Diploma Thesis

Surface Extraction from Curve Networks

in Application to

Sketch-based 3D Modeling

Alexander Kuhn

Department of Simulation and Graphics

Faculty of Computer Science
Otto-von-Guericke Universität Magdeburg
July 2009

Diploma Thesis

Surface Extraction from Curve Networks

in Application to

Sketch-based 3D Modeling

Alexander Kuhn

Course of Studies:	Computational Visualistics
Matriculation Number:	168167
E-Mail:	TheCypher@gmx.net
Supervisors:	Prof. Dr. Holger Theisel Jun.-Prof. Dr.-Ing. Raimund Dachselt
Processing Period:	February 3rd, 2009 - July 3rd, 2009

Department of Simulation and Graphics

Faculty of Computer Science
Otto-von-Guericke Universität Magdeburg

July 2009

ABSTRACT

This thesis deals with the topic of sketch-based 3D shape design as a special case of 3D modeling. Its purpose is to discuss, develop and implement a sketch-based modeling prototype, which is built upon experiences made with earlier systems. The focus thereby is on developing an intuitive, consistent and expressive modeling metaphor. This combines techniques for interactive 3D curve creation from sketch-based input, automatic topology extraction and subsequent surface creation.

In order to achieve these goals general aspects of conceptual and sketch-based modeling approaches will be discussed. This also includes the integration of general psychological and ergonomic design factors, which will motivate basic concept decisions for the prototype and lead to guidelines for the development. Based on those fundamental considerations a structured concept is presented, which will settle the foundation for the implementation. The resulting system will be tested against the general requirements by creating a set of 3D shapes of varying complexity and with respect to different modeling scenarios. Creating those models will also allow for conclusions about the effectiveness and applicability of the proposed shape creation workflow. Finally the thesis concludes with summarizing remarks towards given contributions and an overview about future work to be done.

ZUSAMMENFASSUNG

Die vorliegende Diplomarbeit beschäftigt sich mit der Thematik der skizzenbasierten Erzeugung von 3D Modellen als eigenständiges Teilgebiet der 3D Modellierung. Ziel ist die konzeptionelle Entwicklung und Implementierung eines Prototypes zum Zwecke der skizzenbasierten Formmodellierung. In diese Ausarbeitung werden zusätzlich bestehende Erkenntnisse bereits existierender Ansätze mit einbezogen. Die Arbeit konzentriert sich dabei auf die Erarbeitung einer intuitiven und konsistenten Modellierungsmetapher, die zudem eine hohe Vielfalt bei der Formentwicklung unterstützt. Diese beinhaltet sowohl Techniken zur interaktiven Erzeugung und Gestaltung von 3D Kurven aus skizzenbasierten Eingabedaten, als auch der automatischen Extraktion topologischer Strukturen und der anschließenden Erzeugung von Oberflächen.

Um die beschriebenen Ziele zu erreichen, werden zunächst allgemeine Aspekte der konzeptionellen Formfindung und bestehender skizzenbasierter Modellierungssysteme diskutiert. Dabei werden neben generellen psychologischen auch nutzungsergonomische Faktoren bei der Konzeptfindung mit in Betracht gezogen. Diese fließen schließlich in Form von spezifischen Richtlinien in die praktische Umsetzung mit ein. Aufbauend auf diese Betrachtungen, folgt die Vorstellung eines umfassenden Konzeptes, welches als Grundlage für die weitere Implementierung des Prototypes dient. Das entstandene System wird dann durch Modellierung einer Reihe von unterschiedlich komplexen 3D Objekten mit den oben genannten Zielstellungen verglichen und in verschiedenen Modellierungsszenarien getestet. Auf Basis dieser Erkenntnisse können dann konkrete Aussagen zur Effektivität und Anwendbarkeit des vorgeschlagenen Systems und dem Zusammenspiel seiner Komponenten getroffen werden. Abschließend werden die erzielten Ergebnisse zusammengefasst und ein Ausblick auf zukünftige Schritte zur weiteren Entwicklung präsentiert.

CONTENTS

ACRONYMS

3D	three dimensional
API	Application Programming Interface
B-Rep	Boundary Representation
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computer Numeric Control
CSG	Constructive Solid Geometry
CPU	Central Processing Unit
GPU	Graphics Processing Unit
GLUT	OpenGL Utility Toolkit
HCI	Human Computer Interaction
NPR	Non-photorealistic Rendering
OpenSG	Open Scene Graph
OpenGL	Open Graphics Library
ROI	Region of interest
SBIM	Sketch-based Interfaces for 3D Modeling
WIMP	Window, Icon, Menu, Pointer

Chapter 1

INTRODUCTION

1

INTRODUCTION

1.1 BACKGROUND

Digital three dimensional (3D) contents and their representation have gained significant importance during the last decades. With the advent of powerful computing abilities, graphical applications in various domains are accompanied by visual feedback to convey a broad range of information. Since the origin of first visual computer displays and graphical interaction techniques, one revolutionary application was presented in 1963: "Sketchpad" by Ivan Sutherland [Sut63]. This geometric application can be seen as the early ancestor of modern Computer Aided Design (CAD) systems.

At this point in time all of the planning within technical development was done in 2D using pen and paper. With the introduction of small and affordable computing machines, these processes started to be transferred to the digital domain. This progress marks the beginning in the evolution of CAD applications. The ongoing development has also changed the engineering process itself into a far more interdisciplinary subject: Numerous additional influences besides the pure technical aspect have found their way into the construction process. One major change is that the human designer behind these systems has been moved into the focus of creating engineering systems and therefore involves the explicit consideration of aesthetic, psychological and ergonomic factors.

This leads to the clash of fundamentally different requirements: On the one side precise design, reliable creation, high stability and focussed engineering and on the other side fast, intuitive and creative exploration pushing the bounds of existing developments. This circumstance is also reflected in the way the software for these purposes has changed: Over time the CAD packages have grown to large and powerful engineering systems, capturing a broad range of capabilities and integrating evolved structuring combined into large-scale systems. In the ideal case those are used from the first screw up to the final shipping of the product. Unfortunately these large packages often fail to reflect the requirements of the conceptual development of the first ideas due to the amount of functionality. For this reason one scope of research is the development of modeling and conceptual design techniques which are optimized towards conceptual content development.

Since the first innovative endeavors to extend the conceptual CAD process within its early stages like 3Draw [SRS91] and SKETCH [ZHH96] nearly two decades have passed. They triggered the development of a large variety of innovative and promising systems aiming for the common goal of practical and user-friendly conceptual 3D design. Having a closer look on what remained from those much-noticed concepts and what aspects have made their way into practice the actual utilization is behind the initial expectations.

1.2 MOTIVATION

The development of conceptual 3D modeling techniques raises two major questions: First, what are the reasons for the rather slow integration and limited acceptance of those systems? Second, what are further steps in order to improve existing techniques and promote their practical embedding?

It seems digital conceptual 3D modeling and technical 3D design processes contradict each other, due to their fundamental differences and therefore an integration is not feasible. But having a look at the 2D counterpart, the rather seamless embedding of digital creative and explorative drawing techniques has already become practical reality. This development benefits from specialized hardware devices and optimized 2D painting systems, which are closely oriented towards practical necessities of the designer. Nevertheless transferring those working paradigms efficiently into 3D still remains a challenging task. In relation to the initial question recent research efforts seem to have realized the fundamental problems and offer progressed and promising approaches, which are much closer to what they are actually intended for:

Practical and efficient sketch-based 3D conceptual modeling.

In addition to the creative aspect, the creation of valuable visual 3D content undergoes a complex production procedure, supported by a considerable number of specialized editing tools. Due to the general complexity of this task and the variety of different methodologies these tools have grown to extensive and complex software systems and require a remarkable amount of training in order to work with them efficiently. Up to now it seems that practical barriers still outweigh potential benefits and capabilities for introducing applicable conceptual modeling tools into this content development. Despite those obstacles existing approaches like ILoveSketch [BBS08] and SketchCAD [KS07] made considerable advances towards more usability and expressiveness, getting closer to the realistic integration into a practical 3D design workflow. Exploiting advanced 3D space curve techniques to close the gap between traditional 2D techniques and 3D content, paves the way for new possibilities to think again about sketch-based 3D modeling.

1.3 GOALS

To identify answers and possible solutions for the two initially stated questions is the main objective of this thesis. Therefore the thesis will subject to revise existing approaches and analyze them with respect to those questions. Basing on this discussion the goal is to develop a concept to improve and extend current capabilities and implement an appropriate 3D sketch-modeling prototype.

Further, the applicability of this systems shall be tested towards its efficiency and the needs which have been identified for the conceptual modeling purpose. As 3D space curve techniques have received considerable attention within recent systems and will be argued as natural extension to common approaches, their application will be within the focus of this discussion. Therefore the task of 3D modeling becomes the task of efficiently create curve networks from sketch-input in 3D and evaluate them in order to create valid, suitable and intuitive 3D shapes.

Working with those curve networks involves the further discussion of the following important aspects:

- How to efficiently create and interact with 3D curve in a 3D environment using sketch-input?

- Which steps are necessary to structure those input curves and to build up an appropriate topology?
- Assuming this structure, how to extract, create and shape surfaces with the desired features implied by the sketch-input?
- Finally, what can be said about the quality and achievable complexity of models created with these techniques?

Finding substantial answers to those questions will set the structure and guide the course of the thesis.

1.4 TASK DESCRIPTION

In order to achieve those general goals the thesis will take the following steps:

The first task will be to analyze existing 3D modeling approaches and discuss their their properties, features and basic concepts in chapter 2. Based on this analysis a structural concept will be presented in chapter 3, which captures the inferred considerations from the preceding discussion. The focus will be on remaining challenges in this domain and relate to the initial questions. Additionally, a set of related extensions and system features will be proposed and their choice will be motivated for the implementation of the prototype.

Following the concept, the necessary steps for implementing the system will be explained in section 4. This includes the description of individual components and their structural relation. Moreover, their integration into a practical modeling workflow is explained and the need for suitable interface control paradigms is substantiated. Concerning the underlying shape representation, the task is to create a predictable, parameterizable and flexible surface structure out of the sketch-based input. Further, necessary restrictions towards input and possible results have to be considered, as well as the properties and features of existing approaches.

In chapter 5 the resulting system properties will be presented and a set of generic sketching scenarios for 3D shapes of varying complexity will be presented. This represents the basis for the subsequent discussion of the applicability of the system and comparison with the initially stated goals. Additionally, the interface decisions, resulting workflow and appropriateness of chosen interaction techniques will be reflected. In order to provide an objective and profund discussion of the actually implemented components, the comparison to existing systems will be presented at the end of this chapter.

The thesis will conclude with chapter 6, providing final remarks on remaining challenges and give an overview about future research activities.

Chapter 2

FUNDAMENTALS

2

FUNDAMENTALS

The domain of digital modeling of 3D objects has become an important and active research area during its ongoing development. To provide the theoretical foundations for a conceptual modeling prototype, this chapter will present a discussion of 3D modeling in general and existing modeling approaches. The given overview will focus on systems for conceptual 3D design exploiting sketch-based control paradigms and put them into context to existing 3D modeling environments. Therefore, basic concepts and experiences made with existing 3D sketch systems will be explained. The final comparison of the presented systems with actually implemented techniques, will be presented together with the results in in section 5.4 of chapter 5.

2.1 3D MODELING IN GENERAL

Following a general definition *3D modeling* is the process of creating or modifying a 3D representation of a digital object with the help of specialized editing tools. The geometric representation of the manipulated object is referred to as *3D model* whereas the modeling software is labeled as *3D modeler* controlled by the *designer*.

Distinct modeling operations are following a so-called *modeling metaphor* which constrains the parameters and operations of the construction process and gives the designer an idea about how to use the operations and their possible results. The whole process is usually carried out *iteratively*, which means that each modeling step subsequently builds on previous editing operations. This goes hand in hand with an increasing geometric and structural complexity of the modeled object.

Following a general taxonomy as proposed by Olsen et al. [OSSJ09] the modeling process can be split up into three main phases:

- *Creation*

  During the creation phase geometry is constructed without any previous reference or context information. Usually this is done by inserting a standard template object which undergoes a set of geometric operations and further processing. In general the creation process is limited to one defined shape representation. Switching the representation requires an explicit conversion of the geometric model.

- *Augmentation*

  Inserting or editing additional features of an existing reference object and changing its topological structure can be referred to as augmentation phase. This also can involve structural changes, which fundamentally change the appearance and complexity of the object. Furthermore, the changes can be defined within different geometric representations in order to simplify the shape operation, which also requires an explicit conversion afterwards to merge the augmented geometric information with the original object. Additionally, the outcome of previous modeling steps has to be taken into account and delivers contextual information for subsequent shape operations.

- *Deformation*

> If no structural changes occur and only existing features and parameters of an object are modified, the operation can be considered as deformation. This can be defined mathematically as a closed function, which transforms an existing object into a modified version. These operations can be applied in a global manner, defined independently from the shape or in a local fashion considering local details of the object. As a result, existing topology constrains the results which can be achieved by deformation operations without producing topological inconsistencies.

in order to support the efficient creation of shapes and the exploration of alternatives, the relation of those three phases between each other has to be considered. In addition to this, the modeling process might involve several switches of the previously defined modeling phases. Starting with the initial creation phase, multiple augmentation- and deformation operations are applied, until the desired geometric shape is achieved. The course of this shape development can be referred to as *modeling workflow*.

Modeling Interfaces

To control the course of this workflow, the user interacts with the modeling system via a *modeling interface*. The structure and components of this interface are influenced by the exploited modeling metaphor. The interface consists of static and dynamic control elements which support the modeling process. Static elements are displayed during the entire process and provide basic functionality, which has to be accessible at every point in time working with the system (e.g. file- and system management, addition or deletion of basic structures). In contrast dynamic information is only displayed when needed during the process, to control recent interface operations. Both interface element types can be directly part of the scene (e.g. as widgets or dynamic manipulation objects) or organized in an additional way, independently from the actual content, for example by using additional menu elements or iconic representations.

The complexity of the system interface usually is coupled to the complexity of the construction task, the amount of internal structural dependencies and the manual influence to control the shape creation process. Two well-established representative interface classes are Window, Icon, Menu, Pointer (WIMP) and Sketch-based Interfaces for 3D Modeling (SBIM). The majority of professional 3D modeling applications exploit interfaces of the first type as they offer explicit control over detailed modeling functionality, provide an overview about integrated parameterized operations and are flexible towards user customization. Hence these interfaces often include significantly more control elements, than actually needed during the usual modeling process since most control elements are static parts of the interface. Therefore a large portion of modeling operation and design time has to be invested into mode switching and interface control operations. In addition to this, static interface components usually separate the shape modeling process from the actual object. On the one hand, this separation between operation and content allows for exact, abstract and complex control operations. On the other hand and in order to efficiently use such functions, the abstract relation between operation and resulting effect has to be learned. In contrast, the majority of the SBIM systems exploits the flexibility of sketched input for controlling shape operations in direct relation to the 3D object, which will be explained in detail in section 2.4.

General Modeling Systems

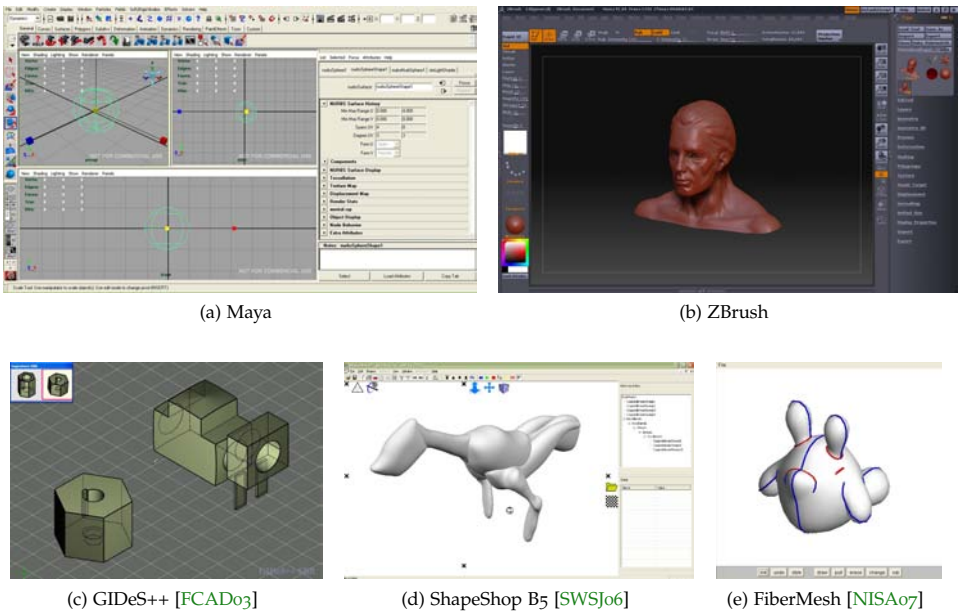
Examples for commercial WIMP systems are large software packages like 3D Studio Max, [1], Maya [2](shown in figure 1a) and Blender [3]. These are tailored towards general 3D design and digital content production and support a broad range of functionality which includes animation, texturing, physics simulation and advanced rendering capabilities. Optimized

1 Autodesk, www.autodesk.com/3dsmax
2 Autodesk, www.autodesk.com/maya
3 Open source 3D content suite, www.blender.org

towards detailed organic surface editing, volumetric sculpting systems, such as ZBrush [4] (figure 1b), MudBox [5] and sculpting modules in Blender have found their way into practical use and have lead to a significant quality improvement of high-resolution 3D models.

Towards specialized technical construction and design a wide range of CAD systems exists, for example Pro/ENGINEER [6], CATIA and SolidWorks [7]. These systems focus on precise technical construction and a parameterized output, which can be used in subsequent production processes including physical Computer Numeric Control (CNC)-manufacturing and distribution management.

For general 3D development a variety of 3D modeling tools exists, while their general purpose and used modeling metaphors are reflected in their interface structure. A small overview of some existing modeling systems and their interface structure is presented in figure 1.



(a) Maya (b) ZBrush

(c) GIDeS++ [FCAD03] (d) ShapeShop B5 [SWSJ06] (e) FiberMesh [NISA07]

Figure 1: Examples of the structure from typical user interfaces of different modeling systems.

The different purposes and related modeling metaphors often directly influence the appearance and complexity of the interface. Compared to figure 1a displaying a general CAD system, the ZBrush interface displayed in 1b benefits from its limitation to a small set of expressive operations and the direct spatial correlation between input and deformation method. Even more simplistic appearance is offered by SBIM systems shown in 1c, 1d and 1e, which usually contain only a minimal set of static elements while most of the shape operations are triggered by the interpretation of the sketch based input.

Extrinsic Factors of 3D Modeling

Considering the conceptual and initial shape design most of the mentioned systems do only provide a minimal or no supporting functionality. For this purpose large general CAD packages usually include sketching modules which mainly serve over-sketching and surface alignment for reconstructing given 2D content. This lack of support for conceptual design

4 Pixologic, www.pixologic.com
5 Autodesk, www.mudbox3d.com
6 PTC, www.ptc.com
7 Dassault Systems, www.solidworks.com

phases introduces a gap between 2D artwork done by professional concept artists with traditional 2D drawing systems and the 3D content production chain.

Besides these inherent technical and structural considerations towards the 3D modeling process a lot of additional external factors come into play. In general the user centered affordance and technical requirements correlate with the development of the shape. While rather simple geometric operations and small number of parameters is enough to steer the initial stages, their number vastly increases with its complexity. Structural dependencies and the interaction of shape features transform the editing into a global problem potentially affecting the whole existing geometry. This circumstance is usually not reflected within the modeling interface especially in static WIMP systems.

In order to identify specific additional requirements and to allow for this factors to be incorporated, not only geometrical aspects have to be considered, but also perceptual, ergonomic and artistic influences affect the underlying modeling process. The aspects of those two important factors are discussed in the next two sections.

2.2 VISUAL PERCEPTION OF 3D SHAPES

Digital representations of 3D objects are often integral part of visual computer applications. The effortlessness of the human visual system to perceive and evaluate images of 3D shapes might lead to the false assumption, that this process can easily be transferred into an algorithmic description. However, this assumption does not hold for the majority of steps during the perceptional transformation process.

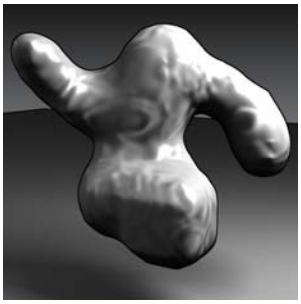
Most digitally modeled 3D objects we encounter in daily life, are represented in 2D media and undergo a complex visual perception process in our brain. This process starts with physical objects that can absorb, reflect or emit light. The resulting differences in light intensity can be perceived by the human eye and are evaluated by our brain. Due to further processing, these *discontinuities* are put into context with previous knowledge and essential abstract features are derived. The differences in the emitted light intensity can be artificially reproduced for digitized objects by combinations of shape features, texturing and lighting conditions. This reproduction can be used to imitate real-world physical lighting processes or optimized towards their contextual perception in so called Non-photorealistic Rendering (NPR) systems.

Recognition and Reconstruction

The association of perceptive low-level information with abstract mental features we already know due to our prior experiences can be referred as *Recognition* [Gol02]. On the other hand our brain is also capable of inferring the shape of 2D representations of 3D objects we have never dealt with before. By following a limited set of so called rules we can 'guess' about the 3D properties of this object as a result of a mental *Reconstruction* process. The reconstruction ability of our brain strongly relates to spatial relations, patterns and assumed continuity of shapes. The assumed reconstruction rules are described by Hoffman et al. [HS97].

The differences between these psychological phenomena are illustrated in figure 2. Figure 2a shows a human torso and belongs to a class of objects, which is familiar to the viewer. Therefore, several additional shape features can be reproduced, even within regions which are not part of the image or covered, like the back of the torso. Our brain automatically assumes additional features following a general minimization principle [HS97]. Although there is no or only little direct correspondence to known objects, this also applies to abstract and unknown shapes, shown in figure 2b, which are interpreted fast and efficiently by our visual system. In the latter case additional shape features are reconstructed and assumptions about back facing or covered parts of the shape are generated. The image shown in picture 2c demonstrates the local effect of the visual reconstruction rules. Although all corners and

(a) Rendering of a human torso



(b) Rendering of an abstract shape

(c) Necker Cube as optical illusion

Figure 2: Three different types of objects and perceived shape properties.

edges are reconstructed locally in correct fashion the global shape does not correspond to the actual expectation of the viewer. Creating this object as a real 3D shape would require an viewing position which does conceal the real object structure. Therefore viewing positions should either be canonical to provide as much as possible recognizable information or be subject to interactive control [BE93]. The referenced visual properties play a central role in the subjective evaluation during the modeling and can influence the course of the process. This view has to be taken into account developing ergonomic modeling tools, by exploiting user centered digital shape-representations and orient their development towards our perceptional understanding of shapes.

Shape Features in Human Perception

As denoted by [HS97] the human visual system uses silhouettes as key index to memorize shapes. Further shape evaluation is influenced by shading and texturing of the object simulating its surface properties. Additional shape cues are provided by so called surface features. Within the area of NPR this definition has been abstracted towards suggestive contours by De Carlo et al. [DFRS03]. Those features describe shape areas that will become silhouette features in nearby viewing positions. The evaluation of multiple viewing positions during the modeling process therefore has crucial influence to the depth perception of modeled objects and valuation of modeled surfaces. Interactive exploration of the scene thereby is an important point how to improve the overall understanding of the scene.

These findings are confirmed by Cole et al. [CGL+08] where it has been shown that certain shape features are constantly used to reproduce specific shape properties, even if the artistic style may differ fundamentally. Recent approaches directly try to use these encoding features for specifying 3D shape information and will be described in further detail in section 2.4.

By this point of view another important aspect has to be considered: The described abstract mental concept is diverse, enriched by individual subjective information, incomplete and can change dynamically during the modeling process. Identifying a one to one digital representation of such an concept is not feasible. Hence the modeling process is also necessary to constrain, limit and objectify this representation to a certain degree, while over-constraining the creation process will influence the structure and number of the final results.

Besides psychological considerations, which mainly focus on perceived information of an object and its representation as mental concept, resulting subjective design decisions have strong influence on the final result. Especially within the conceptual design phase these factors have to be considered developing appropriate tools. Capturing those ergonomic requirements, is an important issue where a lot of effort has been spent on ([VH98], [SOD05], [BBS08], [CGL+08], ...).

A general concept of a 3D modeling tool thereby has to balance two major aspects: On the one hand the modeling should be easy to use, intuitive and reduce the learning effort in order to use it. On the other hand higher model complexities require efficient editing tools with the ability to edit complex structures and should reduce repetitive tasks, in order to optimize the modeling process. The term '*intuitive*' during the modeling procedure thereby consists of three major aspects: Consistency, predictability and the match with general expectations in relation to the perceptional statements made before.

Stages of 3D Design

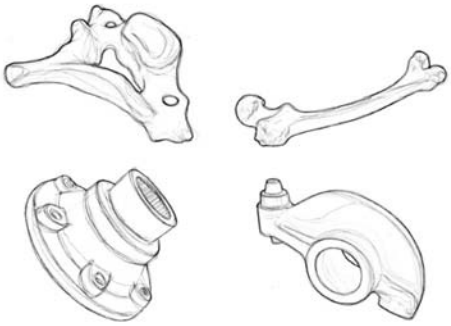
Orienting at the requirements during the conceptual design the overall 3D modeling process can be subdivided in several phases, which mark different states of the model complexity during the process:

1. **Conceptual Phase**
   During this phase new objects are constructed and large topological and geometric modifications occur. It is characterized by the need for quick exploration of different variants and a selection process of suitable variants. Although the exploration is done without reference or explicitly defined goal, consistency and predictable results already have high priority, in order to let the designer control the exploration process. Most of the fundamental design decisions and large structural changes will occur within this phase.

2. **Constructional Phase**
   During subsequent modeling steps, the chosen models are constructed and optimized towards their later purpose. This involves optimization of geometrical and topological aspects of the object. Additionally, structural extensions are applied, such as the introduction of abstract high-level features like smoothness or hierarchical dependencies. Such abstract features ease the handling of complex aggregated objects and still allow for general modifications on high levels of complexity without violating basic requirements. These structural dependencies are one important aspect in modern CAD acCAM systems.

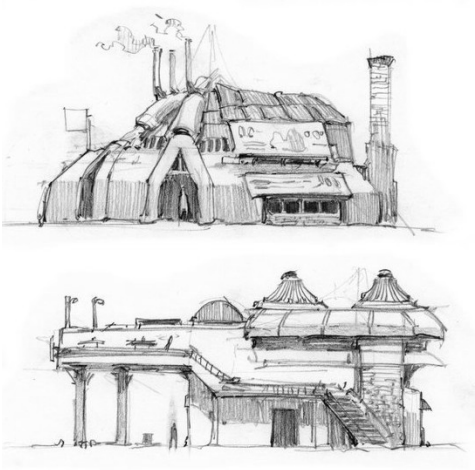
3. **Finalization and Export Phase**
   Once a 3D model is constructed it can be converted into different representations to enhance subsequent processing. In a usual modeling procedure this includes texturing, animation and the addition of external information. Also the arrangement of multiple constructed objects into one setting belongs to this phase, while structural changes are rather minimal and unlikely to occur. In order to prepare the object for further use, the representation is enriched by additional information, which can also be gathered during the preceding steps.

Supporting the creative exploration processes in the first phase thereby is a complex task. Mandatory restrictions to define the general modeling metaphor in this phase might already have major impacts to the final results. Therefore the most of the conceptional work done in this phase is carried out with traditional pen and paper media or digital equivalents. Some examples for this artwork are shown in figure 3.



(a) Sketched objects [CGL+08]

(b) Car concept (Fiat FCC, Sao Paolo 2006)



(c) Building concept sketch [Leo7]

(d) Rough initial sketch [Leo7]

Figure 3: Some examples of concept sketches and line drawings, where 3a shows a averaged map of several drawings of the same objects [CGL+08], 3b and 3c illustrate classical concept sketches preceding a 3D construction process. In contrast 3d represents a very rough shape approximation and exploration sketch.

3D Sketch Conversion

One major problem thereby is the transition from this conceptual 2D representations into a concrete 3D shape. As already indicated 2D sketches drawn by hand are usually characterized by inherent incompleteness and inconsistencies, which can also be conscious part of the creative production process. This circumstance is illustrated by figure 3b and 3c, which mark later stages within the conceptual phase and already concretized most of the intended shape features and already give a quite well defined mental concept of the object. Figure 3d shows a very rough sketch from the same artists as the last picture, but denoting an earlier conceptual stage. Although there are features which can be interpreted spatially, most of the shape is still ambiguous and could not be captured in a concrete representation without losing the central exploration aspect. This 'explorative effect' or 'conceptual blurring' can also be interpreted as an extended utilization of our perceptional reconstruction abilities mentioned in the previous section.

In contrast, a 3D model in most descriptions has to be exact, consistently defined and follow the inherently given structural rules of the underlying description. In order to transfer these representations a selection of sustainable shape features has to be made. This also implies the reduction of features which were used to expand the exploration space of the object and inconsistencies occurring between multiple conceptual variants of the same objects.

As already mentioned artistic styles and techniques may vary fundamentally in every aspect even for the same objects but in different design scenarios. Due to this fact the conversion usually has to be done by hand as one additional task in between the first two phases. The related initial construction of so called *base meshes*, approximating the artwork for further detail modeling is usually subject to tedious vertex editing or polygon subdivision paradigms. Despite fundamental differences Cole et al. [CGL+08] have shown, that professional artists still use common shape properties, such as silhouettes and surface features to visually represent 3D objects. It also turned out that most of these artistic styles are rather consistent, as similar features on different objects are expressed in a similar fashion. Applying the mostly strict constructional rules of common 3D model representations in-between the conceptual exploration will restrict the number of possible results and therefore be counterproductive during a creative process.

Following these considerations a conceptual modeling tool has to reflect this crossing from a conceptually blurred sketch into a concrete and valid description. Ideally the designer is also visually aware of parts containing inconsistencies or insufficiently defined regions in order to refine those within the conceptual phase using his own mental concept.

2.4 SKETCH-BASED 3D MODELING

As already outlined, the standard workflow constructing a concept for a 3D model, involves the creation of an extended set of 2D descriptions beforehand. This 2D design is usually carried out using traditional media like pen and paper techniques or in a digital domain using similar input hardware and appropriate 2D painting applications. The traditional pen-and-paper approach offers distinct advantages compared to the digital counterpart: It is immediately available, offers a vast amount of different expressive styles, as well as tactile feedback and it can be considered very intuitive.

On the other hand a digital support of those processes also has interesting benefits: Digital data can be copied, stored and modified, always offering the possibility to step back and forth at all stages of the creation process. Additionally, aiding computations allow for much higher exactness, working on all levels of detail or the application of high-level operations during the design process which can speed up repetitive processes tremendously. Further it is possible to guide the design process towards requirements of later modeling stages and hint on incomplete, inconsistent or ambiguous parts of the drawing.

Sketch System Taxonomy

Motivated by the ease, effortlessness and intuitive use offered by this media many systems developed similar control mechanisms to steer the overall construction process. To provide an overview about the existing SBIM approaches which have been developed over the last years, the classification of section 2.1 will be exploited again. Additionally, they will be put into context how sketch input is processed within the individual systems. The discussion comprises classifications proposed by Olsen et al. [OSSJ09] and Kara et al. [KS07].

A general taxonomy to structure the existing approaches towards the interaction aspect is outlined in figure 4.

Starting with the basic sketch input, there can be identified three major possibilities to process this input: As general *augmentation* operation, by *creation* of new geometry from scratch or *deformation* of existing shapes. Within all classes sketch like data can either be used

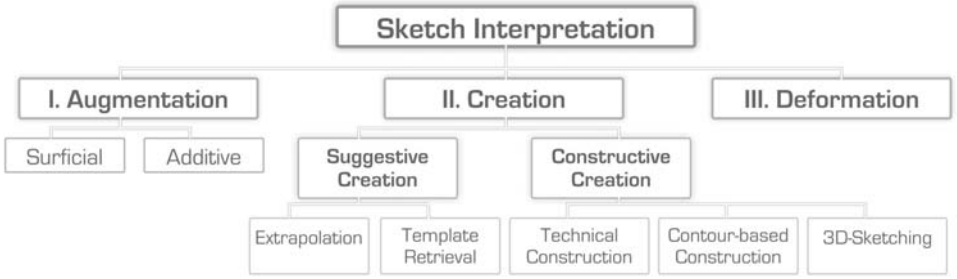


Figure 4: A Taxonomy of Sketch-based Modeling approaches following [OSSJ09].

to control abstract operations indirectly, using abstract *command gestures* and parameterized strokes or directly relate to the given geometry.

Augmentation

If reference information is already available sketch-based methods can be used to augment the existing geometry by additional features changing the underlying object structure. The existing data can be a result of previous modeling iterations or may be imported from another system. Augmentation can involve the editing of small geometric surface details and range up to complete restructuring of the initial object topology. Either way the existing data has to be taken into account and can constrain further operations. This usually limits the number of possible outcomes, but might also specify general expectations of the designer towards results of the operations.

Augmenting additions to the existing model geometry can be made *surficial*, which denotes changes that directly depend on a given surface structure or *additive* which is similar to a constructional geometry extension plus additional context information from the existing geometry. One prominent class of the first area are so called Shape-from-Shading techniques which interpret sketch input as local surface discontinuity. Recent examples are ShapePalettes [WTBS07] and the system proposed by Gingold and Zorin [GZ08]. In the first system the user can define a Region of interest (ROI) on the object and additionally provides the direction of the normals via a spherical reference, which is used to refine the existing surface towards the given normal constraints. The second system acts as an extension to FiberMesh [NISA07] and allows for the definition of additional linear shape features, which are interpreted as shading discontinuity resulting in indirect normal constraints and again local mesh refinement.

Taking existing shape features as contours and silhouettes into account, sketches can be used to modify these in an expressive way. One example for this procedure is given by Zimmermann et al. [ZNA07] where in a first step a ROI is selected and further silhouette points are identified. Those points are moved towards constrained positions on the input line and the resulting surface within the ROI is refined using Laplacian mesh optimization.

Additional surface features can also include extensive structural changes on the surface like holes or branching geometric structures. Representative examples are extrusion and cutting operations as used in Teddy [IMT06] and ShapeShop [SWSJ06]. In the latter system the use of implicit functions enables those additions to be consistently defined in a Constructive Solid Geometry (CSG) like manner. This allows for the aggregation of complex shapes by iteratively blending simpler geometric entities. An hierarchical extension to this system to handle complex arrangements and to express structural dependencies has been presented by defining so called Surface Trees [SS08]. A volumetric interpretation of this procedure is provided by [OHH+08] as input strokes use volumetric context data and its cutting planes for

mapping deformations and the addition of geometry.

The second major taxonomy class includes approaches which utilize given input sketches for the direct creation of geometry from the scratch without additional context information in the scene. These approaches can be further divided in *suggestive* and *constructive* approaches.

Suggestive Creation

As the initial construction of geometry is an ambiguous task due to the missing reference data, the first class captures approaches which let the user make major decisions during the construction. The most likely choices are presented to the designer by directly displaying them as dynamic part of the interface like in ShapeShop[SWSJ06], which provides a set of standard interpretation modes (e.g. sweep surfaces, inflation or extrusion). Another way is to suggest specific geometry based solutions during the construction interactively as in GiDes++ [FCAD03]. Additionally, the selection of suitable solutions requires the definition of a general measure and a rule based framework in order to construct possible solutions and evaluate computed results.

The decision when to compute geometric suggestions can either be triggered by a set of command strokes or automatically inferred by the structure of the input data. The latter variant is limited by the complexity of the input data as too many suggestions will not be feasible during the modeling process.

Besides just trying to interpret input data already given it is also possible to guess about the next modeling steps and extrapolate geometric structures out of the given input. One example is the Chateau System [IH01] where simple linear elements are used to construct basic models and the system provides further geometry by applying heuristics which shape configurations can be used to construct further elements. As the number of possible results is limited due to the use of linear axis aligned elements, they are blended on the screen within separate frames. For freeform models with more complex geometric features, the number of possible extensions usually is too large or requires too many user decisions to be used in general.

Another basic way to ease the geometric construction of complex objects is to provide a database of predefined shapes and indicate which objects should be used in which parametrization and location in the scene. Retrieving indexed shapes from known objects is similar to the recognition task performed by the human visual system described in section 2.2.

As the number of shapes in the database can be very large a flexible, extensible and expressive method is to use sketch input for template retrieval. In order to retrieve models from the database, the models have to be associated with distinct patterns and an appropriate similarity measure. Reasonable choices to encode object features are for example silhouettes and Suggestive Contours [DFRS03] or approximated skeletons [SSGD03], whereas the requirements towards complexity of this retrieval correlates with the number of given shapes. If sufficient templates are available modeling can easily be reformulated as an assembly task towards more complex shape structures. One basic challenge is to find an expressive minimal set of shapes and retrieval commands, which can be utilized to construct a reasonable complex geometry.

One way to tackle this problem with sketch based input is given by Owada et al. [ONI06]. As certain shapes are characterized by repetitive geometric features to a high degree (e.g. walls or fences) the system uses curved line projections and the resulting parametrization to place them within a scene. Additionally, specific geometric features or more complex shape entities can be inserted by indicating location and scale with additional sketch gestures.

Many systems also provide a simple set of basic initial shape primitives (e.g. spheres or cuboids) in order to provide a first context to the shape creation process. One example using this technique is SketchCAD [KS07]. It is also stated, that with the effectiveness of subsequent

deformations and augmentation steps the choice of the initial base template does not have too much impact.

The use of template patterns can significantly speed up the creation and insertion process, especially for repetitive elements or multiple complex predefined entities. Furthermore, the general ambiguity of the sketch-line input can be reduced and initial proportions, as well as depth information are provided.

Constructive Creation

Besides using predefined data, another way is to completely create the initial geometry from scratch without reference models, forming the class of *constructive* approaches. The definition of shape features thereby is performed without having additional context information and can be seen in analogy to the Reconstruction aspect of the visual perception system in section 2.2. Thereby three important sub-classes can be identified:

The first class captures approaches interpreting technical designs. Many classical sketch based approaches like [ZHH96] and [SRS91] were developed under the aspect of technical drawings and try to interpret the sketches made by the user in 2D and map them towards a reasonable 3D geometry structure. To do so, usually a rule based heuristic and subsequent surface optimization steps are required. A good overview over systems developed using this metaphor is given in [CPC04].

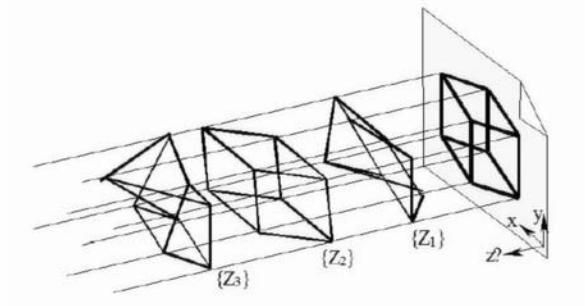


Figure 5: Illustration of the one to many mapping-problem resulting from single view 2D images [ML07].

The Mapping-Problem

Technical drawings usually provide multiple views of one object from one or more directions. Given a single drawing the resulting ambiguity can be referred to as *mapping-problem* illustrated in figure 5. As single perspective 2D images are usually not enough to clarify its 3D geometry additional constraints have to be defined. These rules can be taken from technical drawings (e.g. hidden lines are dashed, drawing perspective, crossings equal edge intersections), but still stable 3D extraction is limited to certain special configurations fulfilling those constraints. Examples for such systems are presented by Lipson et al. [LS07], Ku et al. [KQW06] and Naya et al. [NCC+03]. For organic, natural shapes or freeform descriptions a similar procedure is possible, but limited to simple shape configurations (e.g. SmoothSketch [KH06]).

The second class is formed by contour based systems, which mainly process the input data as contour or silhouette information with typically some kind of inflation operation to extract the 3D shape. As already mentioned in section 2.2 silhouettes are one essential element which is used by our neuronal system to classify and reconstruct shapes.

Due to this fact, a reasonable choice to use silhouettes or sketches of silhouettes as input in order to construct basic shape features. Popular approaches utilize an inflation like metaphor

to construct resulting shapes, for example Teddy [IMT06]. In order to apply the inflation operation, the given planar closed contour is discretized and a skeleton approximation is extracted by triangulating the given contour. The skeleton reference allows to define amplitudes for the inflation depending on the 2D contour distance. In a similar fashion the distance values within a closed shape can be directly interpreted as high-field as proposed by Bernhardt et al. [BPCB08]. An approach handling overlapping contours is realized by Karpenko et al. [KH06]. In the latter system hidden parts of the contour and intersection points are identified. Based on this a closed contour and the resulting shape are reconstructed in 3D using minimization principles. In contrast ShapeShop [SWSJ06] offers three differen main interpretation types of the sketch contour: Inflation, extrusion and sweep objects which are augmented by shape parametrization functions for increasing the scale, extrusion height or round corners. This definition leads to a redundant definition of similar shape classes, further surface discontinuities can only be approximated with the underlying implicit representation.

As a major advantage these approaches allow rapid shape development combined with minimal learning and application effort. This upside comes on the cost of the resulting limited complexity of shapes that can be created with these techniques. Furthermore, the resulting surfaces are very smooth and have a blob like appearance. Especially for objects with dominant sharp features as technical designs, the respective modeling metaphors are not flexible enough.

In order to extend the expressiveness of these procedures, one idea is to define multiple contours from different directions to approximate more complex shapes. Furthermore, the silhouette and contour lines are not mandatory to be smooth, but can also describe sharp feature edges or even blend both edge types. This is addressed by FiberMesh [NISA07] introducing a combined technique which starts with the inflation based initial shape construction and iterative refinement on the surface whereas additional sketch lines can be flagged as creased features. This is shown in figure 1e with red denoting creased features and blue smooth edges on the object. This allows for more complex shapes and extends the basic expressiveness of the system towards more general shapes. The structural abstraction and topological analysis for sets of contour lines defined in space intuitively leads to the definition of 3D contour networks.

3D Sketch Construction

Finally the last sub-class frames approaches were the construction is defined more independently from restrictions, such as technical-drawing standards or planar silhouette curves only and tries to extract general shape information out of 3D space curves.

Within these systems space curves might serve to describe boundaries from certain perspectives as well as other surface features or suggestive contours. This usually requires to draw directly in 3D space or a rather seamless 3D mapping of 2D strokes. One recent example is ILoveSketch [BBS08] which can be considered as an 3D line drawing application highly optimized towards its ergonomic use. Due to the combination of efficient mapping techniques and the use of symmetry properties, the system facilitates the 3D line construction in a significant way. The line mapping is supported by a small set of command strokes which allow for the definition of active drawing planes or extrusion surfaces.

General methods to construct 3D curves usually require to change perspective in order to give a second reference line for reconstructing 3D positions. An alternative method is proposed by Cohen et al. [CMZ+99] using directional projections of the curve on a visible surface. These projections can be interpreted as curve shadows, whereas efficient curve manipulation by editing its projected version can be a demanding task. Another way is to use existing lines and tangent directions to align given space curves as described by Michalik et al. [MKB02]. In the proposed system input sketches are used for a variety of shape operations including sweep surfaces, curve extrusion and line blending, allowing high-quality patch

based surface creation and editing.

One general target for those systems is to construct surface descriptions out of curves stemming from a sketch based interactive process. One System which takes closed space curves of this kind as input is proposed by Rose et al. [RSW+07] wrapping developable surfaces around a set of closed given boundary curves exploiting a iterative local extraction of convex areas. Doing so the system identifies ambiguities and produces multiple shapes which can result from the given input curves.

In medical application a similar, but more specific problem is already known, as typical segmentation applications produce 2D line contours from volumetric data sets. As a closed and smooth 3D shape is required, these slices have to be registered and combined as done in [BV07] and [JWC+05]. To do so in general a registration and reparametrization is required in order to match two subsequent slices. This can be done iteratively by defining a set of error measures which incorporate surface constrains for the resulting surface. Applying a minimizing deformation in relation to the defined measures will approximate the optimal shape results for the given input data. A more general approach has been chosen by [LLB+08] in order to extract surfaces from non-parallel Curve networks and even extract and maintain different structures.

Although 3D space curve definition is more demanding towards the user and the system interface than 2D sketching, they allow for complex and expressive creation of 3D geometry.

Deformation

Besides direct curve creation, the majority of existing shape deformation techniques can be controlled and parameterized with the help of 2D sketch input. Therefore they cannot only be understood in a purely constructive way, but also as control operations to circumvent complex user interfaces for complex operations. A simple curve already offers a wide variety of possible parameterizations, which can be used to steer the output of design operations. Using curves to control deformations single parts or the complete geometry can be edited, without changing the underlying topology. Deformations controlled with input strokes can be defined globally, so that they are applied in an equal fashion to all parts of the geometry (e.g. translation, scaling, rotation) or locally which requires more elaborate deformation descriptions and the definition of an active ROI.

An intensively studied feature are so called *gestural interfaces* which maps sketch-input to a set of predefined command strokes or gestures. This interface technique can help to avoid the inherent switch between software control operations and the actual modeling process. Further static interface elements can be avoided and unnecessary changes of the editing metaphor during the modeling process are reduced.

The application of these techniques raises two major challenges: First, in a completely sketch based environment gestures have to be clearly separated from actual modeling strokes, which in general cannot be guaranteed. Second, the actual capabilities of respective commands are not automatically clear when applied in different modeling situations. The more complex the operation becomes, the more important is a deterministic behavior towards different input situations.

In order to avoid the self-disclosure problem of a purely gestural interface and avoid interference with constructive strokes, visual feedback can be displayed to guide the parametrization process. So called *widgets* or handles can be used to visually associate and display the set of available shaping operations and can be found in many modeling systems. The integration of such elements into an existing interface has been examined by Schmidt et al. [SSB08] and the results of these efforts have been integrated into the ShapeShop system. Analyzing

principal directions and the curvature directions of input gestures allows to define small sets of distinct modeling operations, which are used for general transformations, shape operations and axis definement. Another given reason for choosing sketch-driven interfaces can be the simple limitation of input devices as some systems are designed for working on large tabletop displays or screens, where frequent mouse and menu operations are not feasible.

2.5 SHAPE REPRESENTATIONS FOR SKETCH-BASED 3D MODELS

Besides discussing existing approaches in SBIM focusing on the interaction aspect another central aspect is the underlying geometric and topological representation of objects.

As the main input consists of sketch like data, the question has to be resolved how to correlate the input with the representation of the shape. In one system usually multiple different representations are exploited, which are converted during the modeling process. The main focus thereby is on seamless conversion, effective manipulation and compact feature representation.

The representing of a digital 3D shape involves two major aspects: The *Topology* and the *Geometry* of the shape. While the topology of an object stores inherent structural information, the geometry encodes the actual shape information. Although it does not directly influence the visual appearance of the object, the topology usually dictates the overall performance and further editing possibilities. As the requirements towards performance, visual appearance and further editing are varying depending on the application the selection should reflect the importance of those aspects.

In order to find an appropriate representation, an overview of common approaches will be presented to motivate the choice for the conceptual prototype. In general two main categories for shape representations can be found: *boundary-* and *volumetric* representations.

Boundary Representations

The final aim for the majority of SBIM applications is the integration of the object into a rendering pipeline. The dominant representations in these systems are surface based and oriented at the outer hull of a shape. According to boundary representations following subcategories can be found:

- **Polygonal Mesh Representations**

  The mesh structure consists of a collection of polygons, where each polygon is stored as set of vertices storing the geometry and interconnecting edges, which implies the topology. Usually the number of vertices for every polygon is constant and minimal, as more complex polygons will affect the computational effort.

  Within this class triangular polygonal representations are common in many sketch-based systems as they are the standard way to render and process the final shape result. Usually, this also requires a conversion step between the preceding structural description and the triangular mesh. Triangular meshes allow for discrete approximation of any shape, whereas the complexity of the shape implies the number of triangles to use. For further optimizations additional quality measures can be defined discretely in order to assure certain visual properties of the shape. As most graphics hardware and further editing operations are tuned towards meshes, they also offer a standardized exchange interface towards other systems.

  The direct manipulation of complex polygonal meshes can be a cumbersome task, depending on the number of entities in the network, although it allows for absolute control. Therefore one focus in SBIM systems is to develop efficient high-level manipulation techniques. As there is a huge body of work on the topic of Boundary Representation (B-

Rep), a general overview can be obtained from Foley et al. [FvDFH95] or Botsch et al. [BPK+07].

- **Parametric Surfaces and Curves**

  In order to describe complex and smooth shape features linear control structures can be used to describe parameterized curves. In general a parametric curve can be defined as a map of the parameter interval $D = [t_0, t_1] \in \mathbb{R}$ into 3D Euclidean space such that:

$$t \rightarrow \mathbf{c}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}, \mathbf{c}(t) \in \mathbb{R}^3 \tag{2.1}$$

  In order to represent shapes the curve description is accompanied by a sequence of control points $\mathbf{y}_0, \mathbf{y}_1, \ldots \mathbf{y}_n \in \mathbb{R}^3$ with an associated knot sequence $i \in [0, n]$ and $t_{i-1} \leq t_i \leq t_{i+1}$ and $t_0 \neq t_n$.

  The sequence of control points thereby defines a linear control structure and the curve topology which allows to acquire the geometric position of the final curve $\mathbf{c}(t)$ in space using an interpolation scheme. The scheme defines whether the control points are interpolated directly (e.g. with Catmull Rom splines [CR74]) or used to approximate the location of the curve given defined continuity and differentiability constraints (e.g. general Bézier Curves and Splines, NURBS, ... [Faro2]).

  This concept can be extended towards surfaces considering the parameter domain $D = [u_0, u_1]x[v_0, v_1] \subseteq \mathbb{R}^2$ which leads to:

$$(u, v) \rightarrow \mathbf{s}(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \mathbf{s}(t) \in \mathbb{R}^3 \tag{2.2}$$

  In contrast to polygonal surface descriptions, parametric descriptions allow for continuous definition of 2D surfaces within 3D space. Therefore smooth analytical shapes can be approximated with higher exactness and more compact. One downside is the description of discontinuities along surfaces requires the definition of surface patches, which have to be carefully aligned along the discontinuity. Additionally, the closed analytical description of complex geometric objects usually is not feasible as well es the modeling wich such complex descriptions. Therefore the handling of the control structures usually is handled internally with external geometric and topological constrains given by the user.

- **Subdivision Surfaces**

  Another common standard among high resolution 3D surface modeling are subdivision surfaces, that follow a multi-scale approach representing 3D shapes. Given an initial geometric control structure, further shape details are acquired using a subdivision scheme. Applying the subdivision step once produces a refined version of the previous mesh structure, while an infinite number of subdivision steps lets the base mesh converge towards the final surface. This can be seen in analogy to subdivision schemes for constructing parametric curves [Faro2]. The control structure usually consists of a coarse polygonal mesh while the subdivision scheme can be thought as rules how to construct new geometry points and where to place it. They formulate the operations for a single refinement step and determine the properties of the final shape. In general, subdivision schemes are used to produce smooth surfaces from the given base mesh. A profund overview about existing techniques is presented by Zorin et al. [ZS00].

Complex control structures can further be described by aggregating curve networks which can store inherent topological information, as well as geometric data along curve positions. Examples how to achieve complex surface structures out of general curve networks are described by Schaefer et al. [SWZ04] and Levin [Lev99]. Both approaches define control structures over a given curve network and construct smooth surfaces which converge against the given boundaries of the network. The general topology of the network has to be defined by the user.

Volumetric Representations

In contrast, volumetric descriptions allow for the natural definition of closed solid shapes, which define a partition of the Euclidian space. This description can be used to define an 'outside' and an 'inside' subspace, while the transition between these two regions represents the object surface. For rendering purpose the interior of an object usually is not of further concern, therefore the visualization of the objects usually implies an conversion into a B-Rep for visualization.

As the representation of complex objects by an closed analytic function can become very complex, objects can be approximated by multiple spherical objects which are merged together. This procedure has been described by Turk et al. [TO99] and is a common approach to approximate 3D shapes from sketch-boundaries (e.g. [AGB04] and [KHR02]). Nevertheless for modeling purposes the availability of volumetric information offers distinct advantages.

- **Implicit Representations**

  Another well established approach is to use implicit functions for representing the geometric features of the object ([AGB04], [IH03], [KHR02]...).

  As modeling a complex shape with one single closed implicit function is not efficient, the majority of approaches uses approximations of 3D shapes by using multiple simple implicit functions (e.g. spheres) blending into one solid description. The benefit of this definition is the easy combination and availability of boolean CSG operations like Union, Difference and Intersection. The surface extraction of these definitions is accomplished by sampling the resulting isosurface of the resulting implicit description.

- **Space Partition Techniques**

  In general a shape can also be defined as 3D volumetric cells representing the geometric information. The shape of these cells an their spatial distribution defines their general properties. If those cells are distributed in space following a fixed scheme they can be used to form general grids in 3D space. These grids can either be defined regularly, using a global rasterization or algin locally with predefined shape segments. One grid cell usually contains scalar values denoting the existence of an object in this cell and further shape properties. This definition within the cells can be made binary for solid closed objects but also allows for diffuse object descriptions which can be visualized using ray casting techniques. An example for such systems in the SBIM domain is given by Owada et al.[OHH+08].

  As storing and handling such grid data involves much more effort, their interactive handling while maintaining high resolutions usually is computationally very expensive and limited to smaller areas. A main advantage is that no topology maintenance is necessary in fixed grid structures and spatial neighbor requests can be computed in linear time.

2.6 SUMMARY

This chapter lays down the theoretic foundation for developing the conceptual sketch modeling prototype. Therefore general aspects of the 3D modeling process have been considered as

well as the relation to perceptive and artistic aspects. Based upon this a general overview of existing approaches has been given and properties towards interaction aspects and the general model representation have been discussed.

It has been outlined, that conceptual modeling is an highly interdisciplinary process. Besides just considering the inherent system aspects, the embedding into a common modeling workflow has to be included. Therefore advanced modeling features have to be seen in context to ergonomic and artistic aspects and require an intuitive and consistent application.

Challenges for SBIM Interfaces

Taking these findings from general developments in SBIM into account, there can be defined a set of major challenges which have to be considered creating an appropriate system concept. As stated in [OSSJ09] the most recent challenges for this area remain the following:

Many sketch based interfaces, controlled by command strokes and gestures, do not obviously convey their actual interaction possibilities. This leads to the so called *self disclosure* problem, especially if multiple modeling metaphors are combined. In contrast to traditional WIMP interfaces no exploration of predefined menus can offer feedback about the provided functionality. Additionally, systems which work on a high level of automation tend to conceal most operations before the user. Therefore the user might be confronted with rather unexpected results and is not aware of possible reasons for this failure or how to avoid such situations.

A related problem is the general *lack of feedback* for automated system processes and the need for general constraints, to allow for 3D geometry reconstruction. These limitations, processes and parameterizations have to be clearly defined and expressed, in order to avoid unexpected results which do to match the designers intention. This problem is amplified by inherent switches between different modeling metaphors which also have to be communicated to the user.

A further point is that SBIM systems often do not support direct *integration* into general modeling systems, as they usually also follow completely different interaction and representation approaches. Therefore most of the systems are developed as stand-alone prototypes in a general competitive situation towards standard systems. While there are currently no conceptual 3D SBIM systems anticipated in standard modeling workflows, their stepwise optimization towards an efficient 3D modeling is hard to achieve, as they are often developed and tested in isolated environments. Furthermore, most of the provided functionality is redundant to existing operations. Despite they are more efficient to use, they are required to significantly outperform common standards, in order to legitimate their integration effort.

Generally many basic SBIM systems are often characterized by the general *limitation of available shapes* because of constraints due to the underlying representation. This aspect limits the number of achievable results and hinders the general applicability of such systems in a regular modeling workflow. Another consequence from to restrictive representation might be the inadequate integration of the actual input, as it has to be adapted to the capabilities of the underlying structure.

Conceptual Guidelines

Following these challenges a set of general criteria for conceptual modeling systems can be found as follows:

- **Expressiveness**
  Parametrization should allow the exploration of different shape possibilities for the input sketches. This can also be seen in relation to the user interface, whereas available options shall be in range of general understanding and intuitive to use, with respect to the plausibility of the results.

- **Exactness**
  Every input step of the user needs to be faithfully represented by the system. Although approximation errors are assumed, their correction shall not impede intentionally created shape features. The exactness of the shape results has to be balanced with the general support of individual and expressive modeling.

- **Validity**
  The modeling framework processes should be able to handle input inaccuracies, ambiguity and inconsistencies during the model creation. A general lack of consistency either has to be communicated to the designer or lead to reasonable results, implying that further corrections are necessary.

- **Determinism**
  All operations during the modeling procedure should lead to similar, repeatable and stable results under varying input conditions. This implies, that small changes in the structure of an object, should only lead to small changes in the overall results, in order to enhance the understanding and usability of the provided modeling tools.

- **Compactness**
  Minimizing the number of parameters to control and modify the shape results is important to provide effective modeling metaphors. This requires the minimization of redundant modeling functionality.

These guidelines will provide general rules for creating the concept of the targeted SBIM modeling prototype and can be used further, to evaluate the implemented features towards their general applicability.

Chapter 3

CONCEPT

3

CONCEPT

Having gathered the main aspects and properties, as well as distinct challenges among current systems, the goal for this chapter will be to describe the theoretical framework of the proposed sketch modeling system. In order to derive an implementation framework the individual components and their internal structure are described in the first section. Further dependencies and functional relations of the single modules will be explained and extended towards their practical applicability. Besides these structural considerations, the purpose and functionality within a regular modeling workflow will be explained following the definitions given in chapter 2.1. In addition to this, conclusions drawn from the discussion in the previous chapter will be incorporated.

Following these considerations the general embedding and characteristics of the 3D modeling process will be explained in section 3.2. The resulting processing pipeline and selected shape representations will be explained in section 3.3, followed by the conceptual description of the interaction operations embedded in the system in 3.4. The subsequent surface extraction concept will conclude this chapter together with additional remarks about the selected visualization representation.

3.1 SYSTEM STRUCTURE

The proposed structure of the system components reflects the orientation towards an ergonomic and cyclic workflow during the shape modeling. This also requires fast transitions between crucial modules and interactive evaluation of the input data. The individual components include the following entities which are shown in the figure 6.

The basic elements of the system thereby include the following aspects and serve distinct purposes during the shape development process:

I **Interaction Module**

The input data to control the modeling process can consist of two main input data types: 2D geometrical position information and triggered key events. Additional static interface elements are avoided and replaced by dynamic control elements depending on the modeling situation. The positional information is used to construct the sketch-lines directly in a 3D environment and consists of 2D coordinates sorted in temporal order of their occurrence. Key inputs made by the user indicate mode changes to the system and support the interpretation of the 2D data in relation to its context.

Thereby the system takes a different direction than systems like ILoveSketch [BBS08] and ShapeShop [SWSJ06]. Important system command operations are not mapped to gesture commands and therefore completely sketch driven interface. Stroke gestures cannot be generally guaranteed to be exactly distinguished from the actual modeling input. Despite it is assumed, that a small number of modifier keys will provide an exact interpretation of input commands and improve the direct control speed of the system. Although increasing the initial learning effort for the additional modifier keys,

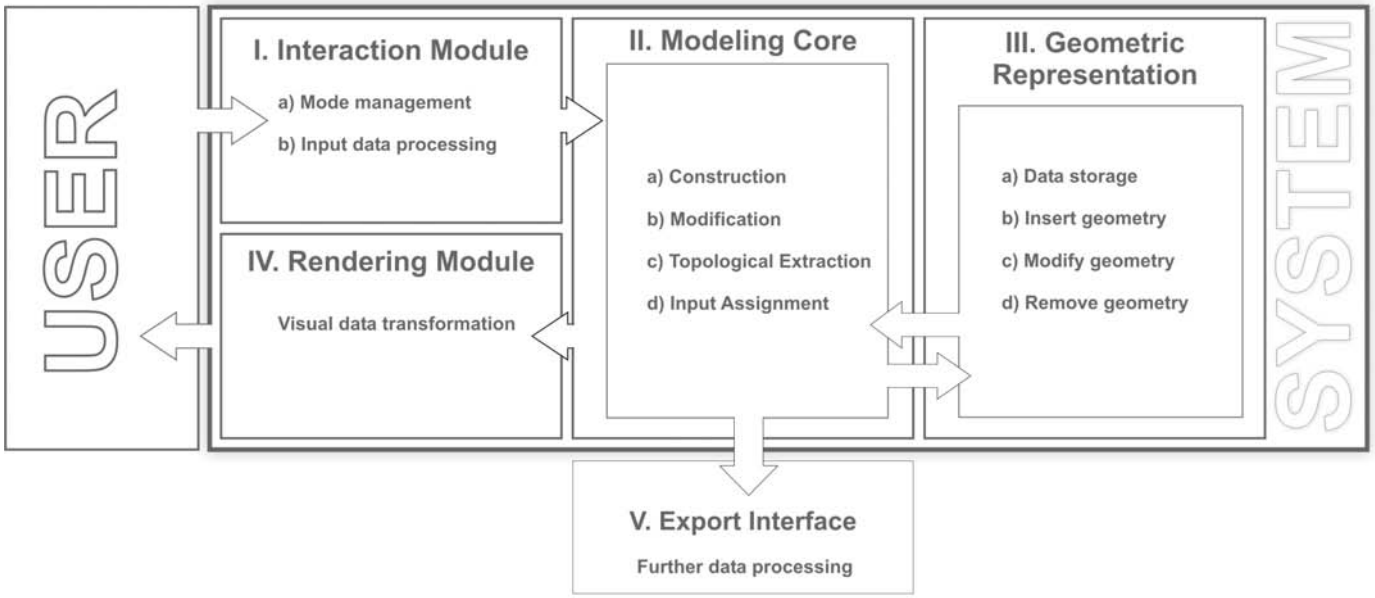


Figure 6: Overview of all system components and their dependencies.

the cognitive demand of a limited number of modifier keys during the modeling session is expected to be negligible.

In addition to the pure acquisition and storage of the primary input data, the Interaction Module initiates first filtering and topological structuring steps. These involve the creation of new primary sketch structures, smoothing and resampling procedures which will be described in chapter 4.3.

II **Modeling Core**

The actual system core distributes and links given input commands to the associated functionality of the underlying geometric representations. During the modeling process it also handles the main surface extraction procedure and maintains the validity of the underlying structural components. This includes the addition, deletion and modification of sketch-lines into a sketch managing module, the detection of intersections and the automatic topological segmentation of the curve network. Further the Modeling Core manages the creation and handling of new topological structures and the determination of structures intended to be modified.

III **Geometric Representation**

In the system there will be used three different basic representations: First, a *curve network* representation to handle the input and infer topological and spatial dependencies. Second, a subdivision based *quad mesh* structure to control the appearance and parametrization of the constructed surfaces resulting from the curve network. Finally, a *triangular mesh* structure for rendering and display purposes. All of these structures allow for interactive insertion, deletion and modification of stored elements and are optimized towards fast conversion during the modeling process. Their motivation and conceptual framework will be discussed in section 3.3.

IV **Export Interface**

As an extended module of the Modeling Core the Export Interface manages the transformation of the internal representation into standard modeling surface representations. On the one side this module takes the permanent data storage of line and surface models and on the other side the import of those external model data into the system. As common standard a set of triangular mesh data formats and general spline descriptions for reconstructing the curve-network are supported.

V **Rendering Module**

As one of two units directly accessible by the user the rendering module takes the visual transformation of the underlying geometry information into the projected image. The main focus lies on representing structural information and shading cues of the constructed shape elements as well as information about the current state of the system. It should also support the understanding of how far the modeling process has advanced, by giving the user feedback about the actual completeness of the edited model parts.

3.2 MODELING WORKFLOW

As already mentioned in section 2.3 one important aspect during the use of 3D modeling tools is the seamless integration of all modeling components into one usable framework and working cycle. Furthermore, the prototype is assumed to be a part of a complex 3D content production chain where every element serves a distinct purpose, so its tasks and process related requirements can be focussed to a limited core functionality. So should be the interface control and corresponding modeling operations.

Final goal for this stage is a *closed*, *solid* and *manifold* surface description of an object. Manifold in this context is reduced to the property, that the final object surface consists of several continuous and closed patches. As it is most common this description will be given as an triangular mesh structure. Conceptual modeling steps may not produce closed, valid or even manifold results, so the system shall guide the user towards a valid description and give feedback which input configurations do not fulfill the requirements of the intended shape class.

As the modeling process itself is intended to be *iterative*, starting from a rough description leading up to a more detailed description of the intended shape, it is important that all modules keep track and comply with the actual development of the 3D model. Iterative also means that ambiguous shape features are concretized during the modeling procedure towards the intended shape results. This general process is shown in figure 7.

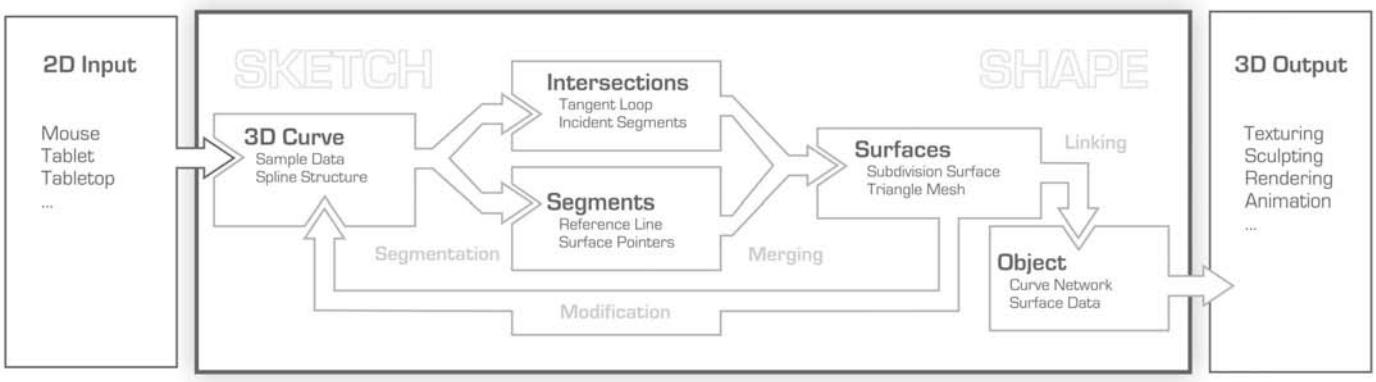


Figure 7: Schematic overview of the intended system workflow.

From this point of view several deductions for the system can be made: All components have to be assured to work interactively with reasonable complex models and without impeding the overall process due to computational interruptions. Single modules automatically have to integrate the given input into the given context, depending on the mode setting currently established in the system. The results of the single processes during the modeling shall be consistent and predictable compared to different input situations. Therefore intuitive results of the surface construction process have a high priority to maintain fast workflow progression. This is further supported by a small set of basic operations minimizing redundancy and structural overhead for the designer.

Following the psychological considerations in section 2.2 and observations made from current systems in section 2.4 the modeling within the prototype is oriented towards shape discontinuities. This means that the input made by the user denotes these discontinuities and locations of special interest on the surface to construct. It also implies that there should be no or only few discontinuous shape features at locations, where no constraints are given.

It is also intended, that during the modeling the amount of mode-switches is minimized. In the optimal case the underlying structure supports creation, deformation and augmentation of the 3D object, while maintaining a consistent modeling metaphor and a maximum of expressiveness. Additionally, the detection of the corresponding system states should be accomplished automatically, in order to assure that given input commands are interpreted in the correct context.

One downside offering too much explorative modeling features is the significant increase of the number of ambiguous cases and unresolvable shape configurations. Intending a valid surface description at the end of the construction process, single model operation steps should be able to deal with this kind of input, but still produce reasonable results, even if not fulfilling all description requirements.

3.3 SHAPE REPRESENTATION

As already shown in section 2.5 the representation of a shape and its geometrical and topological structure offers a wide range of possibilities. In order to find an appropriate representation, the structure and goal of the modeling steps, that are executed with the system have to be considered.

The main goal remains the construction of a *valid* and *expressive* 3D geometrical shape out of 2D input lines. Valid in this context contains two aspects: First the used representation is required to be compatible with standard shape representations such as triangular mesh formats to support further processing with external systems. Second the shape processing has to produce expectable and valuable results for models which can be created with the system. To provide this, the resulting surface structure should avoid holes or self-intersecting elements, unless explicitly defined. Expressiveness denotes the ability to create a wide variety of shapes with the system. Ideally the possible shape space ranges from technical oriented designs with dominant sharp features to organic structures providing smooth transition, as well as hybrid shapes.

In order to store and manage shape information inside of the system and take account for those main aspects, several representation concepts are combined which serve distinct purposes throughout the modeling process:

The general input information is mapped to a *curve network*, holds geometric as well as topological information and provides the transition from unstructured sketch input towards a structured network. In order to express further surface properties, the resulting edge sequences are used to construct a *quad mesh*, that allows for compact and flexible surface representation. For final export and rendering steps the shape is converted into a *triangular*

mesh description. These meshes are a common representation among existing systems for this purpose and are described in chapter 2.5 in more detail.

I **Curve-Network Structure**

In order to handle sketch like input the general assumption is made, that the conceptual drawing can be represented by a set of *parametric freeform curves* as shown in figure 8. Furthermore, the input is interpreted as discontinuity on a later surface, while shading or on-surface details are not part of this primary sketch input. This relates to the observations made from chapter 2.2 and 2.3.

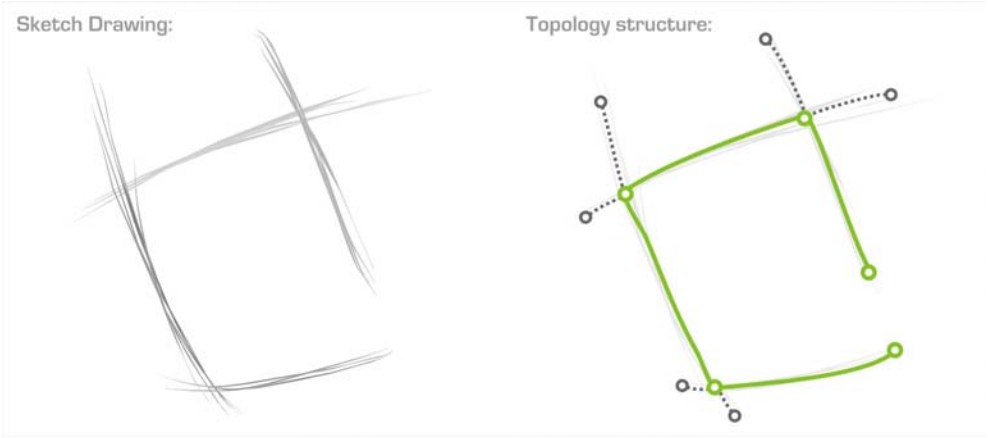


Figure 8: The general correspondence assumption between sketch and topological network.

The set of curves is analyzed towards their spatial properties which are expressed within the topology of the network structure. As main advantage curve networks offer structural similarities to the assumed input and support arbitrary complex shape models. The semantic correspondence of sketch input and elements of the curve network is proposed by the preceding assumption shown in figure 8 and offers a logical extension to existing boundary oriented approaches. One downside is the complexity of the curve network raises with the amount of shape details and is only feasible up to a certain degree of surface complexity. Furthermore, under-determined networks might offer too many ambiguities. Therefore it is assumed, that conceptual objects are characterized by an reasonable complexity which is reflected in the curve-network.

With these preliminaries in mind the first question is how to process and map the 2D sketch input data to the network.

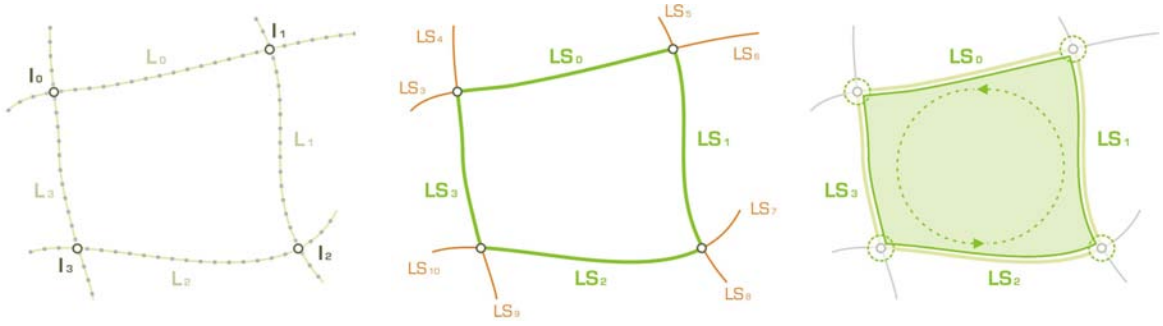
The initial input data is captured as sequence of 2D samples in temporal order and local window coordinates. Projecting them into the scene geometry creates a 3D position for every 2D sample. The sequence of these samples are used to construct a parameterized curve. A freeform-line or 3D space-curve therefore is given by $\mathbf{c}_k = (\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_u)$ with $\mathbf{p} \in \mathbb{R}^3$, whereas a set of curves is given as $\mathbf{C} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_n)$. The details of this construction process and implementation issues will be presented in chapter 4.2.

To support the analysis of the geometric structure, the network is expanded by the definition of curve intersections. An intersection stores a number of pointers to adjacent

curve and a parameter $t \in [0,1]$ describing the local position of the intersection on the line. This leads to $\mathbf{i}_j = (\mathbf{c}_0, tl_0, \mathbf{c}_1, tl_1, \cdot, \mathbf{c}_r, tl_r)$ with $tl_i \in \mathbb{R}$ describing the parameter location of the intersection on the curve and $\mathbf{I} = (\mathbf{i}_0, \mathbf{i}_1, \dots \mathbf{i}_m)$ denoting the amount of all intersections. The index thereby denotes a local sorting parameter and r the number of adjacent lines. An intersection thereby describes a point, where two or more lines are adjacent in 3D space and therefore define local points of special interest for the subsequent topological analysis of the network. Due to numerical issues the exact coincidence of curve positions is hard to achieve therefore the determination of spatial coherence is threshold based. These threshold values are defined in dependence of the drawing scale, as it will be described in section 3.4.

Furthermore, intersections split up every line into a set of curve-segments with every segment starting and ending at one intersection $\mathbf{cs}_j = (\mathbf{i}_{min}, \mathbf{i}_{max})$ and $\mathbf{CS} = \mathbf{cs}_0, \mathbf{cs}_1, ..., \mathbf{cs}_k$. If $\mathbf{i}_{min} = \mathbf{i}_{max}$ the segment describes a loop in the network and by definition it is guaranteed that the segments do not cross or self-intersect. If all intersections are assumed to have a unique positions their number is limited to $k+1$, where k is limited by the number of curves within the network.

The final aim of the approach is to find surface-patches which can be inferred from the network structure as shown in image 9. Formally a surface is given by a set of subsequent segments by $\mathbf{s}_j = (\mathbf{cs}_0, \mathbf{cs}_1, \dots \mathbf{cs}_v)$ with v defining the number of boundary segments of the surface and $\mathbf{S} = (\mathbf{s}_0, \mathbf{s}_1, ...\mathbf{s}_g)$ capturing the amount of all surfaces. It can be assumed, that the number of surfaces g in the system is limited by two times the number of segments k, as every segment is assumed to be adjacent to two surfaces.

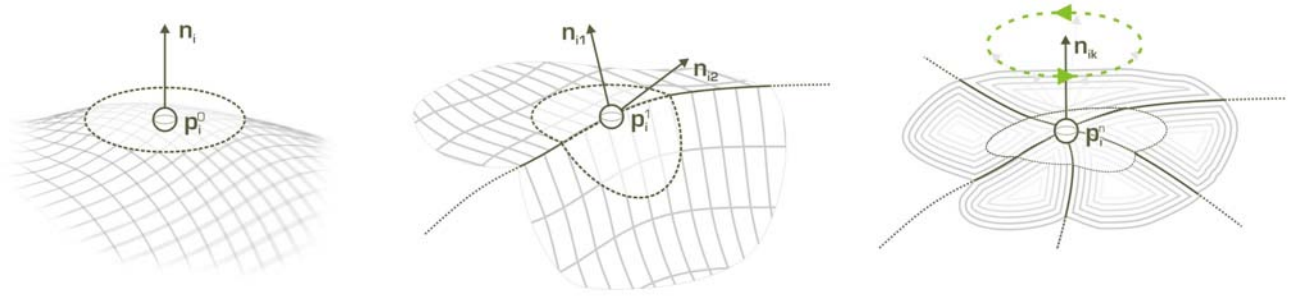


(a) Initial input samples and curve construction (b) Topological Segmentation (c) General Surface Extraction

Figure 9: Different stages of the network during the general shape construction process.

Finally, an object in the sketch scene represented as curve network is given by $\mathbf{CN} = (\mathbf{C}, \mathbf{I}, \mathbf{CS}, \mathbf{S}, \mathbf{M}_{4x4})$ holding all geometric input information in C, the constructed topology information in I, CS and inferred new geometric information in S. To aid later construction processes every curve network object within the scene is associated with a transformation matrix $\mathbf{M}_{4x4}$, that describes general global transformations of the object. This allows the easy definition and control of patterns in the network. General patterns are for example symmetry defined by any plane in the scene or general rotations. Multiple scene objects and patterns can be merged as the correct temporal drawing order and topology is implicitly given by the original object. Note that resulting merged shapes might not comply to all previously stated surface requirements.

As already defined it is assumed that all special features and discontinuities on the later surfaces directly relate to the information in **C**. Discontinuities are defined as locations with multiple normal definitions. Following this definition all locations within the shape, which are characterized by two normals are directly located on a curve, if more than two normals are available in the intersection of multiple curves. This situation is visualized in figure 10.



(a) Object p^0 Point (b) Object p^1 Point (c) Object p^n Point creation

Figure 10: Point types p^n which can occur during surface extraction, where n denotes the number of potential normals minus one.

If the two distinct surface directions in one curve point are equal, the curve can be considered being smooth in in this point. It is assumed, that smoothness is indirectly controlled by the directions of neighboring adjacent curve segments in the network and has to be maintained during the construction process. The overall surface construction process therefore can be reformulated as determining the position of a suitable number of $\mathbf{p}^0$ points to describe the modeled shape. The different point types are illustrated in figure 10.

II **Quad-Mesh Structure**

Assuming an initial surface topology, such as produced by procedures described in section 3.5, a representation has to be found, which allows to construct, parameterize and modify those surfaces. As the curve network only constrains the boundaries and initial tangent information there are still multiple possible configurations for the potential surface-shapes.

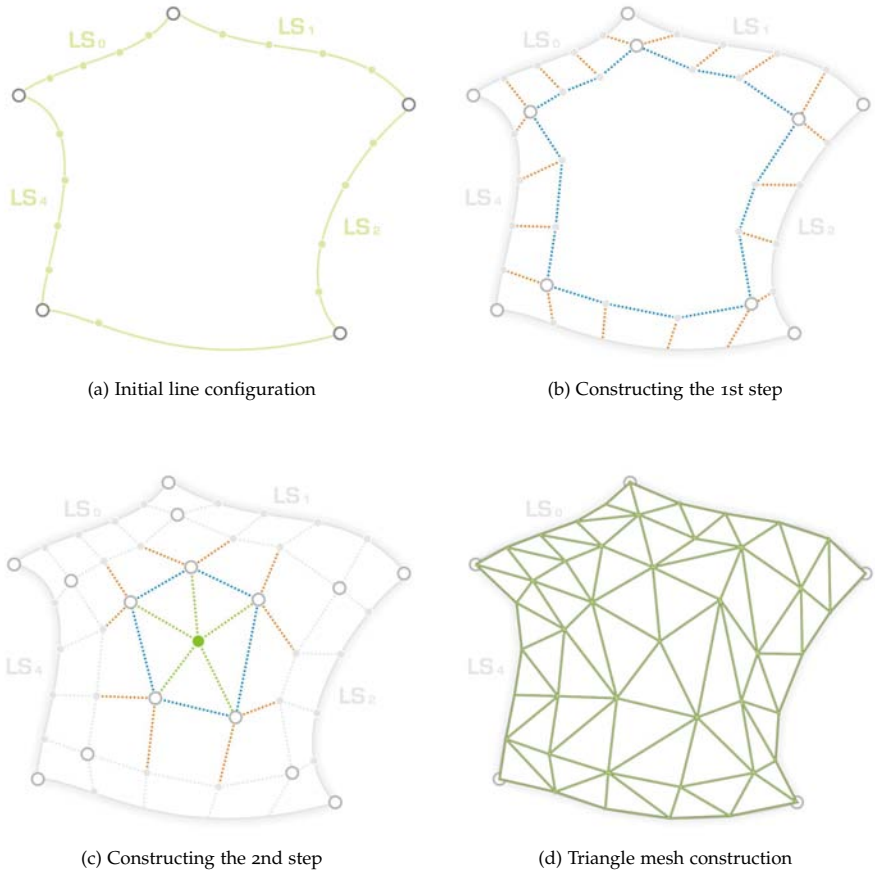
This leads to the following two consequences:

First, a representation has to be found which can handle surface boundary inputs with up to $n \in \mathbb{N}$ boundary curves. This description should be stable against missing or noisy input, as they are quite common during the conceptual modeling process. Noisy in this case relates on the one side to the amount of significant curvature changes on the surface. On the other side to neighboring curve segments, that will not always exactly match within enclosed intersections. This definition will allow gaps between neighbors in the corresponding segment sequence.

Second, a flexible description has to be found which allows steering of the surface outputs under different constraints. On important constraint is the continuity of the surface boundaries, which will dictate the final smoothness along the curve segments. As curves can be smooth and crease, both cases should be reflected in one description.

To address both challenges a subdivision-oriented quad mesh structure is proposed and interpreted by the means of a *spring-force network*. Similar mesh structures have been used by Das et al. [DDGG05] and Schaefer et al. [SWZ04] and have been shown to be well suited for this purpose.

The Quad-Mesh extraction and definition mainly consists of two steps: The initial topology determination and a subsequent force based iteration along the surface positions. The Quad-Mesh $\mathbf{M} = (\mathbf{mp}_0, \mathbf{mp}_1, ..., \mathbf{mp}_l)$ consists of a set of mesh elements $\mathbf{mp} = (p, \mathbf{mp}^*_{up}, \mathbf{mp}^*_{down}, \mathbf{mp}^*_{next}, \mathbf{mp}^*_{previous})$ which store a position $p \in \mathbb{R}^3$ and four links to their neighboring elements in the quad mesh. The interconnecting edges between those elements can be either assumed to be linear or defined as curve segments inheriting geometric properties from the reference curve segments.



(a) Initial line configuration

(b) Constructing the 1st step

(c) Constructing the 2nd step

(d) Triangle mesh construction

Figure 11: Different states during processing the input data

Following the definition of the construction scheme illustrated in figure 11, the points can be sorted in a level-like order. The level defines the number of down elements which are between the current mesh-element and its originating position on the boundary curve. The elements in the first level correspond to $\mathbf{p}^1$ network points, except elements positioned directly in an intersections. These represent $\mathbf{p}^n$ points, whereas all other elements denote $\mathbf{p}^0$ positions, which are characterized by exactly one normal direction.

The quad definition allows for global subdivision procedures increasing the overall resolution of the mesh. The sample resolution along the boundary curves is oriented at the curvature of the boundary. New elements can be inserted between two elements on one level, which requires insertion of the corresponding number of vertices in all levels above and below this element and vice versa for the insertion of elements in between two levels. Note that the valence per element should be kept constant as far as possible in order to maintain the advantages of the quad mesh topology [SWZ04]. For details on this subdivision procedure, please refer to section 4.5.

This definition of the surfaces allows for direct, flexible and force-based control of the whole surface appearance. In order to find expressive and reasonable shape configurations appropriate forces for this quad structure have to be found. From the inherent definition of the links two different types of surface force can be derived: One force describes the attraction forces $\mathbf{f}_{int}$ between elements within one level and the second one attraction forces between the elements of two different levels $\mathbf{f}_{ext}$. For $\mathbf{f}_{ext} = \mathbf{f}_{int}$ the iteration has a smoothing effect on the surface, which can be used to improve the visual quality of the surfaces.

The final translation vectors for an element are constructed as follows:

$$\mathbf{f}^i_{int} = (\mathbf{p}_{next} + \mathbf{p}_{previous}) \cdot \frac{1}{2} - \mathbf{p}_i \tag{3.1}$$

$$\mathbf{f}^i_{ext} = (\mathbf{p}_{up} + \mathbf{p}_{down}) \cdot \frac{1}{2} - \mathbf{p}_i \tag{3.2}$$

The final translation of the control point is described by the weighted sum of all applied forces:

$$\mathbf{p}^*_i = \mathbf{p}_i + \mathbf{f}^i_{ext} \cdot s_{ext} + \mathbf{f}^i_{int} \cdot s_{int} \tag{3.3}$$

One goal of the iterative process and force definition is to find a description which leads to an equilibrium state of the surface positions in which the shape converges to a steady geometric description. For the associated weights this can be achieved by choosing $\sum_{i=0}^{n} s_i = 1$.

Furthermore, there can be defined a set of additional forces to steer the general behavior of the surface. Inspired by definitions given by Farin et al. [FH99] forces can be expressed by discrete surface operators producing a minimal twist surface between the boundary segments.

An additional definition allow to approximate the continuity behavior along the input curves. Considering the normal- and tangent vector of the reference boundary segment, a general direction for the adjacent surfaces can be derived, by computing the cross product of these two given vectors. The derivation of the $\mathbf{p}^0$ elements in the first level from this surface direction, corresponds to the final continuity the surface will have on

this boundary. This effect will be demonstrated in section 5.2.

To avoid self intersection for mostly planar non-convex input patches, additional forces towards a medial or more general chordal axis definition can be defined as proposed in Teddy [IMT06]. Moving all defined quad patch-surface points into the direction of their closest position on the chordal axis, can help to visually improve the result and strive for avoiding intersecting surface elements.

III **Triangular-Mesh Structure**

The final structure to render and export the resulting shape surface is a triangular mesh. It can be reconstructed easily from the given Quad-Mesh by dividing every quadrangle into two corresponding triangles by connecting one pair of opposing quad elements. Doing so the quad boundaries which might have been described by freeform curves before are assumed to be linear. Therefore the resolution of the triangle mesh is oriented at the curvature of the boundary curves. As subdivision-procedures allow for an arbitrary level of detail, the resolution can be chosen in correspondence to the displayed size of the object and the available resolution.

Once the triangle structure is extracted the mesh normals are computed per vertex and stored in an appropriate data structure. Note that this mesh structure will not be subject to further deformation operations, in order to keep the global modeling metaphor. Every time the curve network is changed, corresponding meshes are updated or reconstructed if necessary.

The overall shape construction process and inherent conversion of representations will not be visible to the designer. As the only input to the construction process are 3D space curves, the next section will discuss the efficient creation of such curves within a 3D scene.

3.4 INPUT INTERFACE

As most of the approaches mentioned in section 2.4 the proposed system is sought to work in a general desktop environment using a standard PC using common input devices. There are two main input types, which are supported: Mouse and tablet based input data. In both cases the data consists of 2D sampled points which usually are sampled in window coordinates.

Sketch Curve Construction

A basic property of the proposed system is that the input data is mapped immediately into the current 3D scene. This represents a solution to the mapping problem, but also implies the necessity of curve control operations and additional geometry support to aid the curve construction process.

In order to achieve a higher variety of curve shapes the system will use additional *supporting geometry* to guide the construction of 3D curves during the drawing process. Starting with the first modeling steps during the creation phase this supporting geometry consists of a plane which can be moved and rotated by the user. Subsequently the plane will be aided by geometry created during the design process. With this metaphor a smooth transition between the process of creating new geometry, modifying and augmenting existing data can be assured.

The placement and control of the supporting plane is augmented by four different methods to align the plane within space. Note that the alignment of the plane is characterized by a center position $\mathbf{p}_{plane} \in \mathbb{R}^3$, a normal direction $\mathbf{n}_{plane}$ and two local main directions $\mathbf{v}_x, \mathbf{v}_y$. The user can further directly control the alignment of the plane by defining a rotation angle around both axis and a translation along the normal direction.

a **One-Point Method**

Using a single ray passing through the current camera position and the projected window coordinate in the scene, the system can determine the closest point from the curve network. For every point $\mathbf{p}_i$ on any curve within the scene at least one main direction can be identified. Using this method the plane is moved into this point and the $\mathbf{n}_{plane}$ is adjusted to the given main direction shown in figure 12a. If the closest position marks a point on a single curve $\mathbf{v}_{plane}$ is matched with the tangent direction of the line, otherwise the averaged normal in the intersection point is used or the plane direction is preserved.

b **Two-Point Method**

Given a second point in the network as illustrated in figure 12b, the plane is aligned, such that both points are positioned on one of the main rotation axis $\mathbf{v}_x$ and p_{plane} is positioned at the center of both points. This leaves one free angle which determines the plane orientation.

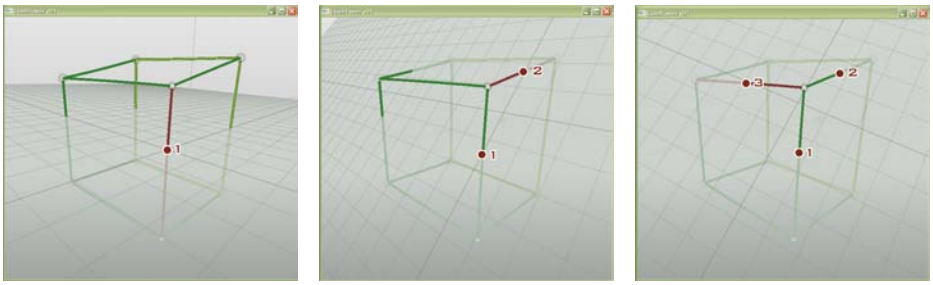
c **Three-Point Method**

One more point will uniquely determine p_{plane} as the centroid of all three points and leaves only two choices for $\mathbf{n}_{plane}$ where one of these normals can be chosen arbitrary. The result is depicted in picture 12c.

d **Extruded Surface method**

In order to achieve non-planar space curves the user can also extrude a supporting geometry surface out of a given space curve.

Given an additional angle $\alpha \in [0, 2 \cdot \pi]$ an extrusion direction can be defined by rotating a standard orthogonal normal direction around the tangent direction of the reference curve. The extruded surface is displayed within the scene and can be used to directly define a new curve on it. This procedure is shown in figure 13a.

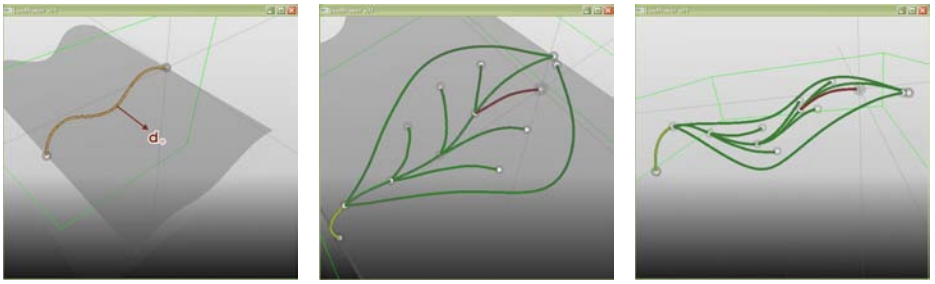


(a) 1-Point Method (b) 2-Point Method (c) 3-Point Method

Figure 12: Three methods of aligning the construction plane, while the choice of the method is determined by the number of active points given in the network.

The application of the described methods is shown in figure 12. The transitions between those interaction methods are seamless and rotations are interpolated by a quaternion description. An animation of this transition visualizes the change of the plane alignment in order to make this positioning procedure more obvious to the user.

Similar and some additional 3D curve placement techniques are presented in ILoveSketch [BBS08] and by Cohen et al. [CMZ+99]. Although the supporting construction geometry has to be controlled directly in the current system, some alignment methods can be inferred automatically while sketching. In order to sketch on the plane, the designer will choose camera positions which offer a large projection area of the construction plane, but avoid



(a) Initial plane extrusion (b) Defining new curves (c) Finished curves

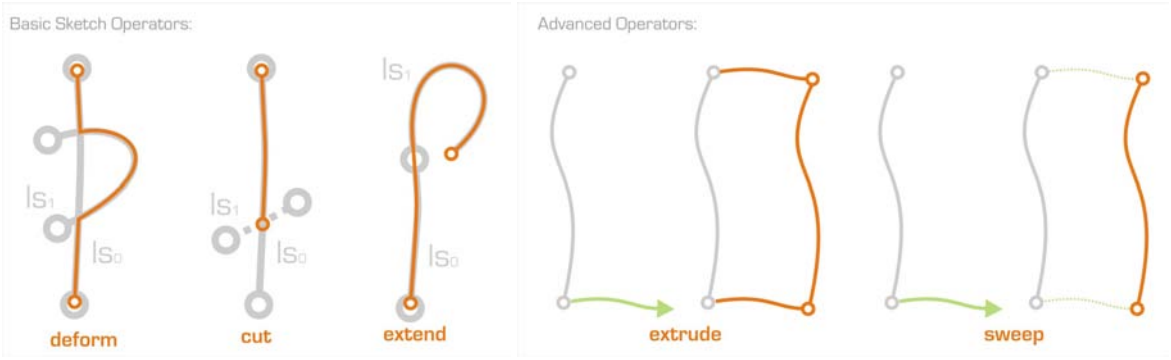
Figure 13: Different states while defining an extrusion drawing surface. The first image 13a shows the initial line and the extruded surface along the given extrusion direction, which is orthogonal to the normal direction. Figure 13b and 13c show the construction of new curves on that surface and their final spatial arrangement.

special configurations, where related lines are covered or only projected to a small area. This correlates to a plane normal $\mathbf{n_p}$ which is adjusted to the current camera viewing direction $\mathbf{v_d}$ plus a derivation depending on the surrounding curves. High angular derivations between $\mathbf{v_d}$ and $\mathbf{n_p}$ will lead to much larger stroke projections viewed from another perspective, so it can be assumed that this affect should be avoided. In general an inadequate drawing positions of the drawing plane can be determined by the associated grid pattern and corrected using direct rotation operations.

Curve Operators

The presented methods represent a fundamental construction toolkit which allows for general shape construction and setup of initial configurations. Basic concepts are extended by the ability to draw additional curves on created surfaces which allows for more complex curve constructions in later stages of the modeling process. In addition to those fundamental descriptions also the parameterized properties of the curve description can be exploited in order to construct a set of basic and advanced sketch operators shown in figure 14. Basic sketch operators can be used to simply interact with existing strokes in the network in order to modify existing shape structures. The *over-sketching* or *deform* method is already a well established feature in many SBIM systems ([ZNA07], [KS07], [BBS08], [CMZ+99], [MKB02]). This also holds for *cut* and *extend* operations which in this case follow out of the internal shape representations. If an intersection is added the corresponding curve is automatically split into two segments, if one of those segments is immediately deleted, this corresponds to the cut operation. If the segments is added to an existing intersections adjacent boundaries are extended by this segment.

Advanced operators shown in figure 14b might speed up and optimize the construction process on the cost of additional parameters and less intuitive application. The extrusion operator allows for copying complete curve structures while maintaining the topological structure. It has been shown to be a very effective and expressive modeling metaphor for conceptual shape structures ([SOD05] [OSD06], [IMT06],[SWSJ06],[FCAD03]). Parameterized sweeps are less appropriate within an optimized intuitive modeling workflow, but allow for fast construction of complex non planar curve configurations (e.g. screw curves or helix structures) which will mainly serve for testing purposes and are demonstrated in section 5.1. Further advanced operators could be defined, but should be balanced with their actual necessity as they will introduce redundancy into the basic configuration.



(a) Simple Operators

(b) Abstract Operators

Figure 14: Basic operators like the deformation or over-sketching, cut and extend operations are used to modifiy existing curves. The advanced operator extrusion displaces the reference line along the secondary curve, while parameterized sweeping can be used to create parallel and correlated curve structures.

In addition to these operator types another concept is introduced, which will be referred as *advanced copy operator*. If sketch-like data given in form of a curve network has to be copied, all 3D samples of the given curves have to be gathered and repositioned into a new location within the 3D scene. This relocation can be seen as a single transformation which is applied to all samples in the same way. This concept can be generalized by using not only one uniform transformation, but defining separate transformations for every sample on the reference curves to copy. A similar procedure has been proposed by Severn et al. [SSS06].

Those distinct translations can be described by mapping all curve elements onto one *copy-stroke*. This curve $\mathbf{cs}(t)$ is characterized by a parameter $t \subseteq [0,1]$, which gives access to the tangent directions $\dot{\mathbf{cs}}_\mathbf{0}(t)$ and stored reference normals acquired during the drawing procedure. This allows the description of all points around this copy stroke in a spherical coordinate system by assigning every points two angles $\alpha_{tangent}, \alpha_{normal}$, the parameter t_{min} for the closest location on the curve and a distance parameter d_{cs} encoding the closest distance to the copy stroke. Storing all points of the reference geometry in this description allows for the definition of unique transformations by simply drawing a second copy stroke $\dot{\mathbf{cs}}_\mathbf{1}(t)$ and transfer the stored spherical coordinates to the new stroke. This procedure can be seen in analogy to barycentric descriptions of general deformations [LBS06] and used for applications similar to operations described by Owada et al. [ONI06]. Taking the length of both copy strokes also into account additional scaling operations can be described with this concept. If the second copy stroke is to complex or characterized by high curvature areas, this method will produce self intersecting geometry. Note that this procedure is very fragile towards noise and irregular parametrization on the copy-strokes, therefore additional smoothing operations should be applied. Some examples for the use of this operator will be presented in section 5.1.

Sketch Filtering

As the main input is done by hand and sampled over the screen grid, the results might contain noise or general sampling artifacts. Therefore the prototype provides basic sketch processing methods for smoothing and restructuring of the input curves in relation to the existing network. The latter group might involve the alignment along general continuity constraints or definition of thresholds for existing curve interaction. In general these methods have to be balanced with the actual artistic and expressive style which might explicitly include those artifacts as part of the shape constructions process.

Although general free expressiveness cannot be provided, one major goal is to minimize the impact of these system related methods towards the conceptual shape creation. One way

to do this is the exploitation of *scale* while working with the system. In general very small features and detail will be unlikely to be drawn from a very high distance, but might require a zooming operation. This reduces the distance of the camera position assuming a constant angular aperture. This parameter can be used to scale general thresholds for noise reduction, sampling resolution and topology recognition. Despite this automatic scaling general system preferences should allow to tune those ratios towards the necessities of the designer.

Final Curve Processing

Basically there are two major approaches how to process the given input curves within the system.

In *batch processing* all input data is collected and only the first two steps of the sketch-modeling workflow in figure 7 are executed. This process is fully automatic and requires no further interaction. Therefore the user constructs the complete curve-network using the available interaction techniques without any interruption by the surface extraction process. Once the user has finished modeling of the intended part or object, the surface construction is triggered.

On the one hand the user does not have to be concerned about surface-construction issues while designing the curve network and the interactivity can be guaranteed at all stages of the network modeling process even for extended set of lines. On the other hand the user has no feedback or direct influence on the surface-extraction process and might be confronted with unintended results, as constructing ambiguous networks is not prohibited by the curve creation system. Furthermore, it can be assumed, that the construction process will consume a certain amount of time up to several seconds, depending on the complexity of the input network.

In contrast during interactive modeling sessions using a *progressive surface construction* seems more promising, because continuously tracking the surface creation process minimizes the risk for unintended results. This progress is semi-automatic, which means that it is constantly triggered while new sketch curves are added. The additional information required from the user is the constructional order in which the input lines are given. This order has to follow two constraints in order to allow for more intuitive results: First, the construction of topological holes in the object is not directly possible, as all closed sequences are immediately interpreted as surface. Second, connecting two different sequences results in a number of special cases, where some of them will not produce a intuitive result in general and require an additional geometrical analysis.

Therefore, this method raises the cognitive effort working with the system compared to the previous approach. As described in section 2.2 creating valid and unambiguous 3D objects is cognitive more demanding than sketching 2D views without caring too much about the topological correspondence and validity in 3D.

Considering working with freeform curves will always include the possibility to produce configurations which will not lead to an intuitive extraction process (e.g. self-intersecting shapes like Necker Cube or a modeled Klein Bottle). On the other hand it can be assumed, that modeling with the system not all special cases have the same probability to occur. For product and basic shape design large and straight curves will dominate the drawing process, whereas smaller curves with higher curvature frequency will be used more frequently for styling and modification of existing curves. Therefore it is useful to make some assumption about the input in order to be able to guarantee a usable result.

Restrictions are mainly resulting from self-intersections of the input data, that will lead to visually unclear curve configurations and ambiguous editing situations. In order to enforce these restrictions, one possibility is to filter and constrain the input data while modeling. Such restrictions are often very general and can be counterproductive when it comes to the support of individual artistic styles. Therefore preprocessing always has to be balanced with the goal of an expressive design workflow.

To reconstruct surfaces out of given curve networks is a well known problem, which has been dealt with in several application areas. Most work stems from the reconstruction of planar closed contour lines, such as segmented medical data sets. In general the goal is a smooth and solid extracted surface (e.g. [JWC+05], [BV07] and [LLB+08]).

Using these Shape representations for sketch based modeling introduces a set of new challenges which have to be considered. One major problem during the surface creation is the extraction of an intuitive topological mapping within the curve network which can be stated as the following question: Given a freeform network by a number of intersections and interconnecting segments, is it possible to find a unique and intuitive surface description which covers a larger set of general cases occurring during the modeling?

First of all, curves and hand drawn strokes do not have to be closed at all. Following section 2.3 it is not even guaranteed, to have a valid or closed configuration after the first design steps. Therefore, it is necessary to add flexibility to the already existing models and to define restrictions which frame the conditions under which an extraction is possible at all.

Problematic network configurations like self intersecting strokes, overlapping regions or high scale variation regions will lead to numerous special cases, where distinct approaches might not be able to produce a predictable result in general. Therefore a general solution of all cases cannot be guaranteed working with unconstrained free-form curves. In addition to this, the constructed input can be used to explore alternative shape configurations or support individual creation styles. This aspect should be generally supported by avoiding too restrictive creation rules and providing general space for the exploration of capabilities and boundaries of the underlying structure.

In order to store and reused former modeling data the system requires interfaces which enable loading previous data. This data can consist of two different kinds of information. The first one contains data concerning the curve network and supply information about geometric properties and its topological structure. As one important fact the curve-network format also has to store the type of curves and their interpolation method in order to reliably reproduce the entire network. The second one refers to the surface representation of the modeled object. This kind of data can be loaded in the beginning of the modeling process to start with an initial reference model to draw on. Therefore it is useful to support standard mesh formats.

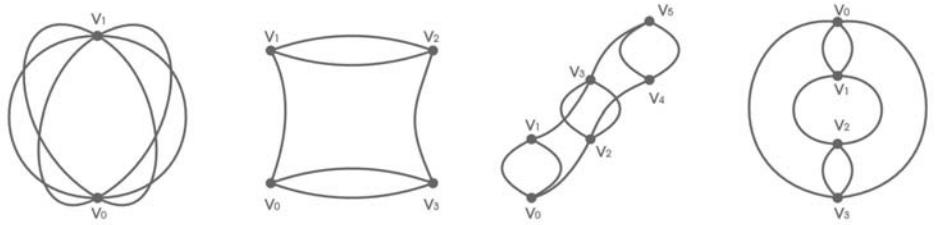
During the course of the thesis different extraction methods have been developed. In order to describe them the classification of batch- and progressive extraction methods will be used again.

Batch Extraction

The batch procedure can be stated as follows: Given a curve network $\mathbf{N} = (\mathbf{C}, \mathbf{I}, \mathbf{CS})$ we try to find a corresponding set of surface-patches $\mathbf{S}$ whereas every patch contains a sequence of adjacent or spatially related curve segments.

The assumption is that these segment sequences can be interpreted as minimal cycles in the undirected graph structure given by the network. The nodes in this graph are represented by the intersections as the curve segments are interpreted as interconnecting edges. A minimal cycle within this graph does not contain any other closed path sequence. This assumption will hold for most cases but will deliver intuitive results in all cases as it is purely topologically driven. Some examples for problematic networks are shown in figure 16. Note that this concept is independent from the dimension, the segment points are defined in.

In practice this method is only feasible for short path lengths. Increasing the length of extracted paths will raise the effort to determine further minimal paths and the risk for extracting counterintuitive surfaces. An intuitive surface hereby denotes patch-configurations we would guess out of the network-configurations. This is the surfaces represent a part of the outer hull of the final object and do not cross interior regions. For the example the network shown in figure 15a would probably expected to be a spherical object limited by six



(a) Example Network 1 (b) Example Network 2 (c) Example Network 3 (d) Example Network 4

Figure 15: Four different examples of basic topology networks. The images already reflect the general ambivalence resulting from the lack of additional geometric context which allows for multiple shape interpretations.

emerging edges and an equal number of boundary surfaces to be extracted. Following the minimal cycle assumption we will find those surfaces plus $n \cdot (n-1) = 30$ further minimal cycles with n denoting the number of edges in the network. One reason for this intuitive assumption is our mental concept of an 'inside' of this object which can only be expressed by geometric means. In general we would not expect any surfaces to pass the interior of the object.

In order to limit the number of surfaces found with this approach we can introduce two additional constrains:

a The number of possible surfaces is limited by the number of edges in the network. For every edge there can be only two surfaces adjacent to it. Therefore $|\mathbf{S}| \leq 2 \cdot n$.

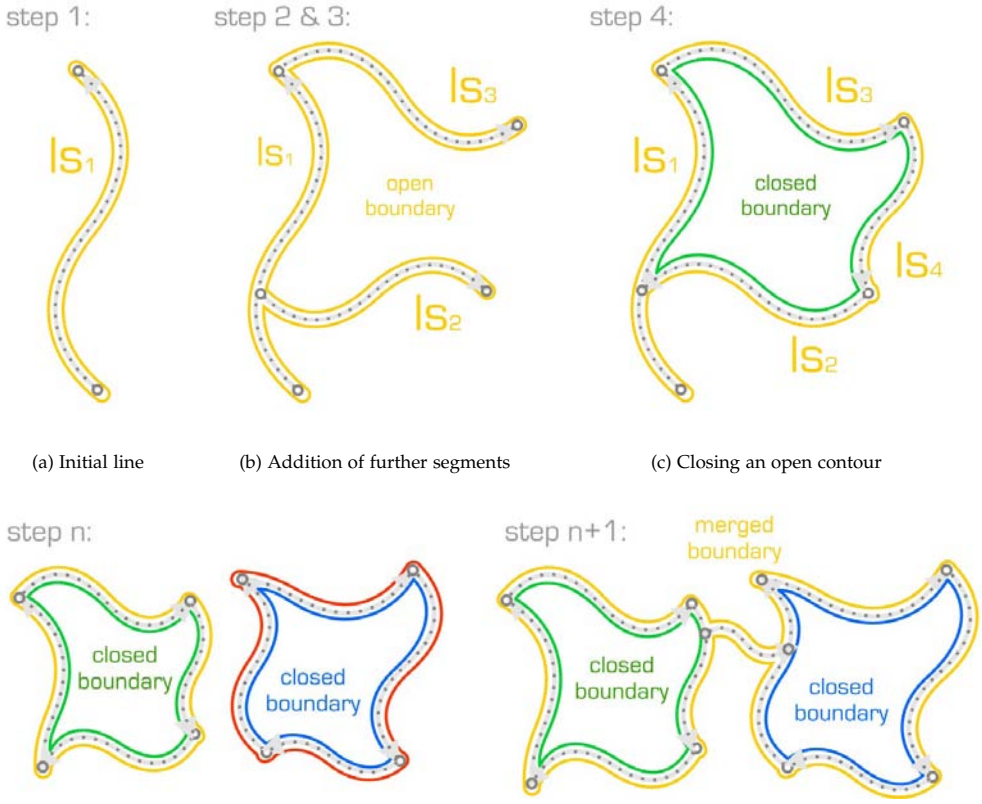
b In order to express the geometric relation of the edges we introduce the concept of the *tangent-loop*. This structure denotes an ordering of edges adjacent to an vertex in the network. All neighboring segments in a patch sequence are required to be neighbors in the associated tangent loop of the enclosed intersection.

This ordering is expressed by the notion of an angle $\alpha_k, k \in [1, |I|]$ around an average normal direction in i_k. In 2D this normal direction can easily be expressed by lifting the intersection position into 3D and rotate around the vector between the original position and its lifted version. In 3D the vector can be approximated by averaging the cross products of all incident edge-tangents. This might fail, if the average cross-product will become exactly zero, which means that multiple surface configurations are possible to be reconstructed. Despite in practical modeling with freeform curves, this issue is rather unlikely to occur. If this case does occur it can be resolved by additional user interaction.

Progressive Extraction

In order to minimize too many counterintuitive results from the batch process, progressive tracking of the surface construction process during the modeling allows for immediate correction. To construct the surfaces their boundary is assumed to be a closed curve segment sequence at all time which is referred to as *active boundary*. Combining existing lines within the network results in extending the existing boundaries by the newly created segments and introduces double-edges in the Active Boundary sequence. This approach extends the concepts presented by Sheng et al.[SWS08]. If an edge connects two vertices of the same border the patch sequence is closed in a circular way including the new edge, which produces a new patch sequence and changes the segments within the existing one. This procedure is illustrated in figure 16.

Although this approach is rather intuitive and easy to process, there are some limitation concerning the shape class of the object. Shape features with holes, large open boundary



(a) Initial line (b) Addition of further segments (c) Closing an open contour

(d) Two separate networks (e) Example Network 3

Figure 16: Progressive surface extraction during different modeling steps. Image 16a and 16a show the basic extension of the active boundary, 16c shows the effect of connecting two boundary vertices. Note that all segment sequences are closed during all steps, occasionally including the same segment multiple times.

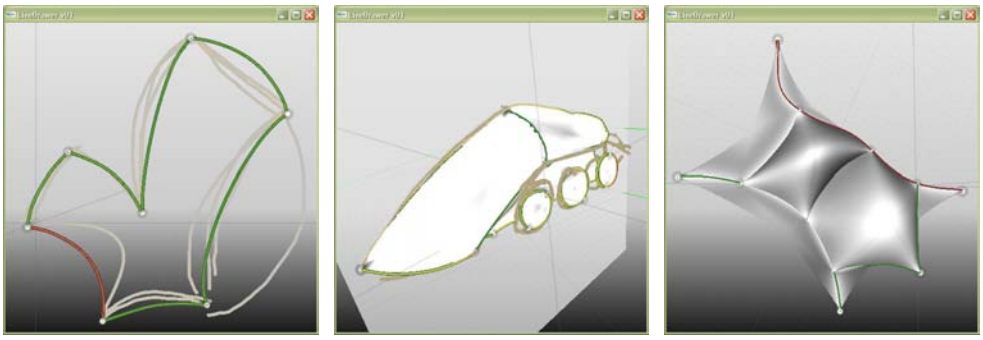
sequences or enclosed edge repetitions might potentially produce unexpected results. An detailed discussion of such objects will be presented in 6.3

Besides the internal shape representation and the interaction procedures the visual representation of the modeled objects are a central aspect of the system. In general, the main purpose of the visual representation is the qualitative and spatial analysis of the constructed geometry. As the user has to decide about further modeling steps based on this representation, it is required to faithfully represent shape features and allow the designer to concentrate on the shape construction process.

The proposed system balances the reduction of visual- and interface complexity against feedback for general processing results. Furthermore, the focus is on fast and accessible correspondence between the network topology structure and the rendered mesh, in order to guarantee interactive editing of the object. Interactivity also allows for improved evaluation of shape properties, as continuous changes of the viewing position support the spatial understanding of the displayed shape. This helps to avoid ambiguous depth perception issues, that may result from unfavorable camera positions like the image shown in figure 2c in chapter 2.2.

Visual Sketch Augmentation

Besides shading and interactive aspects the additional geometry described in section 3.4 further supports the estimation of spatial relations by displaying a semitransparent grid of constant size within the drawing plane. As the plane is rendered semi-transparent parts behind that plane will appear faded, which allows for easy distinction between separated parts of the geometry. This effect can be depicted from the images shown in section 3.4 in figure 12. For more complex objects and modeling of interior structures this might be extended towards a cut-and-slice based approach to completely fade out parts which are located in front of the supporting plane.



(a) Erased curves (b) Pre-Sketch lines (c) Incomplete Surfaces

Figure 17: Three examples for visual augmentation of the construction process. Image 17a shows the embedding of erased sketch curves, similar to figure 17b where pre-sketched reference curves are displayed, whereas figure 17c shows unfinished surfaces during the sketch process.

Another feature inherited from traditional sketching processes with pen-and-paper media has been proposed by Schmidt et al. [SIJ+07]. The authors argue that deleted or corrected sketch lines can support subsequent drawing procedures as they provide reference to previously constructed shape features. Therefore the modeling data is not directly erased, but kept in an additional inactive sketch object and rendered semi-transparently into the scene as shown in illustration 17a. Additionally, styling-strokes used for curve- and surface defor-

mation can be used to pre-sketch shape features, as they are not integrated directly into the shape creation process as illustrated in figure 17b.

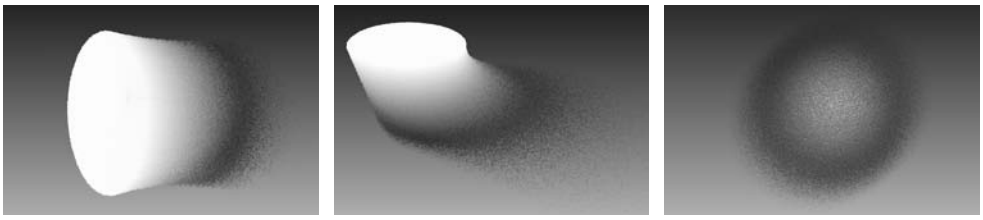
As described in section 3.3 the extracted surface structure is converted into a triangular mesh. This mesh can be rendered using standard shading approaches, whereas the focus is on representing surface features and overall shape properties. Texturing and additional color information is assumed to be added within later workflow stages, this is beyond the scope of this system.

Shape Ambiguity

Another issue which has been dealt with during the course of this thesis is the visualization of incomplete and ambiguous parts of the curve network. First thoughts on fuzzy shape representations are proposed by Reeves et al. [Ree83], more recent work is presented by Lum et al. [LSM02] and is main subject to many NPR visualization techniques. As explorative ambiguity is an integral part of expressive conceptual artwork, their extended application within corresponding modeling systems remains quite promising. Corresponding attempts can also be found in prominent systems such as Teddy [IMT06] and ShapeShop [SWSJ06]. Therefore two general approaches have been tested during the development:

First a general definition of *surface ambiguity* can be used to determine the visual properties of an object. Visual ambiguity can be expressed by objects which offer only a diffuse structure an unclear depth and spatial properties distributed over the region, where the shape might still be represented by geometry. This relates to the observations made in section 2.3 concerning early conceptual explorative development. Within a volumetric particle based representation ambiguity could therefore be captured by continuously fading out shading cues and randomly displacing surface features within the relevant construction region.

Second as this concept cannot be directly transferred to a surface representation one parameter remains the transparency by fading out general shading information and the general behavior of the surface during force based iterations proposed in section 3.3. While the opacity of the surface denotes the distance to the originating construction lines, quad mesh particles with no geometric support are drawn towards existing fixed curve structures. The result of this procedure is presented in image 17c.



(a) Continuous Shape Ambiguity (b) Ambiguous Cylindrical Shape (c) Sphere with Solid Core

Figure 18: Above three test renderings for ambiguous shape representation are illustrated. Image 18a shows increasing shape ambiguity from left to right, figure 18b shows ambiguity distribution over a larger area, while illustration 18c depicts a spherical volumetric object with a solid core.

Despite promising properties those features do not improve the general construction process in a significant way. Although they aid the detection of open shape segments, the surfaces constructed from long, unclosed segment sequences tend to show rather unexpected behavior like self-intersections, due to the missing boundary support. This could be avoided by enforcing further geometric constraints, which would consume a considerable amount of computing

resources. As interactive drawing is within focus of the system these features have not been subject to further development. However, alternative representations (like the particle test renderings in figure 18) might support the active involvement of such features. Especially the numerous parametrization possibilities for resulting network surfaces cannot be captured within the current system sufficiently.

Finally, further examples of the general modeling procedures and their visual representation will be provided by section 5.2.

Chapter 4

IMPLEMENTATION

4

IMPLEMENTATION

Having defined the fundamental concepts and guidelines for the sketch modeling prototype in the previous part, the detailed implementation issues will be subject to this chapter.

The practical implementation first involves the basic decision which framework and tools to use, in order to realize the given concepts. Second, the initial concepts have to be mapped to a basic framework and it has to be decided which components are necessary to provide the desired functionality. The defined concept structure of the system described in section 3.1, is used as reference for implementation. Based upon this, the practical integration of those components will be discussed in section 4.4 focussed on their distinct purpose throughout the shape extraction process. Finally the rendering and visual extraction of the processing results will be explained in section 4.5 concluding the implementation chapter.

4.1 BASIC IMPLEMENTATION FRAMEWORK

To support the basic requirements for interactive speed and flexibility of the system C++ [1] has been chosen as fundamental implementation language. For visual representation the Open Graphics Library (OpenGL) [2] has been selected as common standard Application Programming Interface (API) among visual software development, offering speed as well as enough flexibility to achieve the desired properties. Further the prototype has been developed in Windows XP using Microsoft Visual Studio 2005.

Besides fundamental requirements an additional framework is useful to support basic structural managing of the 3D scene and rendering primitives. For this purpose the Open Scene Graph (OpenSG) [3] framework has been chosen. OpenSG offers convenient features as fast and accessible 3D scene management, interfaces for ex- and importing scene geometry from- and to standard formats, the support of standard modular rendering pipelines and multi core processing. Additionally, a basic mathematical tool set is provided supporting most of the required descriptions for handling general 3D transformations and basic 3D vector space algebra.

Although direct integration of the prototype modeling functionality into an existing 3D modeling system would have been a valuable alternative, the fundamentally different interaction principles and interactive extraction concept does not comply with most scripting interfaces provided by larger packages. Further the internal definition and fast conversion of different shape representations is mostly limited to existing descriptions.

1 Extensive documentation available at http://www.cplusplus.com/
2 SGI, http://www.opengl.org/
3 OpenSG Scene Graph, http://opensg.vrsource.org (accessed at 06.2009)

The final structure of the modeling prototype has been derived from the schematic concept given in figure 6 of chapter 3.1. This structure includes all implemented classes as shown within the overview in 19 to organize the input data and allow for fast data processing.

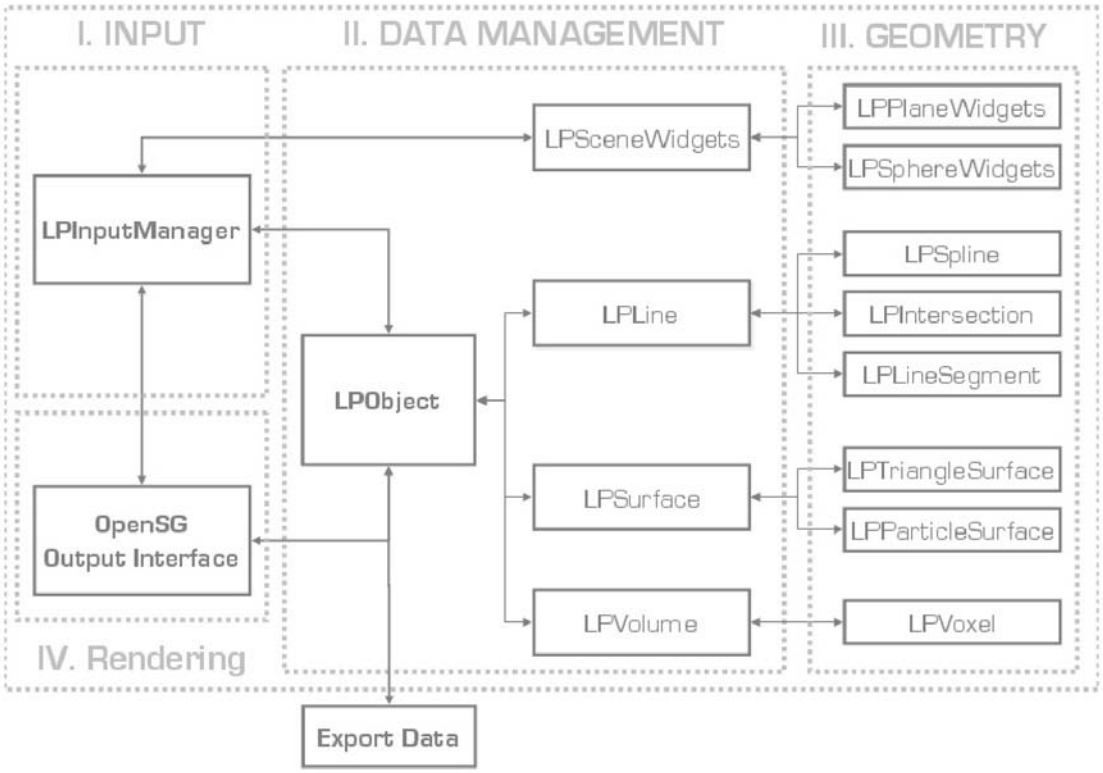


Figure 19: Overview of all implemnted system classes and their inherent relation.

The main components are given as follows:

I `LPInputManager`

This central module provides four main functions: Initiating and loading the scene parameters, handling of triggered key- and mouse motion events, and handling of system state functions, as the behavior during phases with low Central Processing Unit (CPU) and Graphics Processing Unit (GPU) workload. Keyboard- and mouse events are in most cases directly mapped to the corresponding functionality within the scene object or widgets, while the input manager additionally updates visual support objects during the geometry construction. One central aspect is the fast projection of 2D samples into the scene geometry, which is done by means of a ray-intersection test with the active parts of the underlying scene graph. Finishing the drawing process, initiates the construction of new geometry and the addition of those parts into the current network. During processing idle times, general animation updating and surface refinement steps are performed, which do not have a high priority during construction.

II `LPSceneWidgets`

The definition and handling of the augmenting geometry described in section 3.4 is taken by this module. This includes the handling of plane movement events within the scene and the visual support for representing intersections and topologically significant parts of the network.

II `LPObject`

The complete curve network structure of a single object is handled within this class. By standard definition all active drawing lines are handled from one object. Every object is expanded by additional links to subsequent pattern objects (like symmetry or rotational patterns), whereas all geometry operations are directly distributed to those references during execution. Former the object manages general shape operations, which depend on global scene geometry and the addition and deletion of topological and geometrical structures. Besides the main active scene geometry object, additional ones are defined for styling and command strokes and one object for deleted strokes and visual history. These objects do not have immediate impact or reference to internal structures. Additional objects for further geometry layers or imported networks can be defined on demand.

i `LPLine`

As central modeling primitive, the `LPLine` class provides acquisition, storage and shape operation functionality. Every line object handles an internal subclass `LPSpline`, which is used to reconstruct new geometry information and provides basic access to geometric parameterized curve operations. Every `LPLine` instance provides basic geometric requests, like for positions and derivations on the curve, or higher-level operators as distance- or projection requests basing on a local parameter $t \in [t_{min}, t_{max}]$. In general t_{min} is set to zero and t_{max} to one, in order to speed up general requests those parameters can also be limited to certain sub-segments of the line (e.g. by evaluating spatial correlations within the `LPVolume` grid). The further creation of topological entities as `LPIntersection` and `LPLineSegment` is not directly controlled by the input, but depends on the spatial relation between multiple lines.

ii `LPSurface`

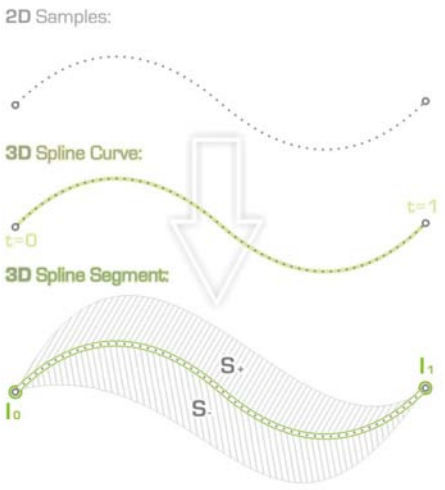
The creation of surface elements is taken by the `LPObject` class and allows for the definition of different types of surfaces. Network related, extracted surfaces are represented as triangular and quadrangular meshes, whereas quad meshes are generalized to topology based particle surfaces. Unstructured particles surfaces are further used to display general augmenting information like surface normals, displacement vectors or sample distributions.

iii `LPVolume`

The volumetric object class contains a general octree and voxel grid structure which can be used used to speed up general geometric requests in more complex networks, merge multiple scene objects or evaluated spatial relations of geometric structures within the scene. The sub-class `LPVoxel` provides general volumetric requests, handling of multiple scalar values as well as splitting and merging functionality.

All objects are structured as implied by the hierarchy given in figure 19, whereas generalized local access of single classes is provided by local back referencing to top-level classes as `LPInputManager`, `LPObject` and `LPSurface`.

The conceptual procedure of the data processing has already been outlined in chapters 3.1 and 3.3. Within this section the handling of the input data and implementation of the sketch modeling workflow is explained in further detail. Thereby the focus will be on the construction of spline representations from the sketch input data and the basic construction of patch surfaces, bounded by spline segments.



(a) Input processing steps

Figure 20: This figure demonstrates the detailed input processing steps for a new line.

Spline Construction

As mentioned before and illustrated in figure 20 the sampled 2D data and the resulting 3D projected points are stored in order to construct a cubic Catmull-Rom Spline [CR74]. This class of splines allows for local control and interpolates the given sample points directly, without the construction of an additional control structure. The only information that it needs to be stored, are the projected 3D points within their temporal order. The system also supports alternative spline descriptions, but as the sampling rate usually is high enough, the internal choice of the curve representation does not have an crucial impact on the visual result.

The construction rule for Catmull-Rom splines assumes the tangent of the interpolating curve in an given point to be the vector between the preceding and the following point on the curve. Formally this can be expressed in matrix form as follows:

$$\mathbf{p}(\mathbf{t}) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -r & 0 & r & 0 \\ 2r & r-3 & 3-2r & -r \\ -r & 2-r & r-2 & r \end{bmatrix} \begin{bmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \end{bmatrix} \tag{4.1}$$

Thereby $t \in [0,1]$ denotes the parameter on the curve and $r \in [0,1]$ the so called tension parameter, which describes the sharpness of the curve following the input samples. Due to the high sampling rates the influence of r on the appearance of the whole curve is minimal, but increases with the lack of available input data. By default value r is set to $\frac{1}{2}$, yielding smooth and visually pleasing results. Given a parameter t the index is given by $i = \lfloor t \cdot n \rfloor$,

while n denotes the number of sample points on the line.

First- and second derivatives can be inferred directly from the given description:

$$\dot{\mathbf{p}}(t) = \begin{bmatrix} 0 & 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & \frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \end{bmatrix} \tag{4.2}$$

$$\ddot{\mathbf{p}}(t) = \begin{bmatrix} 0 & 0 & 2 & 6t \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & \frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p_{i-2} \\ p_{i-1} \\ p_i \\ p_{i+1} \end{bmatrix} \tag{4.3}$$

Given equation 4.2 the tangent of the curve at position $t \in [0, 1]$ can be computed directly, whereas equation 4.3 denotes the second directional vector for constructing the osculating plane normal at spline position t. Note that the direction of this plane changes in extremal points of the curve and behaves rather unexpected in noisy input data as only $\mathcal{C}^1$ continuity is guaranteed by the Catmull-Rom definition. This fact makes it not a good choice for bordering normal conditions on multiple patches. Instead the normal direction for boundary constraints are gathered during the ray casting procedure and recomputed during the acquisition of the input data. Note that for points, which do not have local support such as p_1 and p_{n-1}, the missing positions are mirrored around the linear tangent direction of the first, respectively the last point, while p_0 and p_n will not be moved at all.

Spline Operators

This local definition allows access to interpolated curve positions and derivations in constant time, independent from the number of curve samples. Further this parameterized description offers additional functionality which can improve the interaction with the given curve data:

- **Iterative Smoothing**

  Assuming that every sample point lies directly on a spline description, this can be used to smoothen given input. To do so for a given curve $\mathbf{L} = (p_0, p_0, \ldots, p_n)$ all points are iterated for every point $p_i, i \in [2, n-2]$ a translation vector v_i is determined by computing a new position p_i^* with equation 4.1 by using $p_{i-2}, p_{i-1}, p_{i+1}, p_{i+2}$ and $t = \frac{1}{2}$ as input.

  If p_0 and p_n are fixed, this has two effects: First, the sample distances will converge to a constant value $c = \sum_0^n \frac{1}{n-1}$, the more often the smoothing operation is applied. Second, the curvature variation of the curve will reduce with every smoothing step, vanishing high frequency features and noise within the first iterations. Applying the operation more often, the curve will converge towards a straight line between p_0 and p_n, where the number of necessary iterations depends on the number of samples and the initial curvature.

- **Resampling and Sample Reduction**

  In analogy to the smoothing operation, also new points can be added to an existing curve using the same scheme. The only difference is, that a new point is inserted into the curve at the computed position $t = \frac{1}{2}$. Note that depending on the initial curve

sampling, this will amplify over-sampling effects in areas where already many samples are defined.

As the sample frequency for the initial curves is already sufficient in most cases, this procedure should only be necessary for external curve data, stemming from different spline descriptions or repeated excessive styling operations. More likely, the initial sampling even produces locally much more samples than required, therefore the number of points can be reduced by leaving out distinct points during the spline construction, which should be oriented at curvature and sampling distance.

- **Distance and Length Approximation**

  As for the computation of curve distances and intersections fast processing times are crucial, this parameterized description offers distinct advantages. The straight-forward distance computation approach would assume straight lines between all input points p_i, determining the closest point within $\mathbf{c}_j$ and then compute the closest point on the next and previous line segment.

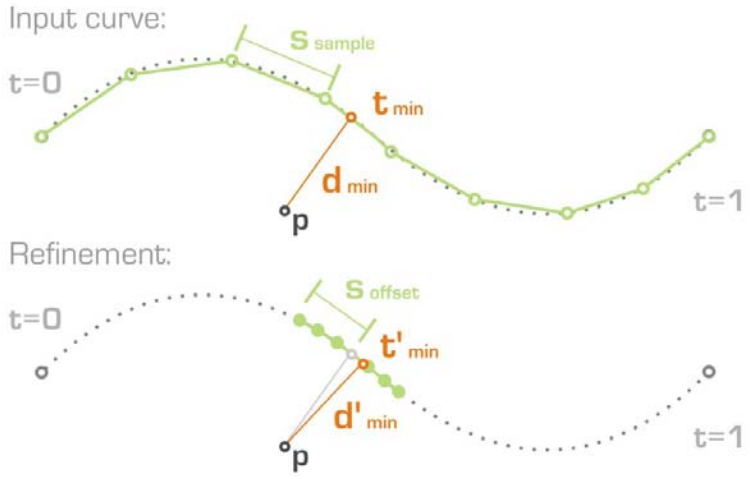


Figure 21: Fast distance approximation procedure for 3D curves.

Depending on the number of samples, this can increase the processing time, as this approach does not take the length and the curvature of the processed curve into account. To speed up this process the procedure illustrated in figure 21 is applied within the current system. First a subdivision factor $sd_i \in [0,1]$ is defined. Second the curve is split up into $1/d_i$ line segments using the Catmull-Rom scheme and the minimal distance d_{min} to the reference point and its associated curve parameter t_{min} is determined. Using a predefined offset $so \in [0,1]$ the same procedure is applied again to the curve segment from parameter $t = max(t_{min} - so, 0)$ to $t = min(t_{min} - so, 1)$. This allows for fast and successive approximation of the curve distance in constant time without depending on the number of curve samples.

Note that sd_i and so should be chosen with respect to length and curvature of the current curve once the curve is created or modified. If the step size sd_i is chosen too large, sampling artifacts are more likely to occur and might lead to inappropriate results.

- **Line and Surface styling**

The parameterized description of the strokes also allows for blending operations of two strokes. These operations can be used for curve styling of already defined network lines, which follows the concept of over-sketching presented in chapter 3.4.

To accomplish this behavior, a reference curve in the network $\mathbf{l}_{ref}(t_r)$ and a styling curve $\mathbf{l}_{style}(t_s)$ has to be defined. In the first step, the closest reference positions of $\mathbf{l}_{style}(0)$ and $\mathbf{l}_{style}(1)$ is acquired, using the previously described method. The reference positions define two parameters on the target curve (t_r^{min}) and $(t_r^{max}) \in [0,1]$ which describe a new curve segment with a local parameter, that is interpreted as ROI for the styling process.

All samples lying on this segment are translated using a predefined style blending function $\mathbf{b}(t)$ with $t \in [0,1]$ to avoid sudden changes in the original target curve. The function given in formula 4.4 describes the mapping of the segment parameter t to an blend value $b \in [0,1]$, which encodes how far the target curve is translated towards the reference style curve, while b_{max} denotes the maximum translation value and $d \in [0, \frac{1}{2}]$ the extend of the blending region. The graph of this function is shown in image 22.

$$\mathbf{b}(t) = \begin{cases} b_{max} \cdot (-\frac{1}{2} \cdot \cos(t \cdot \frac{\pi}{d}) + \frac{1}{2}) & 0 \leq t \leq d \\ b_{max} & d < t < 1-d \\ b_{max} \cdot (-\frac{1}{2} \cdot \cos(((1-t) \cdot \frac{\pi}{d}) + \frac{1}{2}) & 1-d \leq t \leq 1 \end{cases} \tag{4.4}$$

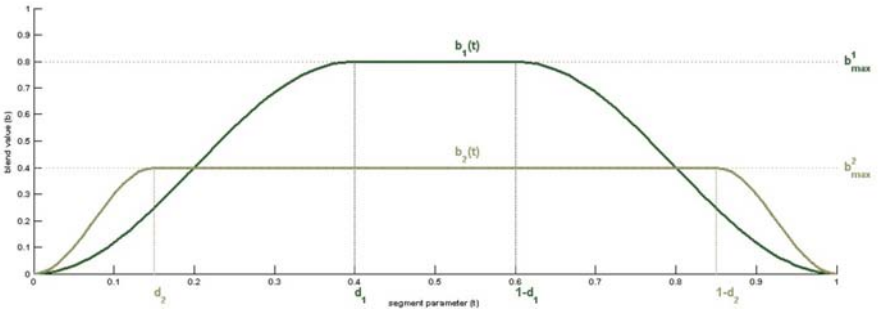


Figure 22: Two examples of blending functions $\mathbf{b}_1(t)$ and $\mathbf{b}_2(t)$ which can be used for lines and surface styling.

A similar procedure can be used to interact with already existing surface within the network. Therefore an additional force $\mathbf{f}_{style}$ can be defined correspondence to the quad mesh definition in section 3.3, which is used to translate the surface samples towards reference line positions. The results of this procedure will be presented in chapter .

Repeated application of this operation of this restyling operator might necessitate subsequent resampling of the corresponding curve or surface structure.

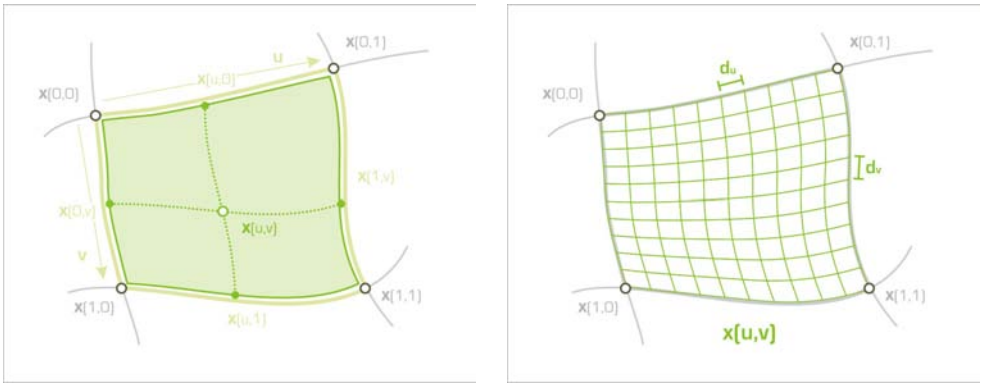
This central definition of space curves within the system allows for the construction of more complex geometric objects which result from spatial relations of the defined input curves. Those spatial relation are estimated immediately as a new curve is inserted to the network structure and are expressed within the `LPIntersection` class, introduced in the previous section. The details of this spatial relation analysis and the resulting topological extraction will be presented within the next section.

Once the lines are inserted to the network the surface construction process can be triggered. As presented in section 3.4 a set of cyclic edge sequences can be found, which serve as boundaries for subsequent surface descriptions and hold all points in $\mathbf{P}^k, k \geq 1 \cap k \in \mathbf{N}$, while n_b denotes the number of bounding segments. Note that each line segment should not occur more than twice within two different patch cycles. In a first approach surfaces bounded by four or less input curves can be represented as so called Discrete Coons-Patches [FH99] as illustrated in figure 23. As they offer high quality parametric surfaces, they have been selected as a first approach to render the extracted shape information.

A discrete Coons-Patches $\mathbf{x}(u,v) \in [0,1]x[0,1]$ with four boundary curves $\mathbf{x}(0,v)$, $\mathbf{x}(1,v)$, $\mathbf{x}(0,u)$ and $\mathbf{x}(1,u)$ is defined as follows:

$$\mathbf{x}'(u,v) = \begin{bmatrix}(1-u) & u\end{bmatrix} \begin{bmatrix}\mathbf{x}(0,0) & \mathbf{x}(0,1)\\ \mathbf{x}(1,0) & \mathbf{x}(1,1)\end{bmatrix} \begin{bmatrix}(1-v)\\ v\end{bmatrix} \tag{4.5}$$

$$\mathbf{x}(u,v) = \begin{bmatrix}(1-u) & u & (1-v) & v\end{bmatrix} \cdot \begin{bmatrix}\mathbf{x}(0,v)\\ \mathbf{x}(1,v)\\ \mathbf{x}(u,0)\\ \mathbf{x}(u,1)\end{bmatrix} - \mathbf{x}'(u,v) \tag{4.6}$$



(a) Patch Defintion (b) Patch Construction

Figure 23: The construction of an discrete Coons-Patch out of four freeform boundary normals. The parameters d_u and d_v denote sampling rate in both directions and therefore the resolution of the mesh.

Now a set of sampling points $\mathbb{P}^0$ can be defined using a given sampling rate d_u and d_v, which complies to the previously stated definition. Based upon this a triangular mesh structure is constructed on the sampled points. The normals of the resulting surface can computed directly by using the first derivations in u and v direction of equation 4.6 and **??** and compute the cross-product out of both vectors.

The resulting net of points can be expressed by means of a matrix:

$$\mathbb{P}^0 = \begin{bmatrix} \mathbf{p}_{1,1} & \mathbf{p}_{2,1} & \cdots & \mathbf{p}_{n-1,1} \\ \mathbf{p}_{1,2} & \mathbf{p}_{2,2} & & \mathbf{p}_{n-1,2} \\ \vdots & & \ddots & \vdots \\ \mathbf{p}_{1,m-1} & \mathbf{p}_{2,m-1} & \cdots & \mathbf{p}_{n-1,m-1} \end{bmatrix} \tag{4.7}$$

Patch Surface Operators

This description allows the intuitive application of a discrete filter operator, by moving the position for every point $\mathbf{p}_{i,j}$ with $i \in [1, n-1], j \in [1, m-1]$ towards the centroid of the four neighboring points. Note that the points on the boundary stay fixed, as they have to be interpolated. Formally this can be stated as:

$$\mathbf{v}_{i,j} = \frac{\mathbf{p}_{i+1,j} + \mathbf{p}_{i,j+1} + \mathbf{p}_{i-1,j} + \mathbf{p}_{i,j-1}}{4} - \mathbf{p}_{i,j} \tag{4.8}$$

$$\mathbf{p}_{i,j} = \mathbf{p}_{i,j} + \mathbf{v}_{i,j} \cdot s_{i,j} \tag{4.9}$$

On the one side this definition can be used as a smoothing operator to achieve a smoother surface appearances in case the input boundary still contain noise or jittering. This can be applied to all Quad-Mesh definitions (see chapter 3.3) made throughout the thesis, independently from the number of boundary curves. On the other side applying this filter multiple times choosing $s_{i,j} = 1$ lets the corresponding surface converge against a *Minimal Spanning Surface*, which means that the distances between all points on the mesh are globally minimized [Far02]. An example is shown in figure 24b. Despite this is not the most intuitive parametrization it offers distinct advantages: Missing boundary information can be handled easily by inserting additional points, but not fixing their position. This can be used to approximate incomplete surfaces as done in chapter 3.6 figure 17c. Further smaller changes within the course of the boundary will only have minimal impact on the whole surface, it will generally avoid self-intersections and will be positioned within the convex hull of the boundary contour.

Once the basic topology of all $\mathbf{p}^0$ points is determined their final position depends on the parametrization of the Coons-Patch. In order to define an appropriate parametrization oriented at the definition of the Coons-Patches the following assumption is made: We are starting from the linear quadrangle defined by $\mathbf{x}(0,0), \mathbf{x}(0,1), \mathbf{x}(1,0)$ and $\mathbf{x}(1,1)$ where every position can be determined by $\mathbf{x}'(u,v)$. On every edge of this quadrangle a freeform-curve is defined which might differ in its positions from the associated line, but has exactly the same position within the corner points.

This variance from the base line is expressed as vector $\mathbf{n}(u,v)$ for example for boundary $\mathbf{x}(u,0)$ by $\mathbf{n}(u,0) = \mathbf{x}(u,0) - (u \cdot \mathbf{x}(1,0) + (1-u) \cdot \mathbf{x}(0,0))$. This vector can be defined for all points on the base quadrangle by means of bilinear interpolation for all boundary curves. This circumstance is expressed in equation 4.10. Note that the length of this vector is weighted with the parameter distance of the base position, therefore the resulting patch will always

interpolate the four boundary curves.

$$\mathbf{x}(u,v) = \mathbf{x}'(u,v) + \begin{bmatrix} v & (1-v) & u & (1-u) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{n}(u,1) \\ \mathbf{n}(u,0) \\ \mathbf{n}(1,v) \\ \mathbf{n}(0,v) \end{bmatrix} + \mathbf{bv}(u) + \mathbf{bu}(v) \tag{4.10}$$

$$\mathbf{bv}(u) = (u \cdot \mathbf{n}(1,v) + (1-u) \cdot \mathbf{n}(0,v)) \cdot (|\mathbf{n}(u,0)| + |\mathbf{n}(u,1)|) \cdot \frac{\beta \cdot k}{2} \tag{4.11}$$

$$\mathbf{bu}(v) = (v \cdot \mathbf{n}(u,1) + (1-v) \cdot \mathbf{n}(u,0)) \cdot (|\mathbf{n}(1,v)| + |\mathbf{n}(0,v)|) \cdot \frac{\alpha \cdot k}{2} \tag{4.12}$$

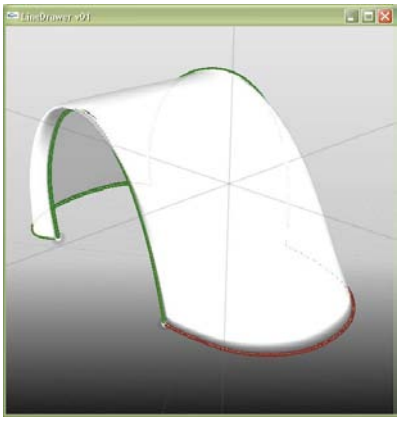
Once this vector is computed it is interpreted as displacement of the original base plane positions. To further gain control over the final shape two parameters α and $\beta \in \mathbb{R}$ are introduced which weight the influence of the boundary curves in v and u direction. For $\alpha = \beta = 0$ we will get our original Coons-Patch. An alternative parametrization has been described in the original publication [FH99] which has been modified for our purpose. Further a scaling factor $k \in \mathbb{R}$ is defined to limit the impact of the displacement. The result of this procedure are shown in figure 24, where k has been set to $\frac{1}{10}$.

Balancing Parametrization

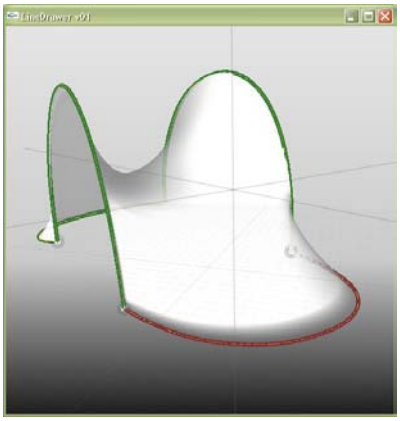
The results of those procedure already proof the tremendous impact of the parametrization on the visual appearance of the final shape result. As it has been argued by Farin et al. [FH99] those parameterizations can be used to achieve "good" shape results, which refers to the presence of defined boundary features in the final result. For example the original Coons-Patch shown in figure 24a tends to have a linear silhouette features, in contrast to example 24c where boundary features are preserved, looking at the shape from different perspectives. Even worse with regard to this would be example 24b, as boundary features are vanished in favor of the minimal twist property [Far02]. Exploiting the capabilities of the parametrization even more shape variants can be produced and might even influence the original interpretation of the initial curve network.

On the one side this might be used for further surface styling procedures as it is very flexible towards its final appearance. On the other side this would violate the initial assumption, that features should be represented within our network and not produced by our parametrization. This fact is emphasized within the last two examples 24g and 24h, where the boundary has been slightly modified using the line styling tool. Especially for the latter configuration 24h, which has a similar parametrization as our previously positive example 24c, slight changes now have an tremendous impact over the whole surface introducing a variety of new features. Therefore the geometric possibilities have to be balanced with maintaining our original modeling metaphor.

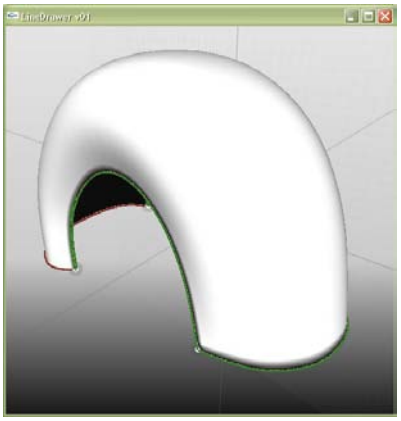
This discussion leads to two major consequences: First minimal twist surfaces have been selected to be the standard surface parametrization in order to get closest to our previously defined shape goals. This also underlines that all features have to be *explicitly* modeled although this raises the overall modeling effort the more detailed the intended surface shall be. Second and directly related to the first statement, the network is should be rather well defined within later stages of the modeling process in order to minimize the impact of surface ambiguity which correlates to the number of possible surface configurations.



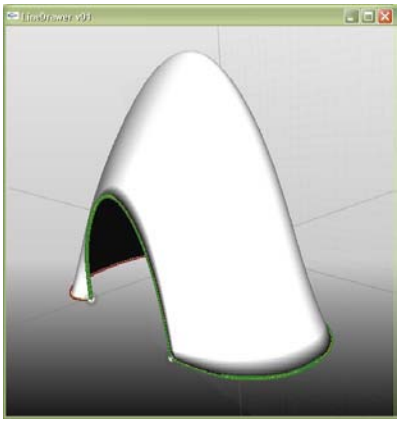
(a) Initial Patch ($\alpha, \beta = 0$)



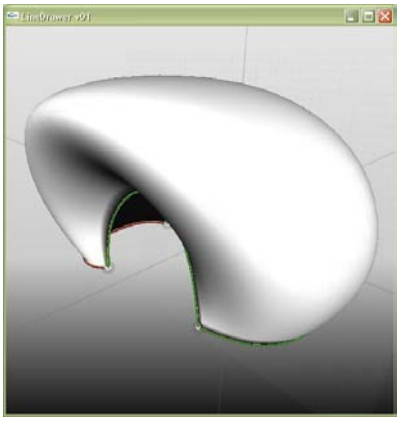
(b) Minimal Spanning Configuration



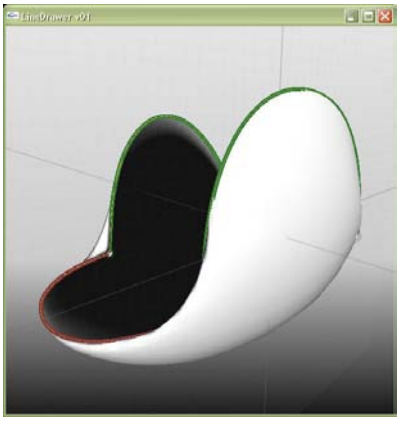
(c) $\alpha = -0.33, \beta = 0.55$



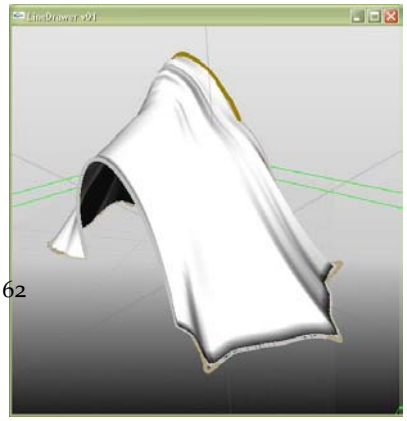
(d) $\alpha = -0.99, \beta = 0.72$



(e) $\alpha = 1.33, \beta = 0.90$



(f) $\alpha = 1.66, \beta = -2.19$



(g) Modified $\alpha = 0.76, \beta = 0.06$

(h) Modified $\alpha = -0.13, \beta = 0.45$

Figure 24: Parameterizations for simple networks inspired by Farin et al. [FH99].

In the previous section the general processing of the data and associated representations have been discussed in order to give a feeling about the general complexity of the shape reconstruction task, even for simple input conditions. In order to be able to handle more elaborate shape constructions the topology analysis of the corresponding network structure is inevitable and can support later construction stages.

The major element for topological structure analysis within the network are *intersections of curves*, which is represented within the `LPIntersection` class defined in section 4.2. By the definitions given in section 3.3 the intersection $\mathbf{i}_j$ references to a number of r adjacent lines which all start or end with a local tangent direction within the intersection. Further the intersections split the complete network into a set of line segments **LS** where every line segment is limited by two intersections. Using a scale-sensitive threshold value and the distance to the reference curve network of a new sketch curve $\mathbf{L}_i(t)$ by the means of the the distances to $\mathbf{L}_i(0)$ and $\mathbf{L}_i(1)$ are computed an intersections are inserted into the existing curves which are below this threshold distance. Note that this is in first place only done for the line start- and end point but can be expanded to general intersections of crossing curves, keeping in mind that this will potentially produce high computational effort (e.g. a jittered curve drawn over a straight one).

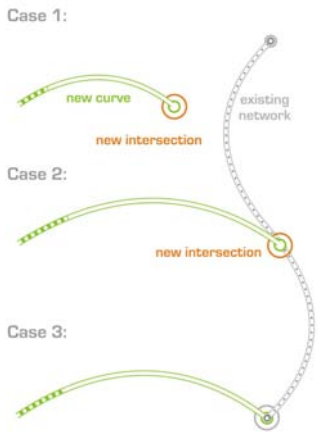


Figure 25: General Merging cases for one end of a line inserted into an existing network.

For inserting a line into a network in general three different cases can be defined per line ending, which are shown in figure 25. As it crucial for further processing and allows for insight in the overall programming structure, an overview about the `finishActiveLine()` method is also presented in pseudo-code in 4.1 at the end of this section.

Those cases are defined as follows:

1 The active ending does not hit any line.
Per default an new intersection at the ending is created pointing only at the created reference line. In general there are now affected surface structures, accept the ones where the line has been drawn on or which might be spatially related.

2 The active ending hits a reference curve at $0 < t_{ref} < 1$.
This results in a splitting of the existing reference curve into two segments from and the corresponding surface structure. The resulting new intersection points to two different lines and three line segments.

3 The active ending hits another intersection.
In this case no intersection is created, only line- and segment references as well as the corresponding surface structures have to be updated.

One key assumption of the network surface extraction states, that we want to create a manifold and therefore patch surface structure at the end of the shape creation process. As a result on every $\mathbf{p}_i^0$ point of our surface we can define two principle surface directions denoted as $\mathbf{u}_i$ and $\mathbf{v}_i$, while the cross product denotes the normal of the surface within this point. Note that the cross product is not commutative which leaves two principle choices for the normal, as we cannot directly define an "inside" or "outside" due to the ambiguous nature of the sketch input.

Within an intersection point $\mathbf{p}_i$ the situation can be stated as follows:
The number of normals occurring in an intersection is at maximum equal to the number of incident segments r, whereas every normal can be defined as the cross-product of the tangent direction vectors belonging to the segments. If one of the incident segments is smooth the number of normals is further reduced by one.
One problem which arises, is that we cannot mathematically define the normal in the point which directly lies within the intersection to lie on a surface, as we have multiple normal candidates. On the other hand this problem is practically circumvented by defining multiple surfaces coinciding in this point. In most cases it is also possible to define a average normal direction $\mathbf{n}_a$, despite some configurations will lead to a zero average direction.

In order to still be able to handle those complex intersection points we assume the following: All incident normal directions can be sorted by angle if projected to a plane through $\mathbf{p}_i$ and defined by $\mathbf{n}_a$. This sorting allows to determine a local neighborhood of tangents and their associated segments. This results in a data structure, which is maintained during the modeling process and will be denoted as *Tangent-Loop*.

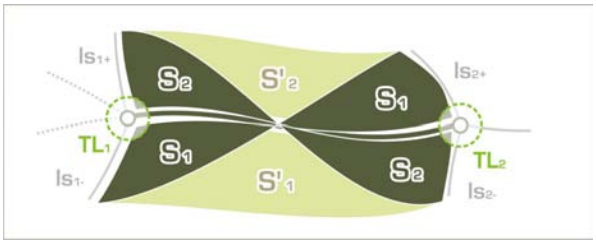
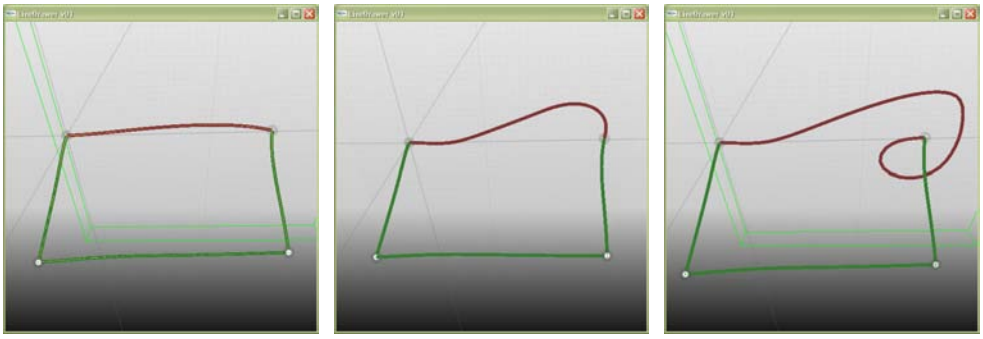


Figure 26: Illustration of the general "Twist" problem.

Given all Tangent-Loops in every intersection and the line topology by resolving the interconnecting segments, the problem of finding a surface topology on the network breaks down to a binary problem per segment. One alternative surface extraction method to the minimal cycle extraction thereby would be to assume the following procedure: Given a curve network all Tangent-Loops are computed and every segment $\mathbf{ls}_i$ is associated with two surfaces $\mathbf{s}_i^+$ and $\mathbf{s}_i^-$ which only contain the segment itself once. As we have two tangent loops associated to this segment $\mathbf{tl}_{min}$ and $\mathbf{tl}_{max}$ we can get the neighboring information per tangent loop and extend all surface-sequences by the next segment surface in in the loop. Doing so we get a set of surfaces containing two line segments each.



(a) Initial Configuration (b) 90° Normal Rotation (c) 360° Normal Rotation

Figure 27: Illustration of the general problem of deciding if an edge is contains a twist or not.

Now problem can be formulated as the following question: Given a segment $\mathbf{ls}_i$, its two adjacent tangent-loops $\mathbf{tl}_{max}$ and $\mathbf{tl}_{min}$ and for each intersection two neighboring segments surfaces $\mathbf{s}^+$ and $\mathbf{s}^-$, we have to decide which enclosed surface has to be connected to which opposite surface. Visually this can be interpreted as a twist or straight surface connection on ls_i which is shown in figure 26. Although this decision is binary per intersection plus incident segment and can be easily resolved in 2D, it cannot be solved locally in general manner within $\mathbb{R}^3$.

Despite this problem when deciding surface connections while modeling the network, the *Batch-Processing* and minimum cycle extraction method described in section 3.5 profits from the definition of the tangent loop, as it provides additional constrains to exclude counterintuitive surfaces from construction. For a *Progressive Extraction* procedures this is more problematic due to its global nature, as the addition of a single line might change the normal direction of extensive parts of the network or invalidate previous surface constructions.

Due to this unresolved ambiguity problem the proposed *Active-Boundary* method has been implemented to achieve stable and fast extraction during the modeling process and allows for local surface decisions. Assuming that we want to achieve a closed solid surface object, we can interpret a line as an addition to an existing boundary. Similar to a surface cycle, it is defined as a sequence of line segments with some additional properties: A boundary has always to be closed, meaning that the first intersection of the first stroke and the last intersection of the last stroke are identical. A boundary can have an arbitrary number of segments and can include repetitions of segments and intersections. These repetitions occur, if the boundary either includes segments which are not matched up to the current point, are only connected to an unmatched segment or if they are combining circular boundary structures.

Adding a new segment into a Active-Boundary sequence can result of four different cases in analogy to the previously mentioned general cases:

1 The new segment does not hit any boundary at all. If so, the segment creates a new boundary which contains only itself.

2 Only the start or the end of the new segment coincide with an active boundary sequence. In this case the existing boundary is expanded by the segment by inserting it into the sequence at the adjacent intersection point.

3 The new segment combines two points which are part of the same active boundary. This case creates an closed loop on the boundary, which can be separated from the edited sequence.

4 The new segment connects two points on two different boundary loops. If this occurs, both sequences respectively surfaces are combined by the new segment and included into one single surface.

As this procedure is purely topologically driven, there are some cases in which the result can be rather unexpected. Those exception include unmatched segments within closed surface loops and the general addition of segments into closed surfaces. Furthermore geometric holes within an objects surface, going through an object, have to be defined explicitly similar to Teddy [IMT06] and ShapeShop [SWSJ06]. This might lead to the necessity of interpreting the geometric context of the edited boundary sequences, especially to resolve to which boundary a stroke is added and extend the existing boundary or surface incorporating the new stroke.

In the current implementation this problem is circumvented by defining only one Active-Boundary sequence which strokes can be added to. If an existing surfaces should be edited it is simply deleted. This operation initiates the surface construction for the current Active-Boundary sequence and sets the closed boundary sequence of the deleted surface as editable. By this procedure we can guarantee one ore more closed surface network objects at all points of the modeling procedure. This concept finally leads to a stable and predictive modeling metaphor, which is limited by a set of exceptions, but in general provides a wide expressiveness as the results in chapter 5.2 will show.

Listing 4.1: Pseudocode of the Function finishActiveLine()

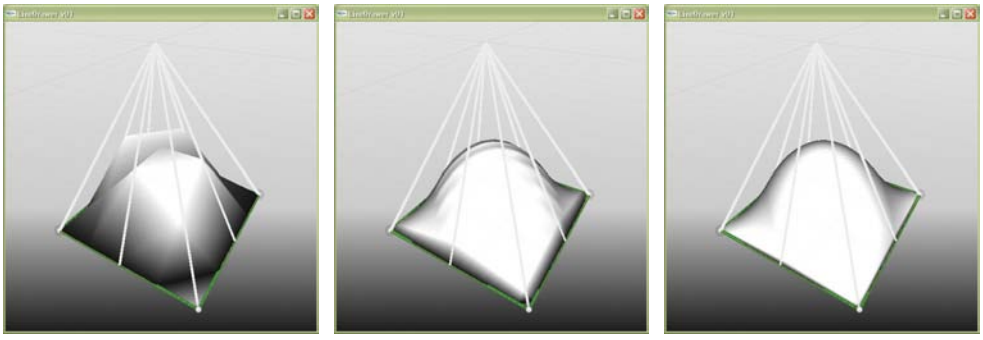
```
void LPObject::finishActiveLine(void)
{
   if( NO_POINTS ) return;
   Define initial Parameters;

   if ( lines.size() >= 1 && IS_ACTIVE)
   {
      float threshold = getScaledDistanceThreshold();
      for all intersections
      {
         pi = Position of Intersection;
         float end_distance = distance(pi-line[1]);
         float start_distance = distance(pi-line[0]);

         if ( start_distance < threshold)
         {
            Add line to intersection references;
            Split corresponding surfaces;
            Move start position into intersection;
            //...
         }
         if ( end_distance < threshold)
         {
            Add line to intersection references;
            Split corresponding surfaces;
            Move end position into intersection;
            //...
         }
      }

      for all lines and no intersections found
      {
         cur_line = Current line;
         float line_end_dist = cur_line distance to line[1];
         float line_start_dist = cur_line distance to line[0];

         if ( line_end_dist < threshold )
         {
            Create new start intersection and set links;
            Move start point into intersection;
            Update line segments and references;
            Merge existing boundaries;
            // ...
         }
         if ( line_start_dist < threshold )
         {
            Create new end intersection and set links;
            Move end point into intersection;
            Update line segments and references;
            Merge existing boundaries;
            // ...
         }
      }
   }
   Recreate line segments depending on found intersections;
   Update and check all affected boundaries;
   Update visuals;
   Select current line as active line;
   Apply operation to pattern objects;
   // ...
}
```

The topological extraction of the input network finally delivers a set of closed boundary surfaces which are expressed as a cyclic sequence of line segments and enclosed intersections. The *Batch-Processing* produces general Patches up to four boundary segments, which can be constructed with the Coons-Patch definition described in chapter 4.3. The progressive *Active-Boundary* method delivers sequences of length $n \in (N)$ which are sought to be converted into a general parameterized surface description. A general concept of how this can be achieved using general subdivision methods has been outlined in chapter 3.3 and will be enriched by further detail within this section.

In analogy to the Coons-Patch construction a general surface can be constructed in two steps: First a appropriate topology has to be defined which reflects the basic shape properties of the surface. The more topological samples are used, the higher the resolution and computational managing effort will be. Second basing on this initial topology the position of samples has to be defined using an appropriate parametrization and the constrains given by the reference boundary curves.

According to the first task a general quad mesh subdivision scheme is used to create the initial topological structure. The general procedure has already been outlined in section 3.3 and is illustrated in figure 11.

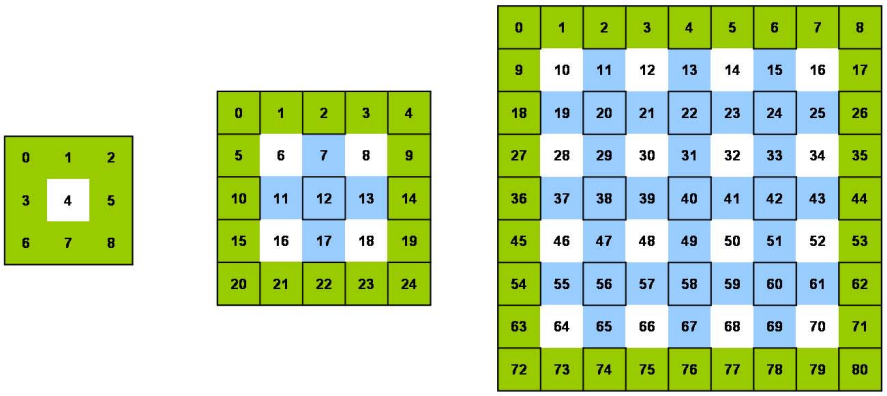
The Algorithmic steps to create a surface out of a n-sided polygon is similar to the method proposed in [DDGG05] including changes and modifications to optimize it towards the use of the spring-mass system. The input into the surface construction process can be an arbitrary number of freeform curves, which don't have to be connected. In the first processing step a number of subdivisions per line and resulting depth-levels has to be determined. The indexing scheme for the quad subdivision and for polygons is illustrated in image 29, while the result of the subdivision procedure is shown in figure 28. Those indexing patterns determine the topology within further subdivision steps, while the geometry can be reconstructed using the initial control net and the neighboring positions of the previous level plus additional deformation patterns.



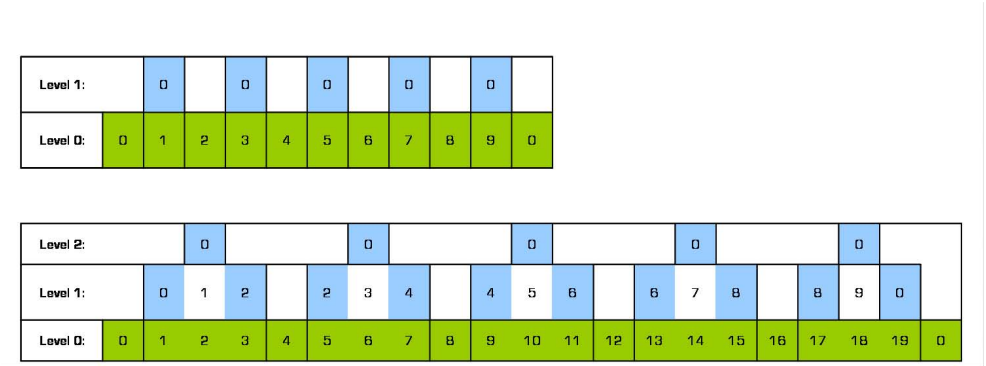
(a) Quad-Patch 5x5 (b) Quad-Patch 17x17 (c) Quad-Patch 65x65

Figure 28: Subdividing a linear Quad-Patch from the initial control structure which is indicated by the brighter lines.

First on all initial lines a umber of starting points is set including the corner points where two lines are adjacent. All initial points are set to be static, which means that they cannot be moved due to further optimization or force iteration steps. Additionally, every point holds four reference links to the neighboring points. In each following step the number of points in the level is reduced by the number of corner points, as the corners are not linked to the next



(a) Quad Subdivision Pattern



(b) Pentagon Subdivision Pattern

Figure 29: Illustration of the indexing scheme for different subdivision patterns. Note that the Pentagon pattern can be extended to arbitrary Polygons.

level. To avoid the necessity of unused links the number of initial points on each line should be chosen even. Doing so the last level contains only as many points as there are corners in the first level. For mostly convex boundary configurations, the last level is assumed to be planar and is connected by one additional point with valence equal to the number of adjacent points in the last level.

For concave or high curvature derivations this assumption will produce self-intersecting surfaces in the final triangulation step. Therefore the last interconnecting point should be extended by an medial- or chordal axis approximation in order to approximate self-intersections avoiding forces similar to the procedure described in [IMT06] and [LG07]. All points are saved within an additional level-order data structure oriented at the indexing scheme illustrated in figure 29b. This supports efficient force iteration as well as optimization steps and the creation of triangular mesh representations for final rendering. In the current system interconnecting edges between elements are linear but can be expanded to an freeform description in order to reduce approximation errors and resulting visual clutter.

The main advantages of this procedure are the ability to process arbitrary input polygons with a running time that depends on the number of boundary curves. As image 34 within the next chapter shows, also open mostly convex line configurations in arbitrary order can be handled, as long as they are known to be part of one surface. As the line number is expected to

be between three and ten this procedure is fully interactive and allows visual feedback during the ongoing optimization. Higher number of boundary curves are Further the description allows easy further subdivision to support higher surface qualities. For speeding up the optimization process the optimization can be performed on the initially constructed quad mesh and if finished, the quad mesh can be subdivided enhancing the visual smoothness of the resulting surface.

Chapter 5

RESULTS

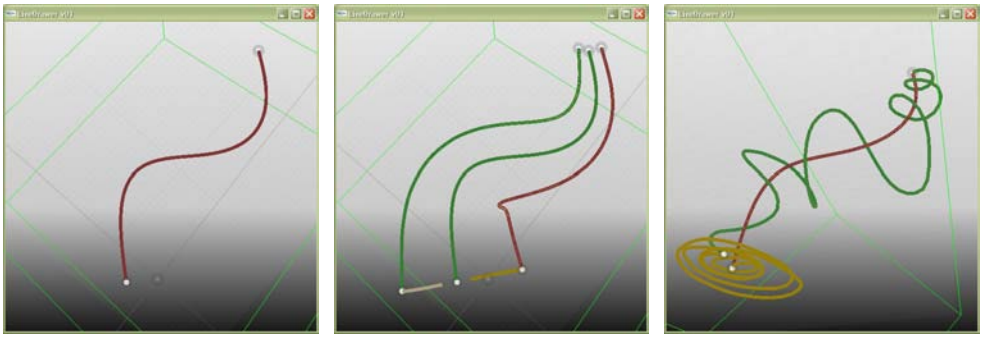
5

RESULTS AND COMPARISON

The implementation of the modeling prototype presented in the previous chapter now allows for testing the provided functionality against our initially stated requirements. The purpose of this chapter will be to trace the general construction process of 3D shapes within the prototype and revise the applicability of the proposed modeling metaphors within different shape creation scenarios. The results and final features of the prototype will be used as basis for a concluding comparison between our approach and selected systems presented in section 2.4.

5.1 INTERACTION PROCESSING

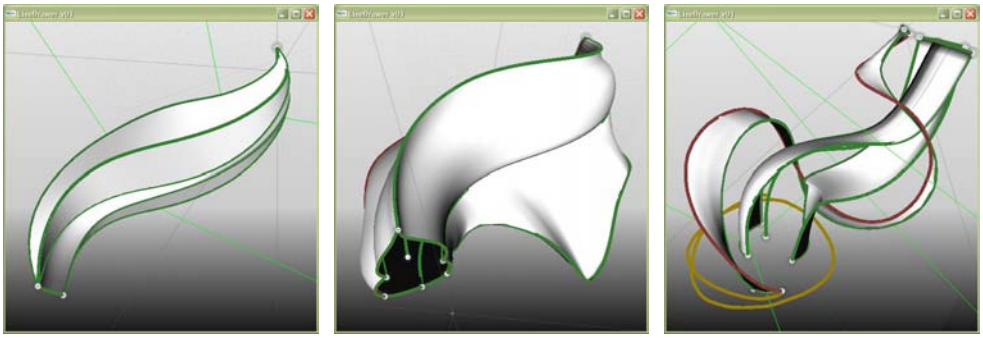
As stated in section 4.3 parametric properties of the input curves can be exploited for a tremendous range of shape operations. This fact can be shown extending the sweep concept from section 32 towards *parameterized sweeps*. The general procedure and shapes which can be constructed with this concept, are shown in figure 30 and 31. Although the concept claims to be very flexible and allows for rapid construction of complex configurations like screw lines and helix structures shown in figure 30c, the direct and intuitive control of this construction method is limited, as the spatial correspondence between reference, construction and resulting sweep curve can be quite complex, especially if intended features are to be achieved. Therefore additional correction, using styling strokes can aid this curve construction process.



(a) Initial Reference Line (b) Linear Sweep Lines (c) Complex Circular Sweep

Figure 30: Parameterized sweeps use a reference line $\mathbf{c}_{ref}(t)$ as in figure 30a and construction line $\mathbf{c}_{con}(t)$ as the two yellow lines shown in figure 30b, to construct the sweep line $\mathbf{c}_{sweep}(t)$. Thereby the construction lines at t are projected into the tangent plane within the same position t on the reference line.

As extrusion has shown to be a powerful operator to speed up complex construction tasks, as shown by Oh and Stuerzlinger ([OSD06], [SOD05]), it can be intuitively defined



(a) Simple Creased Surface (b) Smooth Sweep Surface (c) Complex Helix Surface

Figure 31: Surface constructions based on parameterized sweep lines. Note that all examples use the same reference line from figure 30a.

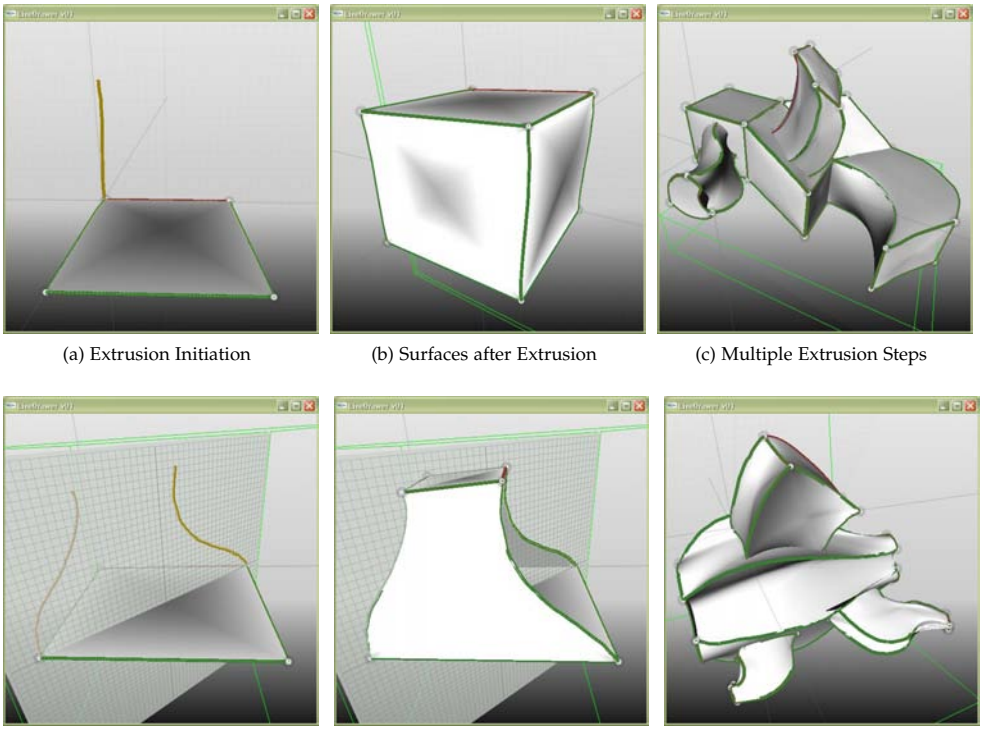
for speeding up network construction tasks as explained in section 3.4 within the prototype concept. The results of two different extrusion operators *linear extrusion* and *bilinear extrusion* are shown in figure 32.

Both operators allow for complex extrusion of arbitrary freeform boundary curves, whereas the plausibility of results is mainly limited by the surface construction process. Less intuitive results are produced for surfaces, that contain unmatched double edges, as they conceptually produce two identical overlapping surfaces but with different normal directions. As the extrusion paths are guaranteed to represent feature lines in the extruded shape, this metaphor complements the overall modeling concept.

In chapter 3.4 the concept of advanced copy operators is outlined. The application of this operator is shown in figure 33. Although this definition is quite powerful, it has two distinct disadvantages: First it is very fragile against noisy input, which leads to over-scaled or self-intersecting results. Second it controls multiple parameters simultaneously and defines an abstract relation between operator and geometry, that might not be very intuitive in its practical use. Due to this significant disadvantages, the operator will not be within further scope of the general discussion. Nevertheless by providing additional visual feedback about intermediate results and improving the stability against unintended output, there are a numerous potential applications for this operator during the sketch-modeling process.

All above mentioned operators integrate into the implemented system by exploiting the parametric properties of the underlying network descriptions. A main advantage is, that those operators have to be only defined on the boundary curves given in the curve network. As long as the resulting operations are continuous, which implies that the original topology and ordering is maintained, the surface extraction methods can be applied directly on the resulting curves. Therefore these operations are significantly faster than applying them directly to the complete surface. Additionally, they do not have to take the preservation of surface features into direct account, as they will be reconstructed from the boundaries, as long as they have not been fundamentally changed.

The support of the visual surface construction and demonstration of the flexibility of the quad mesh description is further underlined by the example given in 34. Furthermore, the methods described in section 4.3 for surface and line styling can be used to change the existing geometry features of curves and surfaces as shown in figure 35. This concept can be extended to surfaces as shown in figure 37. In practice the latter definition has not shown to be very intuitive, as the final shape represents a mixture of various force influences. The ability to follow the styling curve while maintaining a reasonable surface structure, mainly depends on

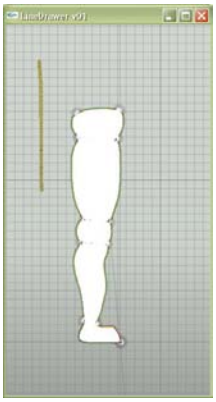


(a) Extrusion Initiation (b) Surfaces after Extrusion (c) Multiple Extrusion Steps

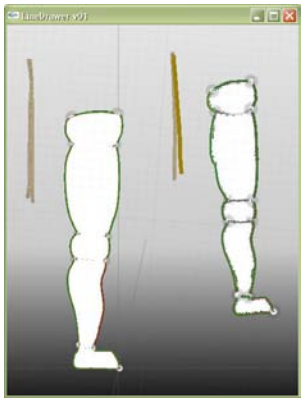
(d) Bilinear Extrusion Initiation (e) Bilinear Extrusion Result (f) Bilinear Extrusion Construction

Figure 32: Linear extrusion translates the initial boundary along the given freeform path (yellow curve) and connects the corresponding intersections, while bilinear extrusion uses the second reference curve to scale and rotate the reference boundary. The example shown in 32c consists of three closed n-sided boundary sequences plus six extrusion paths, while 32f consists of four boundaries and eight paths.

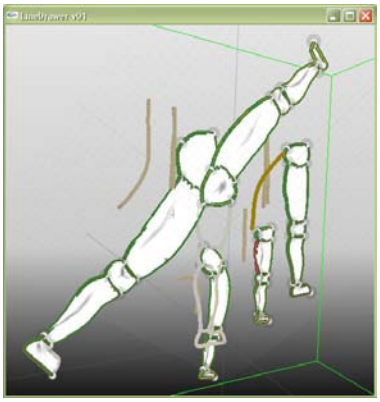
initial boundary configuration and sampling resolution. Further it does not directly comply with the overall modeling concept, as the styling strokes are not always interpolated.



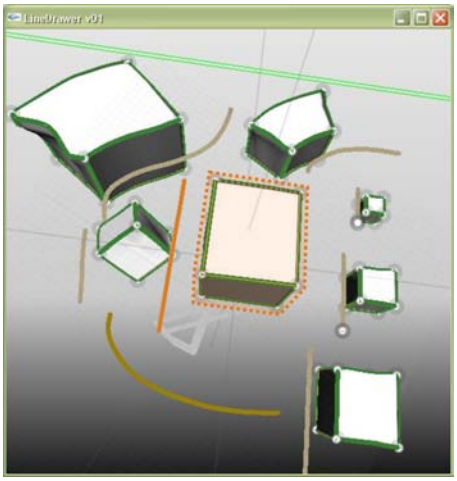
(a) Initial Setting



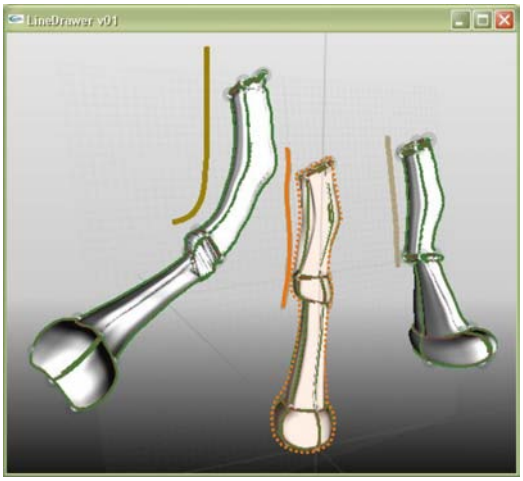
(b) First Copy Instance



(c) Multiple deformed Copies

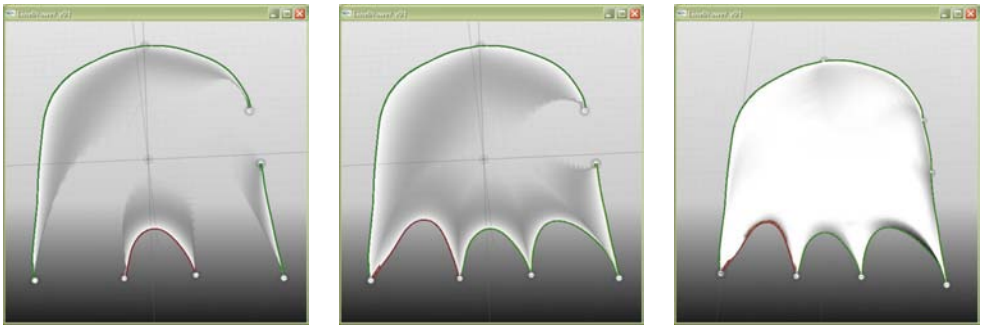


(d) Application to the Cube Example



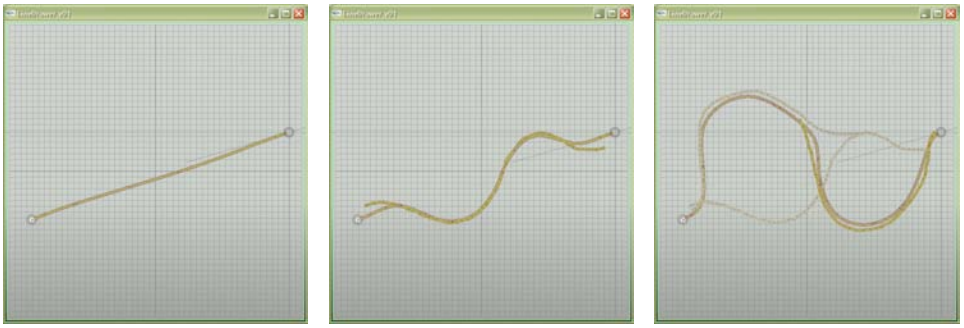
(e) Copying a complex 3D Network

Figure 33: The advanced copy operator takes a reference network and a copy stroke (yellow) shown in 33a. The reference geometry is transformed into the local spherical coordinate system and can be replicated for arbitrary curves illustrated in 33b and 33c. As the a reference normal is acquired while drawing, this can be extended to 3D as shown in 33d and 33e (Original network and copy-stroke are highlighted).



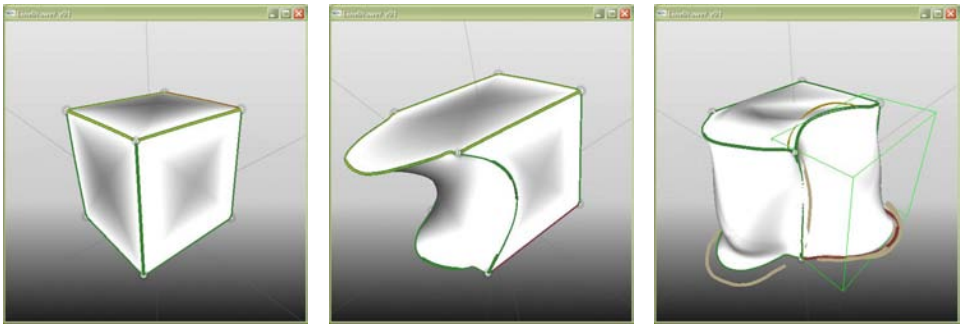
(a) Initial Curves (b) Adding further Curves (c) Closed Surface

Figure 34: The effect of subsequently closing a surface using the polygon subdivision scheme from chapter 4.5. Note that between unmatched intersections straight lines to their closest unmatched neighbor are assumed, while elements on this lines are rendered transparently and not fixed to any boundary.



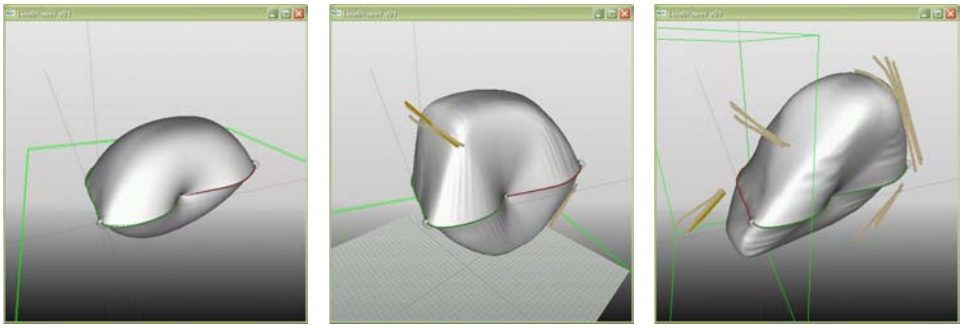
(a) Initial reference curve (b) First modification (c) Subsequent styling

Figure 35: The styling of a single curve using an additional styling curve. Note that only the last styling stroke is used while previous ones are deactivated but still displayed for spatial reference.



(a) Initial cube network (b) Modification one (c) Alternative modification

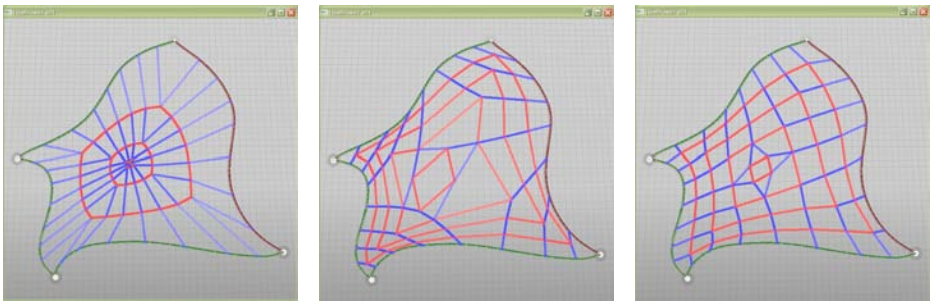
Figure 36: The process of styling for an existing network for a cube network. The boundaries can be modified without changing the structural topology as figure and show two different results of the styling process.



(a) Initial Force Iteration (b) Adding Surface Styling Curves (c) Redefining the Styling Curves

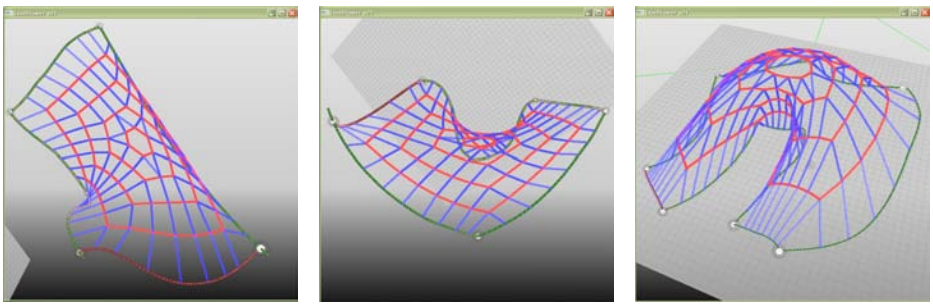
Figure 37: The curve styling concept extended directly to a surface, whereas the styling curves are shown in brighter color.

Before discussing the general results on modeled networks, the quad mesh concept for the underlying surfaces is explained with reference to four practical examples. The definition of this structure is explained in detail in section 3.3, whereas the images shown in figure 38 show the impact of different parameterizations for external- and internal level forces. As seen in figure 39 the parametrization within the spring force model can be used to improve the resulting surface quality and to model various shape effects. Additional forces can be easily defined, as shown in 39c, which depicts the additional application of a force into the global direction $\mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. To allow for more control over the surface shaping, the forces should be defined in context to existing geometry and properties of the adjacent boundary segments, which will be explained in the next example.



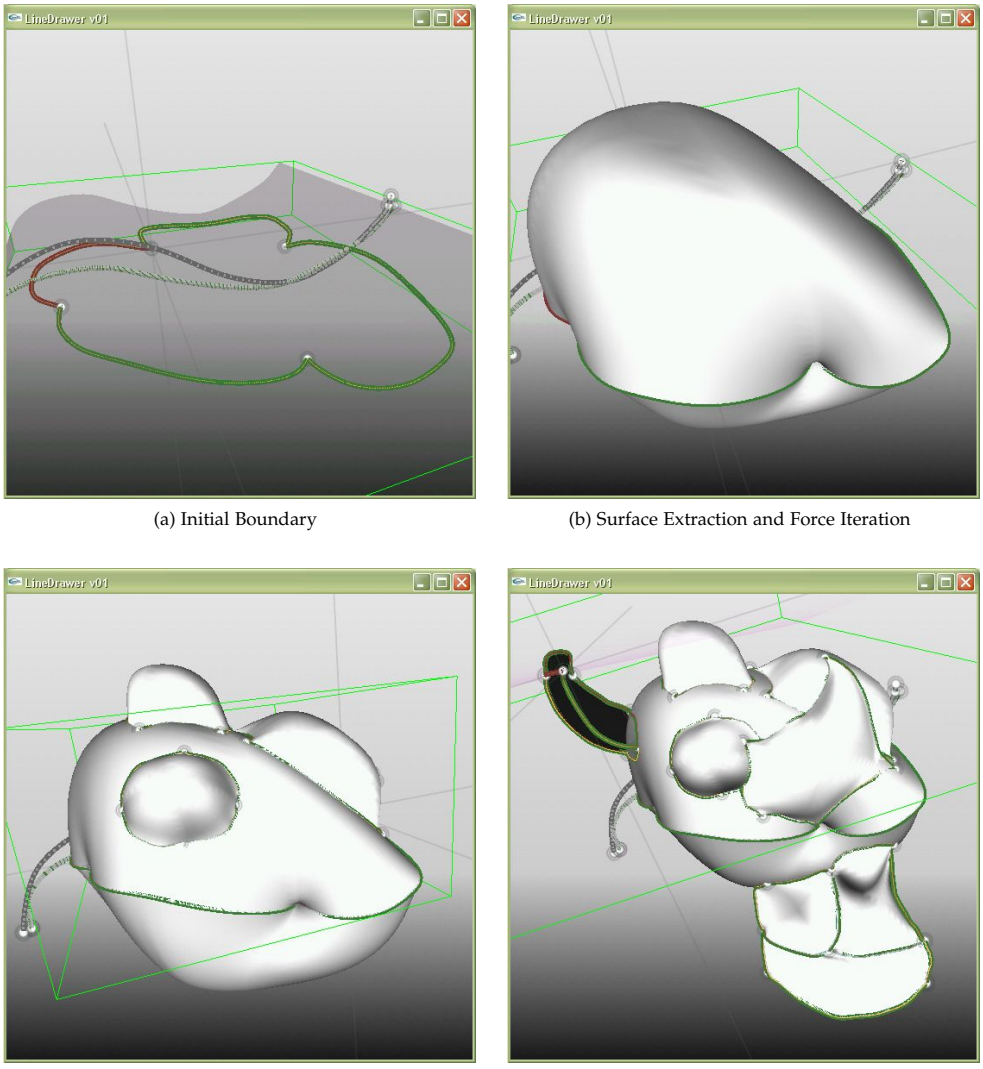
(a) E=0.09 I=0.99 (b) E=0.79 I=0.065 (c) E=0.86 I=0.58

Figure 38: Different results for the parametrization quad-mesh structure. Thereby E denotes the forces between two levels (blue) and I the forces inbetween one level (red).



(a) Iterated line configuration (b) Iterated network (c) Applied directional force

Figure 39: Different results for Parametrization of the triangular Surface



(a) Initial Boundary

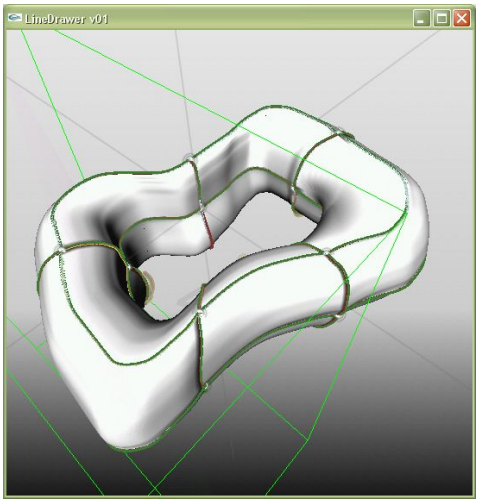
(b) Surface Extraction and Force Iteration

(c) Adding Further Surfaces

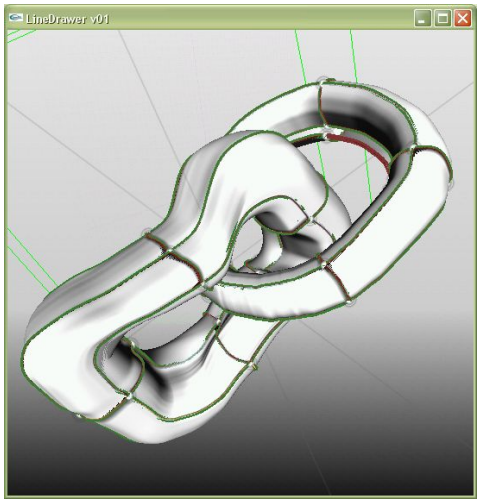
(d) On-Surface drawing of additional Features

Figure 40: Illustration of the general affect of the continuity force iteration on a closed surface. Figure 40b shows the inflation-like effect of the force iteration on a non-planar boundary. Figure 40c and 40d show the general embedding of further patches into the existing scene geometry.

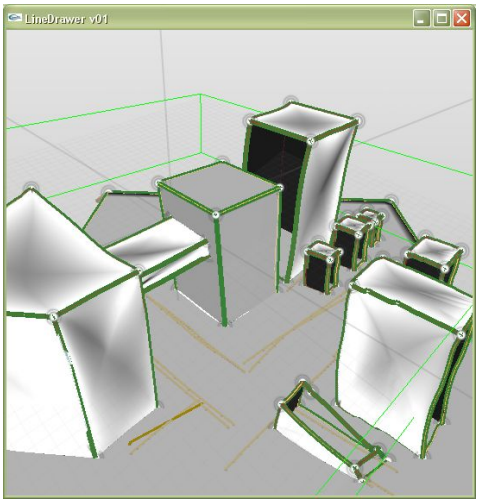
If the defined forces are used to approximate general continuity constrains between bordering surfaces, an inflation like operation can be derived. By defining a single closed boundary sequence and iterating both constructed surfaces multiple times the results shown in figure 40 are achieved. Note that the defined boundary does not have to be planar and was drawn onto an extrusion surface in this example. To achieve the shape configuration shown in 40b the force iteration has been applied 30 times. Drawing new boundary sequences on an existing surface can be used to achieve shape effects as illustrated in figure 40c. As the acquired normal from the drawing process influences the inflation direction, patches drawn on another surface are 'pushed away' from the underlying patch, as the quad mesh elements are moved into an orthogonal normal direction.



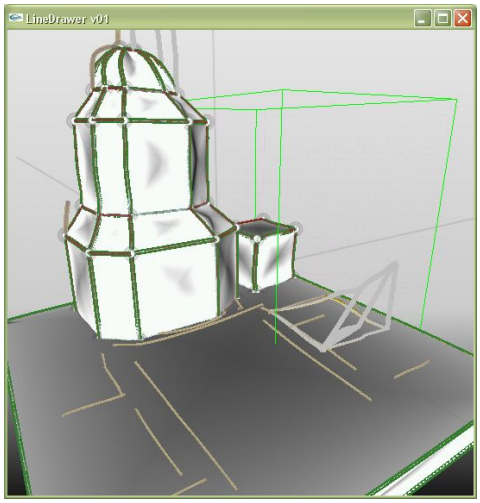
(a) Circular Structure



(b) Extended Circular Structure



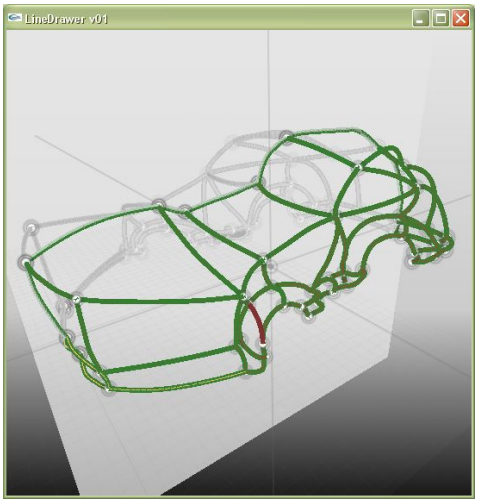
(c) Extrusion Modeling



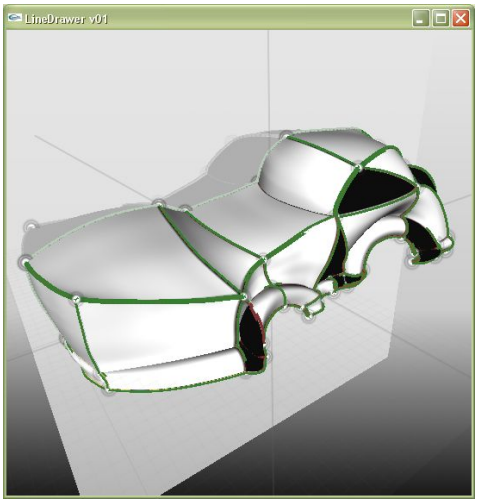
(d) Sketching a simple Building

Figure 41: Example 41a and 41b show the construction of objects with holes, which requires deleting invalid surfaces covering the hole structure. Figure 41c and 41d show the general application of the extrusion operators for simple scene construction.

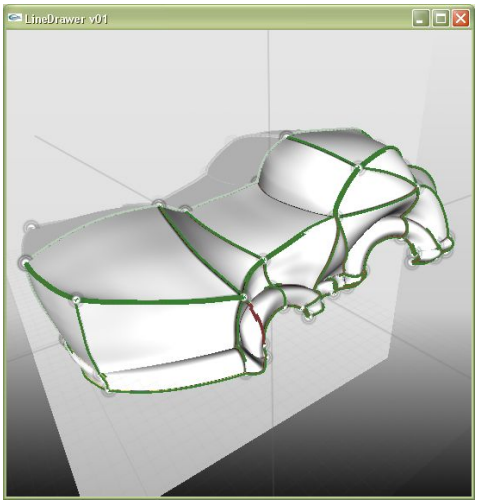
It already has been outlined, that the construction of objects with holes in general produces at least one surface, which is not required. Despite the overall construction of such objects is possible using the batch extraction procedure in combination with the bilinear extrusion operator as shown in figure 41b. In order to get these results the additional surfaces have to be deleted. Example 41c and 41d shown the repeated application of both extrusion operators, in order to constructed rather simple scenes exploiting mostly rectangular objects.



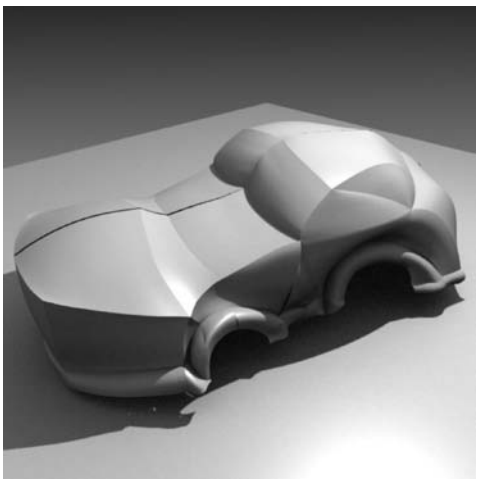
(a) Initial Network



(b) Batch Extraction Result



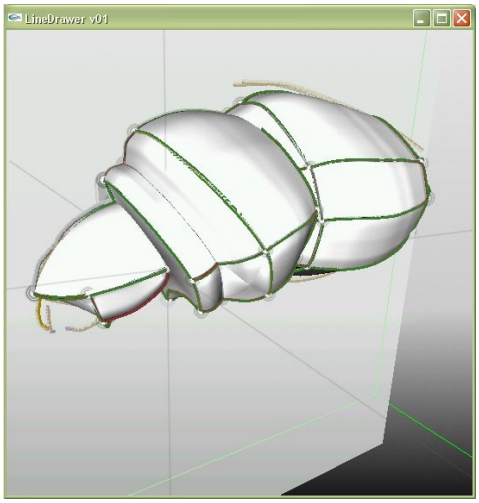
(c) Normal corrected Result



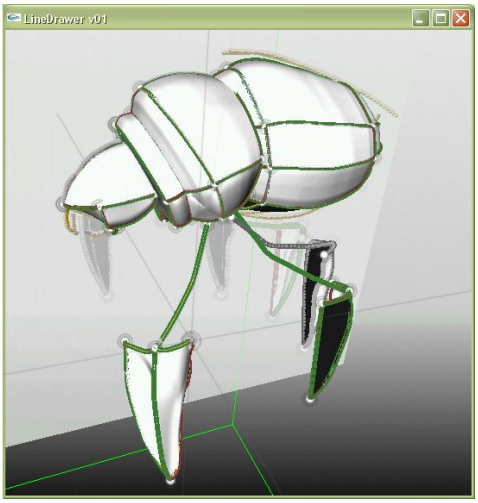
(d) Rendering in Blender

Figure 42: The Extraction result for the Batch processing, which includes the modeled network in figure 42a, and extracted surfaces in 42b. As not correctly determined for all patches some normals are inverted in 42c, while 42d shows the exported and rendered result using the Open-Source package Blender.

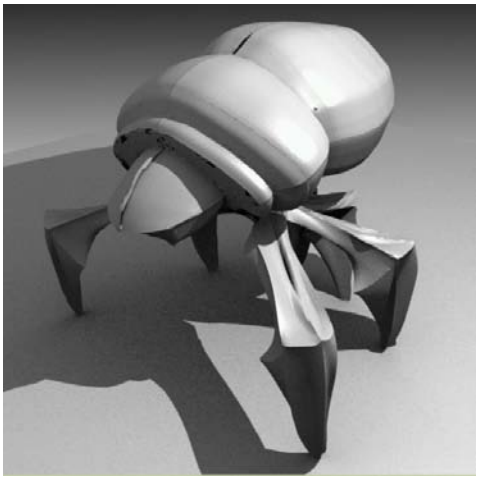
One advantage of the batch procedure, is the predictable and stable surface extraction as shown for example . In order to produce this shape, the network has been constructed to fulfill the constraint, that all closed boundary sequences are quadrangular freeform patches (except of the one boundary of all segments which are only adjacent to one constructed quad surface). This raises the constructional effort significantly and is not feasible during conceptual modeling processes.



(a) First Symmetric Network



(b) Construction of Rotational Patterns

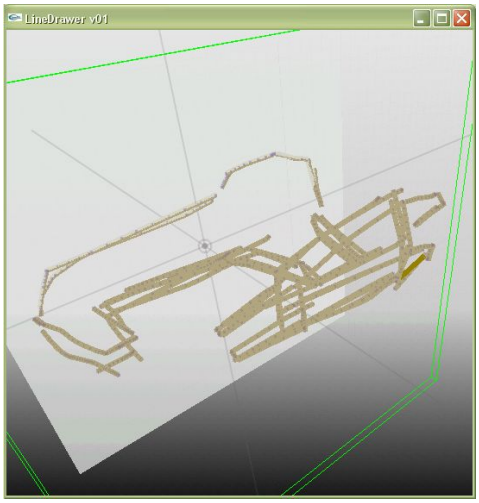


(c) Blender Rendering Front

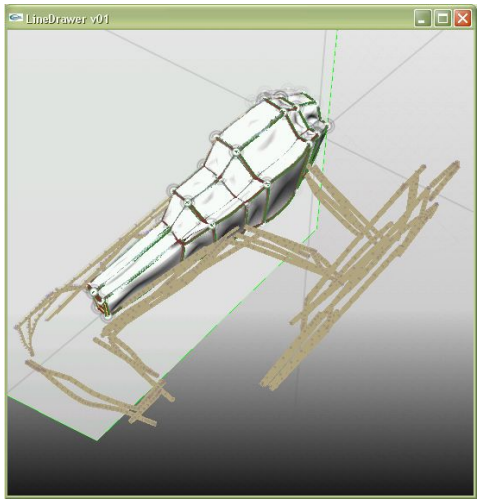
(d) Blender Rendering Back

Figure 43: Exploiting the possibilities of general transformation patterns. Figure 43a shows simple symmetry turned on, while image 43b shows the modeling of one leg, which is further used by two pattern objects (plus two due to symmetry), rotated 30° and 60° around the global z-axis.

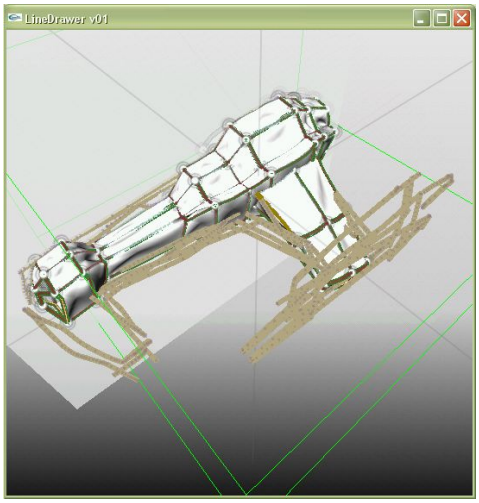
In general all curve network objects within the scene are associated with an additional transformation matrix, which allows easy reuse of existing geometry. The effect of those patterns during the modeling process is illustrated in figure 43. The overall modeling session took 19 minutes, while 32 curves had to be directly defined and additionally six extrusion operations for the body. Note that these patterns have been defined externally and explicitly activated during the modeling session.



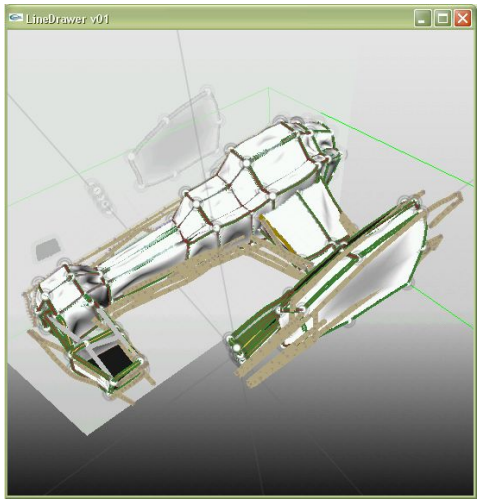
(a) Pre-Sketch



(b) First Extruded Network



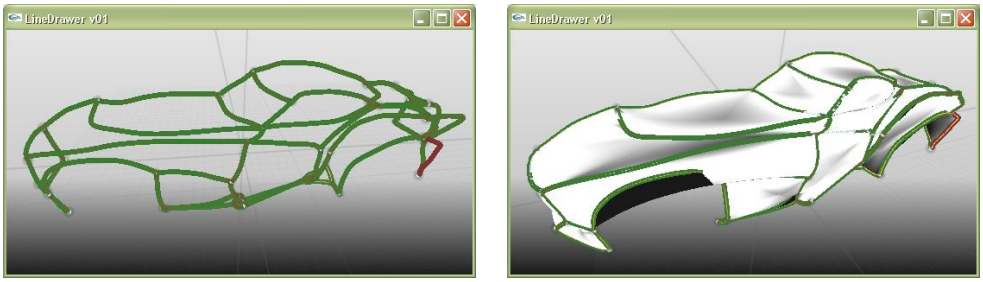
(c) Second Extruded Network



(d) Third Extruded Network

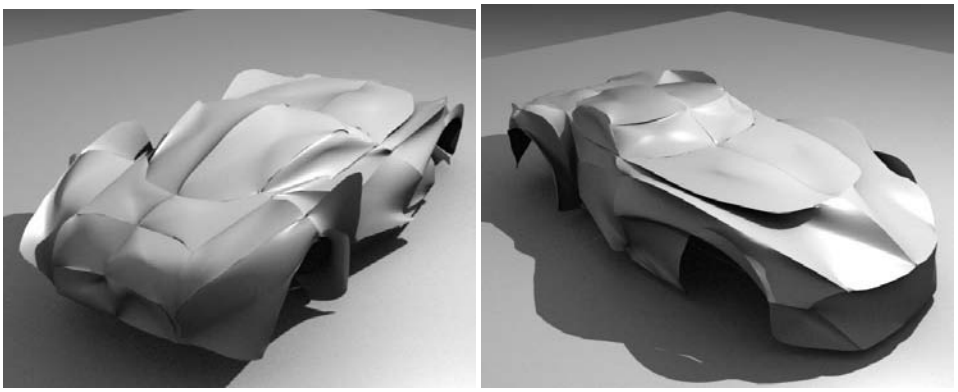
Figure 44: The modeling progress in principle: Starting with a rough pre-sketch in figure 44a, First surfaces are created using the extrusion operator in 44b. Further modeling approximate the initial sketches shown in 44c and 44d.

In order to further demonstrate the overall modeling process in principle, an example of an iterative modeling session is given in figure 44. The final model consists of 220 curve network lines and took 32 minutes for its construction. The images illustrate, that the extrusion operation allowed an appropriate approximation of the initial sketches. Within the last stages shown in figure 44d, the surface construction failed, as some of the curves were below the distant threshold value for the existing intersections and violated the active boundary constraints.



(a) Reconstructed Curve Network

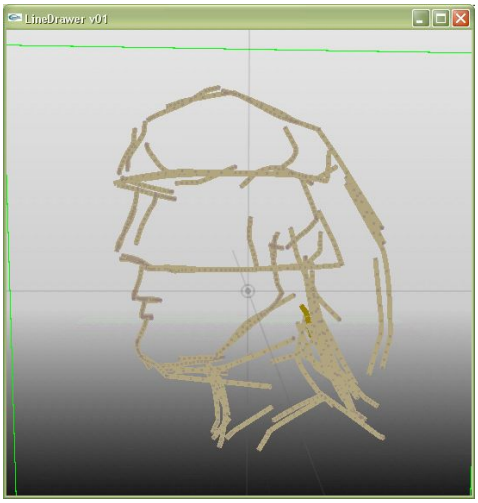
(b) Progressive Extraction Result



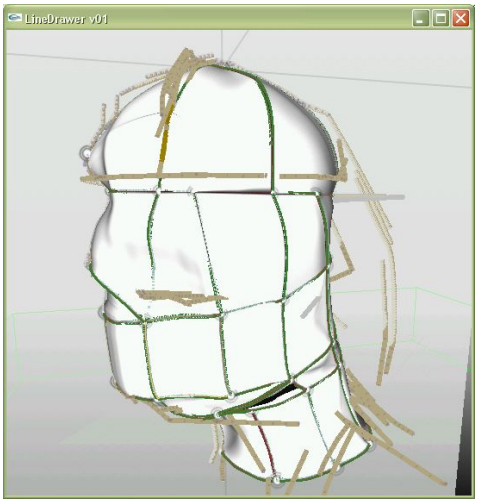
(c) Rendering with Blender

Figure 45: Results of the progressive surface extraction method for the manually redrawn curve network from the "Roadster" data set provided with the ILoveSketch project [BBS08]

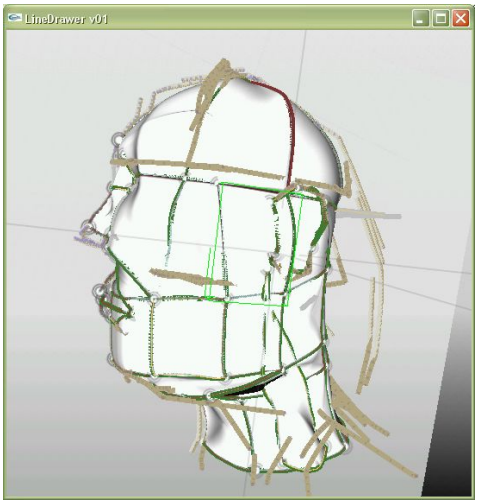
The system ILoveSketch published by Bae et al. [BBS08] takes a similar direction as developed system concerning the representation of the input data. As already mentioned there have been published three data sets, that have been used to test the extraction processing. The given example networks have been drawn by a professional designer and give valuable hints on the structure of this kind of data. Unfortunately the given format is not directly compatible with the definitions made for this prototype. In order to get the result shown in illustration 45, the provided representation (NURBS-Spline plus knot vector in *.ma Maya shape format) had to be converted into the native Catmull-Rom description. Since the drawing order of the data sets does not comply with with the presented extraction methods, contains off-surface features with no direct network incidence and complex freeform boundaries, the network has been reconstructed manually. This process has been logged and took about 31 minutes, five additional curves at the door section to preserve main shape features and another two curves have been left out (one front-lamp and an off surface feature defining the car front, assuming symmetry). Note that the reconstruction of the network is not comparable to the original creation process, as designing the network from scratch is a far more demanding task. Therefore the focus on finding acceptable topological correspondences to aid the surface construction task for this data. Despite those simplifications the reconstructed surfaces still offers significant disadvantages towards preserving the network features, since the minimal spanning surface principle from chapter 4.3 has been applied. Nevertheless all surfaces could have been recovered, keeping to the inherent system construction rules and the reconstructed network captures the complexity of the original.



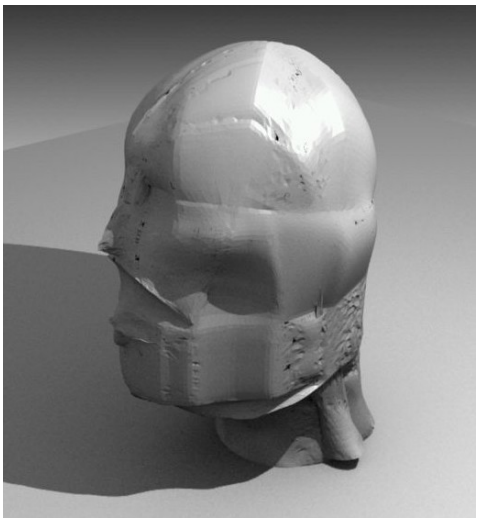
(a) Pre-Sketch



(b) First Head Approximation



(c) Addition of details



(d) Rendering in Blender

Figure 46: A first approach towards very complex objects as an human head. The initial extrusion operations in 46b deliver quite promising results, whereas the addition of further details in 46c remains unsatisfactory. Note that the smaller surface discontinuities in figure 46d stem from multiple surfaces extracted for one patch, due to extraction failures.

The final challenge for every non-technical modeling system is the faithful representation of objects, which are very familiar to the viewer as described in section 2.2. Towards this class of shapes, recognition delivers many additional shape information and therefore very detailed and complex requirements are inferred to corresponding 3D models. Although the first steps in 46b seem quite promising, the current prototype fails to reflect additional complex and specific details of such models. The main reasons for this failure can be seen in the missing correlation between corresponding surface parts and the undefined or erroneous integration of additional curve segments into existing surface structures. Creating the model in its final state took 28 minutes, 48 lines (without symmetry) and three extrusion operations.

The prototype has been tested on several different computers which were characterized by a general CPU speed of at least 2.0 GHz and 1.0GB RAM using a GeForce 7600 Go Graphics Card or better. Ensuring those minimal requirements, the prototype offers interactive modeling speed also for higher model complexities as the preceding examples have shown.

Concerning the general extraction methods provided by the systems, the following statements can be made:

Within the course of the evaluation of the testing data it has been proven that the batch extraction method for four sided patches works stable on quadrangular networks (see example 42). In general the constructional effort for those kind of networks is too high, to be effectively applied. On the other hand this procedure can be applied for both extrusion operators, while they are guaranteed to produce quadrangular surface boundaries along the extruded sequence.

For patch sequences with more than four boundaries, the progressive method delivers acceptable results, but also tends to behave counterintuitive if the current limitations towards the active boundary constraints are violated, such as in example 44.

In reference to the initially stated guidelines for conceptual modeling tools, the system has been shown to produce complex aggregations of surfaces, which still allow an appropriate amount of control to express the intended shapes. Therefore the system benefits from its amount of expressiveness, which also allows the construction of problematic shapes like torus objects in 41. As there are no explicit methods for including predefined geometry others than lines, most shapes only can be approximated using freeform curves. This limits the achievable exactness of the system in a significant way which is especially reflected within the extrusion examples 41d and 41c. Therefore the application is also restricted to distinct conceptual purposes, while further reuse of such data might be problematic, when specific requirements towards size, scale and overall proportions are given. However, further modeling e.g. with subdivision freeform modeling tools is not affected in a significant way, which makes the system a useful tool to construct preceding base meshes.

The requirements towards similarity of shape operations can be fulfilled easily, as basically only a single modeling metaphor is exploited as demonstrated by the deformed cube example in figure 36. In contrast the validity of the proposed methods cannot be guaranteed in general, as further considerations of geometric constraints have to be taken into account, that would impede the interactive aspect of the prototype. Finally storing the shape indirectly within a curve network is a very compact way to represent geometry. The savings made to store the geometric information, have to be reinvested in their interpretation, as the unique results can only be achieved, if exactly the same surface reconstruction and parameterizations methods are applied.

The implementation of the prototype and the general evaluation of achieved results now allows for a detailed discussion with respect to advantages and shortcomings of the system. The given comparison will focus on the developed main features of the system and put them into relation to previously proposed approaches.

3D Sketch-Shape Construction

Considering the implemented features and procedures, there are two approaches which are conceptually closest to the proposed system: The 3D sketching system presented by Das et al. [DDGG05] and similar system introduced by Sheng et al. [SWS08]. Both systems are optimized towards the sketch driven creation of free-form surfaces reconstructed from parameterized boundary curves and deliver valuable insights into the handling of sketch-like data. In contrast to the presented prototype both systems concentrate on the interpretation of 2D line drawings and subsequent 3D reconstruction (see the mapping-problem in section 2.4). The first system [DDGG05] further offers the automatic interpretation of 2D lines as minimal curvature space curves, automatic critical point extraction and constrained triangulation of the boundary contours, whereas the construction scheme is similar to the quad-mesh procedure presented in 4.5. The second approach [SWS08] offers a surface extraction scheme which is similar to the active boundary concept and automatically computes normal directions for the curve network, which are used for reconstruction. Unfortunately, as stated in the publication, this reconstruction is too time consuming for interactive application. In comparison the implemented prototype circumvents the 2D sketch-definition, includes additional high-level operations and therefore allows for more expressive and complex models. Furthermore existing geometry can be taken as context for iterative modeling procedures and the quad-mesh approach allows for surface higher resolution and improved representation of shape features. Another advantage can be seen in the interactivity, which is offered by our system up to a reasonable model complexity.

Conceptual SBIM Modeling

Instead of comparing our prototype to all recently stated conceptual SBIM systems, two prominent ambassadors are selected, that offer a set of features commonly available in related systems: On the one hand ShapeShop, whereas the first basic framework was presented by Schmidt et al. [SWSJ06], and on the other hand SketchCAD by Kara et al. [KS07].

The current version of ShapeShop[1] represents an aggregation of multiple SBIM related contributions (Surface Hierarchies [SS08], Visual NPR History [SIJ+07], Sketch Widgets [SSB08], ...) and can be considered one of the currently most advanced conceptual 3D shape modeling systems. The main properties have already been outlined in section **??**. In comparison to our approach ShapeShop takes another direction, as the intended shape is aggregated by multiple simpler objects, whereas our prototype orients at a closed patch based description. These definitions represent an implicit volumetric approach on the one side and a patch-based B-Rep on the other side, that means in oversimplified terms: The strength of one system is a general weakpoint of the other one and vice versa. The effective implicit representation in ShapeShop offers fast blending of complex shapes and guarantees intersections free, intuitive surfaces at any resolution (limited only by computational effort and meaningful shape representation). At least the first two facts have to be taken into special consideration within our prototype. Furthermore the definition of holes and structured shape hierarchies is provided by the CSG-like nature of this representation. One downside of the implicity approach is, that it fails to reflect sharp or creased surface features at lower levels of resolution and requires the combination of multiple modeling metaphors and operators, to reflect possible shape variations. These distinct metaphors imply a separation of shape deformation and control operations, as there is only an abstract relation between moving a slider and the impacts on the perceived object features and silhouette. Nevertheless the expressiveness provided by both

1 ShapeShop B5, available at http://www.shapeshop3d.com/download.html (last access on June 2009)

systems can be considered high enough, to be effectively applied during conceptual modeling stages.

In contrast SketchCAD orients (as the name implies) on a technical CAD oriented construction process, providing additional styling functionality and interfaces for sketch-based input. The approach basically consists of a set of augmentation and deformation operators, which require for a predefined reference geometry. This approach offers high-quality, flexible surface extraction and intuitive definition of space curves on reference model surface plus additional restyling operators. Moreover the interactive application of the system has been proofed by the construction of elaborate examples, which reflect the strong relation towards technical CAD systems. With respect to the given examples, this approach seems less appropriate for design exploration and conceptual modeling, as the general resulting shape complexity seems to be significantly influenced by the complexity of the template geometry. More expressive augmenting operations as parameterized extrusions, as proposed within our prototype, might be a valuable extension to this approach. Similar technical CAD oriented approaches exploiting expressive modeling metaphors as extrusion (e.g. SESAME [**?**] and the commercial package Google SketchUP) have to balance the resulting multiple parametrization options versus maintaining the reduced simplicity using a consistent modeling metaphor. The developed prototype proposes one way to integrate those operators into one major space-curve metaphor.

Surface Extraction

Another system that handles similar tasks as our prototype, but focusses on the application of developable surfaces is presented by Rose et al. [RSW+07]. On the one hand the system offers the advantage of potentially finding all surface configurations of the underlying network with respect to the developable surface description and can handle more complex shapes than the two previously mentioned approaches. On the other hand it does not propose general methods how to construct those networks, except the use of complex reference geometry and does not consider interactive application within a general design-workflow, but more towards a batch extraction procedure for final curve networks. Although developable surfaces are able to achieve intuitive and high quality surfaces for the given networks, they do not directly offer a smooth transition between different surface configurations.

With regard to the extraction of high quality surfaces there are three approaches which offer excellent results. Namely one system motivated by the medical applications by Liu et al. [LLB+08] and two systems that focus directly on surface extraction from closed curve networks: The subdivision based approach presented by Levin et al. [Lev99] and the lofting approach presented by Schaefer et al. [SWZ04].

The system by Liu et al. presents an advanced extraction method which allows for complex high surface extraction for off-surface curves, if the constructional relation between the input curves is given. The system can handle the extraction of multiple objects and allows the construction and blending of branching shape structures. As the system is not oriented towards the sketch-modeling topic interactivity, handling of different surface features and intuitive spatial boundary descriptions were not of concern and therefore no subject to direct comparison. Nevertheless it offers valuable concepts for further improvement of SBIM surface extraction procedures and evaluation of geometric boundary relations.

With respect to the surface quality the subdivision based approaches by Levin et al. [Lev99] and Schaefer et al. [SWZ04] provide superior surface qualities and feature representation, whereas the latter approach is the more general one. Both approaches proof, that subdivision is a well suited method to enhance current surface extraction results. Due to this fact they should be in focus of further efforts to improve the presented surface quality and especially to express smoothed and creased features with the network. In addition to those concepts the presented prototype offers a progressive, semi-automatic and intuitive SBIM oriented surface

construction, which can deliver the necessary topological structures for these subdivision procedures.

Space Curve Construction

Concentrating on the curve construction aspect the proposed system can be compared to the spline construction system presented by Michalik et al. [MKB02] and the ILoveSketch System presented by Bae et al. [BBS08]. The methods proposed by Michalik et al. are oriented towards the patch based spline construction directly in 3D space and offer first extended operators, how to put given space curves into context without having predefined a complex template geometry. The resulting surfaces provide an high quality and are, due to the provided over-sketching functionality, flexible and expressive. One downside which is shared by the presented prototype, is the fact, that construded surfaces are not taken into a topological relation and therefore higher-level dependencies between surfaces or off surface features cannot be expressed. Our prototype extends the proposed methods with regard to their expressiveness and the abstraction of the resulting surface descriptions towards a more sketch-oriented workflow.

Considering the workflow aspect ILoveSketch by Bae et al. can be considered as the currently most advanced state of the art system. The approach strictly implements a conceptual 2D working metaphor and presents a completely sketch driven interface with semi-automatic 3D navigation concept that is oriented towards the *sketchability* of the provided input. As the given insightful evaluation with a professional artist has shown, some features still require mostly repetitive actions (e.g. the deletion of unmatched line segments) and might be enriched be additional high-level curve extraction methods. Our proposed prototype expands the pattern aspect of instantiated geometry above axis symmetry, proposes basic high level sketch-operators, which might integrate into a similar design workflow and proposes specific methods, how to aid the subsequent topological shape construction process, that has not been part of the ILoveSketch project.

The presented comparison concludes the overview about the final results achieved with the prototype and correlates implemented features to existing systems. These relations allow for the critical reflection of the implemented concepts and profund, substantial statements concerning the major challenges stated at the beginning of this thesis.

Chapter 6

CONCLUSION

6

CONCLUSION

6.1 SUMMARY

In order to reach our goals, stated in chapter 1, the existing fundamental concepts have been discussed and analyzed towards their properties and underlying concepts. The general findings of this discussion have been reformulated as a set of general *guidelines* given in section 2.6, which have been used to set up a concept for an appropriate system framework.

In the first part of the concept chapter an overall system structure is developed, which reflects the necessities of an iterative conceptual workflow, the modular description of essential components and support for a fast an interactive creation of shapes from the given input. It is argued that *3D space curves* and aggregated *parameterized curve networks* represent an appropriate way to capture the characteristics of the input data and support the acquisition of further topology structures.

The formal definition of this network structured is developed in section 3.3. To support the shape extraction from this network description two additional shape representation have been integrated: a subdivision oriented *quad mesh* structure, to obtain a flexible and versatile surface description, and a *triangular mesh* structure to perform further rendering tasks and allow for standardized export of constructed shapes.

In section 3.4 general interaction methods for the effective and expressive creations of curves within a 3D environment have been presented. These involve the use of an additional *support geometry*, a set of basic curve operators for controlling geometric properties and further advanced operators, that allow for fast and abstract construction of complex network structures.

The surface extraction itself is accomplished by introducing two general concepts in section 3.5: The *batch processing* allows for the extraction of surfaces out of a complete network in one single process. Therefore the concept of minimal pathes in a graph structure has been used and augmented by the abstract *tangent-loop* concept. It has been shown, that for a constrained set of networks this procedure delivers acceptable and stable results. The *progressive extraction* method raises the constructional effort during the definition of the network, but offers immediate, fast and intuitive extraction of arbitrary complex surface-structures. The method of the *active boundary* reconstruction supports these properties, although there also have been identified specific limitations regarding the underlying network structure.

To control the resulting patch surfaces, the integrated quad mesh description has been extended by a general force driven parametrization method. Furthermore, additional visual techniques to render and display surfaces of objects during conceptual modeling processes have been proposed.

Based on this overall concept, a prototype has been implemented, whereas detailed formulations and procedures are presented in chapter **??**. This includes details regarding the implemented interaction concepts in section 4.3, the discussion of the the significant impact of the parametrization on the basis of four sided Coons-patches and further details on the topo-

logical extraction process. The final prototype has been tested against different input networks, in order to evaluate the realized concepts and oppose them to the initial propositions. These testing scenarios and their evaluation have been followed by a detailed comparison to directly related approaches, based on the actually implemented features in section 5.4. This direct comparison identified common future challenges and concludes the practical considerations of this thesis.

6.2 CONTRIBUTIONS

The core achievements of this thesis can be summarized as follows:

1 **Methods for intuitive and expressive Curve Construction**
The prototype uses an aggregation of basic and advanced curve operators and controlled augmenting geometry, to intuitively define and control the creation of space curves within a 3D scene. With raising scene complexity, more complex strokes can be defined, exploiting the created geometry.

2 **Consistent iterative Modeling Metaphor**
General parameterized curve networks, together with the curve operators allow for expressive 3D conceptual modeling of a wide variety of shapes. The network supports creation, deformation and augmentation of shapes, ranging from sharp distinct features as well as smooth, organic objects and hybrid configurations. Furthermore, the metaphor offers the possibility to integrate existing modeling approaches into a consistent framework. In analogy to the curve construction, more complex shapes are aggregated by comprising previously created geometry.

3 **Intuitive Surface Construction Procedure**
Together with the network construction an flexible, iterative and topology driven extraction procedure has been proposed, which supports a wide variety of complex configurations and can be applied interactively during the modeling process.

4 **Optimized and Flexible System Framework**
The prototype system framework and the selected shape representations have been developed and optimized towards the flexible description of different sketch-driven input scenarios and the fast evaluation of topological dependencies. Its modular description covers possible extensions for inherent functionality and the integration of new components

Besides those main advantages, the realization of the prototype and its features provide insight into the development of a conceptual SBIM system, in order to derive challenges and effective concepts for further improvement and comparison. The consideration and implementation of psychological, ergonomic and artistic factors has lead to natural extensions for existing approaches and the proposition of a substantial theoretic basis for further discussions in conceptual 3D development.

6.3 LIMITATIONS

The comparison of the developed prototype against the initially stated challenges from section 2.6 facilitates the critical reflection of the general development and reveals aspects, that require further attention in order to improve their applicability. The presented aspects relate to the construction of the example models in section 5.2 and general aspects, that occurred during the implementation.

With regard to the interface, the system shares the mentioned *self disclosure* problem with other SBIM systems. Although the basic drawing- and surface extraction processes can be explored easily, the access to advanced operators requires additional learning effort and abstract understanding of the underlying concepts, which are not provided by the system itself. The extension of the prototype towards an *appropriate interface- and interaction metaphor* remains an open issue, which has not been in focus during the technical development. The main concept stated that the control of the techniques includes a limited set of modifier- and interaction keys, whereas in the current state there is no consistent interface concept for these control operations.

Interface

The interaction with the system reveals a significant weakness when it comes to the *exploration of different surface parametrizations*. In the current system this issue is addressed by exploiting an abstract parameter mapping of input positions and surface parameters, which does not comply with the general modeling metaphor. Especially for the interaction with complex surface patches the general application of minimal spanning surfaces (described in section 4.3) does not offer satisfactory results in a general manner.

Interaction

Although versatile geometric and topological information can be extracted from the network, more detailed styling operations will be necessary to finalize shapes or specify distinct geometric properties. This also includes the faithful representation of continuity constraints between surface patches. Although a general force based method has been proposed to approximate these continuities, their final appearance is not convincing, especially in comparison to related subdivision approaches from Levin et al. [Lev99] and Schaeffer et al. [SWZ04].

Furthermore, the current system does not handle all special cases which can occur using the active boundary concept. Especially the connection of two separate closed boundaries would require significant additional computational effort and the handling of potential problematic configurations. This might be solved applying an extended geometric analysis, although the discussion from section 4.4 implies, that this analysis can be assumed to fail in specific cases. As a result, these cases either have to be explicitly defined or will require further user interaction. Solving this general problem also requires an answer on how to solve the twist problem mentioned in the same section.

A general drawback with respect to the shape reconstruction, is the definition of holes within the object structures. In general they can be constructed, but require elaborate constructional effort. Defining them as separate geometrical structures as done in Teddy [IMT06] or ShapeShop [SWSJ06] might extend the achievable shape space, but also implies new requirements towards the user interface and the surface definition.

Shape Construction

Another open aspect is the missing relation between separate patches of an object. The dependencies between several surfaces within the shape and their spatial correspondences will further require an explicit formulation, in order to improve the overall shape appearance and -handling.

Visual Representation

According to the visual feedback, one major issue is the depiction of the *conceptual incompleteness* of the modeled shape. The used representation still implies a high level of completeness, due to the explicit surface representation. On the one hand, these surface definitions do not reflect the general amount of possible configurations for the given curve configuration. On the other hand, they fail to faithfully capture all surface features. The application of NPR related rendering techniques for the enhancement of silhouette- and suggestive features [DFRS03] potentially improves the perception of depth and general spatial relations within the scene.

In addition to the above mentioned aspects a general *user study* with the system is definitely indispensable to substantiate the claim for intuitive use and construction efforts above theoretical or potentially biased considerations.

6.4 POSSIBLE EXTENSIONS AND FUTURE WORK

The preceding discussion of general weaknesses of the prototype already hints on general aspects, which are valuable to be improved.

As already mentioned, one main future goal will be the practical assessment of the proposed prototype and its developed features. A specific user study with a larger spectrum of users with different 3D modeling background, will inspire further improvements towards the system and get the hole system closer to the goal of its practical embedding.

Another important aspect is the improvement of the resulting surface quality with regard to the faithful representation of inherent shape features, as continuity and the general presence of geometric features within the reconstructed surfaces. Existing subdivision approaches (e.g. [SWZ04], [Lev99]) already offer a good perspective for further efforts within this direction. Concerning further surface interaction techniques, valuable considerations can be done towards the use of inherently available structures within the network. These can lead to general extensions towards edge-flow oriented surface operators, which is a common elaborate working paradigm among detailed subdivision surface design. Therefore a general formulation of those aspects potentially improves the integration into current systems.

As a general modeling metaphor has been proposed, the overall concept is still missing a consistent and appropriate concept for the interface and interaction control operations, which has been outlined within the previous chapter. In order to define a consistent overall system design, the incorporation of further SBIM features, like augmenting widgets or gestures remains promising.

In addition to this, the second part of our second fundamental question from the beginning is still missing a substantial answer. The practical improvement and optimization of conceptual SBIM techniques can only be accomplished, by finally integrating them into a real practical workflow. Therefore the common goal remains to improve existing techniques up to the point, where the resulting benefits are perceived high enough, to motivate their practical integration.

These final remarks will finish the overall considerations of this thesis. It claims to have taken some small steps into the direction of a common goal, which still remains a fascinating, lucrative and multi-faceted research topic:

Practical and efficient sketch-based 3D conceptual modeling.

LIST OF FIGURES

Part I

BIBLIOGRAPHY

BIBLIOGRAPHY

[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. (Cited on page 27.)

[BBS08] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology, pages 151–160, New York, NY, USA, 2008. ACM. (Cited on pages 9, 17, 23, 31, 41, 42, 84, 89, and 97.)

[BE93] Heinrich H. Bülthoff and Shimon Edelman. Evaluating object recognition theories by computer graphics psychophysics. In In, pages 139–164. Wiley, 1993. (Cited on page 16.)

[BPCB08] Adrien Bernhardt, Adeline Pihuit, Marie-Paule Cani, and Loic 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), pages 57–64, Annecy, France, june 2008. (Cited on page 23.)

[BPK+07] Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff, and Christian Rössl. Geometric modeling based on polygonal meshes. In SIGGRAPH '07: ACM SIGGRAPH 2007 courses, page 1, New York, NY, USA, 2007. ACM. (Cited on page 26.)

[BV07] Gill Barequet and Amir Vaxman. Nonlinear interpolation between slices. In SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling, pages 97–107, New York, NY, USA, 2007. ACM. (Cited on pages 24 and 45.)

[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), aug 2008. (Cited on pages 16, 17, 18, 19, and 95.)

[CMZ+99] Jonathan M. Cohen, Lee Markosian, Robert C. Zeleznik, John F. Hughes, and Ronen Barzel. An interface for sketching 3d curves. In SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics, pages 17–21, New York, NY, USA, 1999. ACM Press. (Cited on pages 23, 41, and 42.)

[CPC04] Pedro Company, Ana Piquer, and Manuel Contero. On the evolution of geometrical reconstruction as a core technology to sketch-based modeling. pages 97–106, 2004. (Cited on page 22.)

[CR74] E Catmull and R Rom. A class of local interpolation splines. 1974. (Cited on pages 26 and 55.)

[DDGG05] Koel Das, Pablo Diaz-Gutierrez, and M. Gopi. Sketching free-form surfaces using network of curves. In SBIM '05: Proceedings of the 2nd Eurographics workshop on Sketch-based interfaces and modeling, New York, NY, USA, 2005. ACM. (Cited on pages 38, 68, and 87.)

[DFRS03] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. ACM Transactions on Graphics (Proc. SIGGRAPH), 22(3):848–855, July 2003. (Cited on pages 16, 21, and 94.)

[Far02] Gerald Farin. Curves and surfaces for CAGD: a practical guide. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. (Cited on pages 26, 60, and 61.)

[FCAD03] N. Faria, T. Cardoso, B. Araújo, and F. Dias. Gides++ - 1st usability test at centifme. Technical report, INESC-ID, 2003. (Cited on pages 14, 21, and 42.)

[FH99] Gerald Farin and Dianne Hansford. Discrete coons patches. Comput. Aided Geom. Des., 16(7):691–700, 1999. (Cited on pages 39, 59, 61, 62, and 96.)

[FvDFH95] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. Computer Graphics: Principles and Practice in C. Addison-Wesley Professional, second edition, August 1995. (Cited on page 26.)

[Gol02] E. Bruce Goldstein. Senstation and Perception (Wahrnehmungspsychologie), volume 4th German Edition. Wadsworth Publishing Co., Spektrum Akademischer Verlag, Heidelberg, Berlin, 2002. ISBN 3-8274-1083-5. (Cited on page 15.)

[GZ08] Yotam Gingold and Denis Zorin. Shading-based surface editing. ACM Trans. Graph., 27(3):1–9, 2008. (Cited on page 20.)

[HS97] Donald D. Hoffman and Manish Singh. Salience of visual parts. Cognition, 63(1):29 – 78, 1997. (Cited on pages 15 and 16.)

[IH01] Takeo Igarashi and John F. Hughes. A suggestive interface for 3d drawing. In UIST, pages 173–181, 2001. (Cited on page 21.)

[IH03] Takeo Igarashi and John F. Hughes. Smooth meshes for sketch-based freeform modeling. In I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics, pages 139–142, New York, NY, USA, 2003. ACM. (Cited on page 27.)

[IMT06] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In SIGGRAPH '06: ACM SIGGRAPH 2006 Courses, page 11, New York, NY, USA, 2006. ACM. (Cited on pages 20, 23, 40, 42, 49, 66, 69, and 93.)

[JWC+05] Tao Ju, Joe D. Warren, James Carson, Gregor Eichele, Christina Thaller, Wah Chiu, Musodiq Bello, and Ioannis A. Kakadiaris. Building 3d surface networks from 2d curve networks with application to anatomical modeling. The Visual Computer, 21(8-10):764–773, 2005. (Cited on pages 24 and 45.)

[KH06] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. SIGGRAPH2006, 25(3):589–598, July 2006. (Cited on pages 22 and 23.)

[KHR02] Olga Karpenko, John F. Hughes, and Ramesh Raskar. Free-form sketching with variational implicit surfaces. Computer Graphics Forum, 21:585–594, 2002. (Cited on page 27.)

[KQW06] D C Ku, S F Qin, and D K Wright. K.: A sketching interface for 3d modeling of polyhedrons. In Eurographics Workshop on Sketch-Based Interfaces and Modelling, pages 83–90, 2006. (Cited on page 22.)

[KS07] Levent Burak Kara and Kenji Shimada. Sketch-based 3d-shape creation for industrial styling design. IEEE Comput. Graph. Appl., 27(1):60–71, 2007. (Cited on pages 9, 19, 21, 42, and 87.)

[LBS06] Torsten Langer, Alexander Belyaev, and Hans-Peter Seidel. Spherical barycentric coordinates. In SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, pages 81–88, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. (Cited on page 43.)

[Le07] Khang Le. Sketch gallery. Website, 2007. Available online at `http://www.khangle.net/`; visited on June 2009. (Cited on page 18.)

[Lev99] Adi Levin. Interpolating nets of curves by smooth subdivision surfaces. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pages 57–64, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. (Cited on pages 27, 88, 93, and 94.)

[LG07] Florian Levet and Xavier Granier. Improved skeleton extraction and surface generation for sketch-based modeling. In GI '07: Proceedings of Graphics Interface 2007, pages 27–33, New York, NY, USA, 2007. ACM. (Cited on page 69.)

[LLB+08] Liu, L., Bajaj, C., Deasy, J. O., Low, D. A., Ju, and T. Surface reconstruction from non-parallel curve networks. Computer Graphics Forum, 27(2):155–163, April 2008. (Cited on pages 24, 45, and 88.)

[LS07] H Lipson and M Shpitalni. Optimization-based reconstruction of a 3d object from a single freehand line drawing. In SIGGRAPH '07: ACM SIGGRAPH 2007 courses, page 45, New York, NY, USA, 2007. ACM. (Cited on page 22.)

[LSM02] Eric B. Lum, Aleksander Stompel, and Kwan Liu Ma. Kinetic visualization: a technique for illustrating 3d shape and structure. In VIS '02: Proceedings of the conference on Visualization '02, pages 435–442, Washington, DC, USA, 2002. IEEE Computer Society. (Cited on page 49.)

[MKB02] Paul Michalik, Dae Hyun Kim, and Beat D. Bruderlin. Sketch- and constraint-based design of b-spline surfaces. In SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications, pages 297–304, New York, NY, USA, 2002. ACM. (Cited on pages 23, 42, and 89.)

[ML07] M. Masry and H. Lipson. A sketch-based interface for iterative design and analysis of 3d objects. In SIGGRAPH '07: ACM SIGGRAPH 2007 courses, page 31, New York, NY, USA, 2007. ACM. (Cited on pages 22 and 95.)

[NCC+03] Ferran Naya, Julián Conesa, Manuel Contero, Pedro Company, and Joaquim Jorge. Smart sketch system for 3d reconstruction based modeling. In Lecture Notes in Computer Science, pages 58–68, 2003. (Cited on page 22.)

[NISA07] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. In SIGGRAPH '07: ACM SIGGRAPH 2007 papers, page 41, New York, NY, USA, 2007. ACM. (Cited on pages 14, 20, and 23.)

[OHH+08] Shigeru Owada, Takahiro Harada, Philipp Holzer, , and Takeo Igarashi. Volume painter: Geometry-guided volume modeling by sketching on the cross-section. pages 9–16, 2008. (Cited on pages 20 and 27.)

[ONI06] Shigeru Owada, Frank Nielsen, and Takeo Igarashi. Copy-paste synthesis of 3d geometry with repetitive patterns. In Smart Graphics, pages 184–193, 2006. (Cited on pages 21 and 43.)

[OSD06] Ji-Young Oh, Wolfgang Stuerzlinger, and John Danahy. Sesame: towards better 3d conceptual design systems. In DIS '06: Proceedings of the 6th conference on Designing Interactive systems, pages 80–89, New York, NY, USA, 2006. ACM. (Cited on pages 42 and 72.)

[OSSJ09] L. Olsen, F.F. Samavati, M.C. Sousa, and J. Jorge. Sketch-based modeling: A survey. Computers & Graphics, 33:85–103, 2009. (Cited on pages 12, 19, 20, 28, and 95.)

[Ree83] W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. ACM Trans. Graph., 2(2):91–108, 1983. (Cited on page 49.)

[RSW+07] Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. Developable surfaces from arbitrary sketched boundaries. In SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing, pages 163–172, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. (Cited on pages 24 and 88.)

[SIJ+07] Ryan Schmidt, Tobias Isenberg, Pauline Jepp, Karan Singh, and Brian Wyvill. Sketching, scaffolding, and inking: a visual history for interactive 3d modeling. In NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering, pages 23–32, New York, NY, USA, 2007. ACM. (Cited on pages 48 and 87.)

[SOD05] W. Stuerzlinger, Ji-Young Oh, and John Danahy. Comparing sesame and sketching on paper for conceptual 3d design. In Sketch Based Interfaces and Modeling, number ISSN 1812-3503, pages 81–88. York University, 2005. (Cited on pages 17, 42, and 72.)

[SRS91] Emanuel Sachs, Andrew Roberts, and David Stoops. 3-draw: A tool for designing 3d shapes. IEEE Comput. Graph. Appl., 11(6):18–26, 1991. (Cited on pages 8 and 22.)

[SS08] Ryan Schmidt and Karan Singh. Sketch-based procedural surface modeling and compositing using Surface Trees. Computer Graphics Forum, 27(2):321–330, 2008. Proceedings of Eurographics 2008. (Cited on pages 20 and 87.)

[SSB08] Ryan Schmidt, Karan Singh, and Ravin Balakrishnan. Sketching and composing widgets for 3d manipulation. Computer Graphics Forum, 27(2):301–310, 2008. Proceedings of Eurographics 2008. (Cited on pages 24 and 87.)

[SSGD03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. Shape Modeling and Applications, International Conference on, 0:130, 2003. (Cited on page 21.)

[SSS06] Aaron Severn, Faramarz Samavati, and Mario Costa Sousa. Transformation strokes. In Eurographics Workshop on Sketch-Based Interfaces and Modeling, pages 75–82, September 2006. (Cited on page 43.)

[Sut63] Ivan E. Sutherland. Sketchpad: A Man-Machine Graphical Communication System. PhD thesis, Massachusetts Institute of Technology, Lincoln Lab, 1963. Also published as technical report UCAM-CL-TR-574 of the University of Cambridge, UK, Computer Laboratory. (Cited on page 8.)

[SWS08] Bin Sheng, Enhua Wu, and Hanqiu Sun. Sketching freeform meshes using graph rotation functions. The Visual Computer, 24(7-9):745–752, July 2008. (Cited on pages 46 and 87.)

[SWSJ06] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: sketch-based solid modeling with blobtrees. In SIGGRAPH '06: ACM SIGGRAPH 2006 Courses, page 14, New York, NY, USA, 2006. ACM. (Cited on pages 14, 20, 21, 23, 31, 42, 49, 66, 87, and 93.)

[SWZ04] S. Schaefer, J. Warren, and D. Zorin. Lofting curve networks using subdivision surfaces. In SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 103–114, New York, NY, USA, 2004. ACM. (Cited on pages 27, 38, 39, 88, 93, and 94.)

[TO99] Greg Turk and James F. O'brien. Variational implicit surfaces. Technical report, Georgia Institute of Technology, 1999. (Cited on page 27.)

[VH98] I.M. Verstijnen and J.M. Hennessey. Sketching and creative discovery. Design Studies, (19):519–546, 1998. (Cited on page 17.)

[WTBS07] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Shapepalettes: interactive normal transfer via sketching. In SIGGRAPH '07: ACM SIGGRAPH 2007 papers, page 44, New York, NY, USA, 2007. ACM. (Cited on page 20.)

[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. (Cited on pages 8 and 22.)

[ZNA07] Johannes Zimmermann, Andrew Nealen, and Marc Alexa. Silsketch: automated sketch-based editing of surface meshes. In SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling, pages 23–30, New York, NY, USA, 2007. ACM. (Cited on pages 20 and 42.)

[ZS00] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. Technical report, SIGGRAPH 2000, 2000. Course Notes. (Cited on page 26.)