature. The cones (dashed lines) used to select how much of the parent silhouette should be used to form a sub-crown. The red vectors are the tangents at the ends of the two Hermite curves and the ends of the shared edge section. The blue dots are where the shared edge meets the curves. α_c, d and r are described in the text.

7.3.4 Ensuring the parent crown is covered

The last automatic step before displaying the result and enabling the user to refine the sketch is to check whether the detected sub-silhouettes give a good coverage of the tree. If it is not the case, we generate more 2D branches. Note that if the user drew a very smooth silhouette for the tree, the coverage will typically be quite bad (due to the lack of high curvature features and thus to the small number of branches), so performing this coverage check is necessary.

Checking coverage is based on the inferred sub-silhouettes: we ‘fill-in’ any remaining space by proceeding iteratively as follows, from the base to the top of the parent trunk: at each sample position along the trunk, we iteratively copy the nearest sibling branch and crown and re-orient it according to the BAF. It is then scaled to fit within the parent silhouette and tested for overlap with the existing set of sibling crowns. If the area of overlap is above a threshold (we use 20% of current crown area) then the branch copy is rejected and the next position is tested, still using the sub-silhouette of the nearest, existing branch. Otherwise the new branch is added to the set of siblings and the selection continues. In this way, the density of newly created branches is related to the shape of the silhouette (Figure 7.11).

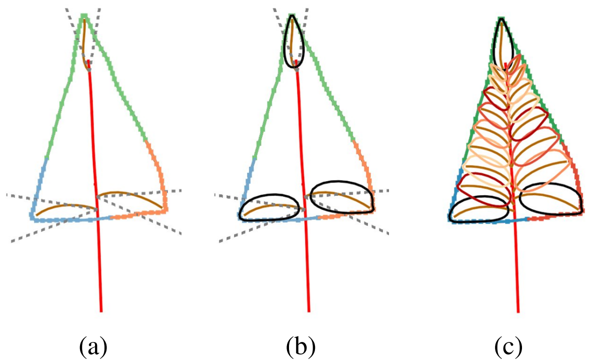


Figure 7.11: When there are few curvature features in the silhouette there may not be enough branches inferred to ensure good coverage. To address this we copy and scale existing branches using an area overlap threshold to control density and adjusting their elevation angle according to the BAF (subsection 7.3.4). **(a)** The only inferred branches, **(b)** The inferred branch silhouettes, **(c)** Additional branches created using coverage criteria.

7.4 Sketch refinement and style transfer

Once the inferred structure (2D branches and associated sub-silhouettes) is displayed, the user may edit the branch style by over-drawing one or two of them and possibly extend the changes to the sibling branches. Then, he/she will typically zoom onto one of the sub-silhouettes to refine it, enabling the inference of sub-branches at a smaller scale. We provide a mechanism for automatically transferring the sub-branch distribution to the other sub-silhouettes, reducing the need for manually refining several of them.

7.4.1 Redrawing branches and transferring their shapes

In addition to being able to draw new branches and delete existing ones (scribble gesture), the user can change the shape of a branch through over-sketching. The new (source) branch S is drawn close to the branch to be edited. The target branch is selected using a cost function based on proximity between root and end points.

Once a branch has been replaced, the user can choose to propagate its new style to the sibling branches. First, the angle and position of the modified branch on the parent trunk are used to add a sample point to the parent BAF (see Figure 7.12). The position of the roots of the sibling branches are automatically repositioned based on this new BAF. Sibling branches are then automatically re-shaped: a spline curve is fit to the new branch S and control points are re-expressed in a local frame based on the span vector of S. These local coordinates are converted to a frame based on the span vector of the target branch, and scaled according to the ratio of span lengths. This defines a new spline curve, giving the new shape for the target branch. The default sub-silhouettes of sibling shapes are automatically modified according to the new branch starting point.

7.4.2 Refining sub-silhouettes and copying style

While the previous editing step is often unnecessary, the user will generally want to refine his sketch by zooming in and redrawing at least one of the sub-silhouettes (the sub-silhouette shape we inferred being used as a construction line). Smaller scale structure will be computed from the sketch exactly as before and the style

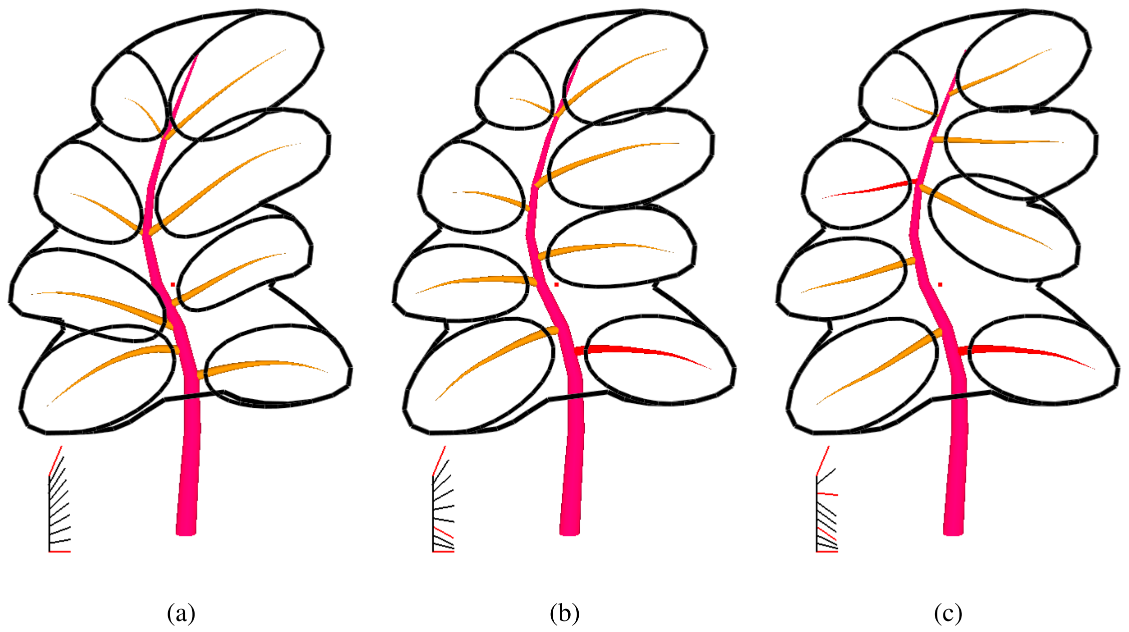


Figure 7.12: **(a)** The automatically inferred sub-branches. Inset - the branch angle function (BAF). **(b)** The red branch has been redrawn by the user, and the shape change is automatically copied to the sibling branches and the sub-silhouette shapes are automatically adjusted. Inset - the altered BAF. **(c)** The user redraws a second branch making it straighter, and again the shape change is copied.

of the new crown can be automatically copied to its siblings. In the tree examples we show, this refinement process is typically performed 3 to 5 times, the finest-scale silhouette sketch being interpreted as an example leaf (indicated by a key press after drawing).

We use a simple metric based on overlap to automatically detect which of the sub-silhouettes the user redrew: the branch which has the largest absolute length covered by the new sub-silhouette (in screen space) is selected.

Once the user is satisfied with the sub-silhouette he redrew and the architecture of the resulting branching system, he may want to copy the same style to its siblings without having to redraw similar silhouette details many times. We provide a method based on extracting and copying parameters describing the distribution of sub-branches along a branch. This again uses prior knowledge of tree growth, detailed next.

From botanical studies [GBYC01], we know that if a section along the beginning of the branch bears no substructure, then it is likely that a similar lack of growth is observed on other sibling branches. So we assign the branch two ranges, an unbranched range at the beginning followed by a ramified (branching) range. The unbranched range is measured as an absolute length which is preserved on other branches. The ramified range is parameterized by a mean and standard deviation (S.D.) of the absolute distances between branching points. A normal distribution is used to generate new branching points on the ramified ranges of sibling branches using these details. The BAF of the source branch is also copied to the target. Branch shapes at new branching points are copied from nearby source child branches, chosen by comparing the normalized t positions along the trunks. The copied child shape is re-oriented according to the BAF and scaled to fit the target silhouette. The different steps of this process are depicted on Figure 7.13.

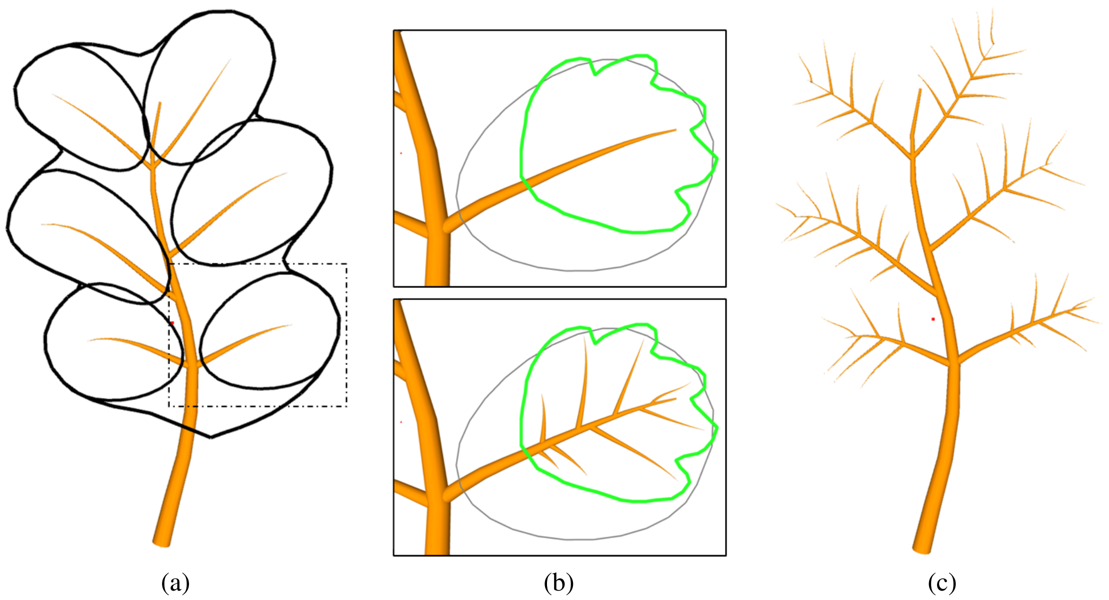


Figure 7.13: The process of copying the style of a branch. **(a)** The initial structure with no L2 branches. **(b)** The bottom right branch silhouette is redefined by the user and the internal branch structure is automatically inferred. **(c)** The branch distribution from the example is used to automatically generate all L2 sub-branches. Note the absolute length of the unramified section at the base of the branches is preserved.

7.5 Positioning organs in 3D

Sketched branches or leaves have a defined position on their parent branch but their 3D azimuthal orientation around their parent branch is not specified in the drawing as we only perceive a projection of the branch in the screen plane. Reconstructing a full 3D interpretation of the branching system thus requires the inference of three types of information from the drawing:

- the azimuthal orientation of the sketched branches
- the existence of extra non-sketched branches
- the shape, length, position and azimuthal orientation of these additional branches.

To solve this problem, two types of constraint must be taken into account. First, the reconstructed plant must be consistent with the sketched information, since the silhouette of the plant imposes constraints on its structure. Second, the solution must also adhere to some botanical laws, depending on the degree of realism the user wants.

We offer a solution which conforms to both sets of constraints (user drawn and botanical) by minimising a cost function which determines where both existing and additional organs (branches or leaves) should be positioned on a parent branch. This section describes how both constraints are specified and used to infer the final 3D tree.

7.5.1 Position and orientation of branches

To represent the position of organs along the trunk, we use a representation that is frequently used for the analysis of phyllotactic patterns and which consists of unfolding the branch cylinder along its carrier curve (see

Figure 7.14). In this system, the insertion point p of each branch on its parent branch is characterized by a set of coordinates (θ, u), where θ is the azimuthal orientation of the branch and u is the curvilinear abscissa of the branch along the carrier curve ($\theta, u \in [0,360] \times [0,1]$). To account for the cyclic nature of the stem cylinder, we assume that the domains of insertion points are cyclic. We also associate with (θ, u) the normalized coordinates (θ, u) such that $\tilde{u} = u/L$ and $\tilde{\theta} = \theta/360$, where L is the length of the parent axial axis.

Constraints may be imposed on the points by defining a positioning domain $D_i = (\left[u_i^{min}, u_i^{max}\right], \left[\theta_i^{min}, \theta_i^{max}\right])$ for each point p_i. Valid branch positioning solutions are thus such that $p_i \in D_i$ for all i.

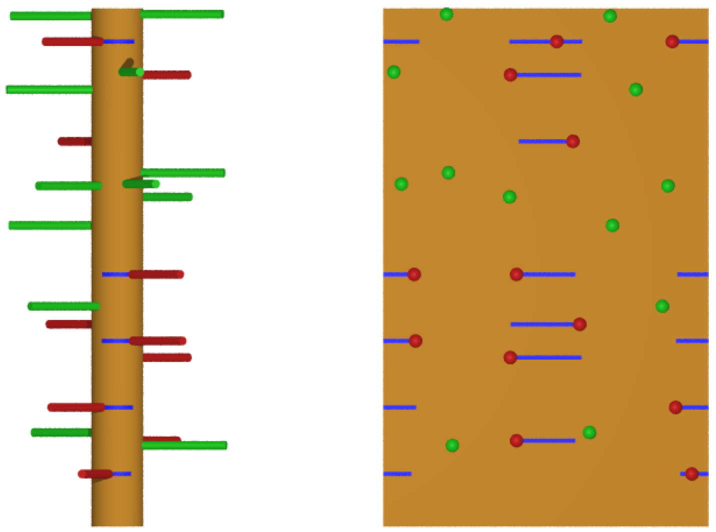


Figure 7.14: Cylindrical (folded) and planar (unfolded) views of the positions of a set of lateral branches. Height corresponds to position along the trunk, width to angle. Red points correspond to positions of sketched branches. Blue segments represent their possible ranges. Green points correspond to positions of added branches. Added branches may be positioned anywhere in the whole domain.

7.5.2 Inference rules and constraints

Addition of new branches

For each sketched silhouette, a first series of branches are inferred (section 7.3). As remarked by [OOI05], *"people tend to draw branches that extend sideways and omit branches extending toward or away from the screen"*. This was confirmed by the botanist who tested our system. Therefore, the density of branches we get on the main stem is generally enough for 2D coverage, but is underestimated for 3D. Consequently, we provide a mechanism for creating additional branches to fill up the 3D space around a parent branch. In our system, density can be incrementally increased in steps using a key. For example the user may increase branch density by 50%, 100% or by a constant number of branches by pressing the appropriate key.

Constraints on the positioning of sketched branches

If we assume with [OOI05] that the user sketches branches that are contained in planes close to parallel with the screen plane, we must respect this by imposing constraints on the position of these branches on the plant. For this, we assume that the coordinates θ of their insertion points are close to either $\theta_{ref} = 0°$ or $180°$ on the left or right sides of the tree respectively. By default we set their positioning domains such that $D = u_0 \pm 0, \theta \pm 45°$. Once the cost function is solved we define a plane with its origin at the branch insertion point and at θ degrees to the parent branch plane. The sketched branch is then projected along the view direction onto this plane, so that it meets both user supplied projection constraint, and the botanical constraint represented by the cost function.

Constraints on the positioning and size of additional branches

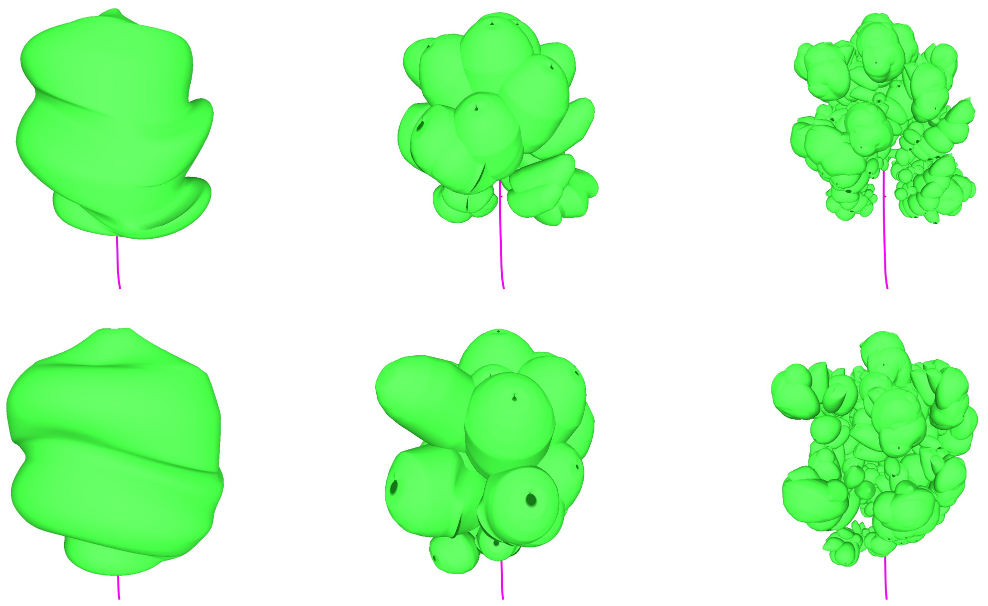


Figure 7.15: **(top row)** The surfaces of revolution, from global to local scale, used to automatically scale inferred child branches. The surfaces are generated from the 2D silhouettes (section 7.5.2). **(bottom row)** The same surfaces from a different viewpoint.

For the additional branches, all u and θ values are available. Once the cost function is minimised we have an azimuthal angle and the u value can be used to supply an elevation angle from the BAF. We chose a branch shape by copying a nearby neighbour on the parent branch, however we still need to scale the branch because there are no projection constraints for these branches we need a way to ensure that they don't extend beyond the user supplied silhouette. This is achieved by creating a 3D surface of revolution from the 2D silhouette that all additional branches are scaled to meet (see Figure 7.15). This surface of revolution is formed by the following procedure:

- Divide the 2D sketched silhouette into two halves, divided in two along the line defined by the base and tip of the branch. Resample the two lines so they have the same number of points.
- Repeat the process, but use the convex hull of the silhouette.
- Take the average of the two half silhouettes of the convex hull.
- Form a surface of revolution by interpolating from the left silhouette (0°), to the right silhouette (180°), going via the average convex hull silhouette (at 90°and 270°).

We chose this 'hybrid' convex hull because it ensures no new branches would exceed the drawn silhouette, while not artificially shortening branches which are close to parallel with the view direction. If we just formed the surface of revolution without the convex hull, then the negative curvature areas from the silhouette would tend to overly shorten new branches, as they would more often be created in positions which coincide with areas of negative curvature. This is because positive curvature areas would already have branches defined (inferred by our method) and so the cost function would tend to place new branches away from these root positions. If we formed the surface of revolution using the convex hull only, then new branches could be placed in a plane close to parallel to the viewplane, and in an area of negative curvature, thus extending beyond the drawn silhouette and breaking the user supplied constraint.

7.5.3 Optimization of branch positioning

The 3D arrangement of branches along a parent stem is controlled by a stochastic point process, known as the *Gibbs process* [Dig83]. In such a process, a pairwise interaction function $f(i,j)$ represents the cost for two insertion points p_i and p_j of being at a relative distance d(i,j) from each other. It can be seen as modelling the interaction/competition between branches. A realization of this process corresponds to minimizing a global cost function defined as the sum of the costs for each pair of points:

$$F = \sum_{i \neq j} f(i,j), \tag{7.1}$$

where f is a function that decreases according to the distance between the two points. The Gibbs process simulates situations of dynamic equilibrium. We first define d a normalized distance:

$$d(i,j) = \left\| (\Delta\tilde{u}_{ij}, \Delta\tilde{\theta}_{ij}) \right\|^2 \tag{7.2}$$

with $\Delta\tilde{u}_{ij} = \tilde{u}_i - \tilde{u}_j$ and $\Delta\tilde{\theta}_{ij} = \tilde{\theta}_i - \tilde{\theta}_j$. Note that $\Delta\tilde{\theta}_{ij}$ and $\Delta\tilde{u}_{ij}$ have to be adjusted to take into account the cyclic nature of the domain. We then define the following closeness penalty function f as the cost function of the system.

$$f(i,j) = \frac{1}{1 + \alpha * d(i,j)} \tag{7.3}$$

where α is a parameter used to weight the contribution of the cost function $d(i,j)$ (set by default to 20). f is thus maximum when i and j are at the same position and decrease with distance.

For some specific tree species, the above cost function must be modified. Additional terms are added to promote specific position angles between branches, for example to stress a particular phyllotactic pattern. A branch arrangement is controlled by a phyllotactic angle ϕ, defining the angle between organs at two successive nodes. To account for whorls we allow a node to bear several organs. The angle between successive organs of the same whorl composed of n elements is generally $\gamma = \frac{360}{n}$. To determine a theoretical angle between two given organs, we also need to know how many nodes are in between. For this, we use an average node length l that can simply be approximated by Ln/N, where N is the total number of branches borne by the stem.

To make regular branching patterns emerge from the cost function, we will assume that a branch i at a given position p_i induces *privileged positions* for the other branches. The cost function between two branches is thus made proportional to the distance to the closest privileged relative position. The new cost function g is:

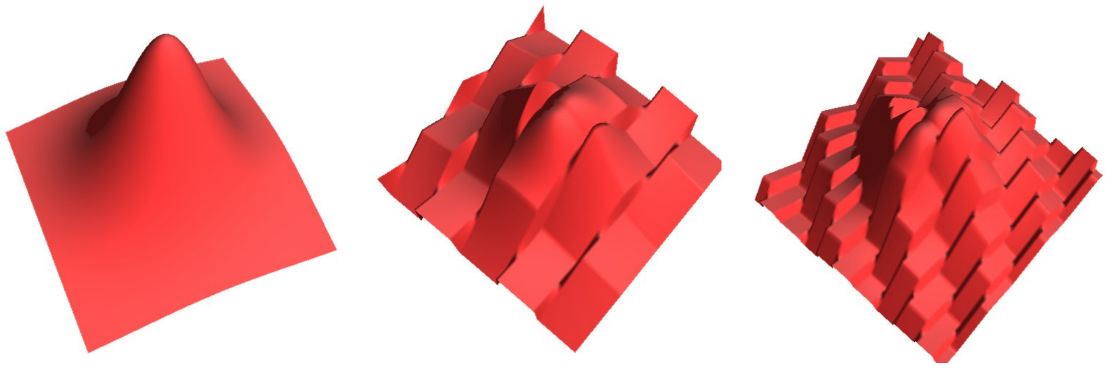


Figure 7.16: Local cost functions used for positioning branches. **(a)** $f(i,j)$ from Equation 7.3. **(b)** $g(i,j)$ from Equation 7.7 with $\gamma = \frac{360}{2}$ and $\phi = 90°$. **(c)** $g(i,j)$ from Equation 7.7 with $\gamma = \frac{360}{3}$ and $\phi = 30°$.

$$d_{node}(i,j) = \lfloor (\frac{\Delta u_{ij}}{l}) \rfloor \qquad (7.4)$$
$$d_{ang}(i,j) = (\Delta\theta_{ij} - d_{node}(i,j) * \phi) \bmod \gamma \qquad (7.5)$$
$$d_{pattern}(i,j) = \frac{\min(d_{ang}(i,j), \gamma - d_{ang}(i,j))}{\gamma} \qquad (7.6)$$
$$g(i,j) = w_1 * f(i,j) + w_2 * d_{pattern}(i,j) \qquad (7.7)$$

where w_1 weights the contribution to the cost due to closeness and w_2 weights the contribution due to the distance, $d_{pattern}$, to the nearest privileged position. To compute this distance, we use d_{node} to define the number of nodes between i and j and d_{ang} to define the variation of angle to the branch position in the pattern at this node distance. In practice to enforce non-overlapping of organ positions, w_1 and w_2 are set to 2 and 1 respectively. This second cost function has thus the same shape as f but with regular local minima and maxima if the relative position of i and j corresponds to or is far from a given phyllotactic pattern respectively (Figure 7.16).

To simulate the Gibbs process, we use the iterative algorithm of *depletion–replacement* [Rip79]. Let P be an initial set of insertion points p_i and F the associated global cost function value. Starting from P, a point is selected at random and its position is changed randomly. The old configuration is replaced by the new one only if the new one has a smaller global cost value F. The process is initiated with a random distribution and iterated until the change in the cost value between consecutive steps remains under a prescribed threshold. During iteration, we observe that F decreases quickly during early iterations and then tends slowly to a local minimum (the optimality of the solution is not of major importance here). In our implementation, we use a maximum of 3000 iterations which is interactive and produces realistic solutions.

7.5.4 Resulting distributions

Figure 7.17 shows different branch arrangements produced by our stochastic process. Red lateral branches are deduced from the sketch and green ones are added to give more volume to the tree.

The first one uses the cost function f. Branches are evenly distributed along the trunk length in terms of both u and θ. The second distribution, called *opposite-decussate*, uses the cost function g and assumes whorls of two branches where each whorl is rotated at 90° from the previous one. We observe an alternating series of two opposed branches. The final distribution is a *whorl* distribution where the whorls are composed of three

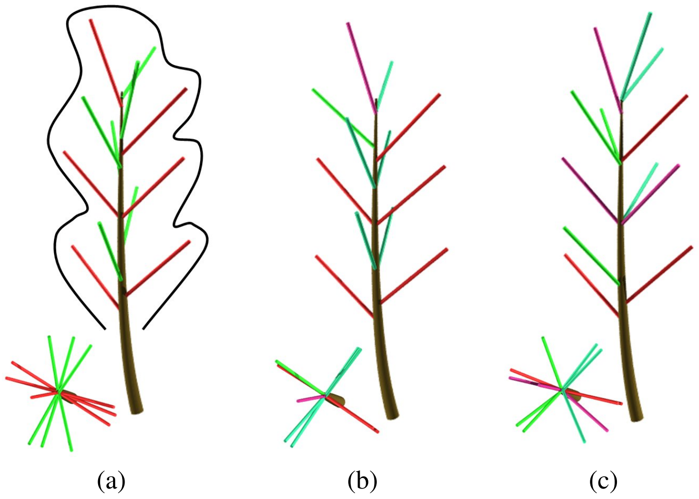


Figure 7.17: Illustration of three different user selectable 3D branch arrangements. Red branches are constrained, green branches are free to move. **(a)** Evenly spread, **(b)** opposite-decussate, **(c)** whorl.

lateral branches with a phyllotactic angle of 60° between whorl branches. Branch colors have been slightly modified to aid whorl identification. Other arrangements can be formed easily by choosing a number of organs per whorl and a phyllotactic angle. In our system, additional arrangements, such as spiral ($\phi = 144°$ and $n = 1$ for whorl), decussate ($\phi = 0°$ and $n = 0$) and horizontal decussate, are available (see Figure 7.18) or can be set by the user. Once the 2D structure has been drawn, the user can explore the various arrangements the system provides and choose the one he prefers (by keyboard selection).

7.6 Discussion and future work

Our aim was to explore the rapid construction of realistic looking trees using a sketch-based interface that was close in spirit to the approach used by artists and botanists in drawing trees. The tree architecture could be specified at multiple scales through either sketching branches directly (as done by [OOI05]) or by inferring sub-branches by sketching crown shapes and using prior knowledge of the structure of trees. To reduce the amount of repetitive detail that users would have to draw we implemented style copying operations, which would generate similar details elsewhere on the tree. When placing existing detail in 3D we respected the positional constraints supplied by the user (so the 3D branches would look the same as the 2D branches when viewed from the construction viewpoint) and we constrained new growth to positions dictated by phyllotaxic rules - which could be specified by the user.

Although non-optimized (written in python using C++ modules), our prototype implementation runs at interactive rates, enabling fluid user interaction. Figure 7.1 and Figure 7.18 give an idea of the variety of trees we can design in a few strokes. Table 7.1 shows the number of strokes required and the resulting complexity. Note that sketching the branching system (as in [OOI05]) instead of hierarchical silhouettes would have taken longer, and might have been difficult to achieve for our dense tree examples. The minimum input to our system, as depicted at top left of Figure 7.18, is one stroke per level of hierarchy. Designing four levels of detail and building the whole hierarchy while zooming out took about two minutes. The example in Figure 7.1, a poplar which required some branch editing, took less than two minutes. The eucalyptus in the middle row of Figure 7.18 (based on a real tree), involved more redrawing to express the variety of branch shapes, and took about ten minutes to design.

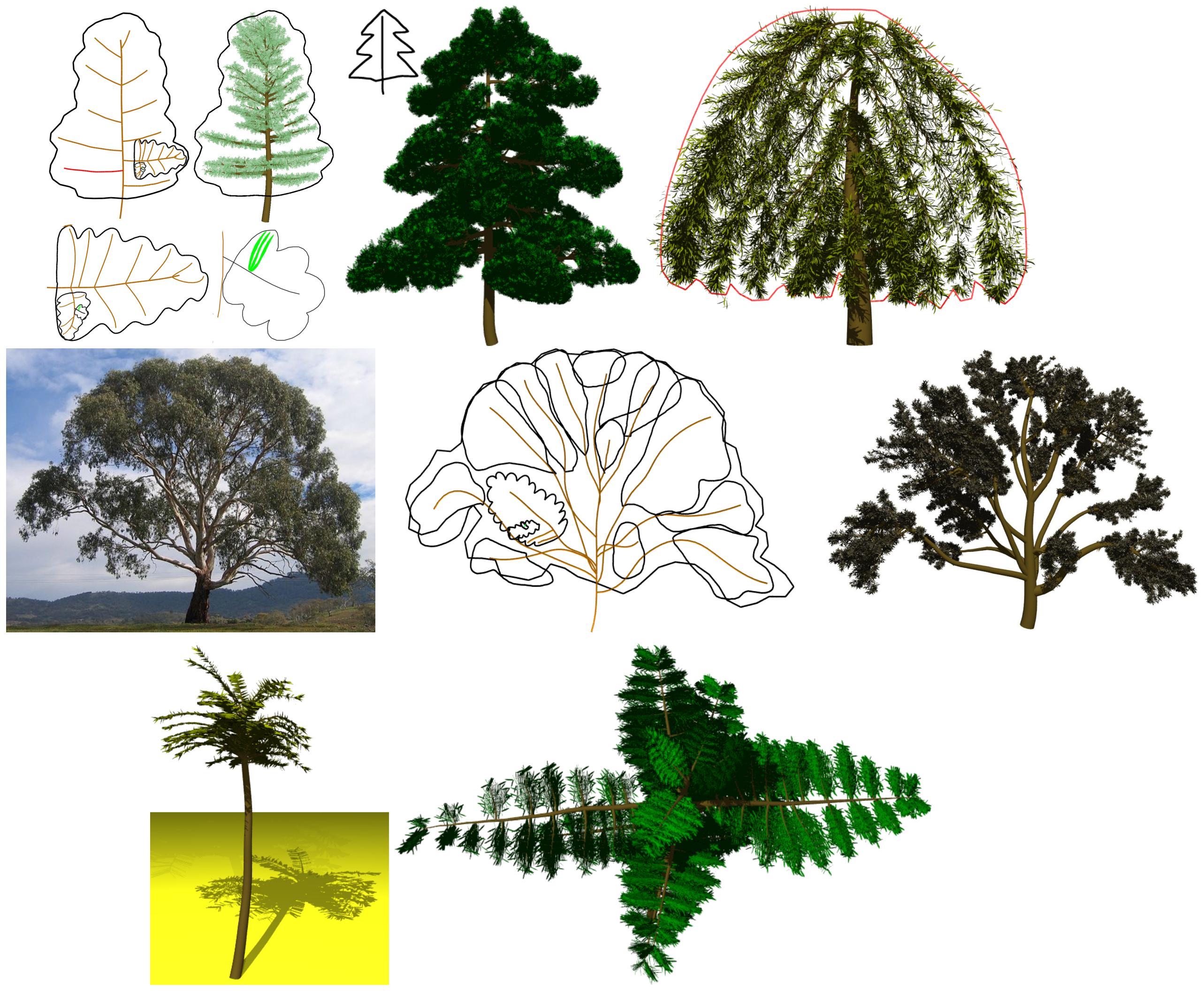


Figure 7.18: Results show the variety of complex trees easily achieved with our system. We used different 3D distributions of branches for designing (from top to bottom and left to right): a young pine tree of low density, a dense Christmas tree (drawn at the initial scale as a child might), a willow, a eucalyptus, palm tree and ferns. The complex eucalyptus example was sketched over a photo of a real tree, and illustrates the way botanists can use our system for expressing their observations on real trees.

We invited an artist and a botanist to use our system. They found that the combination of sketching silhouettes and copying styles to be powerful: with little input, testers achieved results which seemed convincing to them. However, they sometimes felt that it was more natural to sketch silhouettes at smaller scales and to sketch branches directly at larger scales - as they often already had strong ideas about large scale structure of the tree. Giving the outline first took some getting used to, but lead to better proportions and control over the sketch.

The ability to sketch from multiple viewpoints is useful for adding branches in specific locations. It would be of most use when designing a tree like the palm tree (Figure 7.18), where the second level structure is almost parallel to the ground plane - it is natural to move the camera to a top down view point and sketch the silhouette from this position. In our current implementation we support this by rotating by 90°the elements inferred from a side view, for the more intuitive approach we would have to extend the underlying model to support a plane (or a curved surface) for the branch, separate from the silhouette. Our method has limitations.

Tree example	Stroke count	Time (mins)	Element count
poplar	7	2	53,000
pine	7	2	32,000
christmas tree	41	3	40,000
palm	29	3	3800
willow	43	4	35000
eucalyptus	35	10	1294000
fern forest	280	30	90000
tree stand	220	30	440000

Table 7.1: The number of strokes required by the user, the modelling time and the resulting complexity (leaf and branch count) of each tree model.

In our prototype we only support inferring one sub-level of branches from a silhouette, with those branches all attaching to one parent trunk (monopodial structure). However some tree architectures are sympodial (like the LEUWENBERG architecture from Figure 7.2) and exhibit a more recursive bifurcating architecture. If the user has an architecture like this in mind, then the structure inferred from the silhouette will not match his expectation and will require extensive changes. As future work we would like explore methods of inferring more complex architectures from the initial silhouette, perhaps by using more structural information from the geometric skeleton, or allowing the user to draw an example before drawing the silhouette. It would be interesting to infer more than just one sub-level of branches from the initial silhouette.

We currently only use one silhouette per crown shape, and infer the other branches which would be close to parallel to the view direction. A natural extension would be to allow the user to specify another silhouette from another viewpoint (most naturally at 90°to the first). This would give greater control over the 3D appearance of the tree. We experimented with allowing the user to sketch many silhouettes from different positions around a

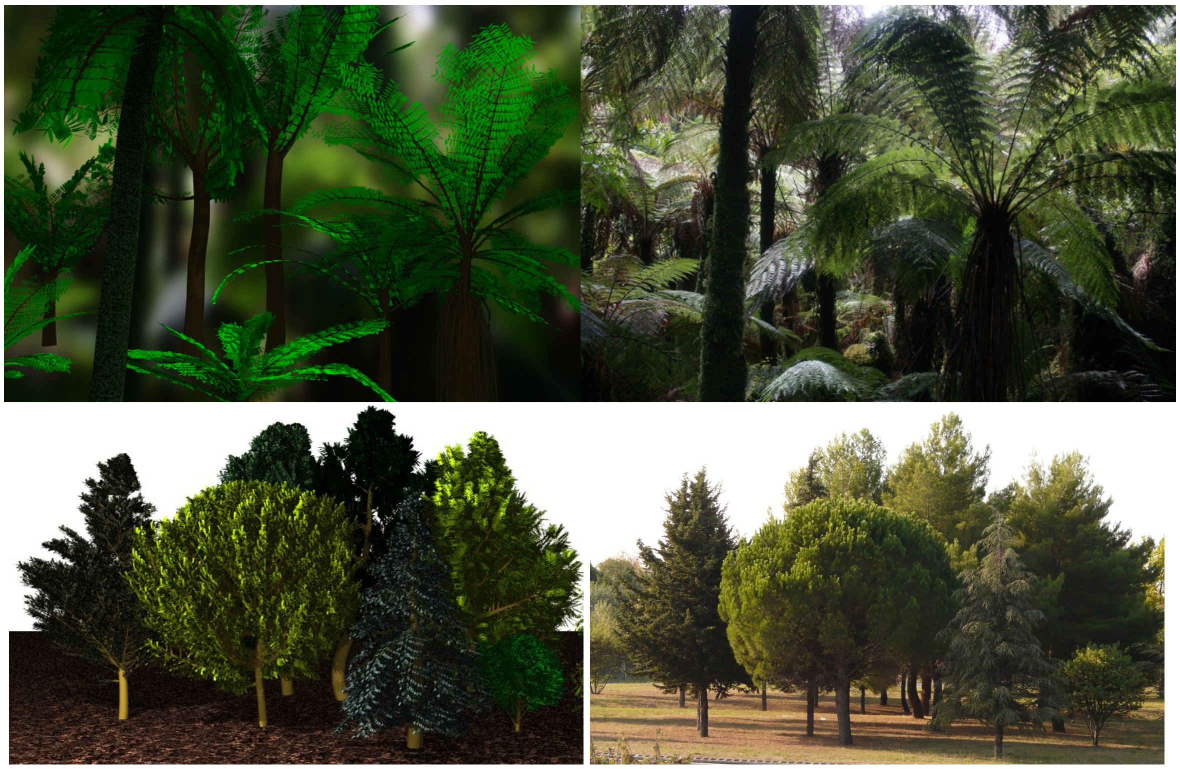


Figure 7.19: **(top)** Application of our method to quickly sketch a fern forest. **(bottom)** A stand of trees. Both scenes were sketched over photographs (right) and took thirty minutes each to model.

crown, and inferring the crown shape using an interpolated surface of revolution. However we found that using many silhouettes was counter intuitive. For example if a new silhouette was sketched at say only a 20°offset from the previous one, and was much smaller - it would introduct a large discontinuity in the crown shape that probably wasn't the users intention. Many such silhouettes created a highly discontinuous shape that became hard to edit. Instead limiting the crown to just two orthogonal planes or four planes at 45°increments, and replacing whichever silhouette was nearest in angle proved a more natural seeming edition process. However the issue of how to incorporate already existing inferred branches when new silhouette information is supplied has not been investigated. The simplest approach would be to just delete and recreate the branches in the sector that the silhouette corresponds to, but perhaps this would not be sophisticated enough.

We restrict branches to be planar. This is an advantage when rendering the tree as billboards can be easily constructed, however obviously real trees do not adhere to this restriction. We could automatically infer a 3D spiral shape for branches using the same constant 3D curvature assumption as [OOI05], or one could imagine improving the system to allow over-sketching of branches from other viewpoints in-order to add complex 3D shapes. This would be a useful line of future research.

Initial leaf density is set by the silhouettes drawn at the scale one level above the leaf (i.e. the number of leaves automatically inferred from the silhouette) - it can then be increased by adding further leaves using the branch positioning process. We offer either doubling or adding a constant number of leaves (or branches) to the currently selected branch via a keypress. However an approach such as spraying leaves [ZS07] might be more natural for fine tuning the density once the tree is in 3D. You could imagine wanting to increase or decrease the leaf density in a local area from a global view without having to zoom in.

The order of edition presented (starting at the coarse scale and moving directly to the finest) is the simplest case of using the software. In reality the user will want to move back and forth between scales, copy some styles to sub-sets of branches and copy styles between scales. Although the operations we define support this - designing an interface which allows the user to apply these operations simply is a complex task that our prototype doesn't adequately solve. For example once a tree has some branches in 3D the selection of subsets of branches becomes difficult. 2D lassoing techniques may not be adequate because they will have to take some degree of depth into account. Also the issue of how much detail to display becomes important. For example we render the initial guideline sub-silhouette shape for the user to over-sketch with the next level silhouette. But when there are hundreds of such shapes on the screen at once, all in arbitrary 3D positions - they become useless as a guide tool (the same argument applies to visualising branches in order to redraw them). A technique for rendering the tree which takes into account the relative importance of the available branches and guides would be useful. One could imagine a screen space based technique which identifies likely candidates for redrawing and renders these solidly, with everything further behind or at large angles to the view direction rendered almost transparently.

It is worth discussing the 'portability' of the interface. The concept of sketching using silhouette and branch strokes at multiple scales and from multiple viewpoints is an attractive one because of its close mapping to the tradition artist approach, making it simple to understand. Could the same approach be used to parameterize an underlying model different from the hierarchy of planes model we chose to implement (for example using underlying L-Systems or procedural models like [WP95, Kru99])? We think it could be, however there would be significant differences. We have chosen an explicit representation for the tree geometry - so the hierarchical position and geometry of each branch are known. This means we can place a branch on a tree exactly where the artist requires. However methods like [WP95] are pseudo-statistical - which means we cannot control exactly where a branch will be positioned. This means the tree can be represented very compactly as a set of parameters for the model, and so it is ideally suited for producing a forest of similar but different trees. If we were to adapt our approach for such an underlying representation then the fine level user control over positioning would be irrelevant. We could measure the parameters required for [WP95] from the sketch similar to the approach we outlined for hair (chapter 5), and then generate a large number of similar trees. However no one tree would exactly match the initial example. This could be an advantage or a disadvantage depending on the application. The same argument applies to L-Systems, except that here the problem is even harder as form is an emergent property of an L-System. Some work has already been done on shaping the result of L-Systems [PJM94, APS08]. The most relevant is the sketching interface of [APS08], which pre-defines some L-System templates and then measures parameters for these templates from a fixed viewpoint sketch. One could imagine a similar approach working for our interface - but the set of predefined templates would have to be large to match the freedom of being able to sketch any shape.

7.7 Positioning our work

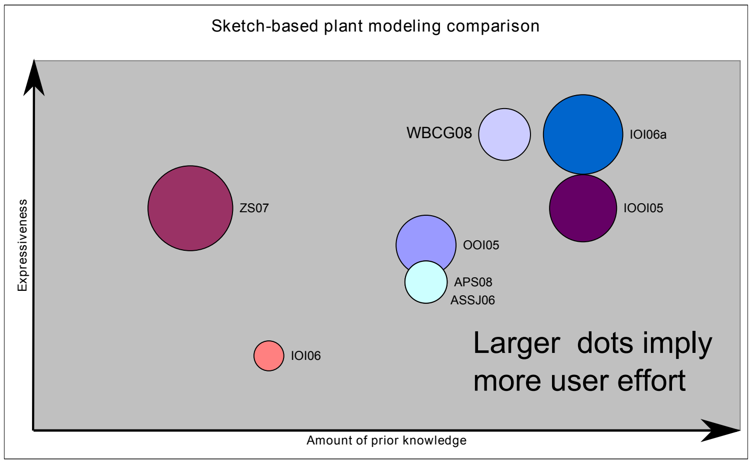


Figure 7.20: Comparing our tree sketching system [WBCG08] with others, according to the categorisation discussed in subsection 2.3.1. Amount of prior knowledge incorporated increasing on x-axis. Expressiveness increasing on y-axis. Larger dots indicate greater amounts of user input required.

In subsection 2.3.4 we presented a chart which categorised the prior work with respect to the categorisation discussed in subsection 2.3.1. Figure 7.20 is the same chart with a data point added representing our evaluation of our tree sketching work [WBCG08]. The most similar previous works are the tree sketching system by Okabe *et al.* [OOI05] which allows the sketching of tree structure directly, and the construction line plant sketching system of Anastacio *et al.* [ASSJ06, APS08] (which doesn't support sketching trees). Our system allows the sketching of direct structure as in [OOI05], but it also allows the rapid construction of many branches at many scales using the structure from silhouettes idea - thus we consider it more expressive than [OOI05] and place it higher on the y-axis. We also support more prior knowledge from the field of botany than [OOI05] in the form of different phyllotaxy arrangements which also respect user supplied position constraints, hence we are further to the right on the x-axis. Finally we think that our system requires less user input for similar types of trees and so the size of the data point is smaller - indicating less user effort required.

CHAPTER 8

Extension: Sketching terrain

As described in subsection 2.3.5, traditional terrain modelling systems work from a top-down viewpoint, usually by applying painting operations on an image which is then interpreted as a height-field. However when we observe a terrain, it is usually from the viewpoint at ground level. It is much easier to distinguish changes in shape and height by viewing them from such vantage point (see Figure 8.1), as opposed to mentally converting pixel intensities into height. This is the way an art director would imagine a terrain when storyboarding a scene in a game or a film. An approach of sketching a terrain from a ground level viewpoint by specifying silhouette information maps naturally to this way of thinking about terrain. The problem with this approach is in inferring the depth of the silhouette, and shaping the deformation of the ground to meet the silhouette. The silhouette alone doesn't provide any depth information, nor does it specify the shape of the terrain along the view-direction.

The two previous landscape sketching systems (subsection 2.3.5) address depth determination by requiring the user begin and end his stroke on the existing landscape, and determining the depth through intersection. This approach is fine when this point is visible, but often it is not. Landscapes can be composed from the back to the front using such an approach, but it isn't then possible to add large mountains behind existing ones. Also when close to the ground, points in the far distance occupy points very close together in screen space due to foreshortening - so for this technique to be accurate the camera must be lifted far above the ground plane and aimed down at it, which is no longer a natural viewpoint.

In [WI04] the profile shape of the deformation is inferred from the silhouette. However often the silhouette shape is not representative of the profile shape and the user would like more control. In both previous works a deformation has an arbitrary area of effect, so different deformations may not blend together smoothly.

We are currently developing an approach for sketching landscapes which addresses some of these shortcomings (see Figure 8.3). The basic primitives for designing the terrain will be a set of 3D space-curves defined

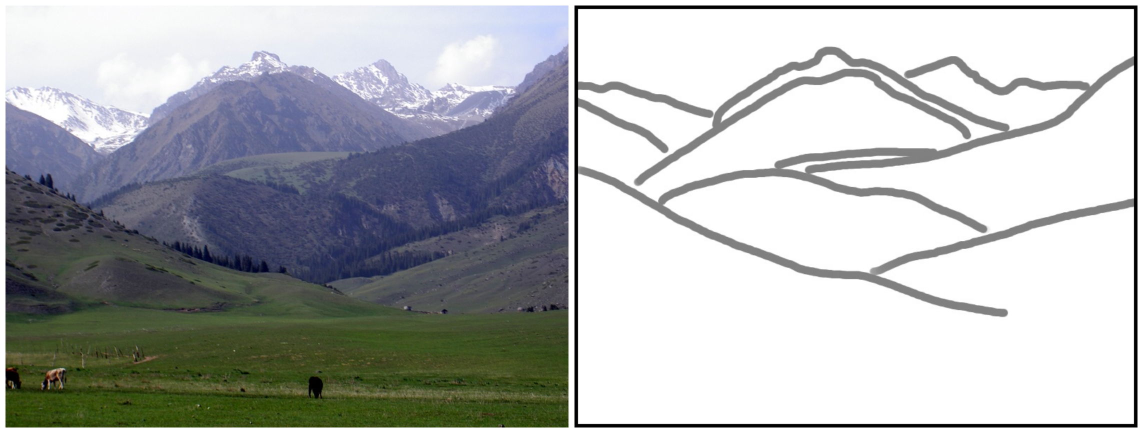


Figure 8.1: The goal of our system would be to infer a plausible terrain based on overlapping silhouettes sketched from a ground level viewpoint. Image courtesy of Michael Bravo.

by the user. These curves will have two types, a silhouette curve, and a profile curve. The silhouette curve is sketched by the user from an arbitrary viewpoint onto a view-plane parallel sketching plane (thus planar to begin with). Depth will be handled in one of three ways:

- The initial depth may be determined by intersection with the ground where this is possible. As for previous systems.
- When the stroke does not intersect the existing terrain, it must start or end near an existing silhouette, forming a T-junction. From perceptual studies (section 1.1) we know that T-junctions can be used to infer relative depth. The depth of the existing feature will constrain the possible depths available to the new silhouette. Given the constraints either a simple heuristic will be used to initially position the curve (for example, place it halfway between two existing features) or a more complex depth embedding based on relative weights could be used (similar to the 1D mass-spring system used in [KH06]), where larger screen space height differences translate to larger z-differences.
- Users could optionally move and redraw silhouette curves from an overhead viewpoint, this allows them fine control over depth positioning and silhouette shape along the view direction, and represents a hybrid approach between the traditional painting approach and sketching from a realistic viewpoint.

The profile shape of the deformation will also be specified by the user from the same viewpoint. Similar to the approach we used for folds in garments section 3.4 the user draws a side view of the profile he desires, but onto the existing viewpoint. Perceptually it is not difficult for the user to imagine the profile line meeting the silhouette line at $90°$(along the view direction), and it saves breaking the workflow to move the camera or draw in a separate viewport. A similar approach was used in Cross-sketch[ASN07] (Figure 8.2), but for sketching specific objects. However in our case the sketched profile does not represent the entire profile of the terrain feature. It just indicates the local deformation close to the silhouette line, the entire profile shape emerges by solving a poisson equation (explained next).

So given a set of silhouette lines placed in 3D by a process of rough initial depth guess followed by optional user deformation from a top view point, and a set of local profile shapes defined orthogonal to these silhouettes, how do we form a height-field to represent the terrain? We propose an approach again similar to the one we used for forming garments from sketches, but based on a much faster and more sophisticated poisson equation solver, presented by Orzan *et al.* [OBW$^+$08]. The final shape of the terrain is formed via a fast diffusion process

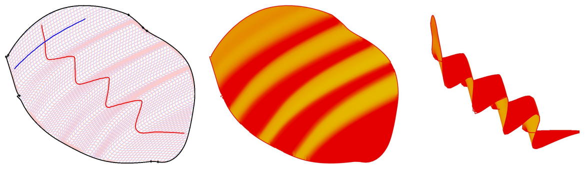


Figure 8.2: Cross-sketch[ASN07] also uses the idea of the user specifying lines which intersect at 90°in the plane which the view direction belongs to.

operating on a discretized grid of height values, which smoothly links features together while respecting the hard constraints on positions and profile shape gradients supplied by the curves. A useful analogy is to imagine a set of coat hanger wires fixed in space, following the silhouette and profile shapes, and then draping a thin cloth sheet over these wires to form the landscape surface. The current prototype is only in the early stages, but the results look promising (Figure 8.3).

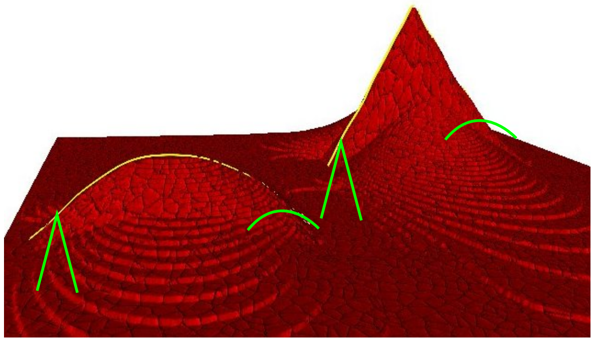


Figure 8.3: An early prototype exploring this idea of a terrain interface. The yellow strokes are the feature silhouettes, the green strokes are the profile shapes which although drawn on the same plane, are treated as orthogonal to the silhouette plane. In this prototype the profile shapes are linearly interpolated along the silhouette. The whole terrain is defined by these strokes, with the surface model emerging from a fast poisson equation solver.

Summary of Part III

In this part we presented sketch-based modelling systems for natural phenomena when there is no pre-existing structure to annotate. We used our prior knowledge of the nature of the shape being modelled in order to incrementally infer supporting supporting structure from sketch silhouettes. We also addressed the multi-scale nature of the phenomena. In the case of clouds large scale shape was inferred from the sketched silhouettes and fine scale details were set via a procedural noise method. In the case of trees the user could sketch detail at any scale within the tree and thus had a large degree of control over the resulting shape. In both cases sketches could be made from arbitrary viewpoints. We ended the part by outlining some work in progress on modelling terrain from sketched silhouettes.

Résumé en Français

Dans cette partie, nous avons présenté des systèmes de sketching pour la modélisation de phénomènes naturels quand il n'y a pas de structure pré-existante à annoter. Nous avons utilisé notre connaissance de la nature de la forme à modéliser pour en déduire progressivement la structure d'appui à partir du dessin de silhouettes. Nous avons également abordé la nature multi-échelle des phénomènes. Dans le cas des nuages, la forme à grande échelle a été déduite des silhouettes esquissées, et à une échelle plus fine, les détails ont été définis par le biais d'une méthode de bruit procédurale. Dans le cas des arbres, l'utilisateur peut esquisser les détails à toutes les échelles de l'arbre et a donc un grand degré de contrôle sur les formes générées. Dans les deux cas, des croquis peuvent être faits à partir de points de vue arbitraires. Nous avons conclu cette partie en soulignant certains travaux en cours sur la modélisation de terrain à partir de sketching de silhouettes.

Conclusion

At the beginning of this thesis we introduced the concept of developing sketch-based interfaces that exploit our prior knowledge of the phenomena to be modelled, similar to the way we ‘construct’ our mental images of the things we perceive visually. This approach can be considered a ‘top-down’ approach as opposed to the ‘bottom-up’ approach of general shape modelling, which exploits only perceptual shape cues used by our visual system to infer shape. We also chose to model our interfaces on the traditional working approaches used by artists, defining rough global detail and then allowing local refinement where necessary. To illustrate this approach we designed specific interfaces for modelling clothing with folds, hairstyles, clouds and trees. We also considered applying the same ideas to terrain modelling. All the approaches used domain specific knowledge to aid in the construction of shape, but in varying amounts. In Part II of the thesis we used a pre-defined supporting surface to help position the inferred shapes (clothes around a body, hair around a head) in 3D from a fixed viewpoint and varied the amount of prior knowledge used in each case (ranging from rough rules of thumb about the fit of a garment, to a detailed physically-based model for hair on a head). In Part III of the thesis the supporting surface was created incrementally as part of the modelling process, structure was inferred from the silhouettes and information could be added from multiple viewpoints.

In each section, the advantages and disadvantages of the specific implementation were discussed. In this conclusion chapter, I would like to take a higher level view and synthesize some general observations based on the specific examples. But first I will address where sketching interfaces fit in with modelling systems today.

Where are all the sketching interfaces?

If sketching is such a useful modelling approach, then why are the most popular modelling tools (3D Studio, Maya) still based on widgets? Where are the sketch-based modelling systems?

First, this thesis doesn’t claim that sketch-based tools are a replacement for the tradition set of tools that modern virtual artists are trained in. Sketching is just one part of a larger toolset, and should be used when appropriate. Sketching is most useful when specifying initial outlines, dimensions and positioning. But once something has been inferred from a sketch, fine scale adjustments may be more appropriate using a traditional widget. For example once you have sketched a lobe on a cloud and you like its shape, but want it positioned slightly to the left, its much easier to slide it along the surface with a movement tool than to resketch a new position. This is especially true when you are happy with the initial shape but not the position. One of the strengths of sketching is that you can simultaneously define the shape and the position, but when you wish to change one aspect without affecting the other more traditional approaches become more appropriate.

Second, traditional modelling tools are designed to be as general as possible. This is why for specific tasks such as modelling trees there are a variety of modular add-ons such as XFrog [Gre08]. These add-ons are often procedural in nature, but in the future specific sketch-based interfaces such as the ones outlined in this thesis could become more widely used. They don't require the user to be instructed in the myriad of parameters available in the model before being able to obtain pleasing results. A sketch-based interface can hide the underlying complexity until the user exceeds the constraints of the interface and wishes to set parameters manually. If inferred parameters are displayed in an easy to understand manner, then the sketch-based interfaces can help the user learn the impact of the many available settings.

Finally, it is only recently that the cost of pen based input technology has fallen to levels that could make such interfaces an appealing alternative. The mouse is a terrible tool to use for sketching, but it is cheap. Pen based input has been possible since the light pen was invented - but only recently have modern alternatives become 'consumer grade' technology, with the benefits in cost reduction that mass production brings. This is especially due to the push from companies like Microsoft to make tablet based PCs a reality. With the massive growth in small devices like phones, PDAs and game systems (Nintendo DS) that use pen input, and with touch screen workstations like the Microsoft Surface on the horizon, sketch-based input methods have surely come of age. The success of the CAD style sketching tool Google Sketch-up [Goo08] is a promising start.

General methodology

Can we identify a general methodology for creating sketched based interfaces for complex models where some prior knowledge is available? Certainly there are some common themes:

Mapping to a procedural model Does an effective procedural model already exist for the thing you are modelling? Perhaps the parameters for this model can be extracted from a sketch. Which parameters have the largest effect on the resulting model? These are the ones which are most important, some of the others could perhaps be fixed (for example - mass per unit volume was fixed to an average natural value in the case of hair, as was radius. Length and curvature were the most important parameters).

Simplifying assumptions Choose your assumptions carefully. You can often reduce the complexity of the problem by a large margin, while only slightly reducing the variety of results possible. For example deciding that layers of clothing may not self-overlap, or in the case of hair - limiting the drawing to a side projection only may be adequate for a first modelling pass. These restrictions could always be relaxed in a second pass if more asymmetry is desired.

Non-intrusive interface All tools which model themselves on the traditional pencil and paper workflow should try and hide the details of the interface, so as not to interrupt the user while they concentrate on the task in hand. For example minimizing the number of mode switches required, and making sure that when they are required they occur at a natural pause in the thought process. Switching from the front to back view of the garment doesn't interrupt the workflow, as the user naturally refocuses their attention at this stage - but requiring selection of a breakpoint and pressing the deletion key would be inappropriate for breakpoint deletion - hence the use of a gesture, which keeps the user focus on the virtual page.

Sketching vs annotation The sketching systems we presented not only illustrate sketching, but also the annotation of a 3D shape serving as a support for the sketch (for clothes and hair - body and head models, for clouds and trees the structure being modelled becomes the support). Relying on the 3D information from the supporting structure in addition to the prior knowledge often makes sketching from a single viewpoint sufficient and thus makes the process much quicker, although other views can easily be added, for example to model the back and front of a garment, or a non-symmetric hairstyle. Further support shapes for annotation may be generated during the process. For example the initially generated garment surface served as a further support model for sketching folds. The method of successive 'coatings' of a base surface could be useful for other situations (such as sketching vegetation or features onto terrain).

Sketching structure directly or inferring structure Both approaches are useful at different times. For example with our tree sketching system both the artist and the botanist expressed a preference for direct control over the primary level branches when they had a strong idea about initial shape, but liked the ease of rapidly specifying many branches at lower levels using the structure from silhouettes idea. If possible a sketching system should support both approaches.

Fidelity to the sketch How closely does the thing being modelled have to match the sketch? If the phenomena is visually complex, it could be that the sketched elements can be considered only an example of the shape required. This eases the problem of constructing a shape from the sketch. For example in the case of hair we created strands which were similar too but not exactly the same as the sketched examples. This isn't a problem, as a head of hair has so many strands that the exact shape of any one is hard to discern.

Fixed or free viewpoint If it's possible to identify a way to infer 3D from only one viewpoint, then a fixed viewpoint system is simpler to use in the initial stages of modelling. However as users gain experience they may wish to take more control over precise positioning of elements. For example in the case of trees adding a branch at a specific azimuthal angle by rotating the camera parallel to the desired drawing plane and drawing the new branch. It might be that you only wish to allow the user to define a stroke at some fixed angle to the view direction, in which case why not allow the user to sketch this from the same viewpoint? For example when drawing folds on garments the user drew a conceptual line representing the size and orientation of the fold profile over each end of the fold line, an approach that we re-used for sketching terrain, and that was also used in [ASN07]. If possible offer a moveable camera as an advanced feature, available to those who need it, but be aware that allowing a moveable viewpoint raises new issues related to visual complexity, discussed next.

Visual complexity With complex phenomena, especially those created incrementally, from multiple viewpoints and at multiple scales, the user could be supplying tens or hundreds of strokes. Often he may wish to return to a stroke and redraw (restate) it. This means that the strokes should be visible in some way. However when there are so many strokes displaying them all at once soon becomes unfeasible. The user becomes overwhelmed by the visual complexity and cannot accurately determine which strokes represent what anymore. This became an issue with our tree sketching system. To address this problem we created menus for the user to select what should be visible, grouping by line type, hierarchy depth and so on. These options should be available to the user - but they add complexity to the interface and make it harder to use. More useful would be a system to automatically decide visibility. For example by considering the current viewpoint and fading (using transparency) those strokes unlikely to be relevant to the artist (for example strokes defined at higher scales and strokes not close to parallel with the current viewport).

Future work

In addition to the future work discussed in the individual chapters, I would like to propose future work on some more general themes.

Composition of sketching interfaces

Usually a sketching interface is focused on modelling just one object. However when modelling a whole scene, especially an outdoor scene like a landscape, the placement of objects relative to each other becomes important. Little previous work has considered how to compose a variety of related sketching interfaces together. Harold [CHZ00] is a system which allows a child like drawing of a landscape to be explored in 3D. I consider the interfaces proposed in this thesis for modelling natural phenomena such as clouds, trees and terrain as good candidates for being composed together to enable the sketching of whole realistic landscapes, allowing the artist to model his environment in real-time. Such a project is a difficult engineering task, as landscapes are large issues such as memory consumption and level of detail become important, however it should be possible. An interesting avenue of research would be to define a whole forest by considering it as being the level of scale above an individual tree. By sketching the forest boundary onto a previously sketched terrain surface, and modelling a few example trees, whole forests of individual trees could be generated by extrapolation or interpolation of the example tree parameter sets. For a compact representation of such a forest the sketching interface should support parameterizing fully procedural tree models such as [WP95] - here we would be sacrificing exact fidelity to the users strokes for a whole range of models which are different but look similar to the given examples - just as we did for the hair sketching interface.

Educational application of sketching interfaces for procedural models

We've put forward the argument that a sketching interface is useful because it hides the complexity of any underlying model. However an expert in the underlying model will be able to manipulate that model more effectively by setting the parameters directly rather than using a sketching interface that may make some simplifying assumptions. So why not use the sketching interface to train people to become experts in the underlying model? One could imagine an interface that allows sketching in one window, while in another displaying in an intuitive and educational manner the changes made to the procedural model underlying the interface. In this way the user quickly determines which parameters are most relevant to manipulate the shape in the way he desires.

Supporting a greater range of sketched line types

In section 1.2 I presented a summary of the types of lines an artist might draw. Currently the only sketch-based modelling system which supports more advanced line types like internal silhouettes (occluding contours) is [KH06]. Why aren't more systems supporting these advanced types of lines? I think one reason is that this is still a young field, and we will see more progress in this area. However there are a few reasons why sketch-based modelling support for such lines might never be widespread. A first reason is that so far the focus for sketch-based modelling is on rapid modelling of shape. The more subtle details that the user has to draw, the longer the whole process takes. A second reason is that it takes artistic skill to understand such line types, and to draw them in a way that is consistent. Even trained artists will draw inconsistent shape cues in a line drawing, and this ambiguity will make it hard for an algorithm to determine the intended shape. A system which supported such lines might not be appealing to those without much drawing skill or training.

Résumé en Français

Au début de cette thèse, nous avons introduit le concept de développement d'interfaces basées sur le sketching qui exploitent des connaissances a priori des phénomènes à modéliser, de la même manière que nous construisons les images mentales de ce que nous percevons visuellement. Cette approche peut être considérée comme ≪descendante≫, par opposition à l'approche ≪montante≫ utilisée généralement en modélisation. Elle exploite uniquement les indices perceptuels utilisés par notre système visuel pour déduire la forme observée. Nous avons également choisi de concevoir nos interfaces en nous basant sur les méthodes de travail traditionnelles utilisées par les artistes, partant d'un croquis général pour rajouter des détails locaux ensuite, si nécessaire. Pour illustrer cette approche, nous avons conçu des interfaces spécifiques pour la modélisation de plis des vêtements, de coiffures, de nuages et d'arbres. Nous avons également examiné la possibilité d'appliquer les mêmes idées pour la modélisation de terrain. Toutes les méthodes utilisées exploitent des connaissances propres au domaine considéré pour aider la construction de la forme, mais en quantités variables. Dans la Partie II de la thèse, nous avons utilisé une surface d'appui pré-définis pour déduire la position des formes en 3D à partir d'un point de vue fixe (vêtements autour d'un corps, cheveux sur une tête), et nous avons utilisé de plus en plus de connaissances (partant de règles rudimentaires sur l'adéquation d'un vêtement pour aller jusqu'à un modèle basé sur la physique dans le cas des cheveux). Dans la partie III de la thèse, la surface d'appui a été créée incrémentalement pendant le processus de modélisation; la structure a été déduite des silhouettes dessinées, et des informations supplémentaires ont pu être ajoutée à partir de plusieurs points de vue.

Dans chaque section, les avantages et les inconvénients de chaque implémentation spécifique ont été discutées. En conclusion de ce chapitre, je voudrais présenter une vue de plus haut niveau et synthétiser quelques observations générales fondées sur des exemples précis. Mais d'abord, je vais préciser à quel niveau les interfaces de sketching s'intègrent dans les systèmes actuels.

Où trouve-t-on les interfaces de sketching ?

Si le dessin est une approche de modélisation si pratique, alors pourquoi les logiciels de modélisation les plus populaires (3D Studio, Maya, etc.) sont-ils toujours basées sur des boites à outils? Où trouve-t-on des systèmes de modélisation basés sur le sketching?

Tout d'abord, cette thèse ne prétend pas que les interfaces de sketching doivent remplacer les outils traditionnels auxquels sont formés les artistes aujourd'hui. Le sketching n'est qu'un élément parmi une palette d'outils à utiliser au moment opportun. Le croquis est très utile pour préciser les premières lignes, les dimensions et le positionnement d'une forme. Mais une fois que l'information de base a été déduite à partir d'un croquis, les ajustements à une échelle plus fine peuvent s'effectuer de manière plus appropriée en utilisant des outils traditionnels. Par exemple, une fois que vous avez tracé un lobe sur un nuage, et que vous êtes satisfait de sa forme mais que vous voulez le déplacer légèrement à gauche, il est beaucoup plus facile d'utiliser un outil de translation plutôt que l'outil de sketching pour le faire glisser vers sa nouvelle position. Ceci est souvent vrai lorsque vous êtes satisfait de la forme initiale, mais pas de la position. Une des forces du croquis est que vous pouvez définir en même temps la forme et la position, mais quand vous voulez changer un aspect sans affecter l'autre, les approches plus traditionnelles sont alors plus adaptées.

Ensuite, les outils de modélisation traditionnels sont conçus pour être aussi généraux que possible. C'est pourquoi, pour des tâches telles que la modélisation d'arbres, il existe une variété de modules tels que Xfrog [Gre08]. Ces modules sont souvent de nature procédurale, mais dans le futur des interfaces spécifiques basées sur le sketching telles que celles décrites dans cette thèse pourraient être plus largement utilisées. Elles ne nécessitent pas que l'utilisateur maîtrise les nombreux paramètres disponibles du modèle avant d'être en mesure d'obtenir de bons résultats. Le sketching peut cacher la complexité sous-jacente du modèle jusqu'à ce que l'utilisateur soit bloqué par les contraintes de l'interface et souhaite modifier les paramètres manuellement. Si les paramètres sont affichés d'une manière suffisamment comprehensible, alors l'interface de sketching peut aider l'utilisateur à appréhender l'impact des nombreux paramètres disponibles.

Enfin, ce n'est que récemment que le coût des tablettes graphiques et autres stylets a chuté à des niveaux qui font de ces interfaces une alternative intéressante. La souris est un outil difficile à utiliser pour le dessin, mais il est bon marché. Les périphériques à base de stylet lumineux sont appropriés, mais ce n'est que récemment que des alternatives modernes de cette technologie deviennent grand public, avec les avantages que la réduction des coûts de production de masse apporte. C'est notamment en raison de la poussée de compagnies telles que Microsoft pour faire des PC à tablette une réalité. Avec la croissance massive des petits appareils comme les téléphones, les PDA et les consoles de jeu portables (Nintendo DS) qui utilisent le stylet et les écrans tactiles (tels que le Microsoft Surface à venir), l'ère des interfaces de sketching est venue. Le succès d'outils de CAO comme Google Sketch-up [Goo08] est un début prometteur.

Méthodologie générale

Peut-on identifier une méthodologie générale pour la création d'interfaces de sketching de modèles complexes, où une certaine connaissance est disponible? Certes, il y a des éléments communs :

Application d'un modèle procédural Existe-t-il déjà un modèle procédural pour l'objet que vous modélisez? Peut-être est-il possible d'extraire les paramètres de ce modèle à partir d'un croquis? Quels sont les paramètres qui ont le plus d'effet sur le modèle? Ces derniers sont les plus importants, alors que d'autres paramètres pourraient être fixés (par exemple dans le cas des cheveux, la masse par unité de volume a été fixée à une valeur moyenne, idem pour leur rayon; la longueur et la courbure étant les paramètres les plus importants).

Hypothèses simplificatrices Choisissez soigneusement vos hypothèses. Il est souvent possible de réduire grandement la complexité du problème tout en ne réduisant que légèrement les solutions possibles. Par exemple, décider que des couches de vêtements ne peuvent pas se chevaucher ou, dans le cas de cheveux, limiter le dessin à un côté de projection seulement peut être suffisant pour une première passe de modélisation. Ces restrictions peuvent ensuite être assouplies dans une seconde passe si l'utilisateur souhaite un résultat plus asymétrique.

Interface non-intrusive Tous les outils conçus sur le modèle traditionnel de travail à base de crayon et papier devraient essayer de masquer les détails de l'interface afin de ne pas détourner l'utilisateur de sa tâche : le dessin à la main. Par exemple en réduisant le nombre de changements de modes requis et en faisant en sorte que, lorsque nécessaires, ils interviennent naturellement dans le processus. Passer de la vue avant à arrière du vêtement n'interrompt pas le travail car l'utilisateur recentre naturellement son attention à ce stade. Mais exiger la sélection d'un point de couture et la pression d'une touche pour le supprimer serait inapproprié. L'utilisation d'un geste qui maintient l'utilisateur concentré sur la page virtuelle est alors requis.

Sketching versus annotation Les systèmes que nous avons présentés illustrent non seulement le sketching, mais aussi les annotations d'une forme 3D servant de support au croquis (pour les vêtements et les cheveux, le corps et la tête; dans le cas des nuages et des arbres, la structure devient le support). S'appuyer sur les informations de structure en plus de connaissance à priori sur le modèle permet souvent de dessiner à partir d'un seul point de vue, et donc rend le processus beaucoup plus rapide, bien que d'autres points de vue peuvent être facilement ajoutés, par exemple pour modéliser l'arrière et l'avant d'un vêtement, ou l'asymétrie d'une coiffure. D'autres formes servant de support pour l'annotation peuvent être générés au cours du processus. Par exemple, la surface du vêtement initialement produite sert ensuite de nouveau support pour esquisser les plis. La méthode de raffinements successifs d'une surface de base peut être utile dans d'autres situations comme le sketching de végétation ou d'aspérités sur un terrain.

Dessiner la structure directement ou la déduire Les deux approches sont utiles à des moments différents. Par exemple, avec notre système de sketching d'arbre, l'artiste et le botaniste expriment tout deux une préférence pour le contrôle direct du niveau primaire de branches pour lequel ils ont une forte idée de la forme initiale, mais aiment la facilité avec laquelle ils peuvent obtenir rapidement de nombreuses branches aux autres niveaux en utilisant l'idée d'inférer leur structure à partir de silhouettes. Si possible, un système de sketching doit supporter les deux approches.

La fidélité au croquis Dans quelle mesure les objets ont-ils été modélisés conformément au croquis? Si le phénomène est visuellement complexe, les éléments tracés peuvent être considérés comme des exemples de la forme requise. Cela facilite le problème de la construction d'une forme à partir d'un croquis. Par exemple, dans le cas des cheveux, nous avons créé des mèches qui étaient semblables, mais pas exactement, aux exemples tracés. Ce n'est pas un problème puisqu'une chevelure a beaucoup de mèches et que la forme exacte de l'une d'elle est difficile à discerner.

Point de vue fixe ou libre S'il est possible d'identifier un système permettant de modéliser la forme 3D à partir d'un seul point de vue, celui-ci est plus simple à utiliser dans les premières étapes de la modélisation. Cependant, lorsque les utilisateurs acquièrent de l'expérience, ils peuvent souhaiter avoir plus de contrôle sur le positionnement précis des éléments. Par exemple, dans le cas des arbres, l'ajout d'une branche à un angle azimutal précis peut être obtenu par rotation de la caméra parallèlement au plan de dessin souhaité. Il se peut que vous vouliez permettre à l'utilisateur de tracer un trait à un angle fixe par rapport à la direction de vue. Dans ce cas, pourquoi ne pas permettre à l'utilisateur de faire son croquis à partir de ce même point de vue? Par exemple, lors de l'élaboration des plis sur les vêtements, l'utilisateur dessine une ligne conceptuelle représentant la taille et l'orientation du profil du pli à chaque extrémité de la ligne de pli; une approche que nous avons ré-utilisée pour le sketching de terrain, ainsi que dans [ASN07]. Si possible, n'offrir la possibilité de bouger la caméra que comme une fonctionnalité avancée à la disposition de ceux qui en ont besoin, mais sachez que le fait d'autoriser un point de vue mobile soulève de nouvelles questions liées à la complexité visuelle, décrites ci-dessous.

La complexité visuelle Avec les phénomènes complexes, en particulier ceux qui se créent progressivement à partir de plusieurs points de vue et à des échelles multiples, l'utilisateur peut fournir des dizaines ou des centaines de traits. Souvent, il souhaite revenir sur un trait pour le redessiner. Cela signifie que les traits doivent être visibles d'une manière ou d'une autre. Cependant, lorsqu'il y a beaucoup de traits, les afficher tous à la fois devient vite impossible. L'utilisateur est submergé par la complexité visuelle et ne peut plus déterminer avec précision ce que représente les traits. Ceci est devenu un problème pour notre système de sketching d'arbres. Pour remédier à ce problème, nous avons créé des menus pour que l'utilisateur puisse sélectionner ce qui doit être visible, en regroupement les traits par type, par hiérarchie de profondeur, et ainsi de suite. Ces options devraient être à la disposition de l'utilisateur, mais elles ajoutent une complexité à l'interface et la rendent plus difficile à utiliser. Il serait plus utile d'avoir un système pour décider automatiquement de la visibilité, par exemple, en examinant le point de vue courant et en estompant (à l'aide de la transparence) les traits qui ont peu de chances d'être pertinents pour l'artiste (par exemple les traits définis à plus haut niveau hiérarchique et les traits qui ne sont pas suffisamment parallèle au plan de la vue courante).

Perspectives

En complément des pistes de travaux futurs présentées dans les différents chapitres, je voudrais proposer quelques perspectives sur certains thèmes plus généraux.

Composition des interfaces de sketching

Habituellement, une interface de sketching est centrée sur la modélisation d'un seul objet. Toutefois, lors de la modélisation d'une scène complète, en particulier une scène d'extérieur comme un paysage, le placement des objets les uns par rapport aux autres devient important. Peu de travaux antérieurs ont examiné la façon de composer un ensemble d'interfaces de sketching. Harold [CHZ00] est un système qui permet à un dessin de paysage fait par un enfant d'être exploré en 3D. Je considère que les interfaces proposées dans cette thèse pour la modélisation de phénomènes naturels (tels que les nuages, les arbres et le terrain) sont de bons candidats pour être composés ensemble et permettre le sketching de paysages réalistes complexes, permettant ainsi à l'artiste de modéliser un environnement complet en temps-réel. Un tel projet est une tâche d'ingénierie difficile, car les scènes lourdes que sont les paysages posent des problèmes tels que la consommation mémoire et la gestion de niveaux de détail, néanmoins cela semble possible. Une voie de recherche intéressante serait de considérer une forêt complète comme étant l'échelle supérieure aux arbres individuels d'un même modèle multi-échelle. En dessinant le contour délimitant la forêt sur la surface d'un terrain déjà esquissé, ainsi que la modélisation de quelques exemples d'arbres, une forêt complète peuplée d'arbres distincts pourrait être générée par extrapolation ou interpolation des jeux de paramètres définissant les exemples d'arbre. Pour garder une représentation compacte d'une telle forêt, l'interface de sketching doit supporter la paramétrisation de modèles procéduraux d'arbres tels que [WP95]. Ici, les traits tracés par l'utilisateur ne seraient pas respectés exactement; tous les arbres seraient différents, mais similaires aux exemples donnés, comme nous l'avons fait pour l'interface de sketching de cheveux.

Applications éducatives des interfaces de sketching pour les modèles procéduraux

Nous avons mis en avant l'argument selon lequel une interface de sketching est utile car elle cache la complexité du modèle sous-jacent. Toutefois, un expert du modèle sous-jacent sera capable de manipuler ce modèle plus efficacement en ajustant ses paramètres directement plutôt qu'en utilisant une interface de sketching basée sur des hypothèses simplificatrices. Alors, pourquoi ne pas utiliser l'interface de sketching pour former les gens à devenir des experts du modèle sous-jacent? On pourrait imaginer une interface qui permette de dessiner dans une fenêtre, alors que dans une autre s'afficherait de manière intuitive et pédagogique les modifications apportées au modèle procédural sous-jacent. De cette façon, l'utilisateur déterminerait rapidement les paramètres les plus pertinents lui permettant de manipuler la forme comme il le souhaite.

Supporter une gamme plus large de types de traits

Dans la section 1.2, j'ai présenté un résumé des types de traits qu'un artiste peut dessiner. Actuellement, le seul système de sketching qui supporte des types de traits avancés, tels que les silhouettes internes (contours occlusifs), est [KH06]. Pourquoi n'y a-t-il pas plus de systèmes supportant des types de traits avancés? Je pense que l'une des raisons est que le domaine est encore jeune, et que nous allons voir à l'avenir d'avantage de progrès sur ce point. Toutefois, il existe quelques raisons pour lesquelles le support de traits avancés dans la modélisation par sketching risque de ne jamais être généralisée. Une première raison est que, jusqu'à présent, le sketching se focalise surtout sur la rapidité de modélisation de la forme. Plus l'utilisateur a de détails à dessiner et plus long est le processus global. Une deuxième raison est qu'il faut des compétences artistiques pour maitriser les types de traits et les utiliser de manière consistante. Même les artistes confirmés peuvent dessiner des traits produisant des indications de forme inconsistantes, et cette ambiguïté rendra difficile la reconnaissance de la forme par un algorithme automatique. Un système qui supporterait de tels traits pourrait ne pas être attrayant pour ceux qui n'ont pas beaucoup de compétence ou d'entrainement en dessin.

CONTENTS

BIBLIOGRAPHY

[ABaO08] Rifat Aras, Barkin Başarankut, Tolga Çapin, and Bülent Özgüç. 3d hair sketching for real-time dynamic & key frame animations. *The Visual Computer*, 24(7):577–585, July 2008.

[ABCG05] Anca Alexe, Loïc Barthe, Marie-Paule Cani, and Véronique Gaildrat. Shape modeling by sketching using convolution surfaces. In *Pacific Graphics,*, Short paper, Macau, Chine, 2005.

[AGB04] A. Alexe, V. Gaildrat, and L. Barthe. Interactive modelling from sketches using spherical implicit functions. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 25–34, New York, NY, USA, 2004. ACM Press.

[Ale05] John Alex. *Hybrid Sketching: A New Middle Ground between 2- and 3-D*. PhD thesis, Massachusetts Institute of Technology. Dept. of Architecture, 2005.

[APS08] Fabricio Anastacio, P. Prusinkiewicz, and Mario Costa Sousa. Sketch-based parameterization of l-systems using illustration-inspired construction lines. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Annecy, France, 2008.

[ASN07] Alexis Andre, Suguru Saito, and Masayuki Nakajima. Crosssketch: freeform surface modeling with details. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 45–52, New York, NY, USA, 2007. ACM.

[ASSJ06] Fabricio Anastacio, Mario C. Sousa, Faramarz Samavati, and Joaquim A. Jorge. Modeling plant structures using concept sketches. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 105–113, New York, NY, USA, 2006. ACM Press.

[AUK92] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proceedings of ACM SIGGRAPH 1992*, Computer Graphics Proceedings, Annual Conference Series, pages 111–120, August 1992.

[Aut08a] Autodesk. 3dsmax (software). `http://autodesk.com/3dsmax`, 2008.

[Aut08b] Autodesk. Maya ncloth (software). `http://autodesk.com/maya`, 2008.

[BAC$^+$06] Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. Super-helices for predicting the dynamics of natural hair. In *Proceedings of SIGGRAPH 2006*, August 2006.

[BAQ$^+$05] Florence Bertails, Basile Audoly, Bernard Querleux, Frédéric Leroy, Jean-Luc Lévêque, and Marie-Paule Cani. Predicting natural hair shapes by solving the statics of flexible rods. In J. Dingliana and F. Ganovelli, editors, *Eurographics (short papers)*. Eurographics, August 2005.

[Bau94] Thomas Baudel. A mark-based interaction paradigm for free-hand drawing. In *UIST 94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 185–192, New York, NY, USA, 1994. ACM Press.

[BCD01] D. Bourguignon, M. P. Cani, and G. Drettakis. Drawing for illustration and annotation in 3d. In A. Chalmers and T. M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 114–122. Blackwell Publishing, 2001.

[Blu67] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[BN04] Antoine Bouthors and Fabrice Neyret. Modeling clouds shape. In *Eurographics (short papers)*, august 2004.

[BNL06] Antoine Bouthors, Fabrice Neyret, and Sylvain Lefebvre. Real-time realistic illumination and shading of stratiform clouds. In *Eurographics Workshop on Natural Phenomena*, sep 2006.

[BNM$^+$08] Antoine Bouthors, Fabrice Neyret, Nelson Max, Eric Bruneton, and Cyril Crassin. Interactive multiple anisotropic scattering in clouds. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and games (I3D)*, 2008.

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, September 2005.

[BPCB08] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loïc Barthe. Matisse: Painting 2D regions for modeling free-form shapes. In Christine Alvarado and Marie-Paule Cani, editors, *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling, SBIM 2008, , 2008*, pages 57–64, Annecy, France, June 2008.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, 1991.

[CGA] CGAL, Computational Geometry Algorithms Library. `http://www.cgal.org`.

[CGL$^+$08] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[CHZ00] J. Cohen, J. Hughes, and R. Zeleznik. Harold: A world made of drawings, 2000.

[CK05] B. Choe and H-S. Ko. A statistical wisp model and pseudophysical approaches for interactive hairstyle generation. *IEEE Transactions on Visualization and Computer Graphics*, 11(2), March 2005.

[Clo00] Jean Clottes. Chauvet cave (ca. 30,000 b.c.). in heilbrunn timeline of art history. new york: The metropolitan museum of art. `http://www.metmuseum.org/toah/hd/chav/hd_chav.htm`, 2000.

[CMZ+99] Jonathan M. Cohen, Lee Markosian, Robert C. Zeleznik, John F. Hughes, and Ronen Barzel. An interface for sketching 3d curves. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 17–21, New York, NY, USA, 1999. ACM.

[Coa08] Drawing Coach. Light and fluffy cloud drawing. http://www.drawingcoach.com/cloud-drawing.html, 2008.

[CPCN05] Pedro Company, Ana Piquer, Manuel Contero, and Ferran Naya. A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computers & Graphics*, 29(6):892–904, December 2005.

[Cry07] Crytek. Crysis (software). http://crytek.com, 2007.

[CSDI99] L. Chen, S. Saeyor, H. Dohi, and M. Ishizuka. A system of 3d hairstyle synthesis based on the wisp model. *The Visual Computer*, 15(4):159–170, 1999.

[DAJ03] B. De Araujo and J. Jorge. Blobmaker: Free-form modeling with variational implicit surfaces. In *Proceedings of the 12th Portuguese Computer Graphics Meeting*, pages 17–26, October 2003.

[DC96] S. Douady and Y. Couder. Phyllotaxis as a dynamical self organizing process. part 1: The spiral modes resulting from time-periodic iterations. *Journal of Theoretical Biology*, 178:255–274, 1996.

[Dig83] P.J. Diggle. *Statistical analysis of spatial point patterns*. Academic Press, London, UK, 1983.

[DJW+06] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum (Eurographics'06 proc.)*, 25(3), September 2006.

[DKY+00] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tsuyoshi Okita, and Tomoyuki Nishita. A simple, efficient method for realistic animation of clouds. In *SIGGRAPH Proceedings*, pages 19–28, July 2000.

[Dur02] Frédo Durand. An invitation to discuss computer depiction. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 111–124, New York, NY, USA, 2002. ACM Press.

[DV70] Leonardo Da Vinci. *The notebooks of leonardo da vinci (2 vols)*. Dover, 1970.

[Edw01] Betty Edwards. *The New Drawing on the Right Side of the Brain*. HarperCollins Publishers Ltd, 2001.

[FHS07] Muhammad A. Fahiem, Shaiq A. Haq, and Farhat Saleemi. A review of 3d reconstruction techniques from 2d orthographic line drawings. In *Geometric Modeling and Imaging, 2007. GMAI '07*, pages 60–66, 2007.

[FS94] Adam Finkelstein and David H. Salesin. Multiresolution curves. *Computer Graphics*, 28(Annual Conference Series):261–268, 1994.

[FWTQ07] Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. Sketching hairstyles. In *Eurographics Workshop on Sketched-based Interfaces and Modeling*, 2007.

[Gar85] Geoffrey Y. Gardner. Visual simulation of clouds. In *SIGGRAPH Proceedings*, volume 19, pages 297–303, July 1985.

[GBYC01] Y. Guédon, D. Barthélémy, Caraglio Y., and E. Costes. Pattern analysis in branching and axillary flowering sequences. *Journal of Theoretical Biology*, 212(4):481–520, 2001.

[Goo08] Google. Sketchup (software). http://sketchup.google.com, 2008.

[Gra08] Daylon Graphics. Leveller (software). `http://www.daylongraphics.com/`, 2008.

[Gre08] Greenworks. Xfrog (software). `http://www.xfrogdownloads.com`, 2008.

[Ham72] Jack Hamm. *Drawing Scenery: landscapes and seascapes*. The Berkley Publishing Group, New York, 1972.

[HB97] J. A. Horst and I. Beichel. A simple algorithm for efficient piecewise linear approximation of space curves. In *Image Processing Proceedings.*, volume 2, pages 744–747, 1997.

[HBSL03] Mark J. Harris, William V. Baxter, Thorsten Scheuermann, and Anselmo Lastra. Simulation of cloud dynamics on graphics hardware. In *Graphics Hardware*, pages 92–101, July 2003.

[HMT00] S. Hadap and N. Magnenat-Thalmann. Interactive hair styler based on fluid flow. In *Computer Animation and Simulation '00*, pages 87–100, August 2000.

[HO70] Hallé and Oldeman. *Essai sur l'architecture et la dynamique de croissance des arbres tropicaux*. Masson, 1970.

[Hof98] Donald D. Hoffman. *Visual Intelligence, How We Create What We See*. W. W. Norton and Company Ltd., 1998.

[HZ00] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 517–526. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[IH02] Takeo Igarashi and John F. Hughes. Clothing manipulation. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 91–100, New York, NY, USA, 2002. ACM.

[IH06] Takeo Igarashi and John F. Hughes. Smooth meshes for sketch-based freeform modeling. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, New York, NY, USA, 2006. ACM.

[IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[IOI06a] T. Ijiri, S. Owada, and T. Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. In *In Proc. of Eurographics*, pages 617–624, 2006.

[IOI06b] T. Ijiri, S. Owada, and T. Igarashi. The sketch l-system: Global control of tree modeling using free-form strokes. In *Smart Graphics*, page 138, 2006.

[IOOI05] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3):720–726, July 2005.

[JJL07] Jr. Joseph J. LaViola. Sketching and gestures 101. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, New York, NY, USA, 2007. ACM.

[JS01] M Jones and R Satherley. Using distance fields for object representation and rendering. In *In Proceedings of the 19th Ann. Conf. of Eurographics (UK Chapter)*, pages 34–44, 2001.

[KH06] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 589–598, New York, NY, USA, 2006. ACM Press.

[KHR02] Olga Karpenko, John F. Hughes, and Ramesh Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21:585–594, 2002.

[KHR04] Olga Karpenko, John F. Hughes, and Ramesh Raskar. Epipolar methods for multi-view sketching . In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 167–173, August 2004.

[KN00] T-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Computer Animation 2000*, IEEE Computer Society, pages 121–128, 2000.

[KN02] T-Y. Kim and U. Neumann. Interactive multiresolution hair modeling and editing. *ACM Transactions on Graphics*, 21(3):620–629, July 2002. Proceedings of ACM SIGGRAPH 2002.

[Kon05] DH Kong. Photo: Flaming hair - side view. `http://www.flickr.com/photos/dhkong/51918474/`, 2005.

[Kra88] Samuel Noah Kramer. *History Begins at Sumer: Thirty Nine Firsts In Recorded History*. 1988.

[Kru99] Paul Kruszewski. An algorithm for sculpting trees. *Computers & Graphics*, 23(5):739–749, October 1999.

[Lev98] David Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 18(3):280–315, March 1968.

[LK01] D. W. Lee and H. S. Ko. Natural hairstyle modeling and animation. *Graphical Models*, 63:67–85, March 2001.

[LZ06] Joseph J. Laviola and Robert C. Zeleznik. Mathpad2: a system for the creation and exploration of mathematical sketches. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, New York, NY, USA, 2006. ACM.

[Mal05] S. Malik. A sketching interface for modeling and editing hairstyles. In *Sketch Based Interfaces and Modeling*, pages 185–194, Dublin, Ireland, 2005. Eurographics Association.

[MIAI05] X. Mao, S. Isobe, K. Anjyo, and A. Imamiya. Sketchy hairstyles. In *CGI '05: Proceedings of the Computer Graphics International 2005*, pages 142–147, Washington, DC, USA, 2005. IEEE Computer Society.

[MKKI02] Xiaoyang Mao, Kouichi Kashio, Hiroyuki Kato, and Atsumi Imamiya. Interactive hairstyle modeling using a sketching interface. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II*, pages 131–140, London, UK, April 2002. Springer-Verlag.

[MP96] R. Mech and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410, New York, NY, USA, 1996. ACM Press.

[MSA03] Katherine McCulloh, John Sperry, and Frederick Adler. Water transport in plants obeys murray's law. *Nature*, 421:939–942, Feb 2003. 10.1038/nature01444.

[MTPS04] Antoine Mcnamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 449–456, New York, NY, USA, 2004. ACM.

[Mun08] Randall Munroe. Xkcd comic. `http://xkcd.com`, 2008.

[Ney97] Fabrice Neyret. Qualitative simulation of connective cloud formation and evolution. In *Eurographics Workshop on Computer Animation and Simulation (SCA)*, 1997.

[NFD07] B. Neubert, T. Franken, and O. Deussen. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph.*, 26(3):88+, 2007.

[NISA07] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph.*, 26(3), 2007.

[OBW$^+$08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008)*, volume 27, 2008.

[ONNI06] Shigeru Owada, Frank Nielsen, Kazuo Nakazawa, and Takeo Igarashi. A sketching interface for modeling the internal structures of 3d shapes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, New York, NY, USA, 2006. ACM Press.

[OOI05] Makoto Okabe, Shigeru Owada, and Takeo Igarash. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum*, 24(3):487–496, 2005.

[PBS04] Sylvain Paris, Hector Briceño, and François Sillion. Capture of hair geometry from multiple images. *ACM Transactions on Graphics (Proceedings of the SIGGRAPH conference)*, 2004.

[Pea03] J. Peacock. *La Mode du XX siecle*. Thames & Hudson, 2003.

[PJM94] P Prusinkiewicz, M James, and R Mech. Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1994. ACM Press.

[Pla08] Planetside. Terragen (software). `http://www.planetside.co.uk`, 2008.

[PMKL01] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. *ACM Computer Graphics (Siggraph'01)*, 22(4):289–300, 2001.

[Pow98] W. F. Powell. *Drawing Trees*. Walter Foster Publishing, Laguna Hill, CA, 1998.

[Pra97] L. Prasad. Morphological analysis of shapes. *CNLS Newsletter*, (139), July 1997.

[PSNW07] Richard Pusch, Faramarz Samavati, Ahmad Nasri, and Brian Wyvill. Improving the sketch-based interface: Forming curves from many small strokes. *The Visual Computer*, 23(9-11):955–962, September 2007.

[RCDF08] Szymon Rusinkiewicz, Forrester Cole, Doug DeCarlo, and Adam Finkelstein. Line drawings from 3d models. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–356, New York, NY, USA, 2008. ACM.

[Rip79] B.D. Ripley. Simulating spatial patterns: dependent samples from a multivariate density. *Applied Statistics*, 28:109–112, 1979.

[RSW$^+$07] Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. Developable surfaces from arbitrary sketched boundaries. In *Eurographics Symposium on Geometry Processing*. Eurographics, 2007.

[Sch90] Philip J. Schneider. *An algorithm for automatically fitting digitized curves*, pages 612–626. Graphics gems. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[SD04] Tevfik Metin Sezgin and Randall Davis. Handling overtraced strokes in hand-drawn sketches. In *Making Pen-Based Interaction Intelligent and Natural*. AAAI Fall Symposium, 2004.

[SGG$^+$07] C. Smith, C. Godin, Y. Guédon, P. Prusinkiewicz, and E. Costes. Simulation of apple tree development using mixed statistical and biomechanical models. In *5th International Workshop on Functional-Structural Plant Models*, pages 31, 1–4, Napier, New Zealand, nov 2007.

[SKP06] Richard S. Smith, Cris Kuhlemeier, , and Przemyslaw Prusinkiewicz. Inhibition fields for phyllotactic pattern formation: a simulation study. *Canadian Journal of Botany*, 84:1635–1649, 2006.

[SLMB05] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, April 2005.

[SRDT01] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. *Computer Graphics and Applications, IEEE*, 21(3):53–61, 2001.

[SSEH03] Joshua Schpok, Joseph Simons, David S. Ebert, and Charles Hansen. A real-time cloud modeling, rendering, and animation system. In *Symposium on Computer Animation*, pages 160–166. Eurographics Association, 2003.

[Sut63] Ivan E. Sutherland. Sketchpad: A Man-Machine Graphical Communication System. In E. Calvin Johnson, editor, *Proceedings of the 1963 Spring Joint Computer Conference*, volume 23 of *AFIPS Conference Proceedings*, pages 329–346, Baltimore, MD, 1963. American Federation of Information Processing Societies, Spartan Books Inc.

[SWG05] Ryan Schmidt, Brian Wyvill, and Eric Galin. Interactive implicit modeling with hierarchical spatial caching. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 104–113, Washington, DC, USA, 2005. IEEE Computer Society.

[SWSJ05] R. Schmidt, B. Wyvill, M. Sousa, and J. Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 53–62, 2005.

[TB02] Andrzej Trembilski and Andreas Broßler. Surface-based efficient cloud visualisation for animation applications. In *Winter School of Computer Graphics (WSCG)*, pages 453–460, 2002.

[TBvdP04] Matthew Thorne, David Burke, and Michiel van de Panne. Motion doodles: an interface for sketching character motion. *ACM Trans. Graph.*, 23(3):424–431, August 2004.

[TCH04] Emmanuel Turquin, Marie-Paule Cani, and John Hughes. Sketching garments for virtual characters. In John Hughes and Joaquim Jorge, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. Eurographics, August 2004.

[TMPS03] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, 2003.

[TWB$^+$06] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John Hughes. A sketch-based interface for clothing virtual characters. *IEEE Computer Graphics & Applications*, 2006.

[TZW$^+$07] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3):87+, 2007.

[WBC07] Jamie Wither, Florence Bertails, and Marie-Paule Cani. Realistic hair from a sketch. In *Shape Modeling International*, June 2007.

[WBC08] Jamie Wither, Antoine Bouthors, and Marie-Paule Cani. Rapid sketch modeling of clouds. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*, 2008.

[WBCG08] Jamie Wither, Frederic Boudon, Marie-Paule Cani, and Christophe Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *submitted to Eurographics 2009*, September 2008.

[WBK$^+$07] Kelly Ward, Florence Bertails, Tae-Yong Kim, Stephen R. Marschner, Marie-Paule Cani, and Ming Lin. A survey on hair modeling: Styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 13(2):213–34, Mar-Apr 2007.

[WEH08] A. Wolin, B. Eoff, and T. Hammond. Shortstraw: A simple and effective corner finder for polylines. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2008.

[WGG99] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, 1999.

[WGL04] K. Ward, N. Galoppo, and M. C. Lin. Modeling hair influenced by water and styling products. In *International Conference on Computer Animation and Social Agents (CASA)*, pages 207–214, May 2004.

[WGL07] K. Ward, N. Galoppo, and M. Lin. Interactive virtual hair salon. In *PRESENCE: Teleoperators & Virtual Environments*, 2007.

[WH96] Lance R. Williams and Allen R. Hanson. Perceptual completion of occluded surfaces. *Computer Vision and Image Understanding: CVIU*, 64(1):1–20, 1996.

[WI04] Nayuko Watanabe and Takeo Igarashi. A sketching interface for terrain modeling. In *SIGGRAPH 2004 Posters*, New York, NY, USA, 2004. ACM.

[Wil97] John Willats. *Art and Representation, New Principles in the Analysis of Pictures*. Princeton University Press, 1997.

[WM07] Haixiong Wang and Lee Markosian. Free-form sketch. In *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 53–58, New York, NY, USA, 2007. ACM.

[WOQS05] Y. Wei, E. Ofek, L. Quan, and H-Y. Shum. Modeling hair from multiple views. In *Proceedings of ACM SIGGRAPH'05*, 2005.

[WP95] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1995. ACM Press.

[WWL07] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *UIST 07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2007. ACM.

[Wyv92] B. Wyvill. Building models with implicit surfaces. *Potentials, IEEE*, 11(3):23–26, 1992.

[XY01] Z. Xu and X. D. Yang. V-hairstudio: an interactive tool for hair design. *IEEE Computer Graphics & Applications*, 21(3):36–42, May / June 2001.

[Yan01] Steven Yantis, editor. *Visual Perception, Essential Readings*. Psychology Press, 2001.

[Yu01] Y. Yu. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics'01*, pages 295–304, October 2001.

[ZHH96] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 163–170, New York, NY, USA, 1996. ACM Press.

[ZS07] Nordin Zakaria and Siti Shukri. A sketch-and-spray interface for modeling trees. In *Smart Graphics*, volume 4569 of *Lecture Notes in Computer Science*, pages 23–35. Springer, 2007.

Appendix - A summary of the book 'Art and Representation', John Willats [Wil97]

Book Description (From the cover)

In Art and Representation, John Willats presents a radically new theory of pictures. To do this, he has developed a precise vocabulary for describing the representational systems in pictures: the ways in which artists, engineers, photographers, mapmakers, and children represent objects. His approach is derived from recent research in visual perception and artificial intelligence, and Willats begins by clarifying the key distinction between the marks in a picture and the features of the scene that these marks represent. The methods he uses are thus closer to those of a modern structural linguist or psycholinguist than to those of an art historian. Using over 150 illustrations, Willats analyzes the representational systems in pictures by artists from a wide variety of periods and cultures. He then relates these systems to the mental processes of picture production, and, displaying an impressive grasp of more than one scholarly discipline, shows how the Greek vase painters, Chinese painters, Giotto, icon painters, Picasso, Paul Klee, and David Hockney have put these systems to work.

But this book is not only about what systems artists use but also about why artists from different periods and cultures have used such different systems, and why drawings by young children look so different from those by adults. Willats argues that the representational systems can serve many different functions beyond that of merely providing a convincing illusion. These include the use of anomalous pictorial devices such as inverted perspective, which may be used for expressive reasons or to distance the viewer from the depicted scene by drawing attention to the picture as a painted surface. Willats concludes that art historical changes, and the developmental changes in children's drawings, are not merely arbitrary, nor are they driven by evolutionary forces. Rather, they are determined by the different functions that the representational systems in pictures can serve.

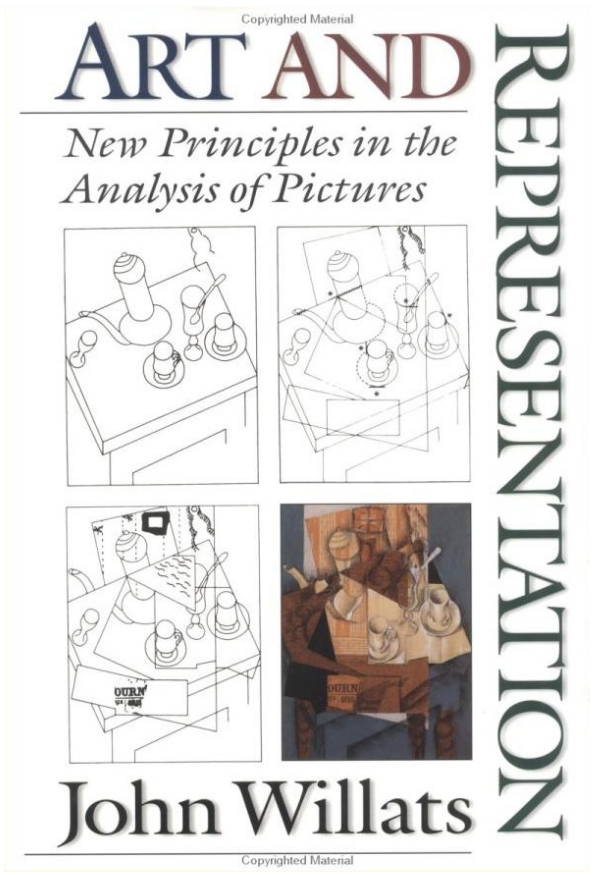


Figure 8.4: Cover

Like readers of Ernst Gombrich's famous Art and Illusion (still available from Princeton University Press), on which Art and Representation makes important theoretical advances, or Rudolf Arnheim's Art and Visual Perception, Willats's readers will find that they will never again return to their old ways of looking at pictures.

My Synopsis

"Why is it that different ages and different nations have represented the visible world in such different ways?"

"Why do drawings by young children look so different from those of adults?"

In order to address such questions one first needs to be able to describe pictures in some meaningful way. Willats book provides a clear account of the representational systems on which pictures are based. He then demonstrates the utility of these systems by using them to describe a wide range of pictures and to suggest hypothesis' behind the mental processes of picture creation.

The following is a brief outline of the book:

Parts 1&2 Drawing systems and Denotation systems A description of the representational systems in pictures. Willat's main aim is to show that the way people represent the visible world can be described in terms of two representational systems: the *drawing system* and the *denotation system*.

Part 3 Picture production Willats proposes that some pictures are derived from *object-centred* descriptions, and that pictures produced using such a mental process can sometimes be identified by characteristic anomalies within the picture.

Part 4 The functions of the representational systems These chapters are concerned with the various functions served by different combinations of drawing and denotation systems. In particular a faithful representation of a particular view of a scene is not necessarily 'better' than one which does not match this definition - it depends on what the picture is to be used for. For example an architectural drawing presents the true shape of the faces of objects in a scene, which is useful for planning and construction - but doesn't give a good sense of how the building will look once complete.

Part 5 Changes in representational systems over time Willat's addresses the question 'Why do artists in different periods and cultures, and children of different ages, use different representational systems'. Based on experiments on children in different age groups, Willats shows that as children grow older they progress from simple to more complex representational systems in order to produce pictures which are *'more effective as shape representations'*. He then proposes a similar motive seems to have prompted changes in styles of depiction during some periods in art history. He concludes that different representational styles are suitable for different purposes and that patterns of development in both childrens drawings and art history are determined by the different functions that the representational systems have to serve.

The rest of this document procedes as a chapter by chapter summary of the key points of the book from the perspective of someone interested in sketching.

Chapter 1 - Introduction

Willat's main aim is to show that the way people represent the visible world can be described in terms of two representational systems: the *drawing system* and the *denotation system*.

The drawing systems are systems that map spatial relations in the scene into corresponding relations in the picture. The three most important such systems are: perspective, oblique projection, orthogonal projection.

The denotation systems are systems that map features of the scene (or scene primitives) into corresponding picture primitives such as regions, lines or points. The three classes of denotation systems are: silhouettes, line drawings and optical denotation systems such as Pointillism. The distinction between *scene* and *picture primitives* is an important one. Scenes provide abstract shape representations of physical objects, whereas pictures provide a 2D shape representation of scenes. The drawing system says where picture primitives should go, the denotation systems say what the picture primitives stand for, refer to or denote.

Many different projection systems can be found in examples of children's drawings, and experiments show that children use different projection systems at different ages. With more faithful representations of a particular view being produced as children grow older.

Willats outlines another important distinction (originally used by Booker) between *primary* and *secondary geometry*.

Primary geometry describes drawing systems in terms of the 3D geometry of the projection of rays from the scene to the eye, and the intersection of these rays with the picture plane. Only used since the beginning of the Renaissance when projection systems we first codified. Secondary geometry describes drawing systems in terms of the 2D geometry of the picture surface, in particular Willats concentrates how orthogonal vectors within the 3D scene are represented in 2D in the picture representation of the scene. Secondary geometry is a consequence of primary geometry. The distinction is important for three reasons:

- Secondary geometry can describe drawing systems such as inverted perspective or route maps which cannot be described in terms of primary geometry (projection of light rays).
- It provides a way to describe the mental processes used to produce pictures.
- Using secondary geometry to describe drawing systems allows more psychologically realistic accounts of children's drawing development and art-historical changes.

Part 1 - Drawing Systems

Chapter 2 - Projection systems

A thorough description of all the true projection systems in terms of primary and secondary geometry. Two further projection systems (naive perspective, inverted perspective) are described which are closely allied to the true projection systems but can only be described in terms of secondary geometry.

[p]

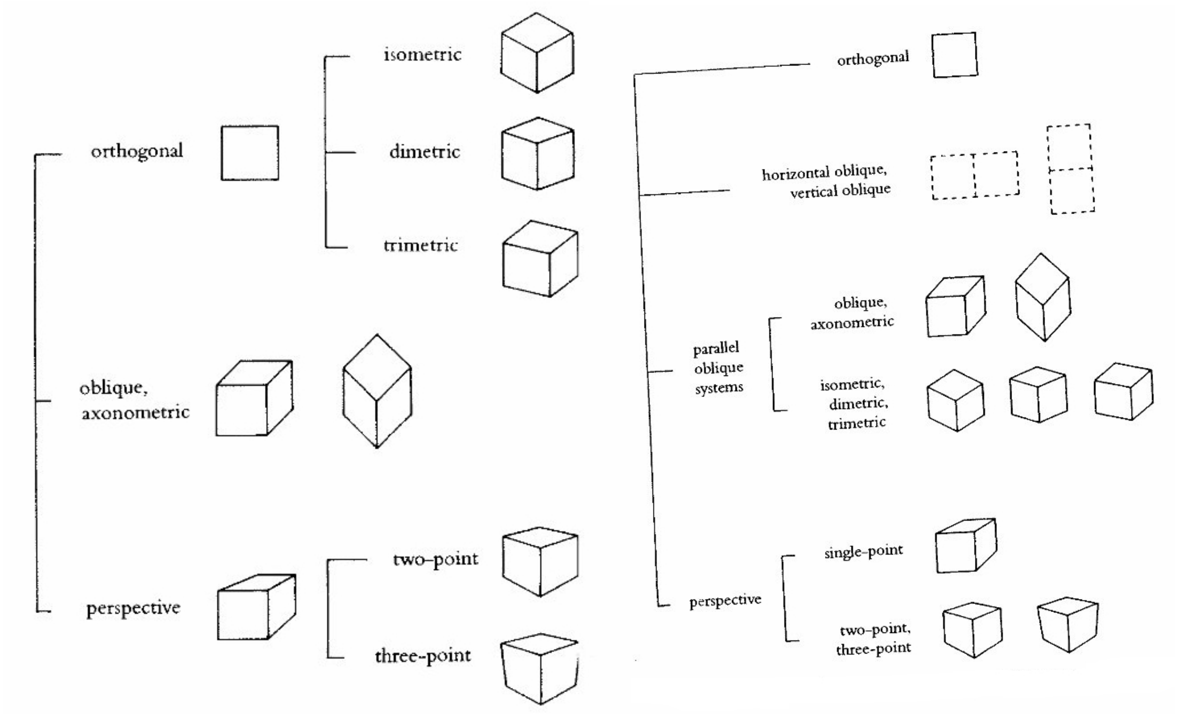


Figure 8.5: Classification scheme for projection systems, based on primary (left) and secondary (right) geometry.

Chapter 3 - Topology and extendedness

In addition to perspective geometry, spatial relations in pictures can be defined in terms of topological transformations. For example the London underground map preserves the properties of order, proximity and connectedness, but not true shapes or lengths. Drawings by young children, cartoons and caricatures can be described in terms of topological geometry. Some topological properties:

[p]

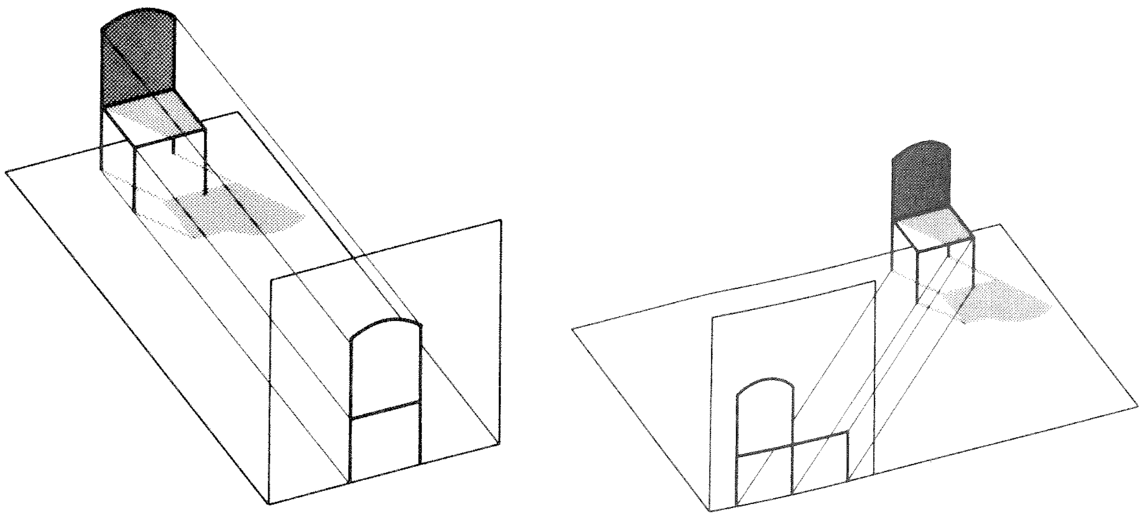


Figure 8.6: Orthogonal projection (left), Horizontal oblique projection (right)

- proximity
- spatial order
- connectedness
- separation
- enclosure

Extendedness is perhaps the most basic of shape properties. Extendedness is a term originating in linguistics and adopted by Willats. It refers to the relative extensions of scene or picture primitives. Informally words like 'round', 'flat' and 'long' refer to extendedness. The key point is the relative extension of the primitive in each dimension. A ball or cube is equally extended in all three dimensions. A stick is extended in only one dimension. Willats uses a notational scheme Xn where X refers to the dimension (can be 0,1,2,3) and each n the relative extendedness is each dimension. Thus 3_{111} describes a ball or cube, 3_{100} a stick and 2_{11} a circle.

Willats makes the case for extendedness being a natural category as children describe items as being balls or sticks if they have the same properties of extendedness (apple, umbrella). However there is no 'natural' symbolic 2D representation for slabs or discs, as they can be represented as either long or round regions. Willats makes a basic statistical argument to demonstrate this.

Secondary shape properties and shape modifiers

There is some evidence to suggest some children represent secondary shape properties such as 'having flat sides', or 'having corners' as separate shape properties to the primary property of extendedness. For example Figure 8.9 showing the same child copying an experimenter drawing a square in two different ways.

Some secondary shape properties (a.k.a. Shape modifiers):

- having flat faces

[p]

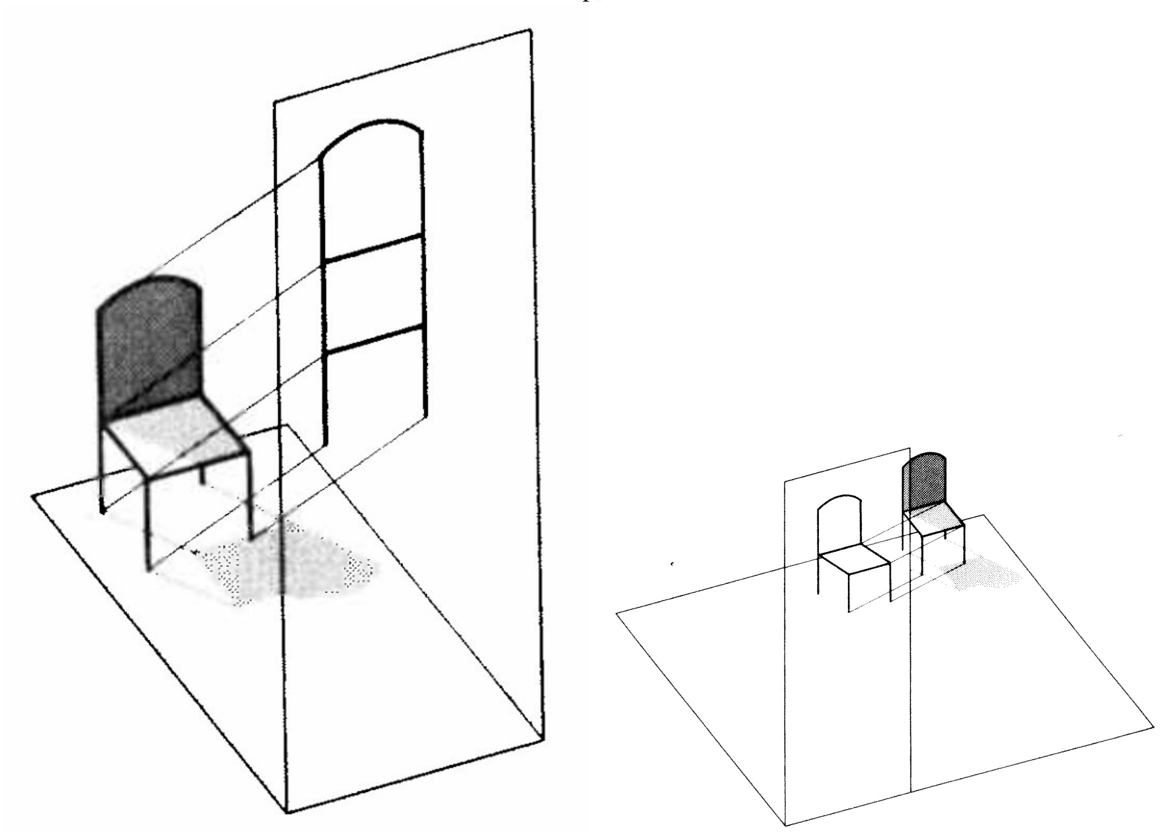


Figure 8.7: Vertical oblique projection (left), Oblique projection (right)

- having corners
- being straight
- being pointed
- being bent

Secondary shape properties have much in common with "non-accidental properties" in theories of scene perception (Biederman 1987). The main idea of Biedermans "recognition-by-components" is that certain shape properties of lines in the 2D images are interpreted by the visual system that the corresponding edges in the 3D scene *have the same properties*. I.e. A straight edge in the picture might be due to viewing a curve from edge-on, but our visual system discards this interpretation as being unlikely (as it is, if you consider all possible views) and assumes it represents a straight edge in the scene.

Biederman describes five non-accidental properties:

- collinearity of points or lines
- curvilinearity
- symmetry
- parallel curves

[p]

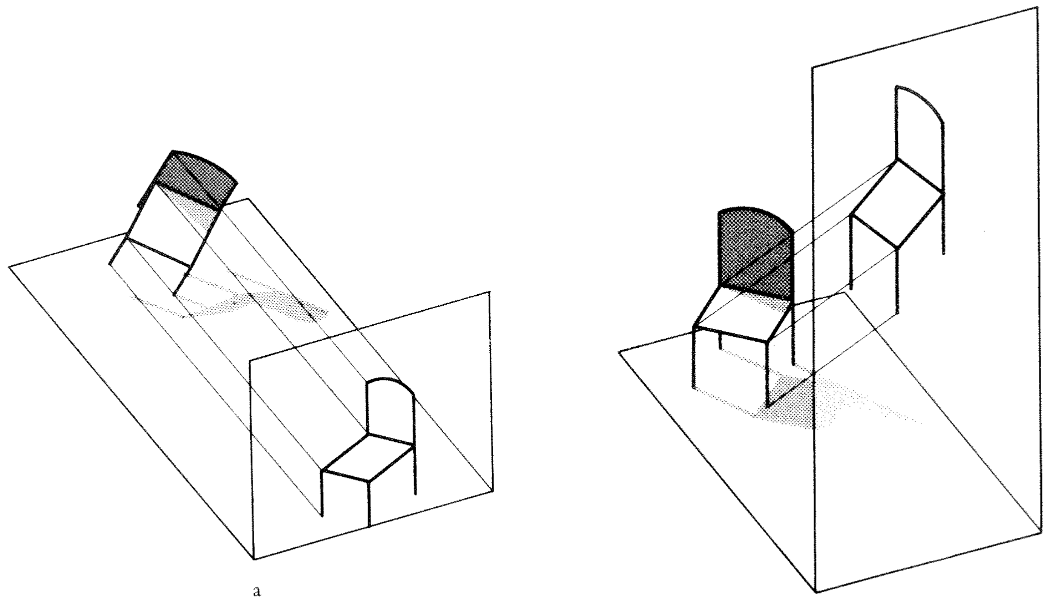


Figure 8.8: Isometric projection (left), Axonometric projection (right)

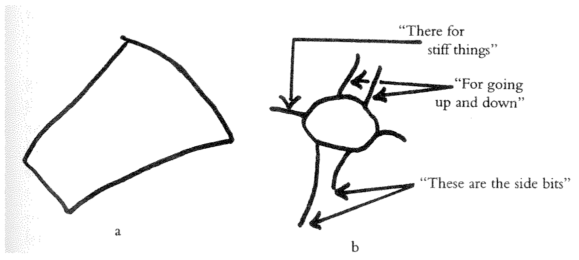


Figure 8.9: A childs copy of a square drawn by the experimenter in two different ways: First using four discontinuous strokes (a), and then by a continuous line (b).

- termination of 2 or more vertices at a common point

Biederman and others argue that the visual system assumes a generic view, or one which reduces the number of accidental alignments.

Finally Willats points out that a full account of spatial relations in pictures should include relations between all kinds of primitives, not only 1D ones such as lines and edges. This is the subject of chapter 4.

Part 2 - Denotation Systems

Chapter 4 - Regions as picture primitives

Willats outlines the three classes of denotation systems: based on regions (2D), lines (1D) and points (0D) as picture primitives. Regions can denote whole volumes, faces of objects, or regions projected into the frontal plane.

Scene primitives may be 0, 1, 2 or 3 dimensional. 3D primitives are volumetric and can be described in terms of extendedness. 2D scene primitives are faces or surfaces in object-centred descriptions and regions in view-centred descriptions. 1D scene primitives are edges. In views of scenes it is necessary to distinguish

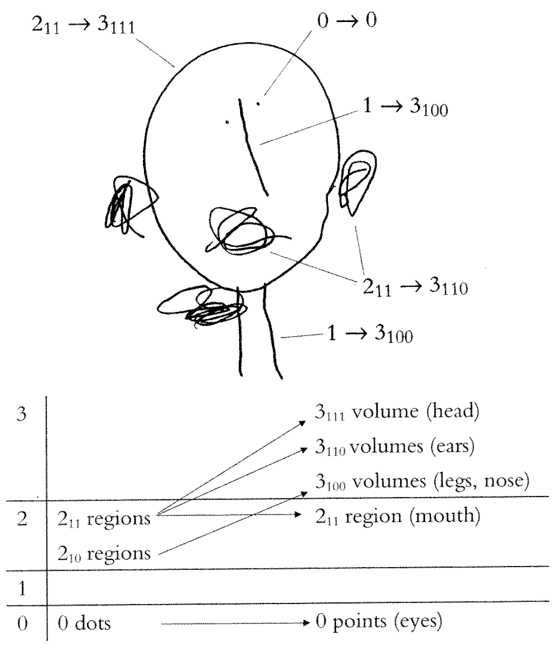


Figure 8.10: Analysis of a drawing of a man drawn by a 5 year old boy

between edges and occluding contours. 0D scene primitives are corners, points of occlusion or 'intercepts of small bundles of light rays'

Picture primitives can be 0,1 or 2 dimensional. 0D primitives are line junctions and points. 1D primitives are lines. Outlines are lines denoting the outer boundaries of the projection of an object. 2D primitives are regions which can be round 2_{11} or long 2_{10}. Picture primitives are represented in pictures by the use of physical marks. Figure 8.10 is an analysis of a childs drawing of a man which maps the picture primitives to the corresponding scene primitives.

Fig 8.11 shows the types of denotation systems. Described here:

Point primitives (a) denote the *intercepts of small bundles of light rays*

Lines (b) denote *occluding contours* and *interior edges*; the convexities and concavites of the *lines* carry information about the shape of the viewed surface

Lines (c) denote *occluding contours* only; if *lines* are carefully drawn they carry useful information about the viewed surface

Regions (d) denote *regions in the frontal plane;* the outlines are too irregular to carry information about the viewed surface, but the shapes of the *regions* carry information about the extendedness of the projection of the object in the frontal plane.

Regions (e) denote *volumes*; the extendedness of the *regions* carries information about the extendedness of the object in the three-dimensional space, and the *line*, as a mark, represents a narrow region as a picture primitive. Foreshortening cannot be represented within this denotation system.

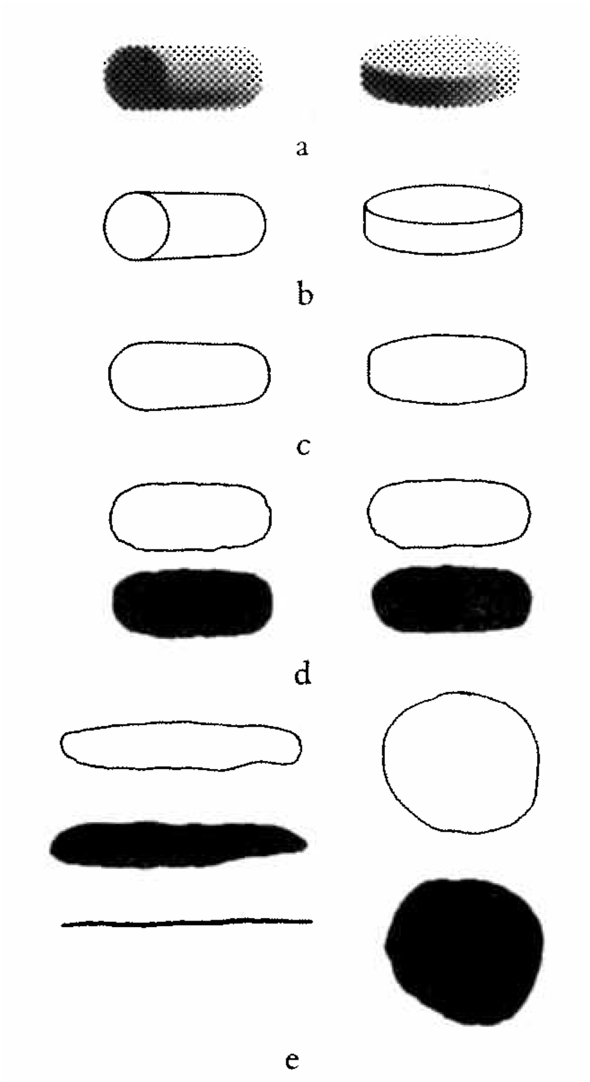


Figure 8.11: Types of denotation systems and associated marks in pictures of partially foreshortened sticks and discs. See text.

Dimension	Picture Primitive	Description
0	line junctions	representing a corner or point of occlusion
0	points	small dots of paint in pointillist paintings
1	lines	only extend in one dimension on picture surface
1	outlines	lines denoting outer boundary of the projection of an object
2	regions	2-11 round region
2	regions	2-10 long region

Table 8.1: Picture primitives

Chapter 5 - Line drawing

Outlines

A brief discussion of what an outline (silhouette) can reveal about the shape of an objects. Discussion of work by Marr and then Koenderink which proves a general rule - convexities in the outline correspond to

Dimension	Scene primitive	Description
0	corners	tangible, like the corner of a vube or table, object-centered description
0	points of occlusion	not tangible, points in the frontal plane where the projection of an edge or a contour passes behind a surface.
0	intercepts of small bundles of light rays	applies to pictures based on optical denotation systems like a newspaper photograph
1	edges	In viewer-centered descriptions - need to distinguish between *edges* which are projections of edges in the visual field, and *occluding contours*, which correspond to the projections of the boundaries of smooth surfaces in the visual field
2	faces/surfaces	object-centered
2	regions	viewer-centered, projections of objects in the frontal plane
3	volumetric	extendedness

Table 8.2: Scene primitives

convex surfaces patches, concavities correspond to saddle shaped surface patches. Consider a saucer - although concave, in silhouette its top edge appears as either a straight line or a convex line. Only saddle shaped surfaces can appear as a concave line. Nevertheless the example given shows an artist drawing a saucer with a concave outline.

Line drawings of objects with plane faces

Willats outlines the work of Huffman and others in line labelling.

Accidental alignments in a scene produce what a painter would call ’false attachments’ e.g. Lines which represent two unrelated edges. Huffman assumes such accidental alignments are not present in the analysed image (It is taken from a general position where a slight change in view does not changes the number or configuration of lines in the picture).

There are only 4 interpretations of a line in a scene made of plane faces:

- Convex edge (+)
- Concave edge (-)
- One of 2 types of occluding edge (plane on one side or the other of the line) (arrow with plane on the right hand side)

There are 12 ways in which corners can be formed (Fig 8.12) and Willats adapts this to create a ’look up list’ of 16 possible types of junctions in a drawing (L-junctions, Arrow junctions, Y-junctions and T-junctions, Fig 8.13).

Line drawings of smooth objects

Contain only T-junctions and End-junctions. C.f. Karpenko ’06 paper. Provided a line drawing contains a few T- and End- junctions and obeys rules for possible configurations of such junctions, it is easily interpreted as a picture of a possible smooth object.

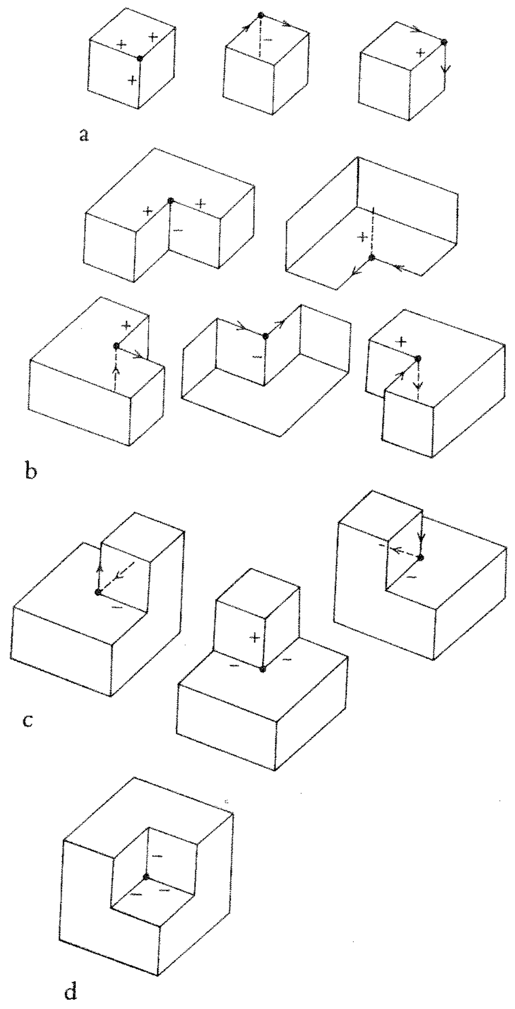


Figure 8.12: Huffmans 12 corner configurations

Features for which lines may and may not stand

Lines can stand for more than just edges or contours. Some of the features which lines can stand for:

- edges
- occluding contours
- thin, wirelike forms (hair)
- cracks
- creases
- the boundaries between areas which differ in local tone or colour. (not universal).
- Surface contours

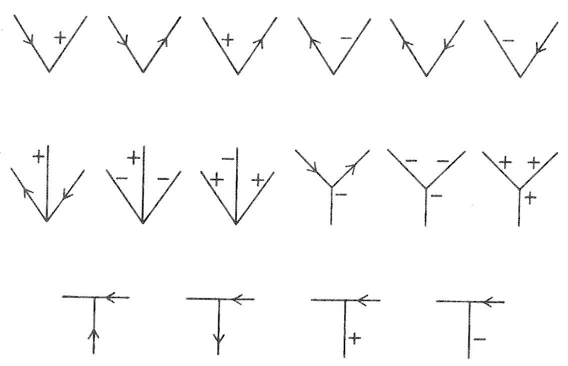


Figure 8.13: 16 junction types, adapted from Huffman

The relation between line drawings and the array of light from a scene

Lines are not generally used to stand for the boundaries of shadows or highlights. Demonstrated by comparing a line drawing by an artist to one generated using an edge detection algorithm.

Chapter 6 - Optical denotation systems

Optical denotation systems are systems where picture primitives denote features of the array of light reaching a sensor or the human eye. Very few artists attempt to replicate the appearance of this array of light directly. A few of the devices used in depicting the effects of light using drawing primitives are outlined:

Tonal modeling (a.k.a shape from shading) the diffuse portion of the Global Illumination equation. Denser lines form darker perceived shades.

Surface contours bracelet shading and hatching

Cast shadow removes ambiguity about position

Atmospheric perspective has four components:
Clarity reduces with distance - Artists sometimes use thicker lines in the foreground and thinner ones in the distance, or use lines of similar width but add less shape detail to distant contours.
Tonal contrast reduces with distance - Foreground contains strong contrast between light and shade, with contrast reducing further away
Change of hue with distance - More distant objects appear bluer (blue more readily scattered by atmosphere)
Change of saturation with distance - Distant colours are less saturated.

Colour modelling Due to colour constancy we tend to see the true colour of objects irrespective of lighting conditions. But in a picture using tonal modelling the colours become desaturateed because of the addition of black and white paint. One solution is to use line drawing for shape representation and pure colour for the surfaces (Japanese prints). Another way (Cezanne) is to use the natural tones of fully saturated colours to achieve tonal modelling, shadows in blue (naturally dark) and surfaces facing the light or viewer in yellows, oranges and pinks (naturally light). Pointillist painter Georges Seurat used small dots of colour placed side by side. The different natural tones of the dots are then exploited. Finally

Cezanne used another technique which exploits the fact that warm colours appear to come forward and cool colours appear to recede.

Willats concludes that in both line drawing and optical denotation the artist employs devices to provide equivalents for the corresponding features in the optic array of light from real scenes, as a picture can never literally deliver an array of light to the eye which is that same as that in a real scene.

Part 3 - Picture production

Chapter 7 - Separate systems

Willats observes that some combinations of drawing and denotation systems commonly occur together, and he explores to what extent they are independent. He also explores the question of how the two systems are related to internal mental representations of shape and space. He considers Marr's proposed three step process modelling the visual system for deriving an object-centred description from the image. The three steps being: 'primal sketch', '2.5-D sketch', '3-D model'. By considering *unusual combinations* he dismisses a direct mapping.

Transcriptions A transcription is similar to a translation in language. An artist redraws a picture keeping the subject matter but changing the drawing or denotation system. Willats looks for examples where either of the systems are kept the same for a transcription to judge how independent each system is from the other.

Systems in conflict: Anomalous combinations of drawing and denotation systems. Certain combinations of drawing and denotation system result in characteristic anomalies. Especially evident in childrens drawings of rectangular objects using a denotation system when regions are used to stand for faces. (Fig 8.14). These errors provide another motivation for describing drawing and denotation systems separately.

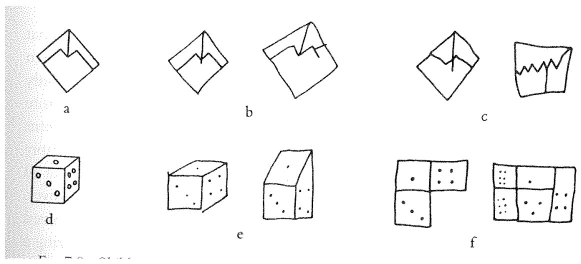


Figure 8.14: Childrens copies of a pattern (top row) and a picture of a cube (bottom row). This experiment tests the extent to which the errors in the drawings are competence rather than performance errors.

Chapter 8 - Picture production as a process

Willats expands on the characteristic anomalies mentioned at the end of the last chapter. He describes the various rules which could be followed by artists when forming a picture and suggests that the characteristic anomalies that occur provide evidence about the mental processes involved in picture production.

This chapter is particularly interesting from a sketching interfaces viewpoint because it raises questions about the ability of people to effectively draw what they see or imagine - especially children. Children younger than 8 tend to draw what they know, and are often unable to draw a view of a scene - rather they draw something corresponding to the mental object model they possess of the item in front of them. After 8 children are better at drawing what they see as a valid viewpoint. This has implications for using sketching interfaces for young children (e.g. a software product based on Teddy for modelling toys).

It's not just children that have trouble overcoming the internal object model when trying to draw a scene. Willats has performed experiments on young adults which show a substantial portion are unable to effectively copy a simple perspective scene placed in front of them (Fig 8.15).

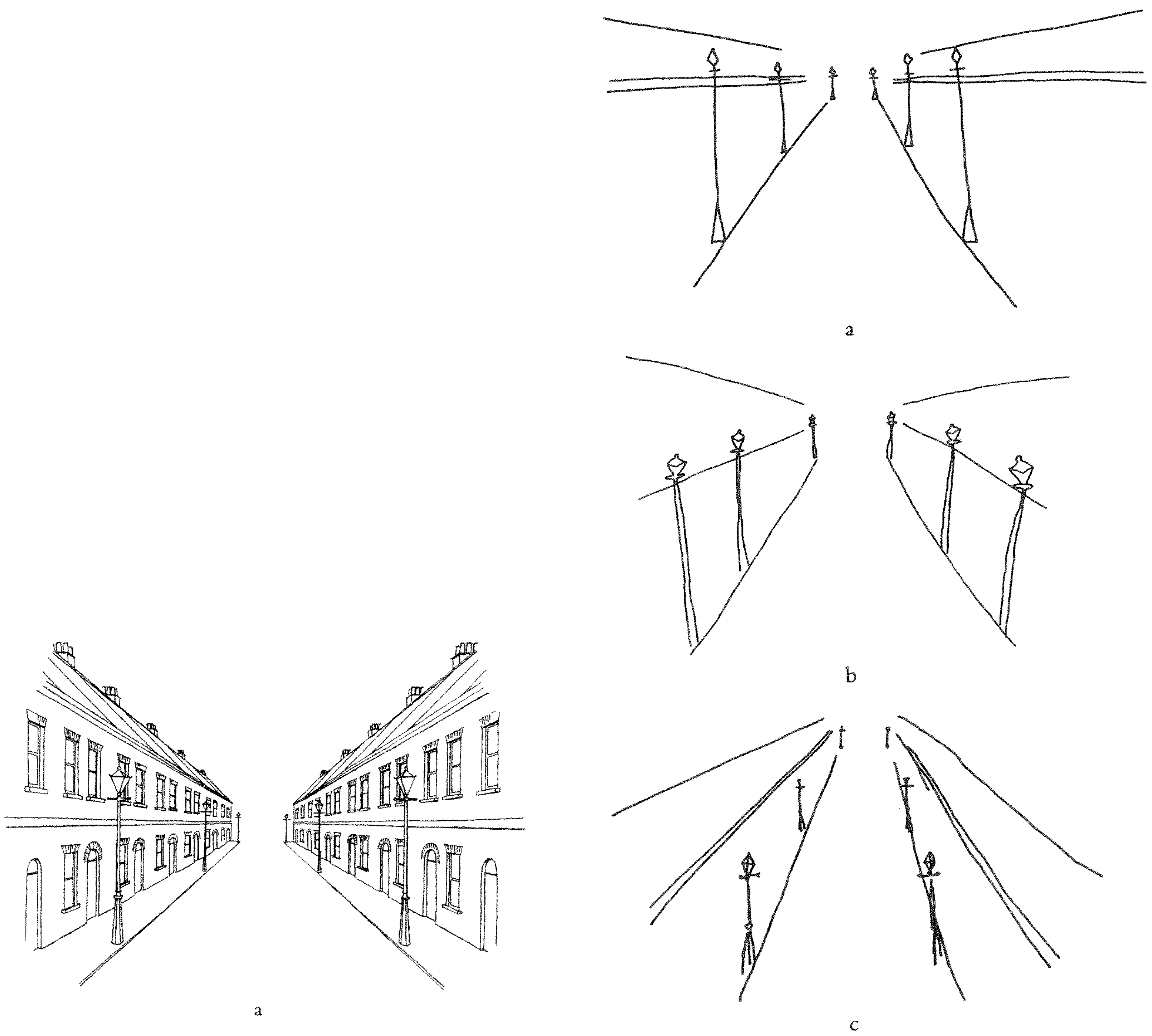


Figure 8.15: Drawing of a street scene used as a stimulus pattern (left), examples of young adults copies (right)

The results suggest that the majority of untrained adults in the West will attempt to draw a scene of rectangular objects by applying rules to an object-centred description of the shapes of the objects, rather than trying to match the edges in a supplied view. This finding impacts any sketching research that assumes the user knows how to draw what he intends to model. It could also help when deciding how to interpret an ambiguous sketch.

Willats goes on to discuss anomalies in trained artists pictures - which shows just how hard it is to overcome our internal representations of objects and sketch only what we see in front of us.

Part 4 - The functions of representational systems

Chapter 9 - Representing shape

Willats describes the constraints that operate on representational systems if they are to provide effective representations of 3D shape. This begins to build his argument that particular representational systems are best suited to serve particular functions.

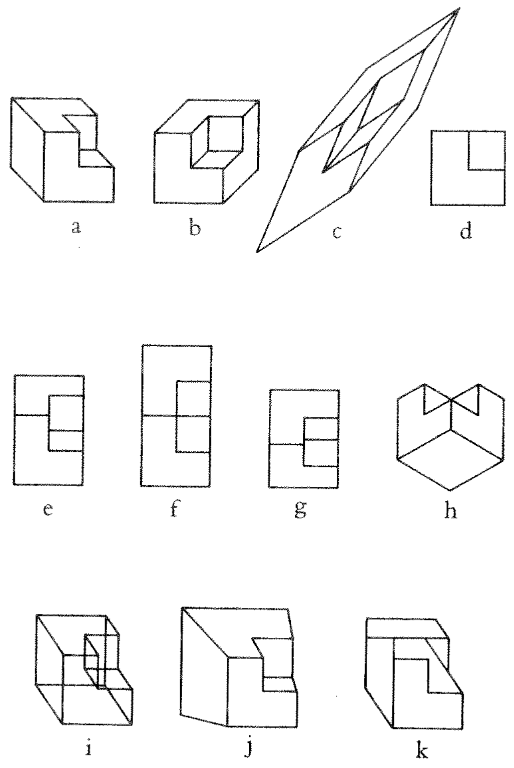


Figure 8.16: Which is the best representation of this cube with a missing corner? (a)

He derives a number of rules for drawings of rectangular and smooth objects. He also considers silhouettes. These rules could be useful in the related and relatively new research area of image saliency (ref. paper presented by franck). The rules in brief:

Line drawings of plane faces: (Fig 8.16)

1. The drawing should show a possible view
2. This view must not be taken from too oblique an angle
3. The view must be taken from a general position
4. The drawing should only show faces, edges and corners seen from a single viewpoint
5. The drawing should contain if possible, at least one T-junction

T-Junctions are important because they reveal information about occlusion and thus relative depth. Fig 8.16a is better than 8.16b at showing the structure of the cube with one corner removed because b could be interpreted as a small rotated cube on top of the larger cube. 8.16a removes the ambiguity.

Line drawings of Smooth Objects:

1. The drawing should show a possible view
2. This view should normally be representative (reflect the extendedness)
3. The view shown must be canonical (reveal all surface undulations)
4. The drawing should if possible contain T- and End-junctions.

Silhouettes:

1. The silhouette should show a representative view
2. The silhouette should show a canonical view

Chapter 10 - Flattening the picture surface

A picture is a surface in it's own right, and a display of information about something else. Willats suggests up to four domains may be present in the perception of pictures and artists may wish to tilt the balance in favour of some of these domains. In this chapter Willats outlines some of the various pictorial devices used to draw attention to the picture surface or its 'flatness'.

Obtrusive marks mosaics, tapestries, cubist, impressionists

The size of the marks larger marks draw attention to picture surface

Uniformity of texture Lack of texture gradient removes a depth cue (linear perspective)

Facture A term used by art historians to describe the manner in which a painting has been executed. Can brush marks be discerned? Are operations of the human visible or suppressed?

Techniques for flattening the picture based on the drawing systems

The geometry of different drawing systems Choose a projection system lacking depth cues (lacking linear perspective). Align the picture axes with the projection system axes (unlikely to happen in real-life). Use inverted perspective.

Mixed drawing systems

Cutting-Off truncate objects using the frame of the picture, the viewer then infers that such truncated objects are close but the lack of motion parallax draws attention to the fact it's a flat picture and not a scene.

The effects of false attachment similarly, false alignments should disappear with a slight change of viewpoint - as this doesn't happen it draws attention to the flatness.

Symmetry use of symmetry is rare. The technique is to align the axis of symmetry of the scene and the picture frame.

The avoidance of atmospheric perspective

The reversal of atmospheric perspective use greastest tonal contrasts and colours in the background to bring it forward

Techniques for flattening the picture based on the denotation systems

Silhouettes and line drawings

Mixed denotation systems

Willats then looks at reasons why artists might want to destroy the illusion of three dimensionality in a picture

Avoidance of idolatry The fear of being accused of worshipping false idols, or encouraging such worship.

Composition drawing on a curved surface (motivated by greek vases) presents problems with true perspective. Hence the continued use of orthogonal projection with no depth cues.

Philosophical and aesthetic balance Motivated by the Chinese values of yin and yang. He mentions an interesting technique to show curvature (cites Rawson) and to suggest depth using layers of contrasting dark and light.

The response to photographic realism once photography was invented artists felt pressured to justify painting as an expressive art form rather than a way to faithfully represent the world.

Chapter 11 - Anomaly in the service of expression

A discussion of examples of art where the artist has deliberate mixed drawing systems in order to enhance the expression of the picture. Willats draws parallels with ungrammatical constructs in poetry used for expressive purposes. Another example is the use of false attachment to emphasise a barrier between Christ and Mary is a particular painting. Further examples are given.

Chapter 12 - Investigating the nature of depiction

A discussion of the use of painting to explore the nature of painting. Introduces the idea of painting as a meta-language, a relatively recent occurance (last 150 years perhaps). Examples from Juan Gris, Paul Klee, David Hockney.

Chapter 13 - Children's drawing development

Willats classifies children's drawings change at different ages in terms of the systems he has outlined, and then discusses how and why these changes occur. One popular theory of child drawing development is that when they are young they draw what they know, as they age they draw what they see. Willats suggests re-phrasing this in terms of Marr's object-centred and viewer-centered internal descriptions. He discusses differing viewpoints and the problems with these and then outlines his own classification of systems used by children in order of complexity of the rules of secondary geometry on which they are based:

Drawing systems (Fig 8.17)

1. Topology and extendedness
2. Othogonal Projection
3. Horizontal and vertical oblique projection
4. Oblique projection
5. Perspective

Denotation systems

1. Regions as picture primitives
2. Lines as picture primitives
3. Line junctions as picture primitives

Experiments are given supporting this classification. First using drawings of an unfamiliar rectangular shape (cube with corner missing Fig 8.18). Then by experiments determining how children represented the foreshortening of an object like a plate, which requires a change in extendedness.

Other topics discussed:

- Children's drawings of people
- Summary of developmental changes
- Children's drawings from other cultures
- Mental processes

In summary, children begin by basing their drawings on topological relations and the representation of extendedness, and a denotation system where regions stand for whole volumes, and they end by combining oblique projection or some form of perspective with line drawing, including the use of T- and End-Junctions. Their motivation in this progression is to produce a recognisable drawing.

Chapter 14 - Historical Changes

Willats describes the use of representational systems and the changes in these systems in two periods of art history: greek vase painting, 8th to 4th century B.C.) and Orthodox Christian art (5th to 15th century A.D.).

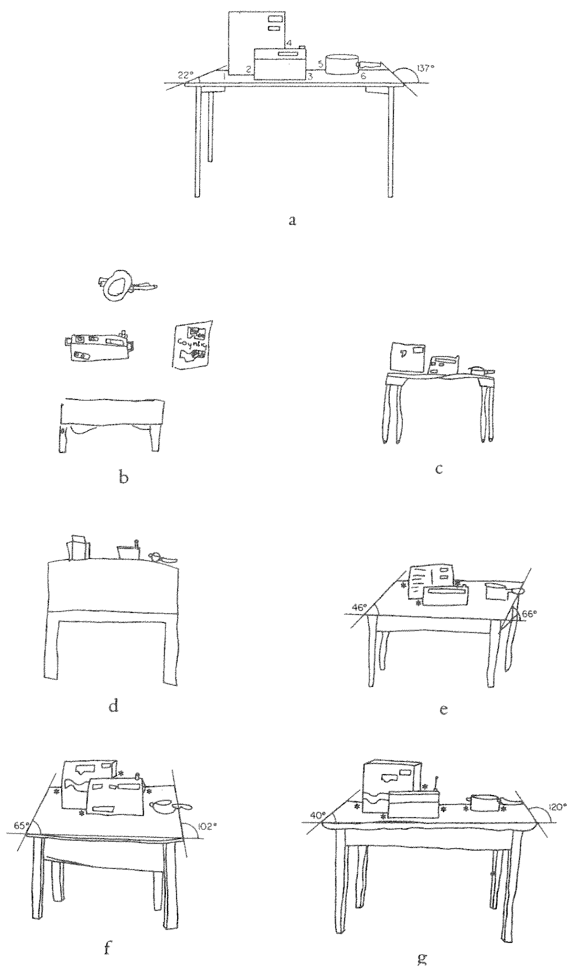


Figure 8.17: Childrens drawings of a table (a). Age increasing from 7.4 (b) to 13.7 years (g)

Greek vase painting

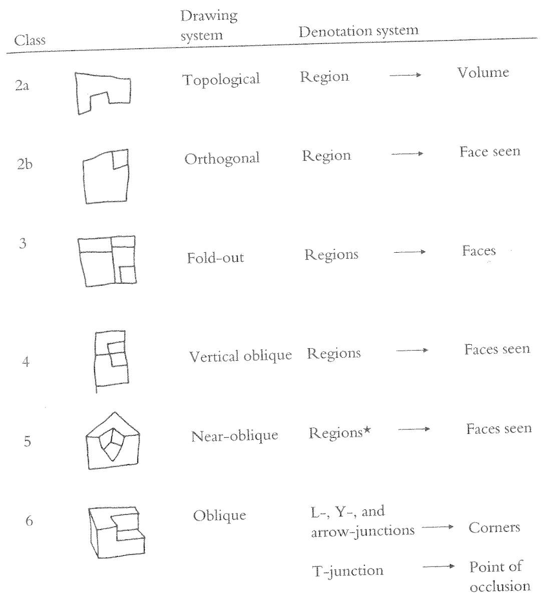
Willats suggests that the aim of the greek vase painters was (as with children) to produce pictures that were more effective as shape representations. The progression of styles:

- pure silhouette
- orientalizing style
- black-figure painting
- red-figure painting
- pure line drawing in white ground style.

All prompted by the need to solve specific representational problems.

Orthodox Art

In contrast Willats believes Orthodox Arts shows no such progress. The reason being the fear of worshiping false idols. Once orthodox art found a way of representing their subjects in a recognisable manner that did not



Class	Drawing system	Denotation system		
2a	Topological	Region	→	Volume
2b	Orthogonal	Region	→	Face seen
3	Fold-out	Regions	→	Faces
4	Vertical oblique	Regions	→	Faces seen
5	Near-oblique	Regions*	→	Faces seen
6	Oblique	L-, Y-, and arrow-junctions	→	Corners
		T-junction	→	Point of occlusion

Figure 8.18: Analysis of childrens drawings of an unfamiliar object (cube with corner missing)

depict them as inhabiting a real, corporeal world then there was no need to change it as it meet the functional needs of their culture.

The direction of art-historical changes

Willats concludes his book by saying that the history of depiction can only be understood in relation to the different functions that representational systems can serve.