Freeform feature recognition and manipulation to support shape design

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. J. T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op donderdag 15 mei 2008 om 10:00 uur
door

Thomas Robin LANGERAK

doctorandus in de informatica,
geboren te Tilburg.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. I. Horváth

Toegevoegd promotor:

Dr. J.S.M. Vergeest

Samenstelling promotiecommissie:

Rector Magnificus	Voorzitter
Prof. dr. I. Horváth	Technische Universiteit Delft, promotor
Dr. J.S.M. Vergeest	Technische Universiteit Delft, toegevoegd promotor
Prof. dr. R.R. Martin	Cardiff University, United Kingdom
Prof. dr. Y.S. Kim	Sungkyunkwan University, Korea
Prof. dr. M. de Berg	Technische Universiteit Eindhoven
Prof. dr. ir. I. S. Sariyildiz	Technische Universiteit Delft
Prof. dr. P.J. Stappers	Technische Universiteit Delft
Prof. dr. P.G. Badke-Schaub	Technische Universiteit Delft, reservelid

This research was financially supported by the Dutch Technology Foundation STW
(projectnumber 06240)

ISBN: 978-90-5155-046-7

Cover designed by Tom Langerak

Acknowledgements

The making of a Ph.D. thesis seems to be the effort of a single person, but there are several people I would like to thank for the support they gave me. First of all, my supervisor Joris Vergeest: the many discussions I had with you over the years contributed much to the direction the promotion research has taken. But most of all I thank you for the opportunity you gave me to determine my own direction. I think I was never a difficult Ph.D. candidate, but that was only possible because you were never a difficult supervisor. Thanks also to Imre Horváth, my promotor, who became involved in my research much too late. If only we would have had more time for heavy discussions, the thesis would have looked differently, and the research would have been more thorough. I don't think I'll easily forget how your eyes started to twinkle whenever we had a discussion that took your mind off managerial tasks for a moment. Thanks also to the reviewers, who helped me with their comments to lift the thesis to a higher level.

Thank you Tom, dad, for designing the cover of this thesis, which for most people will probably be the only part of it they will remember. In addition, your constant 'whining' (you know what I mean) was always a good reality check. Thanks Nel, mom, for promising to not disturb me when I came to work at your place (but never succeeding). Richard, thanks for providing me with a Rubik's Cube when I was in serious need of distraction. All of you, thanks for being my family.

Thanks to Sma for her unconditional love during the making of the thesis. Because she cannot read, I will make it up with special cat dinners.

Thanks to all my colleagues, who provided me with a pleasant and stimulating work environment for four years. Special thanks go out to Wolf, who went out of his way to help me at times, and always proved to be a valuable discussion partner when it came to freeform features.

Finally, I'd like to thank Fieke, my official girlfriend and partner in so many things. The joy of having finished a thesis is nothing compared to the joy of coming home to a happy and safe environment every day and the feeling of being loved.

Legend of symbols

Symbol	Description	Page
A	Area configuration of a feature	57
F	A feature	57
$I\left(F', F''\right)$	Region of intersection of two features	64
k	Number of control point sets	67
n	Number of control points	67
m	Number of parameters of a feature	67
M	Feature shape manipulation	60
p^l	l-th parameter of a feature	57
P	Vector of parameter values	57
P^0	Origin of parameter space	57
P^{area}	Subset of area parameters	72
$\tilde{P}$	Vector of parameter values prior to shape manipulation	60
ROI	Region of interest	100
$e_{i,j}$	Control point in the i-th row and j-th column of the grid	67
μ	Parameter mapping	57
μ_i^l	Mapping function on control point $e_{i,j}$ and parameter p^l	67
ψ	Shape combination function	65
φ	Function mask matrix	67
χ	Correspondence function	74
	Also: Mutation probability	112
Π	Population size	112

σ	Selection size	112
ϕ	Mutation rate	112
E	Shape configuration of a feature	57
E^0	Basic shape configuration of a feature	57
E_{area}	Region of influence of a feature	72
$\tilde{E}$	Shape configuration prior to manipulation	60
Θ	Trajectory curve	123
θ_i	i-th control point on a trajectory curve	123
Υ	Profile curve	123
ρ_i	i-th control point on a profile curve	123
$\wp$	Feature recognition procedure	145
δ	Direction matrix	96
	Also: Distance between two intersection planes	126
λ	Number of control points in a profile curve	126

Table of contents

List of figures

List of definitions

1 Introduction

Imagine that you are an archeologist that is examining an archeological site. After hours of carefully removing soil from an object, you finally hold it in front of you and recognize it as an amphora. Part of this recognition comes from a comparison of the shape of the object to that of other amphorae that you encountered in other archeological projects or that you have read about. From this previous experience you have obtained an idea of the characteristic shape of an amphora, for example with regard to the curvature of its body or the shape of its handles. Although the shape of this particular amphora may differ slightly from this characteristic shape, the resemblance is good enough for recognition. For example, the handles of the amphora may differ from those of other amphorae in size and shape yet are clearly identifiable as handles. Once the object has been recognized, functional, material and historical information that is commonly associated with amphorae can also be associated with this specific example, allowing deductions and speculations regarding its origin. It was, for example, used for storing wine or olive oil and judging by its shape it was manufactured in a specific historical age.
The same archeological site may yield an object that can not so easily be recognized, for example because it is damaged. Still it may have some characteristic shape parts which can be used to identify the object. For example, if a handle is present on an undamaged part of the object, then together with other hints this may lead to the conclusion that the object is an amphora. Although the object can not be identified by a direct relation to existing knowledge, it may be identified by analyzing characteristic shape parts.

These examples demonstrate the concept of *form features*. Form features are characteristic shapes or part thereof to which numerical parameters can be attributed. The values of these parameters correspond to a certain state of the shape of the feature. For example, the height of an amphora can be identified as a parameter, of which the value varies for different amphorae.
The form feature can be identified by reference to existing knowledge. We have, as it were, a ‘list’ of shapes in our mind, each of which incorporates functional information, material information and many other types of information. By recognizing a shape as an item in this list, we are able to classify the shape, but can also attribute to it all the information that is connected to the recognized item in our list. In order to be able to do so, the recognition entails not only the classification of a shape, but also the determination of the value of the parameters defined for the feature. For example, if an amphora has been found, an archeologist will not so much be interested in identifying it as an amphora, but to a larger extent in the exact measurements of its parameters.

As with many feats of the human mind, researchers have tried to automate the recognition of form features. The archeologist, once the joy of the first find has subsided and if he or she is lucky, will possibly discover a large number of other archeological objects at the same site. This means that, in addition to the time it takes to carefully excavate all the objects, it may take years to classify and measure them. If a computer tool would be available that scans each object and then classifies and measures it by using a technique that recognizes the form features on each object, the work could be done considerably faster.

Unfortunately, it is unknown how humans recognize form features or even why humans perceive a shape part as a form feature. Recognizing a feature may seem a simple task for a human, but without knowing how a human does this it is extremely difficult to develop a computer tool such as the one mentioned above. Measuring parameter values such as the height or radius of an amphora is easy for a human, although it may be a tedious process. For a computer, however, this is more difficult: how does the computer know what to measure? Although solutions to this problem exist for very specific cases, as of yet there are few approaches available to the recognition of form features in general.

1.1 Motivation of the research

This thesis presents my promotion research and the results of my work on automatic *freeform feature recognition*, done in the period 2004-2008 at Delft University of Technology. The research was done at the faculty of Industrial Design Engineering and although it targets a computer science problem, the research is motivated by problems in the field of industrial design.
The design process is vital for converting a concept to an actual product. It connects the idea generation phase to that of engineering and manufacturing and lays down, amongst others, the shape properties of a product. The high competitiveness in the consumer market has led to a larger importance of the outer appearance and styling of a product and also reduced the time-to-market. One of the methods to improve the design process in these two aspects is by reusing existing shapes; *shape reuse* allows the designer to not only be inspired by existing digital or physical shapes, but also to quickly introduce these shapes into his/her own design (Funkhouser et al., 2004; Duffy and Ferns, 1999; Vergeest et al, 2001, 2006). In the process of shape reuse, form feature recognition can play an important role, as it can be used to recognize and interpret parts of the shapes that a designer wants to reuse. However, the currently available Computer-Aided Design (CAD) tools do not support shape reuse; neither do they support form feature recognition. In particular, support is lacking in the following scenarios:

Scenario 1: An existing shape can serve as an inspiration when designing a new product. For example, when designing a lamp, one could be inspired by a flower (see Figure 1.1). The flower has certain form features (e.g. the individual petals) and can even be considered to be a form feature in itself (involving characteristics such as height and width). To be able to use these form features in a design, a designer can either reconstruct them, which is time-consuming, or use a form feature recognition method to automatically retrieve them from a digital model of the flower.

Figure 1.1: *A flower and a lamp, the design of which has been inspired by the flower*

Scenario 2: One may want to re-design an existing product, because it has to be adapted to changing consumer wishes or manufacturing requirements. If the original CAD-model for the product is still available, then this can be done with conventional CAD-tools. However, in many cases the original model is not available and a new CAD-model has to be constructed that meets the new requirements. If a physical product is available, then its shape can be reused. In this case, form feature recognition is not used to interpret part of the shape data, as in scenario 1; instead, form feature recognition is used to re-engineer as good as possible the parameters that were available in the original CAD-model.

To be able to introduce a physical object into the computer-aided design process, three activities must be performed. First, the shape of the object needs to be digitized; only then can the shape data be manipulated by CAD tools. Data acquisition techniques such as laser scanning can be used to create digital shape data from a physical object. Second, the acquired shape data has to be interpreted, i.e. the data must be segmented in meaningful features and ‘rest’ data. Then, (design) parameters must be assigned to convert the data to

a CAD-model. Two types of parameters exist: one comes into existence when the geometric data is converted to a parametric surface; these parameters can be used to globally influence the shape data (i.e. both the features and the rest data). For example, these parameters can influence the smoothness of the shape. The other type of parameters is local and is connected to the concept of form features. These local parameters relate to a specific subsection of the shape data, namely the feature. The need to interpret shape data in terms of local parameters after it has been digitized is the motivation for applying feature recognition.

In this thesis, a distinction is made between *regular form features* and *freeform features*. Regular form features, also called *machining features* or *manufacturing features*, are form features of which the shape corresponds to simple manufacturing operations and that can be described using regular surface types, such as planes or cylinders. These surfaces are often positioned perpendicular or parallel to each other, or the relation between them can be described by simple mathematical descriptors, such as angle or distance. Regular form features are often a Boolean combination of geometric primitives. In Figure 1.2a, an example of a regular form feature is given. Although the methodological support of the use and in particular the recognition of regular form features is incomplete, the theory behind regular form features is well-established and is considered to be a closed-off research topic.
Freeform features consist of one or more freeform surfaces. The complexity of the shape of freeform features is larger than that of regular features and therefore more advanced methods are needed to manipulate them. An example of a freeform shape is given in Figure 1.2b. In the existing literature, there exists little theory on freeform features. Consequently, freeform features are poorly supported by current CAD-systems. Shape reuse as it was demonstrated in the mentioned scenarios is therefore only possible through an inefficient and time-intensive process.
For reasons of readability in the remainder of this thesis, unless specifically mentioned otherwise, when ‘feature’ is used ‘freeform feature’ is meant.

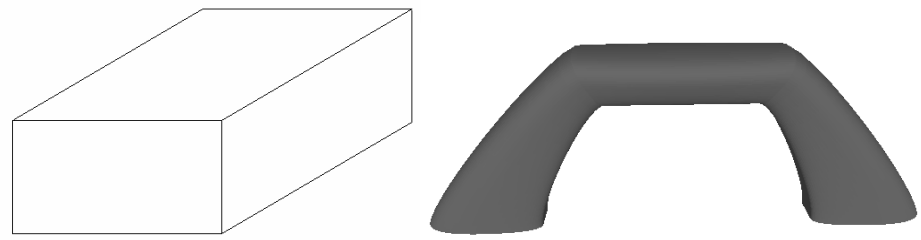


Figure 1.2: *Examples of (a) a regular feature and (b) a freeform feature.*

1.2 Problem statement

As stated in the previous section, there is little theory on the topic of freeform features. In the existing literature, authors use implicit definitions of the freeform feature concept that are (a) fragmented, (b) incomplete and (c) vague. The definitions are fragmented because many of these definitions target a specific application and are based on assumptions that disallow using the definition for other applications. The definitions are incomplete in the sense that they support a specific method and leave the elements that are not relevant in this specific method undefined. Finally, they are vague because they are rarely explicitly given. As a result, these definitions as well as the methods that are developed on the basis of these definitions cannot be validated. The goal of this thesis is to present a freeform feature recognition method that is based on a definition of the freeform feature concept that does not suffer these shortcomings. A general problem statement for this thesis can be formulated as:

How can a freeform feature recognition method be developed on the basis of a complete and sound definition of the freeform feature concept that can be used to locally parameterize freeform shapes?

When the general problem statement is studied in more detail, then the following substatements can be given:

1 Exploration of the freeform feature concept
In order to be able to give a complete and sound definition of the freeform feature concept, it must be known what shapes can be considered to fall under this concept. No objective set of criteria exists with which a freeform shape can be categorized as such. An epistemological description of the freeform feature concept is needed to determine the requirements for both theoretical and methodological support of freeform features. This description focuses on the application of freeform features in CAD.

2 Formal definition of the freeform feature concept
To support operations on all features that fall within the freeform feature concept, a formal, application-independent definition of the freeform feature concept is needed that comprehensively describes the constitution and properties of a freeform feature. Such a definition should be applicable to all sorts of feature-based processes, such as the recognition of existing features, the generation of new features, the manipulation of features or the interrogation of the feature data.

Existing definitions of the freeform feature concept target only a selection of freeform features or a specific application of freeform features, are therefore often contradicting and cannot be combined to a single definition of the freeform feature concept. Existing definitions of the regular form feature concept can not be extended to the freeform domain because they merely address the geometric aspects of the feature, and not its morphological aspects.

3 Formal definition of the freeform feature recognition process

To be able to judge to what situations a freeform feature recognition method can be applied with what result, a description is needed of the information flow in feature-based processes, in particular for the process of feature recognition. Also, the variables that play a role in these processes must be identified and requirements regarding the in- and output of these processes must be defined. Existing freeform feature recognition methods each make use of different assumptions on their input, target different shape representation types, and produce inconsistent results.

4 Development of a freeform feature recognition methodology

A methodology is needed to support the recognition of freeform features. The recognition of a freeform feature, as mentioned above, does not only include the identification of a feature type, but also the determination of its parameter values. Existing methodology on regular form feature recognition cannot naturally be extended to the freeform domain; the few methods that exist in the freeform domain are too situation-specific and not supported by a comprehensive theory.

5 Implementation of freeform feature recognition algorithms and tools

To improve the process of shape reuse, efficient and accurate tools must be developed based on the defined theory and methodology for freeform feature recognition. These tools are not available in current CAD-modeling systems and existing research does not support the development of these tools. The tools must be able to recognize freeform features, but also to reconstruct the parts of a shape that can be perceived as a parametric entity.

In this thesis, all these problem statements will be addressed, resulting in a freeform feature recognition method that supports the local parameterization of freeform shapes.

1.3 Research hypotheses

Existing definitions of the freeform feature concept have been mainly given in the context of *feature-based design*. As a result, existing definitions of the freeform feature concept are based on a geometry-centered view. In a geometry-centered view of features, operations on a feature relate to the shape of the feature only, without addressing all the (parameter) information that is stored with the feature. Theory that is based on a geometry-centered view of freeform features can only support freeform feature-based methodology to a limited extent. The alternative is a theory that is based on a morphology-centered view. The morphology of a feature describes the way in which its shape changes as a result of changing parameter values. In other words, on the basis of the morphology of a feature, its shape can be manipulated by giving parameter values and computing the shape of the feature corresponding to these given parameter values; as a result, the shape of the feature changes but its morphology remains intact. A morphology-centered view on features supports freeform features on a higher level, because not only the shape of a feature, but also the effect of a parameter on the shape can be formalized. We therefore hypothesize:

1. *Freeform feature theory that is based on a morphology-centered view on the freeform feature concept better supports the methodology of freeform feature-based operations.*

In chapter 3, a more detailed discussion of the morphology-centered view on features is given.
Once a view on the freeform feature concept has been established, the challenge arises to develop new theory in which the freeform feature is formally defined. Such a definition fundamentally differs from a definition of the regular form feature concept. Existing, partial, definitions of the freeform feature concept are mainly extensions of the regular form feature concept, but we claim that a paradigm shift is needed in defining the freeform feature concept. We hypothesize that:

2. *Definitions of the regular form feature concept cannot be extended to the domain of freeform feature.*

In addition, existing definitions of the freeform feature concept are fragmented and cannot be merged into one definition. The reason for this is that the existing definitions of the freeform feature concept target a specific application. Therefore, the definitions are based on assumptions that hold only for this specific application and cannot be generalized to other applications. In addition, often only specific feature types are used in

a certain application. If this is the case, then the definitions cannot even be generalized to other feature types. We pose the following hypotheses:

3. *Theories that support different applications of the freeform feature concept can be combined into a comprehensive theory.*

4. *To be able to validate and generalize freeform feature-based methods, they must be based on a comprehensive theory on the freeform feature concept.*

When the freeform feature concept is applied in the practice of industrial design, users will want to be able to define custom features. This means that a freeform feature-based operation should be able to deal with any feature type that a user can come up with. We therefore hypothesize that:

5. *A robust methodology for freeform feature-based operations is able to deal with user-defined features.*

In any practical situation, features are instantiated onto an embedding surface. When a feature is embedded in a surface, then an operation on the feature does not only have an effect on the feature, but also on its relation to the embedding surface. However, we want to be able to distinguish between the two effects and therefore we hypothesize that:

6. *The effect of an operation on a feature can be regarded separately from the effect on the relation between the feature and its embedding surface.*

The goal of this thesis is to apply the freeform feature concept to the problem of feature recognition. Here too, a paradigm shift is needed, because the data structures, heuristics and algorithmic approaches that could be used for recognizing regular form features cannot be extended to the freeform domain. We formulate the following hypotheses:

7. *Because of the diversity of freeform features there is no feasible heuristic approach to feature recognition and at least part of the recognition process therefore depends on brute-force or probabilistic methods.*

All the hypotheses mentioned cast ahead testable expectations of the development of new theory and methodology. They become useful when describing an approach to the generation of new knowledge of the problem of freeform feature recognition and are helpful when formulating a methodological approach to the problems mentioned in the previous section. In section 6.3 we will revisit the research hypotheses and conclude on whether they hold in the theory and methodology that is proposed in this thesis.

1.4 Context and aim of the research

The promotion research that is presented in this thesis was done as part of an STW-funded project that dealt with the development of new methods for the support of freeform features. The aim of the project was to develop a general toolbox that can be used to support the use of freeform features for improved product modeling. One part of this project concerned research that is dedicated to supporting the concept of shape reuse; a large part of it lies at the basis of this thesis. The complementary part of this project was dedicated to the development of new freeform feature-based design methods, and specifically to the role of constraints. Colleagues in this part of the project have concerned themselves with the development of freeform feature library management tools, freeform feature definition methods and freeform feature-based design.

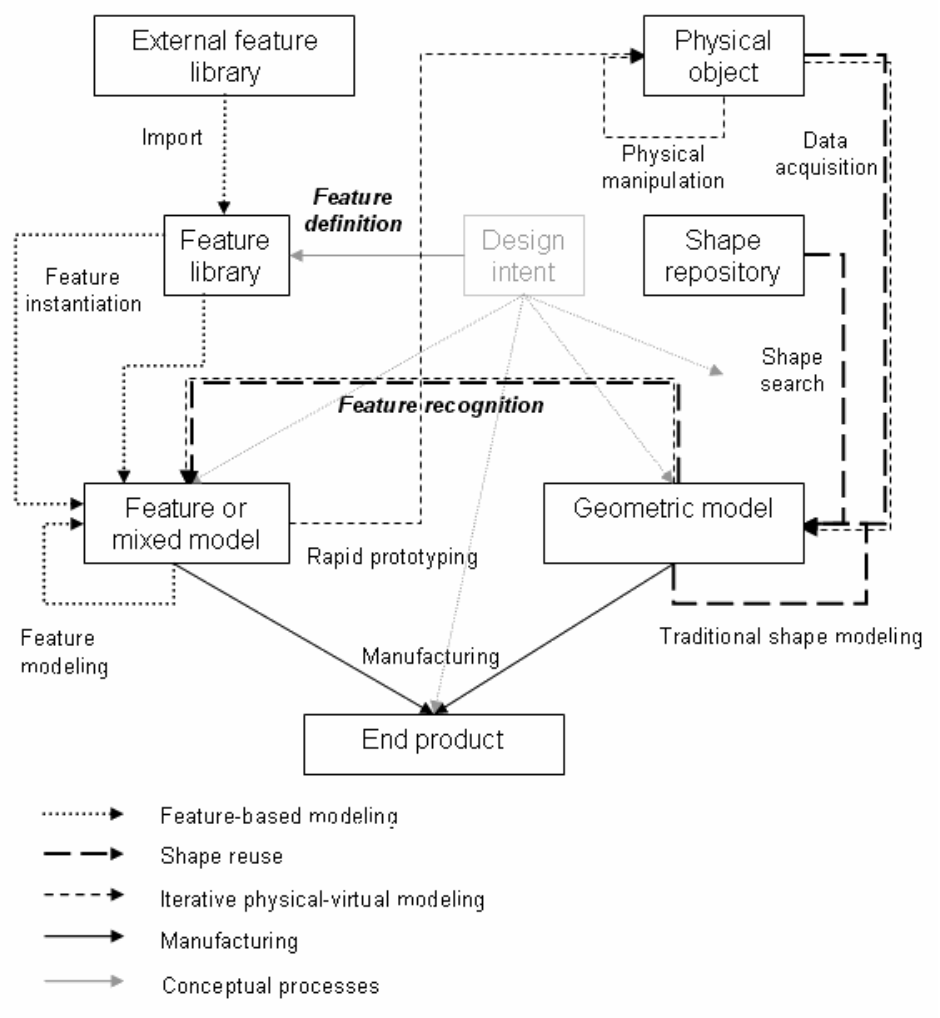


Figure 1.3: *A model of the design process in a feature-based environment with the work presented in this thesis depicted in bold arrows and font.*

The promotion research is also embedded in the Dynamic Shape Advancement (DYNASH) program, in which several efforts to improve the process of CAD for freeform shapes are combined. Within this research program, colleagues have been involved in the process of iterative physical-virtual modeling, in which a manual design process is iteratively coupled to a computer-aided design process, and also with rapid prototyping and data acquisition. Figure 1.3 shows the position of the research presented in this thesis (in bold font) with respect to other research done both in the DYNASH research program and in the STW-funded project.

1.5 Research methodology

To ensure that the work presented in this thesis contributes to research in general and to research on features in particular, the research must be methodologically correct. To support the claim that this is the case, in this section we present a view on the structure of the research.

The research presented in this thesis is both foundational, in the sense that it contributes to the development of new theoretical insight in the feature concept and operational in the sense that it leads to the implementation of new freeform feature recognition techniques. In other words, the generation of new knowledge runs parallel to the generation of new techniques based on this knowledge. A diagrammatic representation of this concept is shown in Figure 1.4.

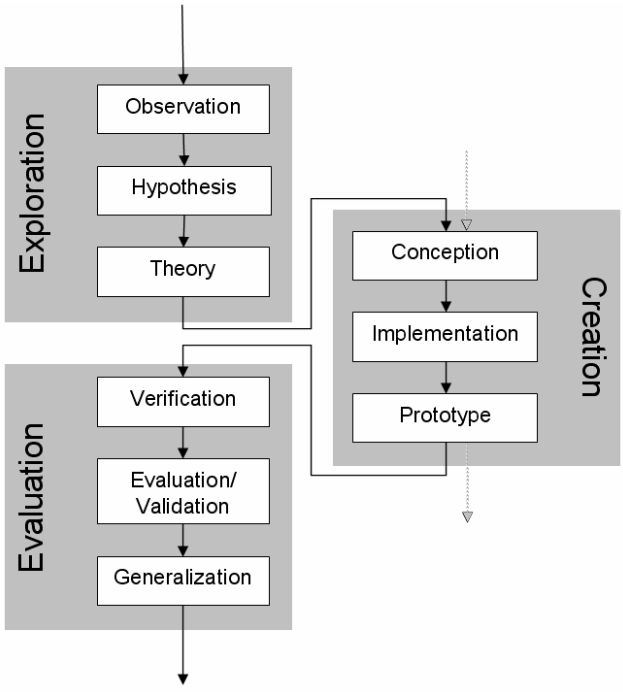


Figure 1.4: *Diagram of the parallel processes of foundational and applied research*

In the exploration phase of the research, a problem statement is formulated that is based on an analysis of the shortcomings of state-of-the-art theory and methods. A literature study is done to identify the relevant issues in feature recognition and feature research in general. Research hypotheses are induced from the existing knowledge and embody assumptions on how the signaled shortcomings can be overcome. These hypotheses were given in section 1.3 and were induced both from the formulated problem statements and from the observation phase of the research. Finally, comprehensive new theory is proposed that forms the basis of the methods and algorithms developed in the creation phase of the research. This theory includes a formal description of the feature concept, as well as a computational model of the freeform feature recognition procedure. Based on the theory, an implementation study was done to find out which parts of the freeform feature recognition process can be automated and what user input can be assumed. The implementations were combined in a single pilot implementation.

In the evaluative phase of the research, it was analyzed based on empirical evidence whether the developed algorithms successfully address the formulated problem statements. An application study was done in which the different algorithms used in the pilot implementation were separately tested and evaluated. The external validity of the research was tested with the help of a user evaluation study, in which it was tested if the input that is assumed for the implemented methods can indeed be given by the user.

Based on a verification of the theory and an analysis of the internal and external validity of the developed methods, a discussion and conclusion are given on whether the developed theory and methodology can be used as a solution to the given problem description.

The research that is presented in this thesis contributes both to the generation of new knowledge and the development of new methodology. First, a comprehensive theory will be given in which the feature concept is defined, and in which all the aspects of the feature concept will be addressed. This theory adds to the existing knowledge in several aspects. First, it connects pieces of existing knowledge that have been generated for specific classes of features or for specific applications of the feature concept. Second, it fills in the gaps in the existing knowledge by addressing all issues that were identified but not supported in existing work. Finally, existing knowledge is analyzed and, in some aspects, challenged or confirmed.

The new methodology that will be presented in this thesis adds to existing knowledge in two aspects. First, due to the fact that it is based on a comprehensive theory, it is shown how methods for specific feature-based applications can also be extended to and connected to methods that deal with related feature-based applications. Second, because the developed methods are based on a formalization of the feature concept, the advantages and shortcomings of the feature concept can be described in more detail than is available in the existing literature.

1.6 Structure of the thesis

The structure of this thesis follows the research methodology as outlined in the previous section. In the second chapter, existing literature is reviewed. An historical overview will be given to place research on freeform feature recognition in the context of past work. A brief overview of existing work on regular form features is given, followed by a critical analysis of state-of-the-art definitions of the freeform feature concept.
Chapter 3 introduces a new theory for freeform features. The theory is based on assumptions that are partly derived from the observations made in chapter 2. The kernel of the theory is a definition of the freeform feature concept and of the general properties of a freeform feature. In addition, an implementation of the theory is given.
In chapter 4, algorithms are given for a freeform feature recognition procedure and the supporting operations, such as the definition of feature types and for the instantiation of these features on target surfaces.
Chapter 5 deals with the verification, validation and generalization of the new theory and methods presented in chapters 3 and 4. The performance of the implementation of the proposed freeform feature recognition algorithms is tested and the advantages and disadvantages are discussed.
In chapter 6, a conclusion is given, the research hypotheses are revisited and directions of future research are discussed.

1.7 Own papers

In the course of the research presented in this thesis, several papers were published in journals and conference proceedings by the author. A large part of this thesis is based on those papers, but each of the papers contains details that are not included in this thesis. Readers who are interested in a specific subtopic of this thesis are referred to the corresponding paper. In this section, an overview of these papers is given, in which individual references to the published papers can be found.

In a first attempt to tackle the problem of freeform recognition, a method was developed to identify styling lines in a freeform shape model (Langerak and Vergeest, 2005). The idea behind this first attempt was that to investigate what aspects play a role in freeform feature recognition, one might as well start with trying to recognize a specific type of feature. Styling lines are elongated regions of discontinuity in a shape that may or may not have been an intentional shape characteristic (meaning that the method is not intended to, but can also be applied to unintentional or even unwanted styling lines). Styling lines are an easily distinguishable feature type, and as such a good starting point for our

exploration of freeform feature recognition. We hoped that additional assumptions could be made that caused styling lines to be an easy starting example of freeform features.
The styling line detection method dynamically sweeps an intersection plane over the target surface to extract the cross-sectional profile of the styling line, as well as its trajectory. Discontinuities in the surface, if available, can be used to further recognize the styling line. In the course of the research it was realized that the styling line can be formalized as a feature with a multi-dimensional parameter and as such the papers on this subject have formed the basis of the curve-based feature recognition method given in section 4.3 of this thesis.
In the development of the styling line detection method, it became clear that a thorough understanding of freeform features was not available in existing literature. As a response, the idea of a foundational theory for features and their definition and recognition emerged. In this theory, it is specified what elements a feature definition should have; in addition, the different aspects of a feature definition and its application to feature recognition are identified (Langerak and Vergeest, 2006, 2007). In addition, it was shown how a feature can be defined and how a defined feature can be instantiated on a shape model (Langerak and Vergeest, 2007). Also, the concept of feature space was introduced, which can be used to further formalize the application of user-defined features in feature-based modeling and feature recognition (Langerak and Vergeest, 2007). All these papers form the basis of chapter 3 of this thesis.
Finally, the characteristics of the feature definition were found to resemble that of objects in an evolutionary computation process, and therefore a new freeform feature recognition method was developed that uses the concept of evolutionary computation to recognize features (Langerak and Vergeest, 2007). This method was improved by introducing a two-step approach that first finds an approximation of the feature and then fine-tunes this definition to the local conditions of the target surface (Langerak and Vergeest, 2008). Finally, it was described how, once a feature has been recognized, it can be deleted from a target surface (Langerak and Vergeest, 2008). These papers form the basis of chapter 4 of this thesis.

2 A review of literature on features

The literature on form features and form feature-based technologies is extensive and covers many different aspects and applications of form features. In this section we embed the problem statements and the research hypotheses in the existing literature. This is done in three regards; foundational aspects of the feature concept are derived from an overview of existing work on regular form features, in- and output constraints of feature recognition are identified by discussing related research topics, and requirements for the definition of the freeform feature concept are conjectured from a critical analysis of recent work on freeform features (see Figure 2.1)

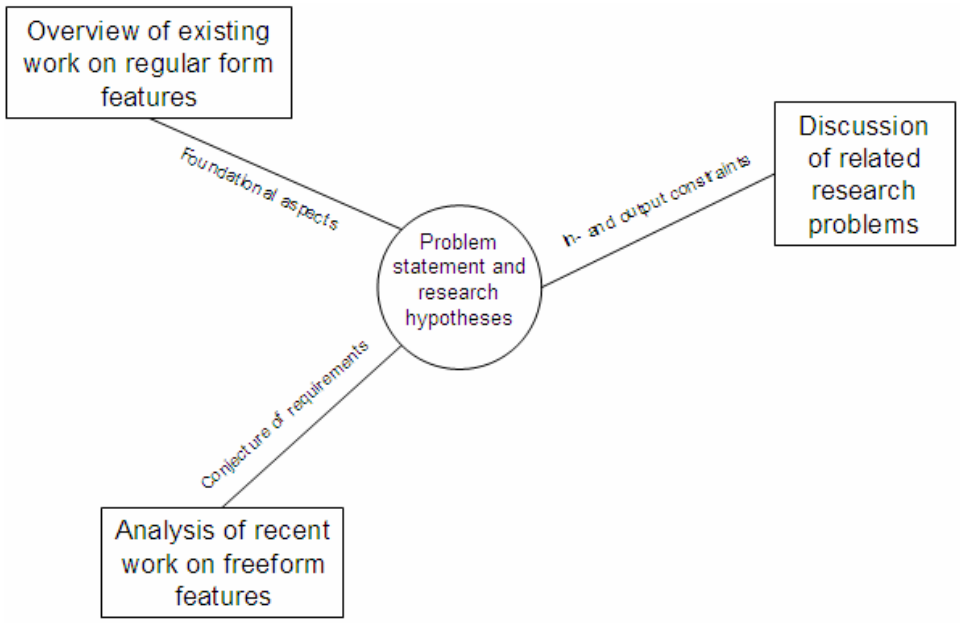


Figure 2.1: *Embedding of the problem statement*

Overview of existing work on regular form features

The interest for the form feature concept originated in the early 1980's, and since then the concept has developed from higher level geometric descriptions to parameterized modeling entities that carry semantics and even design intent. Most of the methods on form features are in the domain of regular form features and are not directly relevant for the promotion research on freeform features presented in this thesis. However, some philosophical or technical debates in the area of regular form features certainly apply to or are relevant in the domain of freeform features as well. For this reason, they must be considered as inspirational for the problems that are addressed in this thesis. In section

2.1, a brief, chronological overview of the form feature concept is given. In this section, we answer the following questions: (i) What are the applications of the feature concept? (ii) What aspects and considerations in the theory and methodology on regular form features are also applicable in the domain of freeform features?, (iii) In a historical perspective, what is the current momentum and direction of form feature research?, (iv) What is the current status of form feature theory and methodology, both in science and in industry?, and (v) What are the open research problems in form feature research and how do they relate to the closed-off research problems in the domain of regular form features?.

Discussion of related research topics
In section 2.2, we give an overview of how research on form feature recognition relates to other research topics. These related research topics are relevant in two regards. First, the solutions to problems in related research topics may serve as an inspiration in the development of feature recognition methods; it may even be the case that they can be directly applied to feature recognition. Second, they are relevant for determining the possible input and the desired output of form feature recognition; the result of a method for a related research topic may server as an input to a feature recognition method or vice versa. In section 2.2, we therefore discuss the related research topics with regard to the following aspects: (i) Do these related research topics, both time-wise and logically, take place earlier than, parallel to or later than form feature recognition problems?, (ii) Do the related research problems imply any requirements on the in- or output of form feature recognition?, and (iii) Are there open issues in the related research problems and if so, how do these relate to the open research problems in form feature recognition?

Analysis of recent work on freeform features
In section 2.3 a critical analysis is given of methods for the definition of freeform features. The methods that are reviewed are analyzed on the basis of the following criteria: (i) The type of shape representation, (ii) Relation of the feature to the embedding surface, (iii) Nature of the feature shape, and (iv) Ability to deal with user-defined features.

2.1 Overview of existing work on regular form feature research

In this section we review the existing work on regular form features. By reviewing the existing work in a chronological order, we give an impression of the momentum and direction of current research on form features, as well as of the current status of feature research. Specifically, we address the open research problems by discussing how they relate to each other as well as to solved research problems

2.1.1 Introduction of the form feature concept

In the past decades, the form feature concept has been gradually developed. When it was introduced, the form feature concept was a response to the increasing complexity of geometric models. Originally, form features had a purely geometric nature: they were macros of low level geometric elements and did not contain parameter information. By using these macros, higher level control over geometric shapes could be provided, because collections of geometric entities could be interrogated or manipulated without having to address each individual geometric element. In addition, macros allowed for information, such as a name or manufacturing information, to be stored with groups of geometric elements.

2.1.2 Applying the form feature concept to feature recognition

Originally, the main application of a geometry-centered view on the form feature concept was in linking CAD to Computer-Aided Manufacturing (CAM) (Grayer, 1975; Kyprianou, 1980). In a process called Computer-Aided Process Planning (CAPP), manufacturing routines are constructed with which a geometric part can be manufactured. In many designs, parts of a geometric part can be standardized with regard to the necessary manufacturing routines. Certain standardized shape parts can be recognized to be instances of one or more predefined form features, which include information on the manufacturing routines that can be used to manufacture these shape parts (Wu and Liu, 1996, Han et al., 2000). Because manufacturing routines are stored with a form feature and can therefore automatically be associated with groups of geometric elements, the efficiency of a CAPP procedure can be improved (Subrahmanyam and Wozny, 1995).
The process of interpreting geometric data in terms of features was deemed feature recognition and can be defined as 'the processing of a geometric model from a CAD system to find portions of the model matching the characteristics of interest for a given application.' (Shah et al., 2001). A good deal of background information on feature technology in general and an overview of feature recognition in particular can be found in the work by Shah and Mäntylä (2005). The three main techniques for feature recognition are graph-based feature recognition, convex hull decomposition and hint-based feature recognition.

Graph-based feature recognition
Graph-based feature recognition is a technique that makes use of the fact that a boundary representation of a three-dimensional shape can alternatively be expressed as a relational graph of geometric entities such as vertices, edges and faces. This allows for the use of graph-based algorithms in analyzing or manipulating the shape model.

Several methods of graph-based feature recognition have been proposed (De Floriani, 1987, 1989; Joshi and Chang, 1988; Chuang and Henderson, 1990. None of these authors go into detail about how a boundary representation can be expressed as a graph, but they do show how, once a graph has been obtained, the model can be analyzed using a pattern recognition approach. De Floriani proposes a feature extraction algorithm that is based on a connectivity analysis of the graph. Joshi and Chang and Chuang and Henderson propose a subgraph matching approach in which not only the model under consideration but also possible features are expressed in the form of a graph. Although subgraph matching is a known NP-complete problem, the complexity of this algorithm is acceptable when the graph under consideration consists of only a small number of nodes.
A large disadvantage of graph-based feature recognition methods is that they are unable to handle feature interference. Sakurai and Gossard (1990) propose an iterative graph-matching approach in which recognized features are temporarily removed from the shape model in order to reveal intersecting features. However, this assumes that features can be recognized despite the fact that they interfere with other features and this can only be claimed for very specific cases.
A solution to the problem of feature interference is the concept of virtual links (Marafat and Kashyap, 1990; Trika and Kashyap, 1993). The idea behind virtual links is that feature interference cannot be detected by subgraph matching, because the interference distorts the structure of the model graph so that it no longer matches a predefined feature graph. By inserting virtual links into the graph, the original structure of individual features can be restored and features can be recognized using the above mentioned graph.
There are significant problems with graph-based feature recognition methods in the domain of regular form features (Venkataraman et al., 2001): these methods only extract the type of a feature, and not its parameter values; in addition, it is difficult to apply them to user-defined features.

Convex hull decomposition

The convex hull decomposition approach, applicable specifically to the Constructive Solid Geometry (CSG) representation, is based on the principle that features are often noticeable as regions of alternating convexity and concavity. In the Alternating Sum of Volumes (ASV) method, a decomposition tree is built by iteratively computing the difference between a volume and its convex hull (Woo, 1982; Kim, 1992). This procedure terminates when the difference volume can be recognized as a feature.
A significant problem of the ASV method is the fact that for some models the ASV method may not converge, i.e. the decomposition of some shapes leads to an infinite decomposition tree (Tang and Woo, 1991); a remedy for this problem was given by Kim (1990). He proposed and implemented the Alternating Sum of Volumes with Partitioning (ASVP) method, which partitions a shape model and performs an ASV routine on the

partitioned parts for which the convergence can be guaranteed. Variants of the ASVP algorithm deal with alternative features such as cylindrical and blend features (Menon and Kim, 1994).

Hint-based feature recognition
Hint-based feature recognition methods are methods that consist of two steps: in the first step, the hint-generation step, an educated guess is made on where features are on the basis of hints that are collected in an analysis of a shape. Examples of possible hints are parallel or perpendicular edges, connection patterns or convexity patterns (Vandenbrande and Requicha, 1993, 1994; Regli et al., 1995; Han and Requicha, 1995; Gao and Shah, 1998). In the second step, the completion step, it is determined whether the hints truly indicate the occurrence of one or possibly more features or are false hints. In addition, the hints are combined and completed in order to be able to extract an entire feature.
Hint-based feature recognition allows for the recognition of interfering features, providing that the completion step is well-implemented. A disadvantage, however, is that the complexity for hint-based feature recognition is large, and that the hint-based approach assumes some preconception of what type of hints are available in a target shape, making the method not generally applicable.

Feature mapping
A problem that is closely related to feature recognition is that of feature mapping, in which features that have been defined in one domain are mapped to a feature definition in another domain (Shah, 1983; Shah et al. 1993, Falcidieno and Giannini, 1991). This is relevant when multiple feature-based applications are used, e.g. when two users are working on the same model using a different feature-based application. Feature mapping has not become a popular topic and is therefore only briefly mentioned here. However, the principles that play a role in the concept of feature mapping have later been used in research on feature exchange.

2.1.3 Introduction of feature-based design

In 1988, the concept of feature-based design was introduced by Samuel Geisberg, who implemented his ideas in the commercial software application Pro/Engineer. By including parameters in the feature concept, it allowed for easy parameter-based manipulation of the shape of form features, thereby solving the problem that to enact a shape manipulation, one has to redesign the shape from scratch. By supporting parametric modeling, features can be included in a CAD-model during the design process, instead of at the end of the design process, in a feature recognition procedure.

The advantage of using features can be summarized as follows (Rossignac, 1990):

- Through the use of features, complex geometry can be quickly included in a design.
- Shape manipulation operations can be associated with the shape of the feature. Through these operations, the shape can be efficiently manipulated.
- Features provide a compact description of a part of a shape's geometry. Through this description, high-level access to the shape can be provided, for example for checking the validity of the model.
- Features can be used to attach manufacturing or other information to a part of the shape's geometry.

The introduction of Pro/Engineer initiated a large change in the CAD industry, and the scientific community soon followed by targeting the many new research problems that parametric modeling introduced (Cunningham and Dixon, 1988). Also in the late 1980's, Pro/Engineer's main competitors UGS and CATIA introduced parameter-based CAD software. In addition to these systems, scientific progress led to the development of several feature-based design systems, most of which have never become commercially viable but where mostly intended to demonstrate specific aspects of feature research.
As a result of its role in the design process, the geometry-centered form feature no longer sufficed. By introducing parameters into the form feature concept, it has developed from a geometric entity to an entity with morphologic aspects. In a morphological view of the feature concept, features are regarded on a more abstract level, on which there is a relation between the parametric structure of a form feature and the geometry of the feature (Horváth, 1996). This relation can be said to embody the semantics of the feature; these become apparent as a symbolic description of the parametric structure of the feature. Although many aspects of features have been extensively researched, there are argumenst to support the claim that research after form features has never become a closed-off research topic (Regli and Pratt, 1996). Many feature research problems, such as, amongst others, feature interference, feature composition, feature mapping, feature classification, and feature database management, are still open (Salomons et al., 1993). Very little work has been done on the development of a general methodology for feature-based design. Some researchers have identified the relevant issues in feature-based design (Pratt, 1988; Rossignac, 1990; Bronsvoort and Jansen, 1993; Salomons et al. 1993; Regli and Pratt, 1996) , but very few of these issues have actually been followed up on. Instead, many researchers have developed their own feature modelers (Shah and Rogers, 1988; Mandorli et al., 1997; Bidarra and Bronsvoort).

Hybrid systems

Feature-based design and feature recognition appear to be complementary applications of feature technology. In feature-based design, models are created in which semantic, functional or other information is built up during the modeling process. In feature recognition, this information is added to the model afterwards. Combining the two approaches seems to be inefficient, but there are several motivations for integrating feature-based design and feature recognition:

- During the modeling process, features may originate that were not intended as such by the user. Because these features were not created intentionally, they incorporate no semantic or functional information and no manipulation handles are available to the user.
- Several aspects of features, such as feature interaction, may unnecessarily complicate the model. Once recognized, these unnecessary complications can be solved by checking the model for possible simplifications. If this is done during the modeling process, the model can be kept as efficient as possible.
- The purpose of a feature-based design is often to eventually manufacture the design. However, planning a manufacturing process requires more effort for a complete model, as issues such as interference must be addressed. By maintaining and augmenting the manufacturing information during the modeling process, the CAPP phase of the design can be made more efficient.

Several systems have been developed that combine the functionality of feature recognition and feature-based design (Laakko and Mäntyläh, 1993; Ko and Park, 1994; De Martino et al., 1994; Han and Requicha, 1997;)

2.1.4 A shift to the freeform domain

In the early 1990's the interest in both feature-based modeling systems and feature recognition methods slowly declined. With the arrival of more advanced manufacturing techniques, the role of feature recognition in CAPP became exhausted and the motivation for researchers to further investigate features decreased. Apart from several overview papers, few papers were published on features during this period.
The interest in features shifted from regular form features to freeform features in the late 1990's and the early 2000's. In a highly competitive industry, shape styling became an important tool to distinguish products from competing products. To support the increasing complexity of the modeling process, new freeform modeling techniques were proposed in rapid succession and among these were methods that used the feature concept. In the late 1990's, new versions of the known commercial systems were available and

several additional commercial software packages, such as SolidWorks, SolidEdge and Autodesk Inventor, where brought to the market. These software packages are competitors of the systems mentioned earlier, with only minor differences in what market segment they are targeting. Although they provide the ability to use freeform features in designs, the support of these features is incomplete and unreliable. The main reason for this is the fact that instead of providing a foundationally new methodology, all these systems apply the regular form feature paradigm to freeform features. Efforts in research on dealing with freeform features have also focused on extending the theory and methodology of regular form feature-based design to the freeform domain (Cavendish and Marin, 1992; Cavendish, 1995).

However, the similarity of the freeform feature concept and the form feature concept as it was proposed in the 1980's is far from obvious (Vosniakos, 1999). In the freeform domain, feature-based methods are much more complicated and benefit little from the insight gained in research on regular features. In addition, a new shape representation, the B-spline surface, must be taken into account when developing feature-based methods. As a result, the shift to the freeform domain caused many of the classical problems of feature-based design to re-emerge; in turn this caused the freeform concept to fall back into a geometry-centered interpretation (Mitchell et al., 2000; Au and Yuen, 2000).

Recently, methods have been proposed to handle features in feature-based design on a morphological level (Van Elsas and Vergeest, 1998; Guillet and Leon, 1998; Pernot, 2004; Pernot et al., 2005), but these methods are not generally applicable and disregard many of the classical problems of feature-based design.

Only recently, the problem of feature recognition has shifted to the freeform domain as well (Vergeest et al., 2001, 2003). Instead of playing a role at the end of the design process, as was the case for regular form feature recognition, freeform feature recognition plays a role during the design process (Vergeest et al., 2004); its main task is not to provide a link between CAD and CAM, but to reconstruct parametric information for non-parameterized geometry that can in turn be used to more efficiently modify a design. For this reason, freeform feature recognition is strongly coupled to the intent of shape manipulation (Song et al., 2005).

So far, the work done on freeform feature recognition has been mostly exploratory; although some theory and methods for freeform feature recognition have been developed, their applicability is limited. As is the case for freeform feature-based design, most freeform feature recognition method are not generally applicable.

2.1.5 Conclusions

Research on form features can be divided in two clearly distinct phases: one in which there was a focus on regular form features and one in which freeform features are the main topic. In both phases one can distinguish an evolution from a geometry-centered view on features to a morphology/semantics-oriented view.
Although most of the topics in the domain of regular form features have been extensively researched, there are still open issues. More importantly, no general consensus has emerged as to what features are and how they are defined (Regli and Pratt, 1996).
The current challenge in research on freeform features is twofold. First, a general theory on freeform features must be defined on the basis of which methodology can be developed. Such a theory is necessary to prevent having to conclude in ten years time that no general consensus has been achieved on the freeform feature. This does not imply that a single theory must be developed that encompasses all feature types and all applications of features, but at least a general theory must support the compatibility of partial theories on and specific applications of freeform features. Second, to develop a more robust methodology that is based on a morphology/semantics-oriented interpretation of freeform features instead of on a geometry-centered interpretation. The work presented in this thesis must be placed in the context of these challenges.

We argue that the theory and methodology on regular form features can not be extended to the freeform domain. This argument is a viable explanation for the slow scientific progress in the domain of freeform features and the poor support of freeform features in commercial systems: freeform feature-based methods that have so far been developed are based on existing work on regular form features and are unable to deal with the complexity of freeform features. This supports hypothesis 2.

2.2 Related research problems

From its earliest applications, the form feature recognition problem has been interwoven with many parallel research topics. Some of these research topics have preceded research on form feature recognition, some tackle problems that are also relevant for form feature recognition, while yet others have benefited from the results of research on form feature recognition. To gain a clear understanding on the connectivity of the different research topics and to embed the research on form feature recognition in the broader domain of computational geometry, we give an overview of the related research problems, in which we focus on the relation of these research problems in aspects of historical timing, relevance for the input- and output of form feature recognition and the relation to the open research problems that was mentioned in the previous section.

The research topics that will be discussed in this section all target computational issues. However, there is also a rich history in research, mainly in the field of psychology, regarding a human-centered view on features (e.g. Biederman, 1987). This research deals with how humans perceive shape and how they recognize shape using the features of a shape. A simple example is that faces can be recognized not by analyzing the face as a whole, but by focusing on specific features of the face (e.g. nose, ears, eyes, mouth). However, the relation between this research and the underlying thesis is weak because the relation between how humans perceive shape and how computers deal with shape is far from obvious. To not confuse the reader with an understanding of features that is too broad, it is being left out here.

2.2.1 Shape recognition and shape data acquisition

An obvious link exists between feature recognition and the more general topic of shape recognition. Shape recognition can generally be described as the process of interpretation of images, in particular of three-dimensional images. This interpretation relies on (structured) knowledge such as a collection of template shapes (Veltkamp, 2002) or an analysis of shape properties (Osada, 2001; Masuda, 2002). For this reason, shape recognition is often taken to be equivalent to shape matching. In many cases, features are used to help in the shape recognition process (Veltkamp et al., 2001). These features are much more general than those used in this thesis and can be any noticeable aspect of a shape.
The images to be interpreted can either be given, for example in the form of digital medical data, or the problem of shape recognition must be preceded by a process of shape data acquisition, i.e. the reconstruction of three-dimensional images from either two-dimensional data (e.g. photos from different viewpoints) or three-dimensional data (e.g. laser range scans) (Besl and McKay, 1992). Although the acquisition of three-dimensional data is a topic of its own, it cannot be seen separately from shape recognition, as the assumptions made during the data acquisition stage can have an effect on the validity and robustness of the shape recognition process (Higuchi et al., 1995). Reversely, shape recognition can benefit from the retrieval of additional information, for example by recording markers in the data acquisition stage. Well-known applications of shape recognition are in the analysis of medical data (Maintz and Viergever, 1998), robot vision (Desouza and Kak, 2002) and face recognition (Scheenstra et al., 2005).

Shape recognition outdates feature recognition, but both topics have their roots in computer vision and, going even further back, in psychology. The topic of feature recognition is a specification of that of shape recognition and the methodology for shape recognition is therefore partially applicable to feature recognition.

Because suitable data acquisition hardware has only become available in the last decades, the topic of data acquisition techniques has only recently been addressed. There are many open research problems on the topic of data acquisition, most of which have to do with optimization and are not directly relevant for feature recognition. The relation between feature recognition and data acquisition was mentioned in chapter 1, where it was assumed that in practical cases feature recognition is applied to the output of a data acquisition procedure.

2.2.2 Shape similarity and retrieval

With the increased availability of three-dimensional models on the Internet, the search for three-dimensional shapes has become a hot topic. Similar to the text-based search of search engines such as Google, based on input given by the user, a shape search procedure retrieves three-dimensional images from the world-wide web or any other collection of knowledge. Well-known examples are the Princeton Search Engine (Funkhouser et al, 2003) and the Shapelab system developed at Purdue University (Iyer et al., 2004; Jiantao and Ramani, 2005). These systems use two-dimensional sketches of the front, top and side view of a shape and compare these sketches to the shapes in the database. The Princeton Search Engine also offers the possibility to find shapes using a three-dimensional sketch or even a full-fledged three-dimensional model.

In both cases, models are found on the basis of shape similarity, either two- or three-dimensional. A big disadvantage of the similarity measures that have been developed both for two-dimensional and three-dimensional shapes (Veltkamp and Hagendoorn, 2000), is that they are too expensive to compute for large datasets. Therefore alternative similarity measures have been developed such as Spherical Harmonics and Shape Distributions (Kazhdan, 2004). However, these similarity measures are not yet well-established and aspects of these measures are still investigated.

Shape similarity plays an important role in feature recognition techniques that are based on a comparison to an existing collection of template feature shapes. The quality of the result of a feature recognition method can be measured using a shape similarity metric (Vergeest et al, 2003). Although there is no direct relation between shape retrieval and feature recognition, the work on shape retrieval that is referenced in this section demonstrates how shape similarity metrics can be applied.

In reverse, often shape retrieval can benefit from feature recognition, as the target of a shape retrieval procedure may not be a shape that is geometrically the most similar to the input of the user, but that exhibits similar features (Cardone, 2005). For example, two Braille letters may have a dissimilar shape, but they are both composed of a pattern of bumps.

2.2.3 Reverse engineering and shape reconstruction

Other research topics that are closely related to the problem of feature recognition are that of reverse engineering (Varady et al., 1997; Fisher, 2004) and shape reconstruction. Reverse engineering has been defined as the extraction of information about the shape of an object that is needed to replicate the object. Reverse engineering is for example a useful tool when trying to modify a shape of which the original model is unavailable. Reverse engineering is also useful when designing shapes that are inspired by already existing shape (Funkhouser, 2004).

Shape reconstruction (Varady, 2005) is the problem of creating a valid and efficient shape representation from incomplete data. This is necessary when the existing shape data is incomplete, invalid, of an unwanted shape representation type or of a low resolution. All of these problems commonly occur when obtaining digital shape data from physical objects.

The topic of reverse engineering relates to feature recognition in the sense that feature recognition is one of the approaches to reverse engineering (Thompson, 1999). The topic of shape reconstruction has some problems in common with the topic of feature recognition; for example, both rely on the curvature, structure and connectivity of a target shape and deal with a region for which these aspects are unknown.

2.2.4 Feature taxonomies, user-defined features and feature library management

In the early days of feature technology, it was believed that all features could be sorted into categories. Each feature would unambiguously fall into one feature class, which could hierarchically be stored in a *feature taxonomy* (a sort of family tree for features). Feature taxonomies would have several advantages. First, they allow for more efficient storage of feature properties: when features share one or multiple properties, it makes sense to assume that they are both subtypes of another feature. In this case, the shared information can be stored within the definition of this more general feature rather than in each feature type definition that is derived from it. Second, they contribute to a more intuitive user interface. In the case of a hierarchical structure of feature classes, users can more effectively browse the collection of available features.

Most work done on feature taxonomies assumes that only simple features should be stored in a feature taxonomy, paired to a mechanism to derive more complex features. In this case, the complexity of the data structure that is used for storing the feature taxonomy can be kept low, without compromising the robustness of the feature classification concept. Compound features, which are composed of multiple simple features, can be derived from the taxonomy but should not be included, as the amount of possible compound feature types is infinite.

Several feature taxonomies have been proposed in the regular feature domain (Wilson and Pratt, 1988; Gindy, 1989; Ovtcharova et al., 1992). Feature taxonomies for the freeform feature domain were also given (Poldermann and Horvath, 1995; Fontana et al, 1999; Nyirenda et al. 2006), but these roughly have the same hierarchical structure as the proposed taxonomies for regular form features, the main difference being that freeform examples are used.
It can be argued that in the early years of feature research, an unambiguous and complete feature taxonomy was an achievable goal. The regular form feature concept could only be meaningfully implemented for a small number of feature types of which an exhaustive categorization could be made, for example based on the necessary manufacturing routines. When feature theory shifted to the freeform domain, more complex features became available and it was found that an exhaustive categorization of feature types was no longer possible. As a result, feature taxonomies are scarcely used in modern feature-based design systems.

A pragmatic solution to the problems of the feature taxonomy is the concept of the *feature library*. The feature library can be defined as a dynamic set of features that can be interactively managed by one or more users (Luby et al. 1986, Grabowski et al., 1991). Contrary to the feature taxonomy, the feature library does not contain an exhaustive categorization of feature types; rather, it contains a set of features that is relevant in a certain domain of application. The relevance of the features in a feature library is maintained by the user, who manages the feature library by adding, deleting or modifying feature type definitions.
The concept of the feature library gives rise to two new research problems. First, the user should be able to define new feature types that are of use in a specific application (Hoffmann and Joan-Arinyo, 1998; Dong and Wozny, 1991). These User-Defined Features (UDF) must be stored in the feature library in a logical way. Second, when the feature library is modified, e.g. when a new feature definition is added, the feature library must be assumed to remain well-organized.
For the construction of user-defined features, two general methods are available: declarative and procedural. When defining new features by declaration, a user specifies a geometric and constraint definition mechanism and a set of constraint validation rules. When a procedural mechanism is used to define a feature, then its parameters and validation rules are derived from existing 'parent' feature definitions using certain inheritance rules (Shah et al., 1988, 1994). Declarative modeling offers more freedom in defining new feature types, but typically requires more effort.
To combine the advantages of the declarative and procedural feature definition methods, a hybrid method can be used. Such a definition method derives some elements from a parent definition, but allows the user to declare additional feature properties (Pratt, 1988).

The work on feature taxonomies and feature library management largely predates the research on form feature recognition and can be considered to be a closed-off research topic. The mechanisms for feature library management are not different for freeform features and no new work has recently been published. The topic of user-defined features, however, is still very relevant for the concept of form feature recognition as well as other feature-based operations. There are many open problems in the area of user-defined feature definition in the freeform domain, and in chapters 3 and 4 of this thesis, we will address some of these problems.

2.2.5 Conclusions

There are many relations between the problem of freeform feature recognition and related research problems. Some of these problems, such as the problem of shape data acquisition, predate the problem of feature recognition and solutions to these problems have an influence on properties of the input data on which a freeform feature recognition method should operate (i.e. the shape representation type or accuracy of the data). Other problems, such as shape reconstruction or shape recognition, are parallel to the problem of freeform feature recognition; these topics have open research problems that are similar or related to those in freeform feature recognition. A final category of related research problems, such as reverse engineering, benefits from solutions to the problem of freeform feature recognition.
Although the analyses given in this section do not directly support any of the research hypotheses, they are instrumental in finding solutions to the given research problems and embed these solutions in the related research.

2.3 Analysis of the freeform feature concept

In this thesis, we propose a new theory for the support of freeform features. On the basis of this theory, a freeform feature recognition methodology can be developed that is a logical continuation of the work on regular form features and that is embedded in related research topics. However, although the previous two sections support a theory of the freeform feature concept, the specifics of such a theory remain to be given. In this section we derive elements of a definition of the freeform feature concept by analyzing recent work on freeform features.

2.3.1 Criteria for the analysis

From existing work, the following aspects of the freeform feature concept can be defined:

Type of parametric control

If parameter-driven control is available to modify the shape of a feature, then what is the relation between parameters and feature shape? For example, if the shape representation type of the feature is a B-spline, then a parameter may control the shape by translating the control points of the B-spline representation. The type of control that is given over a feature shape can be analyzed in terms of simplicity (e.g. is an intermediate structure necessary?), robustness (i.e. what shape changes can be controlled by a parameter?) and composition (can multiple shape manipulation operations be combined?)

Relation to the embedding surface

The shape of a feature can either be additive, subtractive or deforming with regard to its embedding surface, or it is independent of any embedding surface. When a feature shape is additive, then when a feature is instantiated on an embedding surface or when it is manipulated, new geometry is created. In this case, there is a clear distinction between the geometry of the embedding surface and that of the feature. When the feature shape is subtractive (also called eliminative), then it has no shape of itself; instead, it removes geometry from the embedding surface. When it is deforming, then the feature deforms the embedding surface, i.e. its geometry can be expressed as a deformation of the original geometry of the embedding surface (before a feature was instantiated). In this case, the geometry of the feature is a subset of that of the embedding surface and there is no clear transition between the geometry of the feature and the geometry of the embedding surface. Finally, the shape of a feature can be independent of any embedding surface, in which case the feature is free-floating.

Note that, in some of the existing literature on form features, the terms ‘additive’ and ‘subtractive’ are used in a different sense. Namely, additive features are considered to be what above is described as deforming features, but with a deformation that is strictly in the positive direction of the normal vector of the embedding surface. Subtractive features are also deforming features, but in the negative direction of the surface normal. Because technically there is no difference between the two, this notion is discarded here. However, to avoid confusion, the term ‘eliminative’ will be used throughout this thesis.

The relation to the embedding surface is important in determining how the shape change of a feature influences that of the embedding surface. If the feature is additive, then the shape of a feature does not have a direct relation to that of the embedding surface. If the feature is eliminative, then whenever the shape of the feature changes, the embedding surface must be modified to match. In the case of a deforming feature, the relation is even stronger: in this case the morphology of the feature is coupled to that of the embedding surface.

Nature of the feature shape
Are the features semi-freeform or fully freeform? If the shape of a feature is semi-freeform, then some aspect of the feature's shape is freeform while other aspects are not. For example, if the shape is the result of sweeping a regular profile along a freeform trajectory, then the shape of the feature is semi-freeform: its shape can either be modified by changing the regular profile or the freeform trajectory. If the shape of a feature is fully freeform, then only freeform shape manipulation can be applied to it. Whether features are semi-freeform or fully freeform is important when reasoning about the morphological aspects of the feature.

Ability to deal with user-defined features
Is a feature-based method able to deal with a fixed set of predefined feature types, or is it also able to deal with a dynamic set of user-defined features? If a feature-based method only targets a fixed set of pre-defined features, then does it make use of aspects that hold specifically for the features in this set and compromise its general applicability? If the method is able to deal with user-defined features, then are there any constraints on the type of feature that can be defined?
Answers to these questions give insight in the relation between a user-driven feature type definition method and feature-based methods that deal with user-defined features, and in addition give insight in the shortcomings of methods that do not.

Although the proposed criteria are not an exhaustive list of all the conceivable aspects of freeform features, an analysis of these aspects supports several research hypotheses as they were given in the previous chapter. The type of parameter-driven control is relevant for hypothesis 4. An analysis of the relation between feature and embedding surface supports hypothesis 6. The ability to deal with user-defined features supports hypotheses 5 and 6. Finally, an analysis of the nature of the feature shape supports hypotheses 2 and 3.

2.3.2 Review of freeform feature concepts

Cavendish et al. were one of the first to propose combining feature-based design and freeform modeling (Cavendish and Marin, 1992; Cavendish, 1995). They defined feature shapes by relating a curve representing what they call the secondary surface to a curve that represents the primary surface, both with regard to the xy plane (see Figure 2.2a). Both curves are projected onto the xy plane to determine the *transition region*, in which the secondary and primary surfaces are smoothly connected.
A very much similar approach is proposed by Van Elsas and Vergeest (see Figure 2.2b), who describe what they call the displacement feature (Van Elsas and Vergeest, 1998). Their method allows a user to define a custom feature on any target shape by sketching a

closed curve that determines the extent of the feature. The region that is contained within this closed curve is then displaced in the direction of the normal vector of the shape.

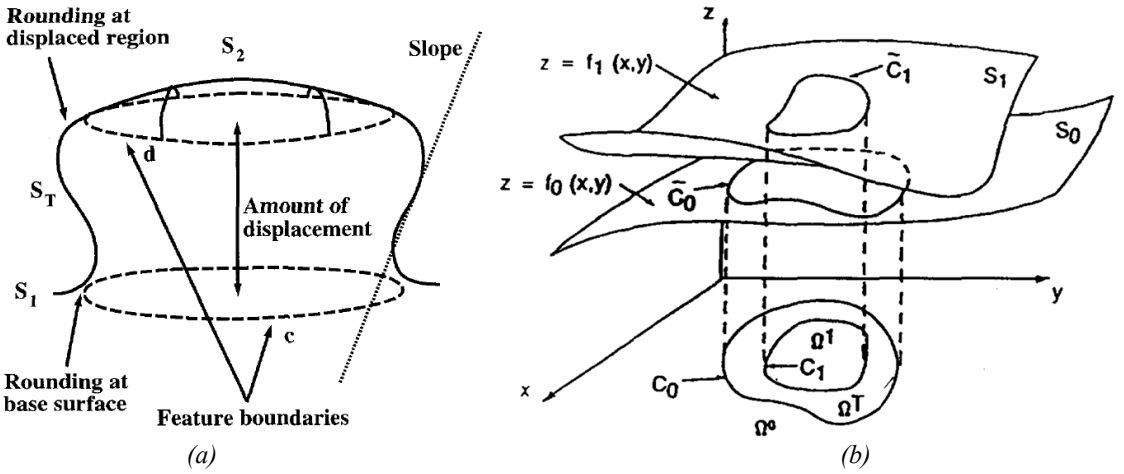


Figure 2.2*: Examples of freeform features definitions by (a) Cavendish and Marin and (b) Van Elsas and Vergeest*

Both the method of Cavendish and Marin and the method of Van Elsas and Vergeest can be used to define semi-freeform features. In both works, the feature shape is restricted to be built up out of three components: subregions of the primary and secondary surface and a transition region. In the work of Van Elsas and Vergeest it is a requirement that the two surfaces are parallel.

Cavendish and Marin do not give a parameter-driven control of the feature shape: the shape of the feature can be modified by manipulating either the primary or secondary surface. Van Elsas and Vergeest give a number of parameters with which a user can control the shape of a feature (see Figure 2.2b). These parameters have an indirect effect on the control points of the B-spline feature shape. Both approaches create features that can be regarded both as additive or as deforming features. The displaced region of a feature is obtained by a deformation of part of the embedding surface, but the transition region of the feature consists of 'new' geometry, i.e. the geometry of the transition region cannot be derived from that of the embedding surface.

Van Elsas and Vergeest specifically mention the importance of user-driven feature definition. In their method, user input is an essential aspect of the definition of a feature type. In the method of Cavendish and Marin, the user is also responsible for the creation of new feature type definitions, but the authors do not give a user-driven feature definition method.

The displacement feature can be used in specific situations. Both Van Elsas and Cavendish have tested their methods in practice and conclude that their method improves the efficiency of the creation of a displacement feature. However, the two approaches are

limited to displacement features and can not be generally used for the definition of freeform features.

Poldermann and Horváth propose a number of definitions for features with a B-spline representation (Poldermann and Horváth, 1995). Although they do not give a general method for defining feature types, their examples are numerous and based on an extensive classification of feature types (Figure 2.3). The authors do not go into the manipulation of feature shape and no statement can be made regarding the nature of the parameter influence. The examples of features given by the authors are user-defined, and although they do not give a mechanism to define new feature types, it is clear that in their view the user has an important role in defining new feature types.
Poldermann and Horváth give a large number of examples of freeform feature definitions, show in particular how features that are relevant in practical applications can be parameterized and demonstrate how these features can be instantiated on a shape model. However, they do not elaborate on the mechanisms with which a feature type can be defined, the properties of a feature or the method to instantiate it on a shape model. The criteria mentioned in this section can therefore only to a limited degree be applied to their work.

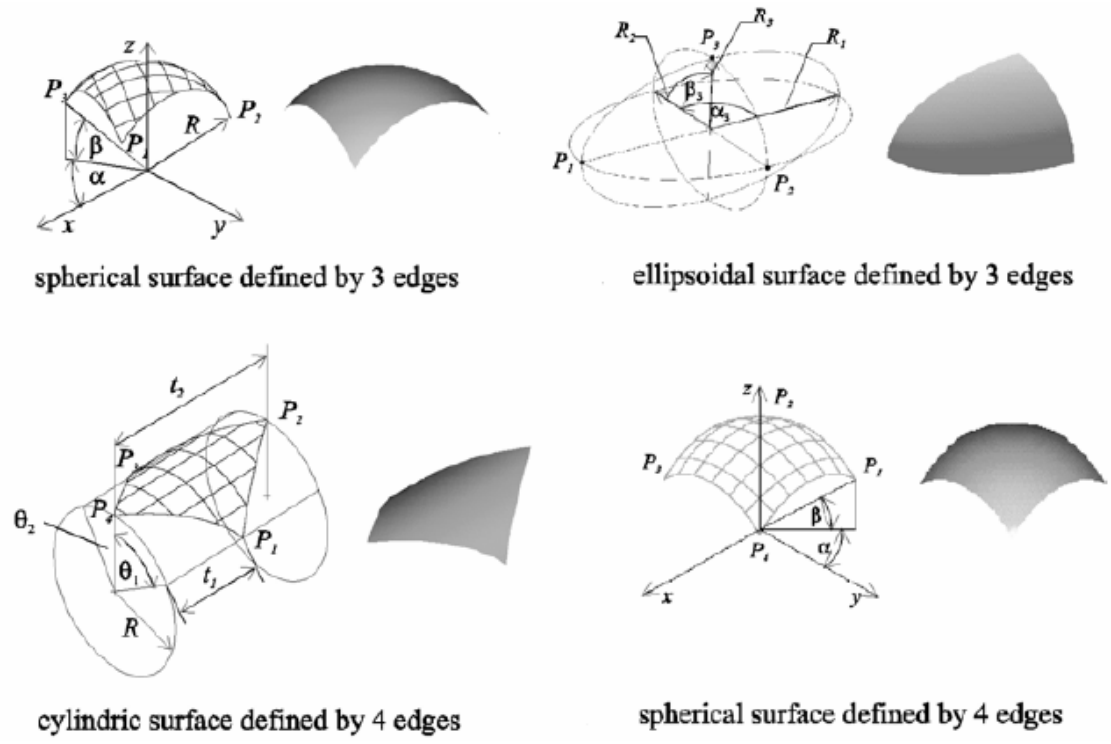


Figure 2.3: *Examples of feature definitions by Poldermann and Horváth*

Vosniakos proposes a feature-based modeling system that uses multiple Bezier surface patches to represent features (Vosniakos, 1999). In his approach, models are built up entirely out of features or, in other words: all parts of a shape model can be expressed as freeform features. Parameters are defined for each individual feature in the shape model and simultaneously control all Bezier patches that contribute to the shape of a feature while maintaining the topological relation between these patches. That is, the parameter-driven control over the shape of a feature does not target individual control points. The features are independent of any embedding surface and being composed entirely of Bezier patches, they are fully freeform. In this approach, features are all user-defined, but defining them requires a large effort from the user.

The approach of Vosniakos (see Figure 2.4) is general and can be used to define fully freeform features. However, a feature type definition is very complex. In addition, the approach requires the user to define the entire shape model as a composition of features, which introduces a lot of redundant data if parts of the model could be more simply described.

An approach to modeling with features that is similar to that of Vosniakos is that of Mitchell et al. and Au and Yuen. They also propose a feature-based modeling system in which a shape model is entirely built up out of features (Mitchell et al., 2000, Au and Yuen, 2000). Both authors do not specify how features can be defined, but focus on respectively the functional and semantic relation between features; these methods will therefore not be extensively reviewed here.

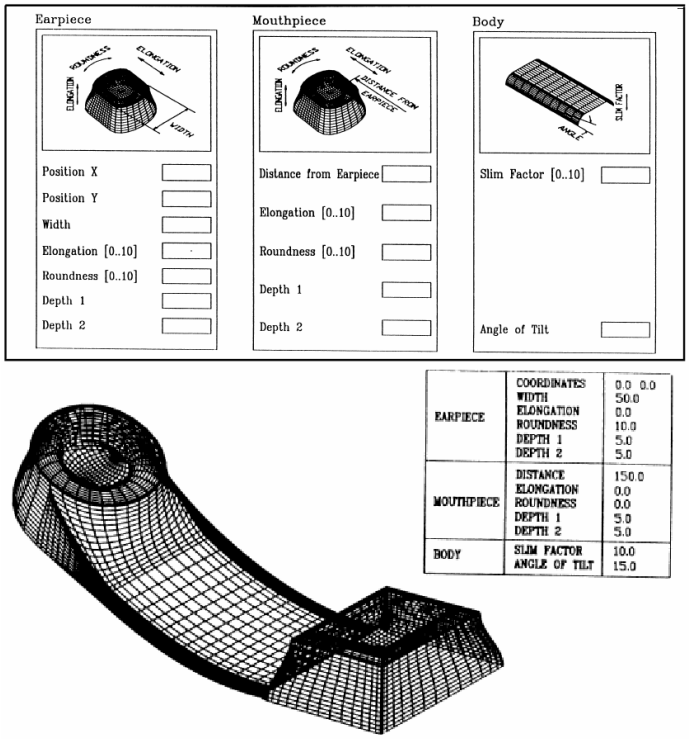


Figure 2.4*: Free form feature definition and an example of a feature-based model by Vosniakos*

Pernot et al. propose a user-driven deformation (Pernot et al., 2005) that is based on a technique proposed by Guillet and Leon (Guillet and Leon, 1998). They couple a bar network to the control polyhedron of a B-spline surface, which enables easy and smooth parameter-driven freeform deformation. The region of a shape that is parametrically controlled is dictated by a target line and is bordered by limiting lines (see Figure 2.5). To compute the configuration of the shape after deformation, first the target line is projected onto the target surface. A deformation is computed that transforms the points that lie close to this projected line onto the target line. Once defined, the feature can be manipulated by modifying limiting- or target lines, properties of the B-spline surface, relaxation constraints on the bar network and the rate of acceptable deformation outside the limiting lines. This means that part of the parametric control over the deformation lies with parameters that do not belong to the feature itself, but to the embedding surface of the feature.

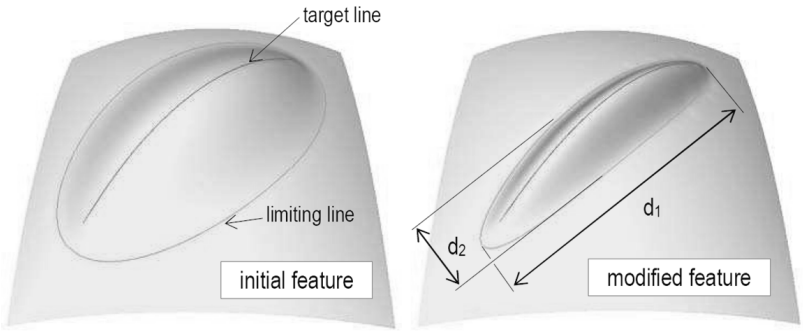


Figure 2.5*: Feature based deformation by Pernot*

An interesting addition proposed by Pernot is the possibility to model with template features (Pernot, 2004). These are predefined sets of limiting and target lines that can be instantiated on a desired target surface. Although it is clear how such predefined templates can contribute to making the proposed technique accessible to non-expert users, no method is proposed to make it possible for users to define features. Pernot mentions this as one of the important topics of future work.
The feature concept as it is proposed by Pernot is not fully freeform: it is assumed that the target line can be projected onto the embedding surface and this assumption limits the range of possible feature shapes. The approach is unable to deal with nested features.

Van den Berg et al. propose a feature definition approach that uses so-called Freeform Feature Definition Points (FFDPs), through which B-spline curves are fitted that define the boundaries of the feature's geometry (Van den Berg et al., 2003). Parameters can be constructed that relate to the relative position of FFDPs. Constraints relate to the distance or angle between these points. Figure 2.6 shows the shape prototype of a curved ridge,

defined with eight FFDPs. The authors do not provide any method to instantiate features on an embedding surface and the features are therefore independent from any embedding surface. The features that can be defined using this approach are fully freeform and are by definition suitable to create user-defined features. However, their use in practice is limited due to the fact that no instantiation method is proposed and due to the fact that the topology of the FFDP's is not a priori known and must therefore be specified by the user during a user-driven feature definition process.

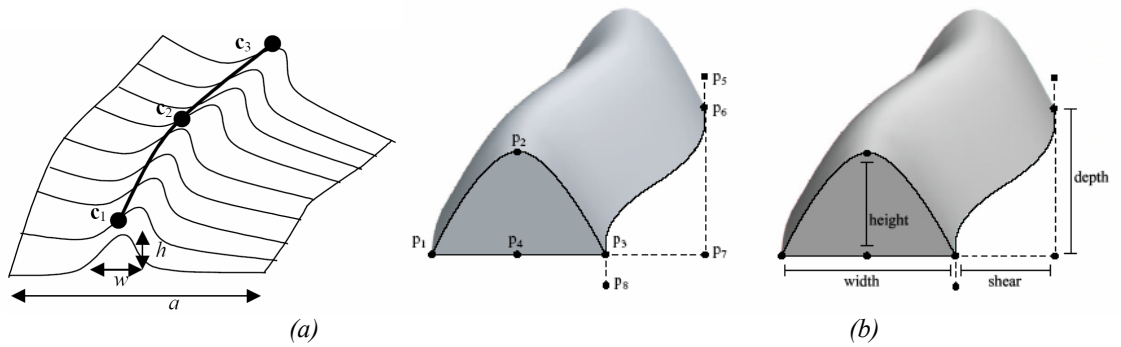


Figure 2.6*: Examples of a feature definition by (a) Van den Berg et al. and (b) Vergeest and Horváth*

Vergeest and Horváth propose a formalism for the parameterization of freeform feature templates (Vergeest and Horváth, 2001). Similar to the method proposed by Van den Berg et al., this method can only be used to define independent features, but in this case the feature templates are intended specifically for use in a feature recognition procedure and for this reason no embedding surface is required. The main difference with the method by Van den Berg et al. is that the points used to define the feature shape are control points of a B-spline. The shape of the feature can therefore be generated more efficiently by the known B-spline algorithms and the parametric control relates directly to the control points. In addition, because a bi-directional grid of control points is used as a data structure, the topology of a feature shape is known. Figure 2.6b shows an example of the definition of a curved ridge.

2.3.3 Conclusions

Although many approaches to freeform shape modeling exist that contain elements of the freeform feature concept, in this section only those approaches were reviewed for which a comprehensive analysis can be given based on specified criteria. In Table 2.1, a summary of the feature definition approaches that were reviewed is presented.

From an analysis of the different freeform feature concept, three groups can be discerned. The first group, in which the approaches of Cavendish and Marin, Van Elsas and Vergeest and Pernot can be placed, focuses on the instantiation of freeform features on a freeform shape. The definition of a freeform feature takes place by an operation directly

on the freeform shape, for example by sketching. The features are semi-freeform because they are limited by the relation of the feature shape to the embedding surface as well as by the instantiation method. The second group, in which the methods by Van den Berg, Vergeest and Horváth and Poldermann and Horváth can be placed, focuses on the definition of feature types without giving a method of instantiating these features on a target shape. Because neither the embedding surface nor the instantiation method has to be taken into account, the feature types that can be defined are fully freeform. The third group, in which the approaches by Vosniakos, Mitchell et al. and Au and Yuen can be placed, also consider features independent from an embedding surface, but they assume that shape models can be entirely built up by features. Hence, there is no need for an instantiation method.

	Type of parametric control	***Relation to the embedding surface***	***Nature of the feature shape definition***	***User-defined features***
Cavendish et al. (1995)	Non-parametric	Deforming	Semi-Freeform	+
Poldermann et al. (1995)	Shape properties	-	Fully Freeform	+
Van Elsas (1998)	Location of control points	Deforming	Semi-Freeform	+
Vosniakos (1999)	Location of groups of control points	Independent	Fully Freeform	+
Vergeest et al. (2001)	Location of control points	Independent	Fully Freeform	+
Van den Berg et al. (2003)	Constraint-based	Independent	Fully Freeform	+
Pernot (2005)	Global parameters	Additive	Semi-Freeform	+

Table 2.1*: Summary of the reviewed feature definition approaches*

The criteria for an analysis of the approaches to the freeform feature concept were chosen because they support the research hypotheses that were given in the previous chapter. Strong support for hypothesis 5 can be found in the analysis of the ability of the reviewed work to deal with user-defined features. In fact, in most of the reviewed work it is argued

or implied that all features should be user-defined. In the most recent work reviewed, that of Pernot (2005), counterarguments can be found for this: although freeform feature methods should be able to deal with user-defined features, overemphasizing this goes at the cost of efficiency and the use of pre-defined or template features must therefore be considered.

Support for hypothesis 4 can be found by analyzing the types of parametric control over the shape of the feature. Most of the approaches to the form feature concept, in all three of the groups, define parameters as having an effect on the location of the control points. This control can be implemented by a simple operator, such as a translation vector or a transformation matrix.

Support for hypotheses 2, 3 and 6 could only sparsely be found: the reviewed methods do not define, or are at the least not specific about (i) an underpinning theory of the freeform feature concept; most approaches address a practical need instead of a theoretical exploration, (ii) the relation between parameters and feature shape; although most authors mention this relation, often it is not specified, and (iii) the relation to the embedding surface; even the work in which a detailed description of the instantiation of a feature on an embedding surface is given do not go into the subsequent managing of this relation, for example as the result of a parameter value change.

In the next chapter, an analysis of these issues will be given and it will be discussed what aspects a theory on freeform features must contain to support these issues as well as the research hypotheses.

3 A foundational theory and computation model for freeform features

In the previous chapter, we have reviewed the existing literature on regular form features, discussed the positioning of the feature concept in related research topics and have given an analysis of definitions of the freeform feature concept. This analysis focused on the relation of a feature to its embedding surface, the type of parametric control over the feature, the nature of the feature shape and the ability to deal with user-defined features. However, these aspects of features are relevant specifically in the context of feature-based design and do not form a comprehensive theory; to support a methodology for freeform feature recognition, a more extensive supporting theory is needed. In this chapter, we present a foundational theory for freeform features, as well as a computational model that targets freeform features in general and freeform feature recognition in particular.

In section 3.1, we first discuss the context and applicability of a foundational theory; it is argued why such a theory is necessary and what the extent of such a theory is. In section 3.2 we discuss the assumptions that form the basis of the development of a foundational theory and conjecture a general structure for a theoretical framework. In section 3.3 we then present a foundational theory, on the basis of which we present and discuss a computational model in section 3.4. Finally, in section 3.5 we discuss a possible implementation of the foundational theory.

3.1 Context of a foundational theory

Given that many researchers consider the topic of regular form features to be closed-off, one would expect that there exist a global supporting theory on the form feature concept that combines the many different theories in one comprehensive theory; without such a theory, feature-based methods cannot be generalized. Preferably this theory would be sound and complete, but it is debatable if such an ‘ultimate’ theory exists. In the domain of regular form features a comprehensive theory could not be found, and indeed: many of the developed methods have not been properly generalized. This was likely caused by the fact that many feature researchers have only concerned themselves with a specific application of features; generalization was therefore not a priority. In the wide range of applications of the form feature concept, many theoretical aspects of features have been addressed, but unfortunately no coherent theory exists.

In the domain of freeform features, the need for a foundational theory is even higher, because:

- The freeform feature is more complex than the regular form feature. As a result, research efforts target relatively smaller research problems; consequently, there is a stronger need for the generalization of the results of these efforts.
- Another consequence of the larger complexity of the freeform feature concept is that more different research problems exist; the effort that is needed for guarding the compatibility of the research results is higher.
- The research after freeform features is young and many research problems still need to be addressed. An established foundational theory would increase the efficiency with which new methods can be developed, because the 'theoretical wheel' does not have to be reinvented.

Several ideas and concepts from the domain of regular form features can be reused in creating a foundational theory for freeform features. However, such a theory cannot simply be copied from the domain of regular form features. This may seem logical, but we explicitly mention it here because many of the methods that have been proposed in the domain of freeform features are based on assumptions that are derived from the regular form domain. For this reason, the applicability of these methods is limited. The theories for regular form features suffer from several shortcomings when applied to the freeform feature domain.

- Theories on regular form features are mostly local: they do not take into account the context of the feature. Although local features can be parametrically deformed, their location and orientation are always fixed, because no relation to an underlying surface is defined.
- Theories on regular form features define features on the level of geometry, and in some cases on the level of topology, but do not or only partly define the form feature on the level of morphology. Although most theories assume that a feature can be parametrically deformed, the relation between parameters and geometry is in many cases ill-defined.

The challenge that is addressed in this chapter is therefore to create a foundational theory that deals with features globally and on the level of morphology and provides a comprehensive and coherent theoretical support for many different problems in the domain of freeform features.

This theory has the potential to:

- Provide solutions to the research problems that are still open in the domain of regular form features, such as feature interference and feature mapping.
- Support the development of a methodology for freeform feature-based methods and in particular freeform feature recognition. Although in this thesis we concentrate on freeform feature recognition and only briefly go into how the theory supports other methods, from an analysis of the existing literature we could not conjecture any argument why the theory is not broadly applicable.
- Formulate problem statements for new research problems. In a critical analysis of the freeform feature concept, new challenges can be conjectured, or the clarity of existing, vaguely defined, problem definitions can be improved.

Even though in our opinion it is clear that a foundational theory for freeform features is necessary and that such a theory has potential, a sound and complete theory cannot be defined within the context of a single thesis, on the one hand because such a theory can only be accepted in the feature research community through extensive discussion. On the other hand, to be able to provide a detailed foundational theory in a limited amount of time and space, several aspects of features had to be left underexposed. Where we believe this to be the case, it will be discussed how the foundational theory can be extended to also cover these aspects.

3.2 Underlying assumptions and general structure of the foundational theory

Before giving a formal definition of the freeform feature concept, first the underlying assumptions of such a definition must be determined. First, we assume that a freeform feature is defined on the level of geometry and on the level of morphology. In section 3.2.1 we give a detailed analysis of what information a feature definition should contain on these two levels. Second, we assume that features are defined with regard to an embedding surface. In section 3.2.2, we discuss this assumption. In section 3.2.3, we discuss the assumption that constraints are available to prevent features from obtaining invalid shapes. Finally, we assume that the parametric control of features is local, linear and one-dimensional. The parametric control of features is discussed in section 3.2.4.
Other assumptions can be made regarding the freeform feature concept, but in this section only assumptions are discussed that are relevant in the application of the freeform feature concept to feature recognition.

3.2.1 Morphological aspects of the freeform feature concept

The shape of a freeform feature can be defined on the level of geometry and on the level of morphology. On the level of geometry, features are defined in terms of the properties of the geometric elements of the feature shape. For example, if the shape of a feature is represented as a polygon mesh, then on a geometric level it can be defined by giving the coordinates of its vertices. This implies that the exact definition of a freeform feature depends on the shape representation type of the feature shape. A feature that is defined only on the level of geometry can therefore not be easily interchanged between different feature-based systems if these systems use different shape representation types. Another disadvantage of dealing with features on the level of geometry is that in defining and operating on a feature, each individual geometric element that is part of a feature's shape must be addressed. This makes the definition of and operation on features on the level of geometry inefficient.

The reason for wanting to define freeform features on the level of geometry despite these disadvantages is straightforward: many operations on freeform features are only possible on the level of geometry. For example, testing for interference with other shapes or testing for self-intersection is only possible if a definition of the geometry of the feature exists. Most of the existing work on features defines them primarily on a geometric level.

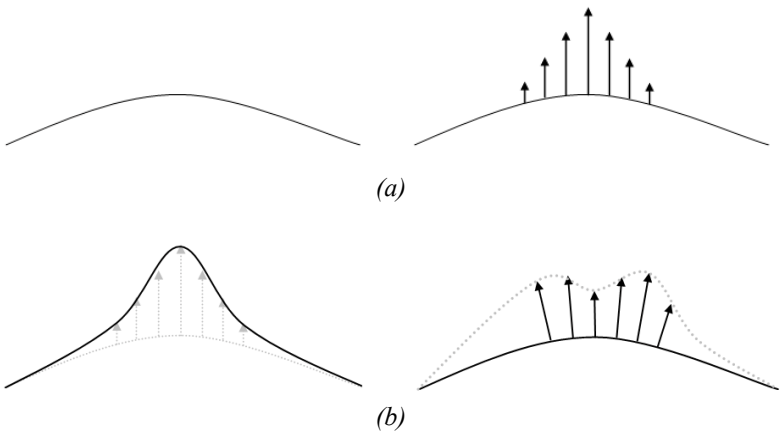


Figure 3.1: *Discrete example of (a) a feature definition on the level of morphology and (b) a geometric (left) and a morphological (right) change.*

If a feature is defined on the level of morphology, then it is specified how its geometry changes under the influence of internal or external factors, such as parameters (internal, see Figure 3.1) or forces (both internal and external). For example, if in a virtual environment someone interacts with an object by pressing a virtual finger into it, then its shape changes as a result of the external forces applied to it, in combination with the internal forces, e.g. stiffness. When a definition of a feature on the level of morphology is

available, then its geometry can be manipulated according to the defined morphology (Figure 3.1b, left). In addition, the morphology itself can be changed. A modification of the morphology changes how the shape can be modified (Figure 3.1b, right). Note that the example used here is a discrete example, because it is difficult to show a continuous morphology.

Although a freeform feature cannot be defined as a parametric entity without addressing it on the level of morphology, many theories on form features assume a morphological structure to be available without defining it explicitly. No examples could be found in the existing literature of a definition of a freeform feature that allows the morphology of the feature to be changed.
In this thesis, we are particularly interested in the definition of the morphology of a feature in relation to its parametric nature. In addition, it is possible that a morphology is defined with respect to other phenomena, such as material properties or physics-based properties. In this case different morphological aspects may interfere or interact; when changing the morphology of a shape in one aspect, the influence of this change on other morphological aspects must be taken into account. However, in the practice of industrial design other morphological aspects are not typically used, and they are not, or only to a very limited extent, supported by the current commercial CAD-systems. We therefore make the following assumption:

Assumption 1: *If a CAD-model or part thereof has been defined on the level of morphology, then this morphology relates to the parametric nature of a feature that has been instantiated on the model.*

This assumption implies that the theory and methodology presented in this theory are applicable primarily to CAD-models or parts of CAD models for which no morphological structure has been defined. The theory and methodology can also be applied to CAD-models or parts thereof that already contain morphological information, but in this case it can not be guaranteed without further research that existing morphological information remains valid when applying feature-based operations.

3.2.2 Relation of a feature to its embedding surface

In the previous chapter, we identified three different approaches to the freeform feature concept. The first, in which features were defined separately from an embedding surface is for obvious reasons not relevant in an application to feature recognition: these features will not occur in practice as self-sufficient shapes. The second, in which shape models are built up entirely out of features, is in our opinion not feasible in the case of feature

recognition for it requires us to recognize an entire model instead of a specific feature. The final approach, in which features are embedded in a surface, will be adopted here.

Assumption 2: *Features are defined with respect to an embedding surface.*

Note that this assumption does not mean that a feature only exist with respect to an embedding surface; it should also be possible to regard the feature in an 'un-instantiated' state providing that an instantiation method is available to impose the feature on an embedding surface. An instantiation method can therefore also be taken to be a method that transforms the feature from an un-instantiated state to an instantiated state.

3.2.3 Constraints

In a CAD-modeling process, there are many shape operations that lead to invalid configurations of a shape. To prevent these configurations from occurring, shape operations are subject to *constraints*, which are limitations to the type or magnitude of the shape operation. Constraints can relate to many variables, such as continuity and curvature, and can be interdependent. Constraint management can be very complicated and is a research topic of its own. Efforts into the investigation of constraints on freeform features are made elsewhere (e.g. Van den Berg, 2007) and we therefore make the following assumption:

Assumption 3: *The parameters of features are constrained such that no parametric configurations occur that correspond to an invalid shape configurations of a feature occur.*

The reason for making this assumption is that in the theory and methods presented in this thesis there are special cases in which the theory does not hold or in which methods do not work. Some of these cases, although theoretically correct, are not meaningful in the practice of working with features and often parameter constraints can be used to prevent these cases from occurring. If this is the case, then without going into details on the constraints it will be assumed that a constraint management system is available that handles the constraints and ensures that the feature shape is valid.

3.2.4 Parametric control of features

There are many ways of controlling freeform shapes. The main advantage of interpreting part of a freeform shape as a feature, is that it enables a high-level control over the shape. To be able to control the feature, a user handles the parameters of the feature. A change in

the status of a parameter leads to a change of the shape of the feature; the parameters function as an interface between user and feature shape.

In the main body of the thesis we will assume that:

Assumption 4: *Parameters have real, numerical values.*

This means that users can control a parameter by giving it a specific value. Under this assumption it is also possible to present parameters as having only a discrete set of values, possibly named after a certain property, e.g. 'sharp' or 'smooth'; this can be done by imposing constraints on what parameter values are possible.
Later in this thesis, to demonstrate that the proposed approach to feature recognition can also be applied to other parameter types, a method will be presented for two-dimensional parameters, or curve-based parameters. Users can control these parameters by sketching a two-dimensional curve, which then influences the shape of a feature. For example, the sketched curve can determine a cross-section of a feature. Other types of parameters are possible as well, but to analyze these requires a separate thesis, and we therefore stick to assumption 4.
Second, the control a user has over a parameters must also be logical, i.e. when the user makes a parameter value change, the change in the feature shape must have a predictable effect. We therefore assume that:

Assumption 5: *Parametric control is linear*.

That is, the change of a parameter value is proportional to the Euclidean distance between a point on the feature shape prior to the change and after the change. Linearity of the parametric control implies that this change can be scaled by multiplying or dividing the parameter values. For example, if a parameter value is increased by a value of 2, then this causes an effect on the shape of the feature that is twice as large as when the parameter value would have been increases by a value of 1.

3.3 A foundational theory for the freeform feature concept

In this section, a foundational theory for freeform features is presented. This foundational theory describes the essence of freeform features, and is based on the assumptions that were given in the previous section. In addition we formalize some theoretical concepts that support using the theory in freeform feature-based applications. In section 3.3.1, we first give supporting definitions of aspects of the freeform feature concept, and then

conclude with a definition of the freeform feature, in section 3.3.2 the concept of shape manipulation is discussed and in section 3.3.3 the concept of feature space is introduced.

3.3.1 Definitions of a freeform feature

The reason behind the use of freeform features is that they provide high level access to shape data. 'High level' in this regard means that a larger part of the shape data can be interrogated or manipulated by a single action. When using features, this single action refers to a value change of the parameters that have been defined for the feature. This parameter value change can be 'translated' into a geometric change by evaluating the morphological aspects of the feature and the parameter values given by the user. This results in what is called the *shape configuration* of the feature. Examples of a shape configuration are an ordered set of B-spline control points or the configuration of vertices and faces in a polygon mesh.

A feature must be regarded as a structure rather than as a shape. The shape of the feature is a manifestation of the feature, but there are others, in particular the *parametric configuration* of the feature. This can be depicted as follows:

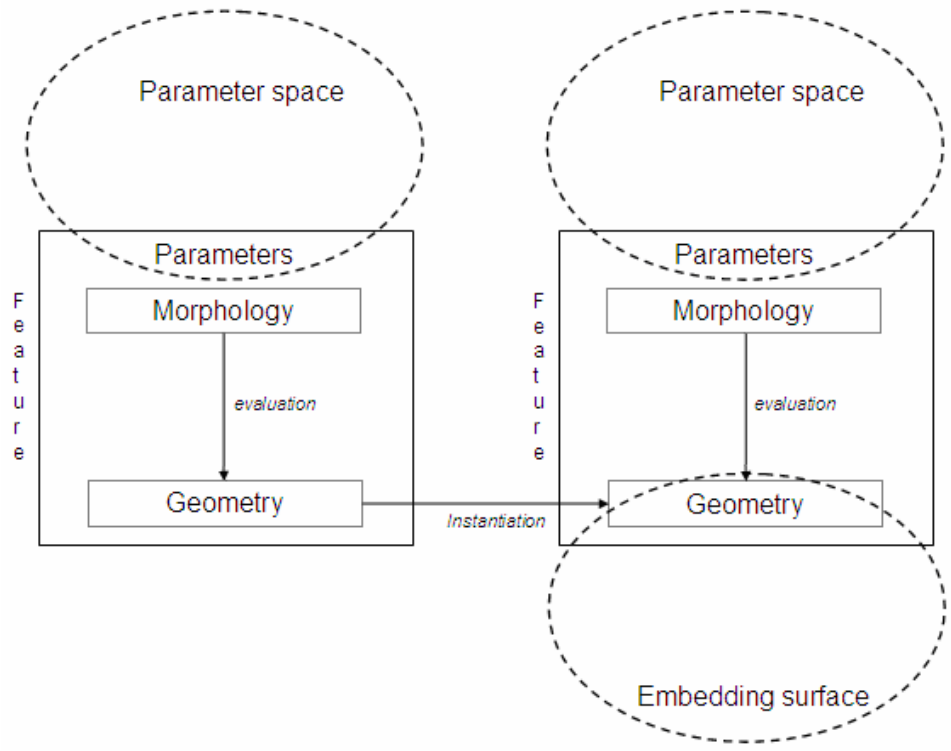


Figure 3.2*: Diagram of the structure of different manifestations of the feature*

On the left of Figure 3.2 is the feature as a pre-defined feature type, for example as it is given in a feature library. The parametric configuration is embedded in the *parameter*

space, which contains all possible parametric configurations for that feature and will be discussed in more detail in the following section. On the right is the feature when it has been instantiated on an embedding surface. In this case, the geometry of the feature is contained in the embedding surface.

Parameter space

In Figure 3.2, the concept of parameter space is introduced, which can be defined as follows:

Definition 3.1: Definition of parameter space
The parameter space of a feature is the set of all possible parametric configurations of the feature.

For a feature with *m* parameters, an *m*-dimensional parameter space exists. The parameter values of a feature can be given in the form of a vector of length *m* and in this case, the parameter space is a vector space. A vector space is a space $\mathbb{R}^m$, in which each coordinate can be described as a vector originating in the origin of the space. The most commonly known example of a vector space is the three-dimensional Euclidean space. In Figure 3.3, examples are given of the two-dimensional parameter space of a bump feature. The shapes of the feature on the left side correspond to positions in parameter space depicted on the right side.

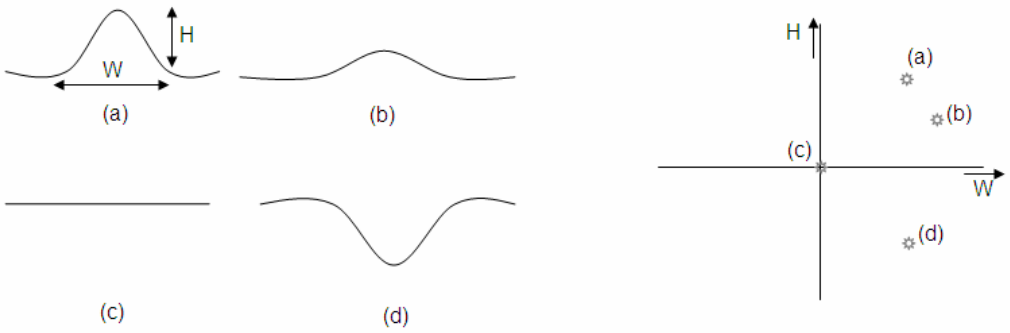


Figure 3.3: *Examples of feature shapes that correspond to positions in two-dimensional parameter space.*

Because the parameter space is a vector space, we can apply mathematical theory on vector spaces. This is particularly useful when applied to the concept of feature shape manipulation. Changing the shape of a feature corresponds to an *m*-dimensional translation vector in parameter space. Because vectors in a vector space can be added and scaled, this also holds for shape manipulation routines.
In vector space, all coordinates relate to the origin of the vector space. This origin corresponds to what is called the *basic shape configuration*, which can be interpreted in

different ways: if instantiated on an embedding surface, then the basic shape configuration is the situation in which a feature has no effect on the geometry of the embedding surface. Alternatively, the basic shape configuration can be said to be the situation in which the evaluation of the morphology and the parameter values of a feature is equivalent to applying the identity function.

Parameter mapping
In the morphological description of a feature, the relation between its parameter values and its geometry is given. We require this relation to be complete, in the sense that every point in the feature's geometry relates to each parameter that is defined for the feature.
The morphological description is given in what is called the parameter mapping and can be defined as follows:

Definition 3.2: Definition of the parameter mapping
The parameter mapping is a set of functions that, given a parametric configuration of a feature, results in a shape configuration of the feature.

The morphology of a feature can be specified by setting the functions in the parameter mapping. In other words, these functions determine how a user can control the feature shape. An alternative view on the parameter mapping is that its functions affect a transformation of the geometry with regard to the basic shape configuration of the feature (see Figure 3.1). This notion corresponds to the concept of shape manipulation, as it takes a shape configuration as a starting point rather than an abstract position in parameter space. The importance of this analogy becomes apparent when regarding it in reverse: a shape manipulation such as is typical in a freeform feature-based modeling process can alternatively be described as a functional relation between the parametric configuration and the target shape configuration. The alternative view on shape manipulation and its implications are further discussed in section 3.3.3.

Relation to the embedding surface
If a feature has been instantiated on an embedding surface, then we say that the feature is attached to the embedding surface and we assume that a feature changes the geometry of its embedding surface.

As was assumed in section 3.2, features are not required to have an embedding surface. A feature is said to be *attached* if it has an embedding surface and *floating* when it does not. A feature can be attached to an embedding surface in any part of the surface. That is, the attachment can be 0-dimensional; in this case the feature is attached to its embedding surface in an *attachment point*. This is for example the case when the feature has a 'central point', for example the bump feature (see Figure 3.4a and b). The attachment can

also be 1-dimensional, in which case there is an *attachment line* or *curve* (see Figure 3.4c and d), or 2-dimensional, in which case there is an *attachment region*. An example of a feature that has an attachment region is the displacement feature (see Figure 2.2). Finally, it may be possible that a feature has multiple attachments, for example in the case of a handle (see Figure 3.4e and f). All cases of attachment can be defined the same way, but before doing so we first introduce the concept of *influence region*.

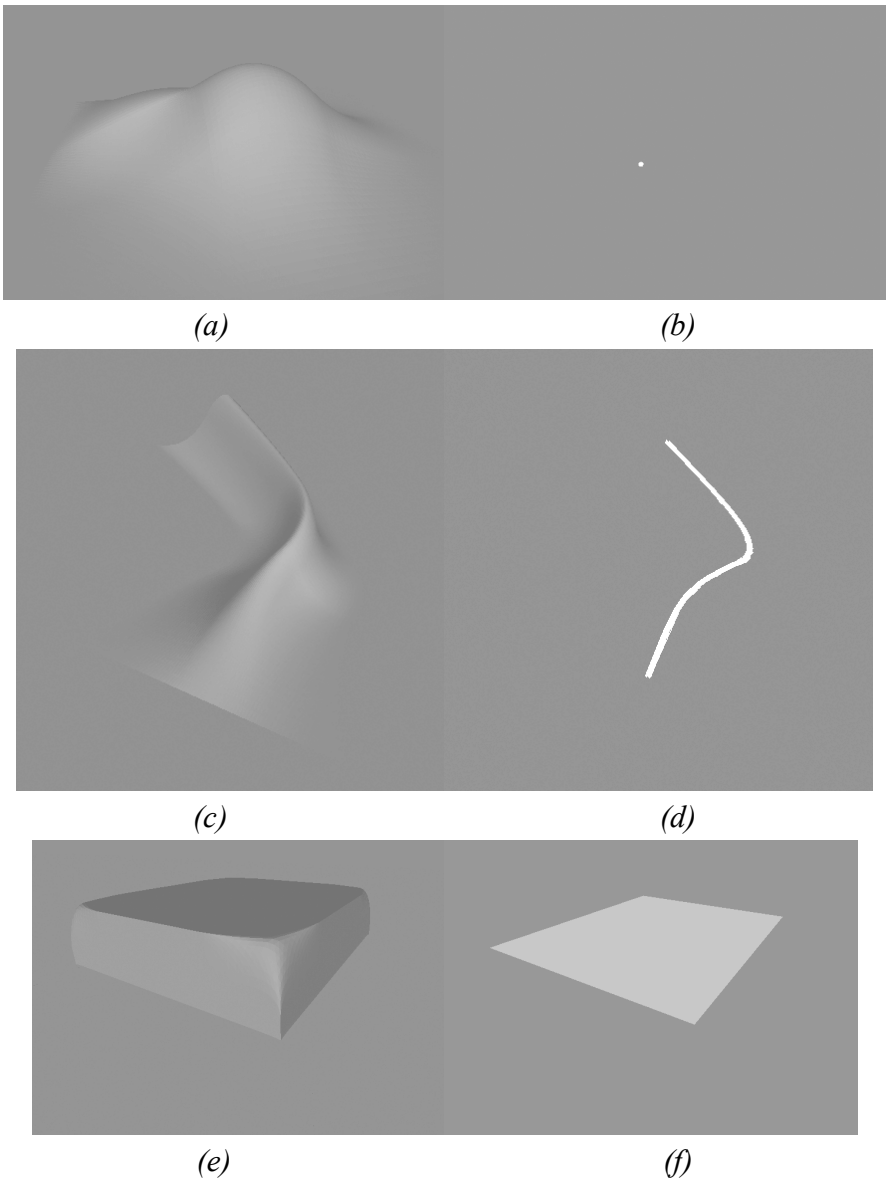


Figure 3.4: *Examples of features and their attachment: (a) a bump-like feature with (b) an attachment point, (c) A ridge-like feature with (d) an attachment curve and (e) a displacement feature with an (f) attachment region*

Apart from the case of an attachment region, the attachment of a feature says nothing about the region of its embedding surface that is deformed by the feature, because this region is determined by the attachment as well as by the parameter values. The influence region can be defined as follows:

Definition 3.3: Definition of the influence region of a feature
The influence region of a feature F is the part of the geometry of its embedding surface that changes as a result of F.

Using this definition, we can define the attachment of a feature as follows:

Definition 3.4: Definition of the attachment of a feature
The attachment of a feature is the influence region of the feature when the values of its parameters approaches zero.

In other words, if the effect of the parameters of F is infinitely small, then the influence region 'shrinks' into the attachment of the feature. If the attachment of a feature lies in a single point, then we consider this point to be the origin of its local coordinate system. Using these two definitions, we can define the basic shape configuration as follows:

Definition 3.5: Definition of the basic shape configuration
The basic shape configuration is the state of the feature in which the influence region of the feature is equivalent to the attachment of the feature.

Following from Definition 3.4, it can be derived that when the feature is in its basic shape configuration, all the parameter values of the feature are 0 (as was shown in Figure 3.1). A formal definition of the freeform feature concept can now be given as follows:

Definition 3.6: Definition of the freeform feature
A freeform feature $F\left(E^0, P, \mu\right) = E$ is a function that, given a basic shape configuration E^0, a parametric configuration P and a parameter mapping μ, results in a shape representation E, where:

- *E^0 is an enumeration of the geometry of the basic shape configuration*
- *$P = \left(p^1, \ldots p^m\right)$ is a vector of parameter values*
- *μ is a morphological description of the feature*
- *E is the geometry that results after changing the basic shape configuration*

In other words, given a vector of parameter values, a feature F applies the function μ to the basic shape configuration of the feature such that $\mu\left(E^0, P\right) = E$.

As said before, the basic shape configuration E^0 occurs when the vector of parameter values $P^0 = (0,\ldots,0)$, which is also the origin of the parameter space (e.g. see Figure 3.3c). It holds that $\mu\left(E^0, P^0\right) = E^0$.

3.3.2 Feature shape manipulation

A specific shape representation of a feature can be derived from its parametric configuration through the mapping functions defined for the feature. This mechanism allows us to reason about shape manipulation both as a transformation between two states of the geometry and as a translation in parameter space. This is depicted in Figure 3.5.

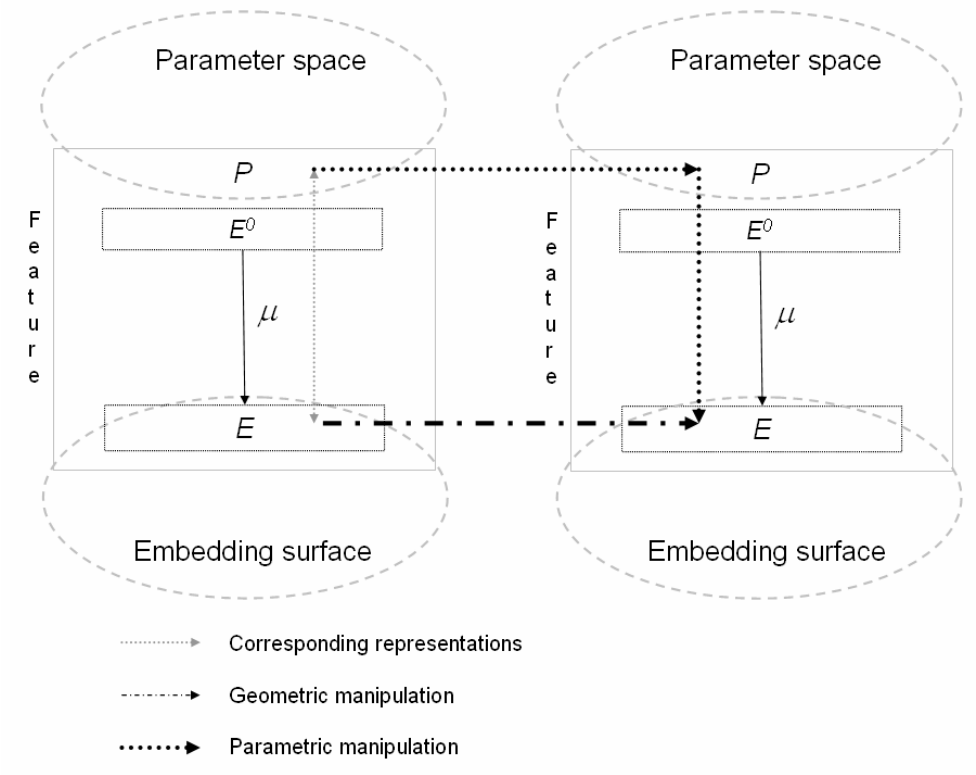


Figure 3.5*: Different information flows in the process of feature manipulation*

Being able to reason about features on the level of parameters increases the efficiency of feature shape manipulation for the following reasons:

- The validity of the shape does not have to be maintained; the result of a parametric manipulation only becomes apparent at the end of the manipulation process.
- A parametric manipulation is more efficient than a geometric manipulation because to obtain a parametric manipulation only input parameter values are needed.
- In a parametric manipulation, the relation of the feature to its embedding surface does not have to be taken into account.

The only disadvantage of a feature manipulation on the level of parameters is that it cannot be visualized without first generating the geometry of the feature. For the process of feature recognition, as will be shown later in this thesis, this is not a problem as only parametric manipulation is used. However, visual feedback is an important aspect of a feature-based modeling process; parametric manipulation can be used, but must be coupled to a visualization process, as is depicted in Figure 3.6.

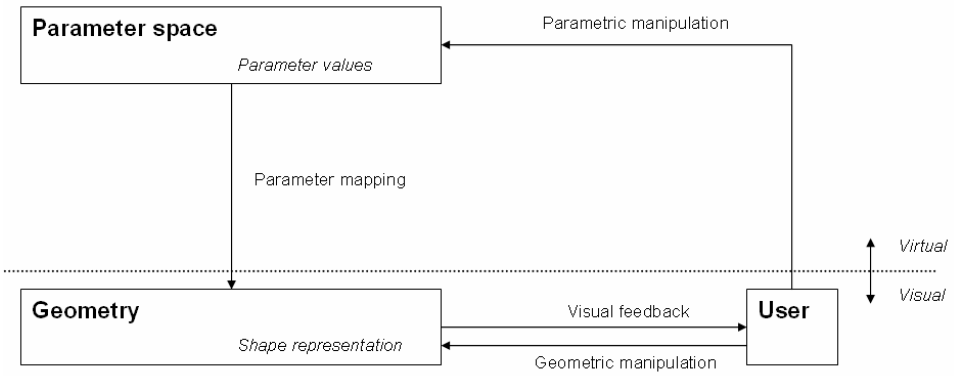


Figure 3.6*: The information flow in a feature-based modeling process*

Although the process of feature recognition will not be defined until the next section, we can look ahead and say that in such a process the optimal configuration of a feature must be defined. To obtain this configuration, several intermediate configurations must be investigated. The difference between these intermediate configurations can be expressed as a parametric feature manipulation, and for this reason a formal description of parametric feature manipulation is of great importance for supporting the process of feature recognition.
Based on Definition 3.5, a parametric feature manipulation can be formalized as follows:

Definition 3.7: Definition of a parametric feature manipulation

A parametric feature manipulation $\mathrm{M}\left(\tilde{E},\Delta P\right)=F\left(\tilde{E},\Delta P,\mu\right)$ *is a change of the shape configuration of a feature as a result of a change* $\Delta P=\left(\Delta p^1,\ldots,\Delta p^m\right)$ *of its parameter values, where* $\tilde{E}=F\left(E^0,\tilde{P},\mu\right)$, $\tilde{E}=\mu\left(E^0,\tilde{P}\right)$ *and* $\tilde{P}$ *are respectively the shape representation, shape configuration and parametric configuration prior to the shape.*

In other words, the manipulation of a feature with a shape $\tilde{E}$ through a change ΔP in parameter values can be expressed as $\mathrm{M}\left(\tilde{E},\Delta P\right)$. By defining parametric feature manipulation this way, we can combine two successive feature shape manipulations $M\left(\tilde{E},\Delta P'\right)=E'$ and $M\left(E',\Delta P''\right)=E''$ to $\mathrm{M}\left(\mathrm{M}\left(\tilde{E},\Delta P'\right),\Delta P''\right)=\mathrm{M}\left(\tilde{E},\Delta P'+\Delta P''\right)=E''$.

As becomes apparent in definition 3.9, two sequential shape manipulations can be combined in such a way that the intermediate shape is not needed to compute the eventual shape. From definitions 3.8 and 3.9 it can be derived that $\mathrm{M}\left(\tilde{E},\Delta P\right)=\mathrm{M}\left(E^0,\tilde{P}+\Delta P\right)$, in other words: each shape configuration can be seen as a transformation of the basic shape configuration. Consequently, in any series of feature manipulations all intermediate configurations, both parametric and shape configurations, are irrelevant for determining the last manipulation in the series (see Figure 3.7).

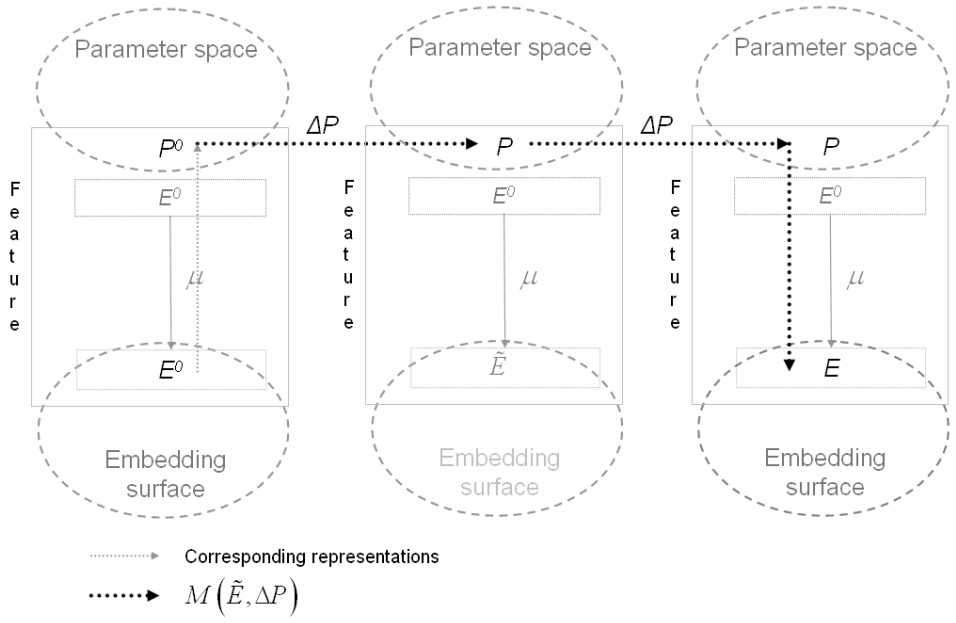


Figure 3.7: *Diagram of the information flow in two successive shape manipulations*

3.3.3 Feature space

In the previous section it was argued that reasoning about feature manipulations can be done on the parameter level with much more efficiency than on the shape level. However, for some aspects of features, it is necessary to regard the shape of features. For example, to determine if two features interfere, their shape must be computed first. The amount of effort that is needed for instantiating a feature on an embedding surface depends to a large extent on the complexity of the surface. It may for example occur that an embedding surface is highly curved or that a feature is instantiated on the border of a surface and therefore only partly contained on the embedding surface. However, in the implementation of the foundational theory that will be given in section 3.4, the morphology of a feature is dependent on the surface normal vector, and we therefore can define the morphology of a feature on any type of embedding surface (independent of the geometry of the surface). For the purpose of calculation, we can simply replace the embedding surface by the least complex embedding surface: a flat surface. A 'mock-up' feature can be instantiated on this alternative embedding surface and can then be used to do the calculations that are, for the original feature, difficult because of the complexity of the original embedding surface.
In order to be able to formalize the concept of a mock-up feature, we introduce the concept of feature space, defined as follows:

Definition 3.8: Definition of feature space
Feature space is a virtual three-dimensional environment in which the xy-plane is the embedding surface of the feature.

To distinguish between feature space and the normal modeling environment we call the latter *modeling space*. A designer that is working with features only sees the modeling space, while the computation is done in feature space. During a feature-based process, for each feature in modeling space, we maintain a virtual copy of the feature in feature space. That is, when a feature is instantiated in modeling space on a surface, then at the same time, not visible to the user, we maintain a copy of the feature in feature space. The copy in feature space can be used to do all the calculation for the feature, while the designer sees only the result in modeling space.
To make sure that the shape manipulation of a feature in feature space also becomes apparent in modeling space, a transposition mechanism is needed between the two spaces. This mechanism guarantees that the correspondence between the copy of the feature in feature space and the copy in modeling space remains intact and valid.
Figure 3.8 shows the information flow when modeling with feature space. The figure shows that, through the user interface, the input from the user is unnoticeably rerouted to feature space, where it effectuates a modification in the configuration of the feature. This

is then transposed to modeling space, where it is visible to the user, providing feedback on the user input.

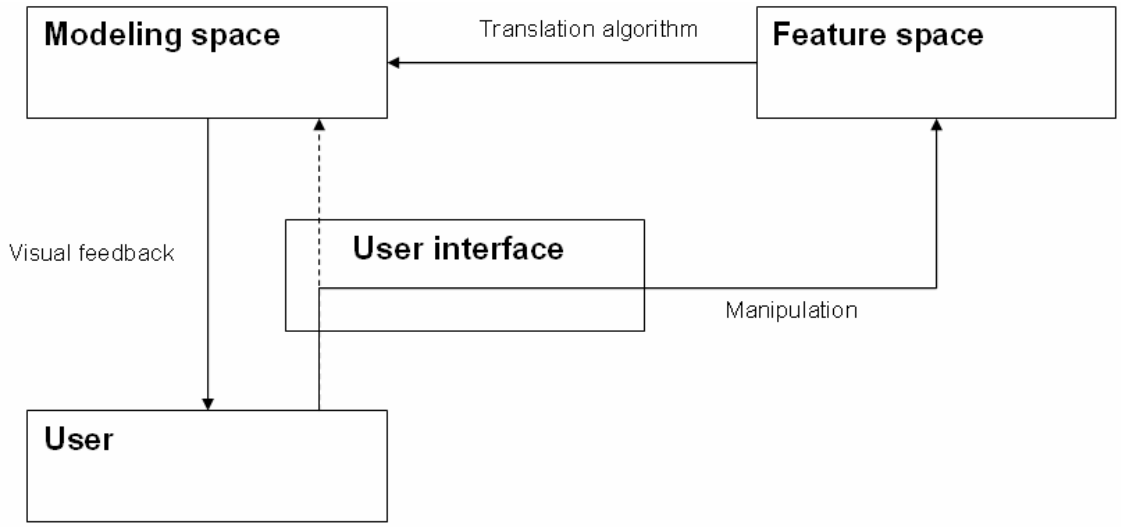


Figure 3.8: *Information flow when using feature space*

If the concept of feature space is used and if a transposition algorithm is available, there is no need to perform any computation on a feature in modeling space. The effect of a modification of the feature can be computed with less or, in the worst case, as much effort in feature space.
For this reason, the use of feature space as a computational environment is not merely useful for an increase of the efficiency of geometric manipulation. It is also possible to change the geometry or, more importantly, the morphology of a feature without having to re-instantiate the feature.

In the remainder of this section, we will look more closely at the different aspects of feature space.

Area and deformation parameters
An attached feature enacts a transformation of the geometry of its embedding surface. This transformation takes place in the influence region of the feature; both the extent of the area and the magnitude of the transformation are determined by parameter values and parameter mappings. We distinguish two parameter types: area parameters determine the extent of the area of the feature and deformation parameters determine the amount of transformation. The difference between the two will be made clearer when we go into the implementation of the theory.

Direction vector and normal vector

To be able to perform any calculation on the feature we must know the correspondence between modeling space and feature space. For this reason, we require a feature to have an origin on its embedding surface. If the feature has an attachment point, then this is automatically the origin of the feature; if the attachment is in the form of a curve or region, then the attachment point is respectively contained in the curve or region. The correspondence between the coordinate systems that are used in feature space and modeling space can be defined with regard to the normal and tangent of the embedding surface; the third axis of the coordinate system is assumed to be given by the user and is named the *direction vector*. In feature space, the surface normal vector is always the vector (0,0,1). The direction vector is considered to be the y-axis of the coordinate system and the x-axis can be computed as the cross-vector of both axes. The same can be done in modeling space: the z-axis of the local coordinate system of a feature in modeling space is the surface normal vector of the embedding surface in the origin of the feature. The direction is a user-given vector that is by definition tangent to the embedding surface in the origin of the feature; the x-axis can then be computed with regard to these vectors.

By controlling the direction vector of a feature, a user can rotate the feature around its z-axis.

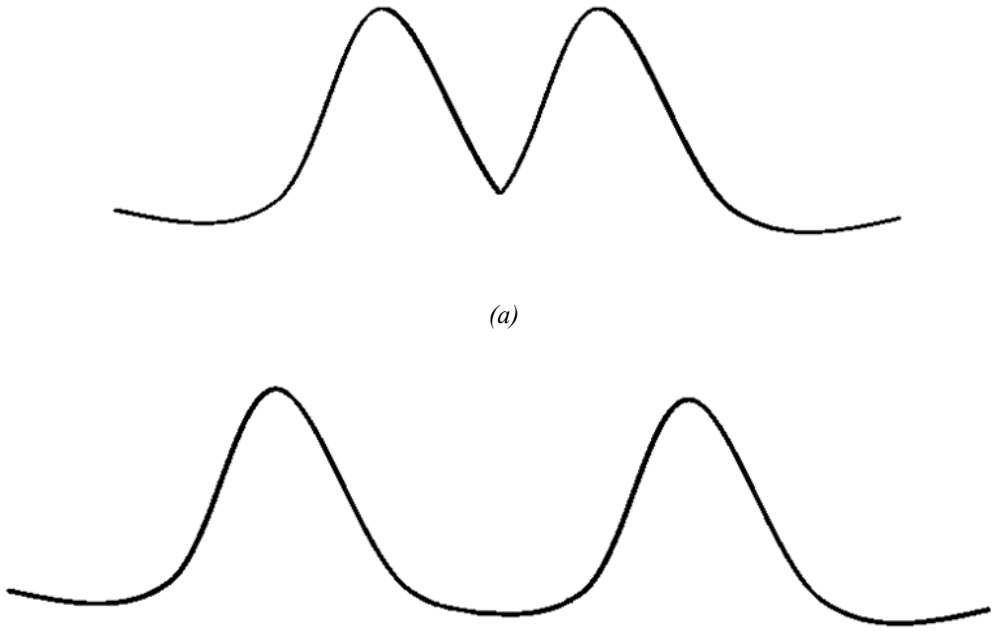
Note that this assumes that the origin of a feature has a well-defined surface normal vector. If this is not the case, then the local coordinate system of a feature cannot be computed as described above. This problem can be solved by moving the feature origin a small amount until the surface normal can be determined. However, this remains a theoretical exercise, as such a situation rarely occurs in the daily practice of industrial design. Solving this problem is left as an open issue.

3.3.4 Feature Interference and Interaction

One of the main problems with the application of feature-based methods is how to deal with interfering features (Regli and Pratt, 1996). Feature interference is ill-defined in current literature; however, a comprehensive theory of freeform features should be able to describe and explain the phenomenon of feature interference. Therefore, we give definitions for the different types of relations between features in this section. On a morphological level, features can be related in two different ways (see Figure 3.9) :

- Both features have the same embedding surface and their influence regions intersect (see Figure 3.9a). This is a case of feature interference: the part of the embedding surface where the areas intersect is influenced by parameters of both features. The two features can be combined in a *compound feature*.
- Both features have the same embedding surface but their influence regions do not intersect. This is a case of *feature interaction*: the morphology of the two

features is related, but there is no point on the embedding surface that is influenced by both features. The two features can be combined in a *pattern feature* (see Figure 3.9b).



(a)

(b)

Figure 3.9*: Examples of different types of relations between features on the morphology level: (a) interfering features and (b) interacting features.*

These two cases can be defined as follows:

Definition 3.9: Definition of feature interference
Feature interference between two features F' *and* F'' *occurs when there is a point* a' *in the area* A' *of* F' *and a point* a'' *in the area* A'' *of* F'' *such that* $a' = a''$.

Definition 3.10: Definition of feature interaction
Feature interaction between two features F' *and* F'' *occurs if there exists an inter-feature relational constraint between the two features.*

Note that two features can both interfere and interact at the same time. In this case the interference takes place in the interference region, while the feature interaction takes place outside the interference region, which can be defined as follows:

Definition 3.11: Definition of the interference region
The interference region $I(F',F'')$ *of two features* F' *and* F'' *is the intersection* $A' \cap A''$ *of their areas.*

When two features interfere or interact, then the status of a shape configuration element is influenced by parameters from both features. However, the influences of more than one parameter can be combined in different ways, depending on the circumstances in which the effect occurs. For example, if a point $e^0 = (0,0)$ in the shape of a feature is displaced by $\mu'(e,p') = e + p'(2,0)$ as well as by $\mu''(e,p'') = e + p''(0,1)$, then the influences of both parameters can be combined in different ways. For example, if the two influences are added, then for $p' = 5$ and $p'' = 3$, it holds that $e = (10,3)$; taking only the largest parameter influence into account gives $e = (10,0)$ (see Figure 3.10).

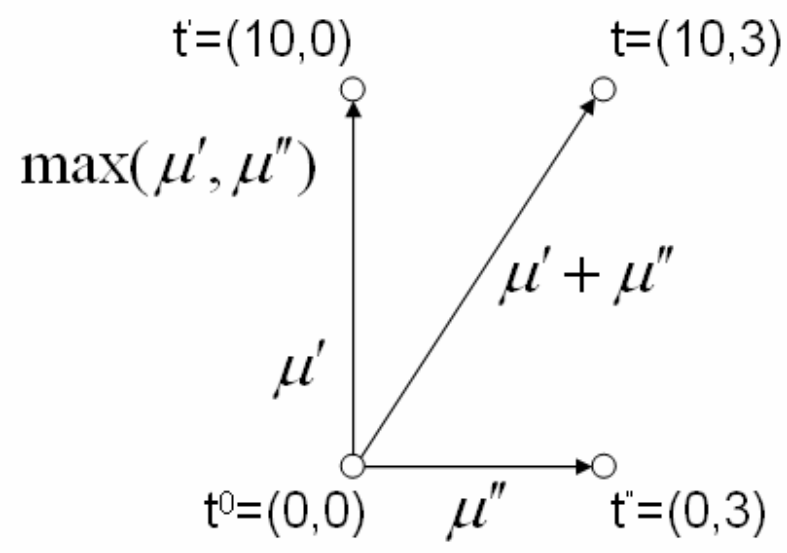


Figure 3.10: *Two-dimensional example of the combination of parameters*

In general, a function is needed in which the method of combination of two feature influences is defined.

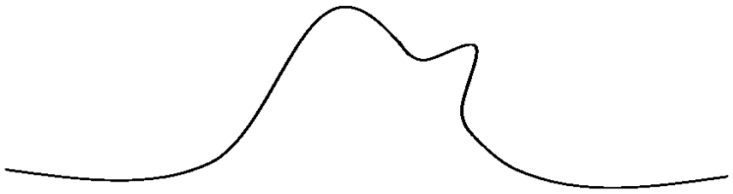
Definition 3.12: Definition of the feature interference combination function
If two features $F'(E'^0, P', \mu')$ *and* $F''(E''^0, P'', \mu'')$ *interfere then a combination function* ψ *exists, that gives the shape representation* E^I *of the interference region* $I(F',F'')$ *as* $E^I = \psi\left(\mu'\left(a, P'^{deform}\right), \mu''\left(a, P''^{deform}\right)\right)$.

In other words, the combination function ψ combines the deformations of an element in the interference region of F' and F''.

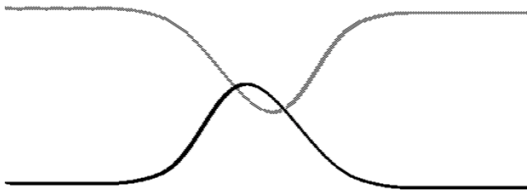
There are two additional ways in which features can be related, but in these cases the relation is on the geometry level, not on a morphology level:

- The shape representation of one feature can (partly) be the embedding surface of another feature (see Figure 3.11a). This is a case of nested features; the shape of a nested feature can only be evaluated after that of the feature on which it is nested.

- The shape representations of two features can intersect without the two features being defined on the same embedding surface (see Figure 3.11b). This is a case of feature intersection.



(a)



(b)

Figure 3.11: *Examples of different types of relations between features on the geometry level: (a) nested features (b) feature intersection*

3.4 Implementation of the foundational theory

To be able to apply the foundational theory in a methodology for freeform feature-based operations, an implementation of the different aspects of the theory must be given. In this section we propose such an implementation and discuss the advantages and shortcomings of the proposed implementation.

3.4.1 Implementation of the freeform feature concept

Recall that in section 3.3.1, the freeform feature was defined as consisting of a basic shape configuration, a vector of parameter values and a parameter mapping. As an implementation of the shape configuration of a feature we use a bidirectional grid of control points. The reasons for doing so are:

- The control points can be used to create different shape representation types; they can for example be used to generate a B-spline surface, a Bezier surface, a Catmull-Clark surface or a polygon mesh.
- There is a strong theoretical and methodological support for control point-based shape representations.
- The resolution of a bi-directional grid of control points can be easily enlarged without modifying the evaluated shape.

Implementation of the shape configuration

The shape configuration of a feature is implemented as k sets of control points, so that $E = \bigcup_k E_k$ with subsets E_k that can be organized in bi-directional grids $E_k = \begin{pmatrix} e_{1,1,k} & \cdots & e_{1,c_k,k} \\ \vdots & \ddots & \vdots \\ e_{r_k,1,k} & \cdots & e_{r_k,c_k,k} \end{pmatrix}$, r_k and c_k being respectively the number of rows and columns of E_k. The reason for implementing the shape configuration as multiple sets of control points is that it allows us to define feature with disjoint shape parts. In the remainder of this thesis, the control points are referred to as *shape configuration elements*.

Initially, we will assume that $k = 1$. Later in this thesis it will be shown how multiple sets of control points can be used to compose a new feature. For the sake of clarity, if $k = 1$, it will be omitted; E will be used short for E_k, and $e_{i,j}$ will be used short for $e_{i,j,k}$. To convert the control points to shape we make use of the weighted de Casteljau's algorithm for degree 3 and with a uniform knot vector. Of course we can provide a user with the ability to modify the degree or the knot vector, but this has no influence on the feature recognition methods proposed later in the thesis and will therefore be left out here.

All control points are given in four-dimensional space in homogenous coordinates as $E_{i,j,k} = \begin{pmatrix} x_{i,j,k} \\ y_{i,j,k} \\ z_{i,j,k} \\ w_{i,j,k} \\ 1 \end{pmatrix}$, where x, y and z are the three-dimensional Cartesian coordinates of the control point and w is its weight. Four-dimensional coordinates allow us to transform both the coordinates and the weight of a control point with a single transformation; expressing these in homogenous coordinates allows us to also use rotations.

Implementation of the parameter mapping

The parameter mapping of features can be implemented as a set of mapping functions $\mu_{i,j}^{l}$ of type $(e, p) \to e$. If the shape configuration of a feature is defined as a grid of *r* by *c* control points and *m* parameters, then a parameter mapping μ is a set of $r \times c \times m$ mapping functions $\mu_{0<i<r,0<j<c}^{0<l<m}$, each of which applies a 5×5 transformation matrix $T_{i,j}^{l}$ to the control point $e_{i,j}$ such that $\mu_{i,j}^{l}(e_{i,j}, p^{l}) = T_{i,j}^{l} e_{i,j}$. The actual state of a control point $e_{i,j}$ can be defined as $\prod_{l=0}^{m} T_{i,j}^{l} e_{i,j}^{0}$. A parameter mapping treats the coordinates of a shape configuration element as a vector and multiplies it with a transformation matrix, resulting in a new coordinate vector. In other words, the feature F maps all functions in μ to the elements of E^0 and results in a new shape configuration E, such that for each shape configuration element $e_{i,j}$ it holds that $e_{i,j} = \mu_{i,j}^{1}\left(\mu_{i,j}^{2}\left(\ldots \mu_{i,j}^{m}\left(e_{i,j}^{0}, p^{m}\right)\ldots, p^{2}\right), p^{1}\right)$. Because the parameter mapping is complete, a mapping function exists for each combination of a shape configuration element and a parameter. This may seem somewhat strict because a parameter does not necessarily influence the entire geometry of a feature. However, to be able to check for validity, we have to also identify this non-influence. If the parameter does not relate to a certain shape configuration element, this information must be visible in the parameter mapping, for example as a zero value.

However, this implementation runs into trouble when the influence of parameter values is also taken into account. For example, if a parameter mapping $\mu_{i,j}^{l}$ imposes the

transformation matrix $T_{i,j}^{l} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ on a point $e_{i,j}$, then this causes a translation of $e_{i,j}$. From the parameter value p^l, the magnitude of the transformation can be determined. However, if the transformation matrix is simply multiplied by p^l, then this results in a matrix $p^l T_{i,j}^{l} = \begin{pmatrix} p^l & 0 & 0 & 0 & p^l \\ 0 & p^l & 0 & 0 & p^l \\ 0 & 0 & p^l & 0 & p^l \\ 0 & 0 & 0 & p^l & p^l \\ 0 & 0 & 0 & 0 & p^l \end{pmatrix}$. This matrix does not only impose a translation effect, but an unwanted scaling effect has also been introduced. Instead, we want a multiplication by the parameter value p^l to lead to $p^l T_{i,j}^{l} = \begin{pmatrix} 1 & 0 & 0 & 0 & p^l \\ 0 & 1 & 0 & 0 & p^l \\ 0 & 0 & 1 & 0 & p^l \\ 0 & 0 & 0 & 1 & p^l \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$. The parameter value should in this case only have an effect on the translation components of the matrix, not on the diagonal components of the matrix. As a solution to this problem, two additional matrices are introduced: the *direction matrix* δ, which determines the direction of the influence of the transformation matrix and the *function mask matrix* φ, which determines the type of influence. These are not algebraic matrices in the sense that they can be added or multiplied. A function mask matrix is a matrix of functions that take the parameter value and an element of the direction matrix as an input. The translation matrix can be computed as:

$$T_{i,j}^{l} = \begin{pmatrix} \varphi_{11}(p^l, \delta_{11}) & \varphi_{12}(p^l, \delta_{12}) & \varphi_{13}(p^l, \delta_{13}) & \varphi_{14}(p^l, \delta_{14}) & \varphi_{15}(p^l, \delta_{15}) \\ \varphi_{21}(p^l, \delta_{21}) & \varphi_{22}(p^l, \delta_{22}) & \varphi_{23}(p^l, \delta_{23}) & \varphi_{24}(p^l, \delta_{24}) & \varphi_{25}(p^l, \delta_{25}) \\ \varphi_{31}(p^l, \delta_{31}) & \varphi_{32}(p^l, \delta_{32}) & \varphi_{33}(p^l, \delta_{33}) & \varphi_{34}(p^l, \delta_{34}) & \varphi_{35}(p^l, \delta_{35}) \\ \varphi_{41}(p^l, \delta_{41}) & \varphi_{42}(p^l, \delta_{42}) & \varphi_{43}(p^l, \delta_{43}) & \varphi_{44}(p^l, \delta_{44}) & \varphi_{45}(p^l, \delta_{45}) \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

To obtain the transformation matrix that was mentioned earlier, the direction matrix and function mask matrix can be given as:

$$\delta^{l}_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \varphi^{l}_{i,j} = \begin{pmatrix} const() & const() & const() & const() & \text{var}() \\ const() & const) & const() & const() & \text{var}() \\ const() & const() & const() & const() & \text{var}() \\ const() & const() & const() & const() & \text{var}() \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

where $const(p,x) = x$ and $\text{var}(p,x) = px$. Using two additional matrices, the influence of a parameter can be defined to be different for different matrix elements. Because the influence of parameters is already captured in the transformation matrix, the parameter values can be left out of the definition of parameter mappings.

The direction and function mask matrix are not only useful for defining a translational effect; together, they are a powerful method of defining transformation effects. To demonstrate this, two additional examples of direction and function mask matrices will be given.

Rotational transformations are important in the process of feature recognition, as will become clear in the next chapter. Parameter mappings with a rotational effect can be obtained by using the appropriate functions in the function mask matrix of the mapping. For example, the following configuration of both matrices controls a rotation around the x-axis:

$$\delta^{l}_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \phi^{l}_{i,j} = \begin{pmatrix} const & const & const & const & const \\ const & s & c & const & const \\ const & -c & s & const & const \\ const & const & const & const & const \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where $const(p,x) = x$, $c(p,x) = \cos(px)$ and $s(p,x) = \sin(px)$.

From these two matrices, the following transformation matrix can be computed:

$$T^l_{i,j} = \begin{pmatrix} const\left(p^l,1\right) & const\left(p^l,0\right) & const\left(p^l,0\right) & const\left(p^l,0\right) & const\left(p^l,0\right) \\ const\left(p^l,0\right) & \sin\left(p^l,1\right) & \cos\left(p^l,1\right) & const\left(p^l,0\right) & const\left(p^l,0\right) \\ const\left(p^l,0\right) & -\cos\left(p^l,1\right) & \sin\left(p^l,1\right) & const\left(p^l,0\right) & const\left(p^l,0\right) \\ const\left(p^l,0\right) & const\left(p^l,0\right) & const\left(p^l,0\right) & const\left(p^l,1\right) & const\left(p^l,0\right) \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \sin\left(p^l\right) & \cos\left(p^l\right) & 0 & 0 \\ 0 & -\cos\left(p^l\right) & \sin\left(p^l\right) & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Another function mask matrix is needed to define a helical or spiral effect, which is for example important when defining or recognizing features with a shape that resembles a screw thread. In this case, not only the rotational effect has to be implemented, but also the rate of vertical displacement which can be controlled linearly. A helical effect around the z-axis would be given by:

$$\delta^l_{i,j} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \phi^l_{i,j} = \begin{pmatrix} \mathrm{c} & -\mathrm{s} & const & const & const \\ \mathrm{s} & \mathrm{c} & const & const & const \\ const & const & \mathrm{var} & const & \mathrm{var} \\ const & const & const & \mathrm{var} & const \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The effect of a transformation can be split over multiple parameters. In the example of a helical effect, vertical displacement and rotation can be alternatively defined by using two different parameters. By coupling these two parameters through an intra-feature parameter constraint, an effect identical to that of the direction and function mask matrices given above is still achieved, but the relative magnitude of the vertical displacement compared to the amount of rotation can be controlled as well.

3.4.2 Definition of area, deformation and weight parameters

As was mentioned earlier, we can distinguish two types of parameters: area and deformation parameters. Now that we have introduced an implementation of the freeform feature, we can define these two parameter types in more detail.

The notion of area parameter can best be understood in feature space. In feature space, area parameters cause a horizontal displacement of a control point, i.e. all parameter mapping functions have zero influence on the z-coordinate of a control point. If only the area parameters of a feature are evaluated, then the resulting shape configuration is called the *area configuration*. Note that the area configuration is not equivalent to the influence region. The influence region is the region of the embedding surface that changes as a result of the feature, where the area configuration is a sub-region of the influence region for which a morphology is dictated by the feature.
In contrast, the configuration of a feature in which all parameters are evaluated is called the *full configuration.* Because the area parameters do not cause any vertical displacement, the control points of an area configuration lie on the embedding surface of a feature: the xy-plane. The corresponding shape representation of the feature is the area of the feature and is defined as $A(u,v) = \dfrac{\sum_{i=1}^{r}\sum_{j=1}^{c} N_{i,3}(u)N_{j,3}(v)w_{i,j}^{area}(x_{i,j}^{area}, y_{i,j}^{area}, z_{i,j}^{area})}{\sum_{i=1}^{r}\sum_{j=1}^{c} N_{i,3}(u)N_{j,3}(v)w_{i,j}^{area}}$,

where $e_{i,j}^{area} = \left(x_{i,j}^{area}, y_{i,j}^{area}, z_{i,j}^{area}, w_{i,j}^{area}\right) = \sum_{p^l \in P^{area}} T_{i,j}^{l} e_{i,j}^{0}$, and P^{area} is the set of area parameters.

In other words, the area of a feature is the weighted de Casteljau's algorithm applied to the area configuration.
Deformation parameters control the transformation of a control point that is not parallel to the embedding surface. Once the area of a feature has been computed, the deformation parameters define a transformation away from the embedding surface.
A third type of parameter can be distinguished based on the type of transformation it enacts: the weight parameter. By modifying the weight of control points, a shape can be manipulated without modifying the spatial coordinates of any control point. A parameter p^l is a weight parameter when for all function mask matrices $\varphi_{i,j}^{l}$ it holds that

$$\varphi_{i,j}^{l} = \begin{pmatrix} const() & const() & const() & const() & const() \\ const() & const() & const() & const() & const() \\ const() & const() & const() & const() & const() \\ const() & const() & const() & const() & \text{var}() \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.4.3 Definition of generic, specific and basic parameters

As was demonstrated in the previous section, parameters can be defined that apply a rotational transformation on the control points of a feature. With these parameters, the orientation of a feature can be manipulated. Likewise, parameters can be defined with which the location of a feature can be manipulated. To give a user optimal control over the location and orientation of a feature, we state that these parameters must be defined for any feature type. We call these parameters basic parameters and in the remainder of this thesis we require feature type definitions to include parameters that control the location and orientation of a feature: similar to those given for the x-rotation in section 3.4.1, direction and function mask matrices can be given for the y- and z-rotation, as well as for the x-, y- and z-translation. With the use of these parameters, any rigid body movement can be defined for the feature. Because we want to be able to parametrically control any operation that has an influence on the shape representation of a feature, we express rotation and translation in the form of parameters instead of treating them as non-parametric operations on the feature shape.

What basic parameters are defined for a feature depends on whether it is a floating feature or an attached feature. Floating features, having no attachment to any surface, can be rotated and translated freely. For a six degree-of-freedom motion, six basic parameters must be defined: x-, y-, and z-rotation and x-, y-, and z-translation. For attached features a rotation around the x- or y-axis has no meaning, as the feature is attached to its embedding surface and its orientation is determined by that surface. Rotation around the z-axis of the local coordinate system of the feature (i.e. the normal of the attachment point) does make sense, as it determines the orientation of the feature on the embedding surface. At the same time, the z-translation does not make sense for attached features, as it would move the feature away from its embedding surface. X- and y-translations do make sense, as these can be used to determine the position of the feature on the embedding surface, relative to the attachment point. This is depicted in Figure 3.12.

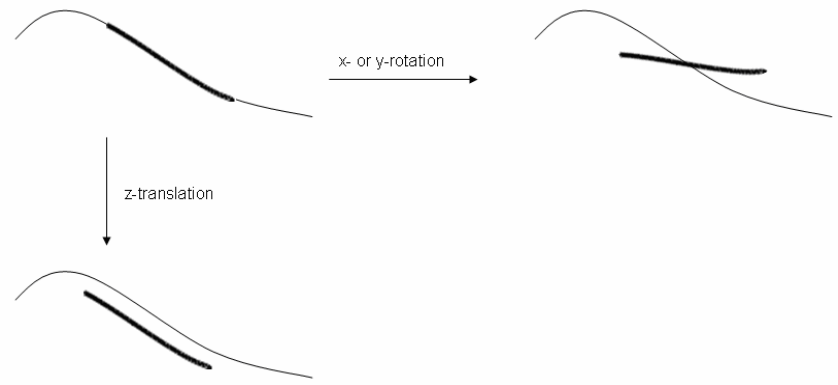


Figure 3.12*: Problems that occur when applying some default parameters for attached features*

As a result, only three basic parameters are considered for attached features: z-rotation, x-translation and y-translation.

Another distinction can be made between generic parameters and specific parameters. Generic parameters are parameters that have a similar effect on all shape configuration elements, i.e. for which it holds that $\delta^l_{i,j} = \delta^l_{i',j'}$ and $\varphi^l_{i,j} = \varphi^l_{i',j'}$ for all $0 < i, i' < r$ and $0 < j, j' < c$.
A parameter that is not generic is said to be a specific parameter. Logically, specific parameters can have a different influence on individual shape representation elements.

3.4.4 Definition of embedded features and corresponding features

When a feature is attached, this means that its shape representation corresponds to a portion of its embedding surface. However, up till now we have not claimed that the shape representation of a feature is a subsection of the geometry of its embedding surface, merely that there is a correspondence without specifying the nature of the correspondence. Here, we distinguish between *embedded features* and *corresponding features*. An embedded feature is a feature of which the shape representation is a subsection of the geometry of its embedding surface. A corresponding feature is a feature of which the shape representation corresponds with the geometry of its embedding surface through a *correspondence function* χ. Note that the distinction between embedded features and corresponding features can be made only when a feature has been instantiated on an embedding surface.
Figure 3.13 shows the difference between an embedded and a corresponding feature. The need for distinguishing corresponding and embedded feature can be explained by looking ahead at the feature recognition problem: in this problem, embedded features cannot be used to recognize a target feature, because the embedding surface of the target feature is not known. This argument will be given in more detail in the next chapter.

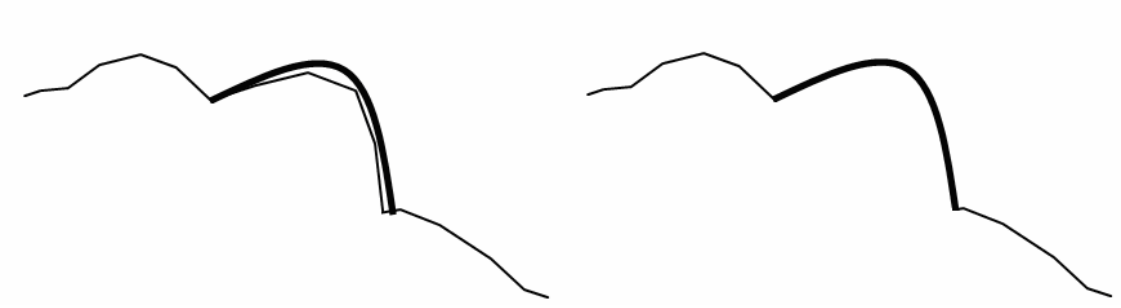


Figure 3.13: *Two-dimensional clarification of the difference between an embedded (right) and a corresponding feature (left) on a polygonal embedding surface*

Embedded and corresponding features each have their advantages and shortcomings:

Instantiation

The advantage of a corresponding feature is that its instantiation is efficient and that it leaves the geometry of the embedding surface intact. If a feature is instantiated as a corresponding feature on an embedding surface, then a correspondence function must be constructed between the geometry of the feature and the geometry of the embedding surface.

The disadvantages of an embedded feature are that its instantiation is inefficient, that the geometry of its embedding surface is changed and that part of the geometry of the embedding surface is lost (or must be stored somewhere). If a feature is instantiated as an embedded feature, then part of the geometry of the embedding surface must be removed and replaced by the geometry of the feature. A transition must be made between the geometry of the feature and the remaining geometry of the embedding surface. In other words, if an embedded feature is created, then a hole is cut into the embedding surface, and the feature is glued into this hole. The advantage of embedded features is that the transition between a feature and its embedding surface (e.g. with regard to the smoothness of the transition) can be managed independently from the geometry of the feature.

Relation to the embedding surface

In the case of a corresponding feature, the shape representation type of a feature is independent of that of the embedding surface. For example, if the embedding surface is a polygon mesh or a point cloud, then it is possible for the feature to have a B-spline representation as long as there is a suitable correspondence function.

In the case of an embedded feature, the geometry of a feature becomes part of that of its embedding surface and is in most applications required to be of the same shape representation type as its embedding surface. For example, a feature that is represented as a B-spline cannot be instantiated on a polygon mesh without changing the shape representation type of either the feature or the embedding surface.

Feature manipulation

The disadvantage of corresponding features is that they can only be used to manipulate their embedding surface through the correspondence function defined for the feature. This makes the manipulation of corresponding features less efficient than that of embedded features, for which this problem does not occur. In addition, the correspondence function must be reconstructed whenever the geometry of the embedding surface changes.

If the area configuration of the feature changes as a result of a feature manipulation, then in the case of a corresponding feature, the correspondence function must be adapted, regardless of whether the area configuration has been made smaller or larger. In the case of an embedded feature, it is much more difficult to adapt to a changed feature area. If the area of an embedded feature surface grows larger, then additional geometry must be removed from the embedding surface; if the area grows smaller, then some of the original geometry of the embedding surface must be restored.

Deletion of a feature
If a corresponding feature is deleted, then this can be done without changing the geometry of the embedding surface. However, if an embedded feature is deleted, then this leaves a hole: the original embedding surface (as it was before the feature was instantiated) must be restored. Therefore, the deletion of a corresponding feature is much more efficient than the deletion of an embedded feature.

3.4.5 Examples of feature definitions

In Figure 3.14, an example is given of a 'Bump' feature. In order to demonstrate the different aspects of the proposed implementation, several instances of the same feature type are shown. Although the morphology of the feature cannot be graphically shown, by showing different instances of the Bump feature, an impression of its morphology can nonetheless be given. The instances of the bump feature shown are in the form of embedded features on a flat plane.
Figure 3.14 shows not only the shape representation of the feature but also its shape configuration, i.e. its control point grid. The control points are visible as small rectangular boxes.
The Bump feature was defined using a grid of 5×5 control points. In the basic shape configuration of the feature, all these control points coincide in the origin of its local coordinate system, i.e. in the attachment point of the feature. One area parameter, the Radius parameter, has been defined for the feature, which moves the control points outward from the attachment point. Figure 3.15b shows a configuration of the feature in which the value of this parameter has been modified with respect to the feature in Figure 3.15a. One deformation parameter has been defined for the feature: the Height parameter, which only has an influence on the central point, which it translates vertically, i.e. its transformation matrix only has a z-component. Figure 3.15a shows a feature with an increased value of the height parameter. Finally, there are two weight parameters, the Top Roundness and Bottom Roundness parameter. The Top Roundness parameter influences the weight of the central control point only, such that a high value for this parameter causes the bump to be very pointy. Figure 3.15c shows an instance of the bump feature with an increased value for the top roundness parameter.

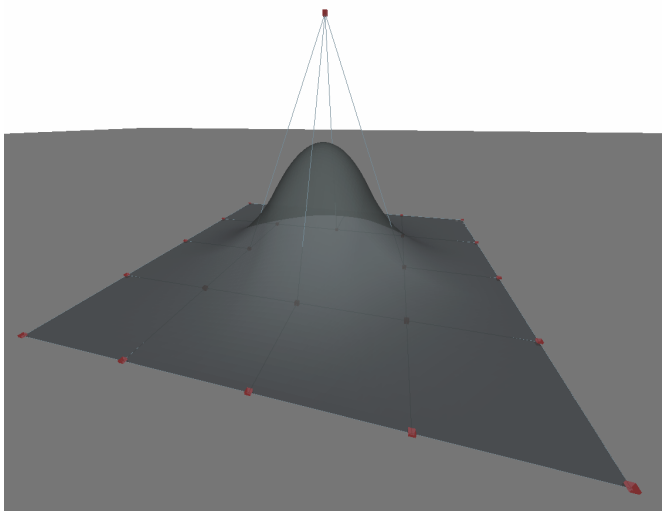
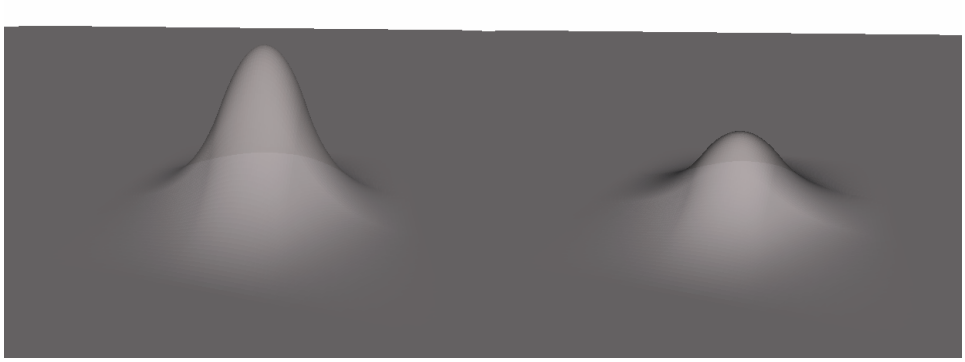


Figure 3.14: The shape configuration of an instance of the Bump feature



(a) *(b)*

(b) *(d)*

Figure 3.15*: Various instances of the bump feature with different parameter values*

The Bottom Roundness parameter influences the weight of the control points immediately adjacent to the central control point. A low value for this parameter causes

the transition between the bump and its embedding surface to be smooth; a high value causes the transition to be abrupt. Figure 3.15d shows an instance of the Bump feature with a large value for the bottom roundness parameter.

3.5 A computational model for feature-based operations

In section 3.3 a theory for the freeform feature concept was given. On the basis of this theory, in this section we give a computational model for feature-based operations. In such a computational model, a set of allowable operations on the freeform feature concept is given, combined with a model of the information flow. Because the process of feature-based modeling falls outside the scope of this thesis, we will focus on feature recognition and the processes that play a role prior to, parallel to or after a feature recognition procedure.

3.5.1 Feature type definition

The starting point for the instantiation of a feature often lies in the feature library, which is a set of pre-defined feature types. The benefit of maintaining such a set of pre-defined feature types is that functional, parametric, semantic and other information can be stored with a certain feature beforehand. This means that the definition of feature types can be done by an expert instead of by an end-user of the feature; in addition, much-used features only have to be defined once. However, as was already discussed in chapter 2, in some applications user-defined features are needed. For this reason, a user must be able to customize the feature library with user-defined features. The concept of the feature library is therefore coupled to a feature definition method.
If a new feature is instantiated, then this feature is a copy of one of the feature types in the feature library. By defining and instantiating features in separate procedures, both can be kept independent, efficient and simple. A new feature type can be defined in three ways: by defining it from scratch, by composing a new feature type definition from two or more existing definitions or by manipulating the parameter mapping and basic shape configuration of an existing feature (see Figure 3.16).

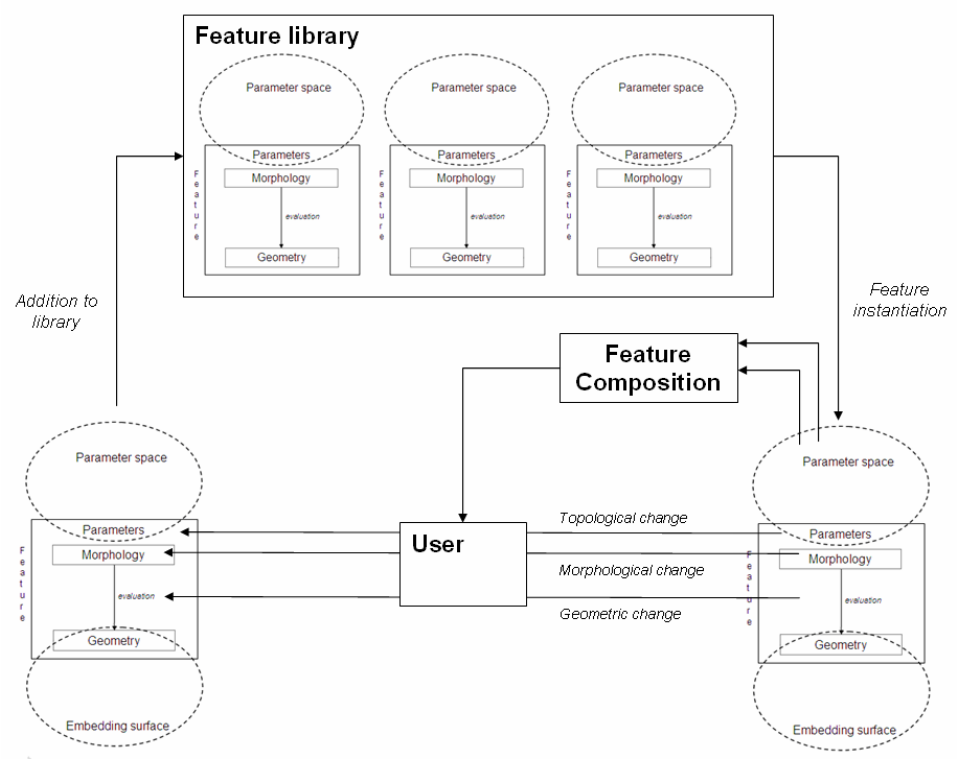


Figure 3.16: *Computation model for feature type definition*

3.5.2 Feature instantiation

Instantiating a feature on an embedding surface is a conceptually very simple task. However, computationally spoken, feature instantiation is a difficult and complex process. As was discussed earlier, it makes a big difference whether a feature is an embedded feature or a corresponding feature, but in both cases similar mechanisms exist, which can therefore generally be used in the process of feature instantiation. In both cases, the first step in instantiating a feature is determining the area configuration of the feature. In the case of an embedded feature, the area configuration determines the extent to which material must be removed from the embedding surface. In the case of corresponding features, the area configuration determines the extent to which a correspondence function must be constructed.

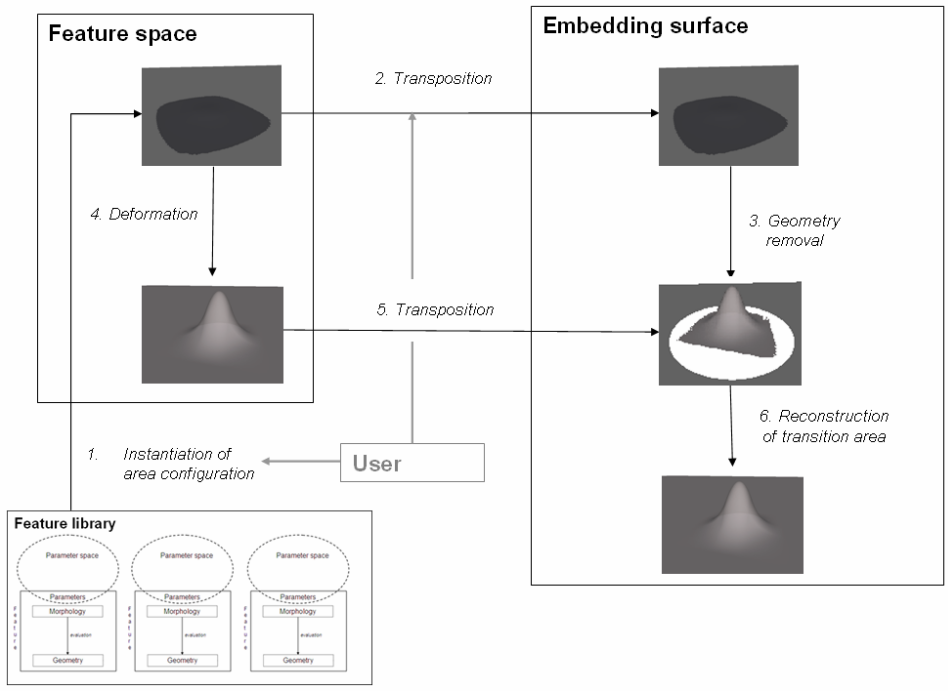


Figure 3.17*: Computational model for the instantiation of an embedded feature*

In Figure 3.17, a computational model for the instantiation of an embedded feature is given. In this figure, it is shown that a feature can first be instantiated in feature space. In this instantiation procedure, an area configuration is computed from the basic shape configuration, the parameter mapping defined for the feature and a vector of parameters given by the user. It is assumed that an attachment point, curve or region has been provided by the user. The geometry in the part of the embedding surface that corresponds to the area configuration of the feature is then removed and stored so it can later be used to restore (part of) the embedding surface as discussed in section 3.4.4. Then, the shape configuration of a feature in feature space is deformed, i.e. the deformation parameters are evaluated with regard to the area configuration. This shape configuration of the feature can then be transposed to the embedding surface, i.e. adapted to the local conditions of the embedding surface at the point of attachment. In a final step, the transition region between the feature and the embedding surface is reconstructed.

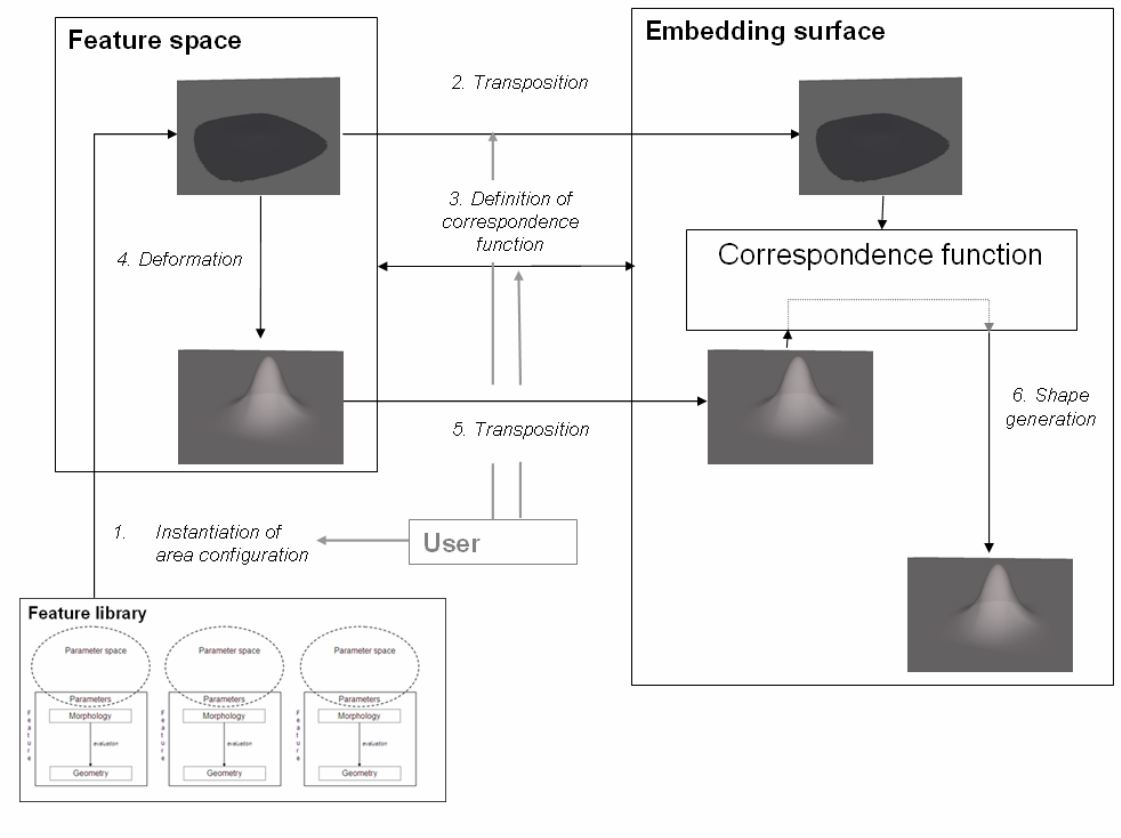


Figure 3.18: *Computational model of the instantiation of a corresponding feature*

In Figure 3.18, a computational model for the instantiation of a corresponding feature is given. It resembles Figure 3.17 a great deal, but is different mainly in two aspects: First, instead of generating the feature shape on the embedding space, it is dependent on the correspondence function defined by the user. Second, no reconstruction of the transition region is needed.

3.5.3 Transposition from feature space to modeling space

Although the computational models given in the previous section specify the different steps that are needed to instantiate a feature on an embedding surface, the actual transposition from the shape configuration of a feature from feature space to modeling space was assumed to be given. In this section, we will give a computational model for this transposition process.

To be able to translate a feature from feature space to modeling space, the shape configuration of the feature must be adapted to the local conditions of the embedding surface. In other words, the geometric properties of the embedding surface are used to modify the basic shape configuration and parameter mapping of the feature. The parameter values of the feature do not change. Figure 3.19 shows a computational model

for the transposition of a feature from feature space to modeling space, where M1 and M2 are the transformation matrices which are needed to transform the feature in feature space with respect to its local coordinate system to the feature in modeling space with respect to its local coordinate system in modeling space. The user has an influence on how this transformation takes place, and therefore also has an influence on M1 and M2.

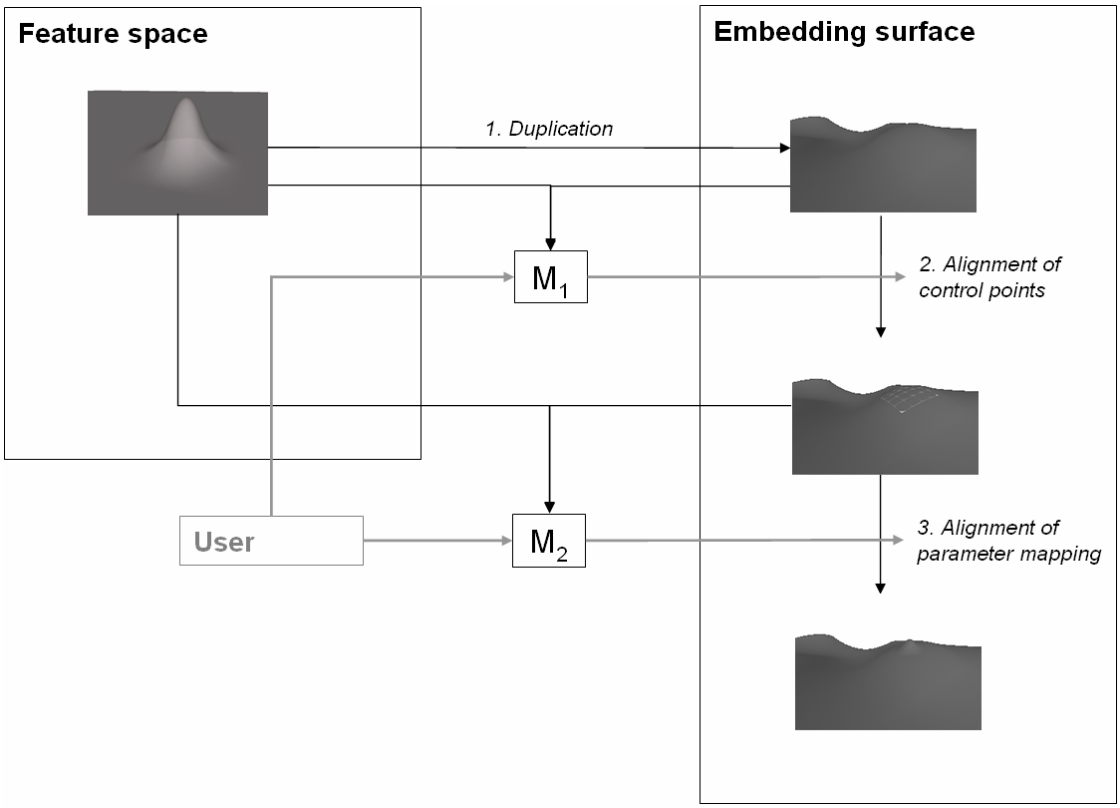


Figure 3.19*: Computational model of the transposition from feature space to modeling space*

Although the computational model given here is directed at the transposition of features from feature space, it is in fact a computational model for copying a feature from any embedding surface to another surface. Although features in feature space have a simplified embedding surface, i.e. the xy-plane, this surface is as valid as any other surface. The computational steps for feature copying are therefore the same as for translating a feature from feature space, although the methodology may be slightly different.

The first step in the computational model is a direct duplication of the feature instance in feature space. That is, a copy of the feature in feature space is made in modeling space without changing its location or orientation. Then, a transformation matrix is computed that aligns the local coordinate system of the feature with the local coordinate system at the attachment point. The user plays a role here in determining a direction vector for the

feature. This transformation matrix is then used to align the feature at the attachment point, i.e. to adapt its location and orientation to the embedding surface at the attachment point. In the third step, a transformation matrix is computed that modifies the parameter mapping of the feature such that it corresponds to the local conditions of the embedding surface.

3.5.4 Feature recognition

The main topic in this thesis is feature recognition. Many problems exist in connection with the problem of feature recognition. Now that computational models have been defined for these problems in the previous sections, we can turn to defining a computational model for the problem of feature recognition. Feature recognition can be roughly defined as the recognition of a part of a target surface as an instance of a pre-defined feature (of course this will be discussed in more detail in the next chapter)

Figure 3.20 shows a computational model of feature recognition. The process of feature recognition is fairly simple: in the first step a region of interest is selected, i.e. a region on the target surface is determined in which the feature to be recognized must lie. In the second step, the type of the feature to be recognized is identified. In both these processes, the user can be involved, but they can also be performed automatically.

Once the feature type and the region of interest have been determined, the feature recognition procedure enters a loop. First, the identified feature type is instantiated in feature space and the resulting feature instance is translated to modeling space. There it is compared to the target surface to determine whether it is an acceptable result of the feature recognition procedure. If it is not, then based on an analysis of the feature either its parameter configuration can be altered or its parameter mapping can be changed. In both cases, a modification is made to the instance in feature space. Calculations on the feature instance can de done here, such as a validity check or a detection of feature interference. Then, to maintain the validity of the correspondence between the copy of the feature in feature space and the copy in modeling space it is again translated to modeling space and the cycle is repeated until an acceptable solution to the feature recognition problem has been found.

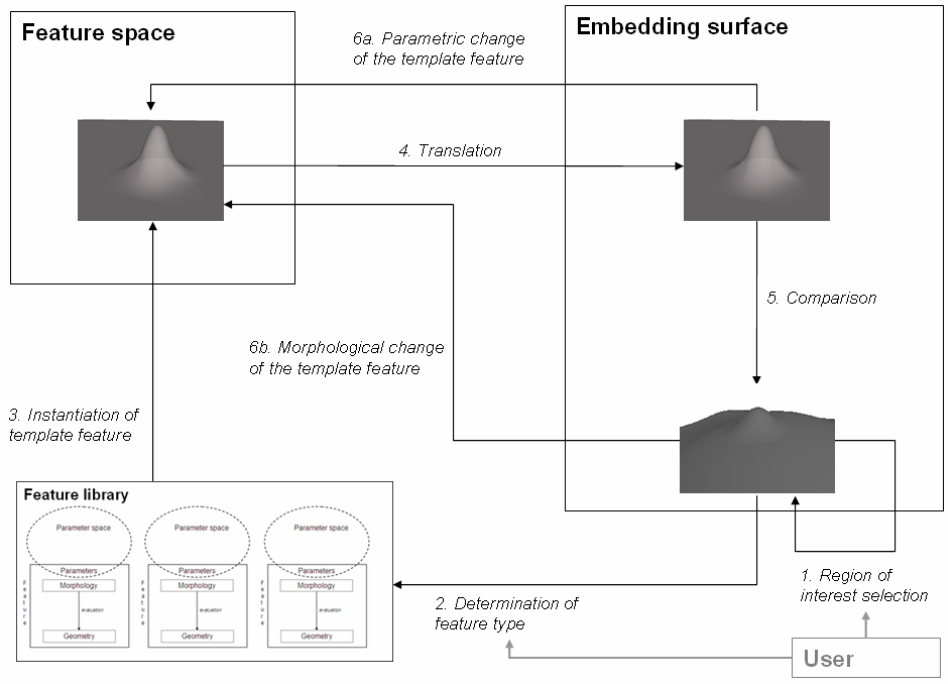


Figure 3.20: *Computational model of feature recognition*

4 Algorithms and implementation

The theory presented in the previous chapter forms a formal basis for several feature-based applications. Because the scope of this thesis is limited to the problem of freeform feature recognition, algorithms will be given specifically for the applications for which a computational model was given in the previous section. Not all issues regarding feature definition and instantiation will be addressed here: only those issues that are relevant for the support of the feature recognition methods that will be given later in the thesis are discussed. In this chapter, algorithmic details will be given. In section 4.1, the definition and instantiation of a feature and its transposition from feature space to modeling space will be addressed. In section 4.2, an algorithm will be given for the recognition of freeform features that makes use of the concept of evolutionary computation. In section 4.3, an algorithm will be given for a freeform feature method that uses multidimensional parameters.

4.1 Algorithms for the definition, instantiation and transposition of freeform features

As was stated in the previous chapter, the definition of a new feature type can either be done from scratch, by changing an existing feature type definition, or by combining two or more existing definitions. Defining from scratch or changing an existing definition can be done by specifying values for each element of the feature type definition, i.e. a basic shape configuration and a parameter mapping. This process needs user input, but no algorithmic support. However, for the combination of two or more feature type definitions an algorithm is given in section 4.1.1.

The concept of feature space that was introduced in the previous chapter can be used to reason about features in a simplified environment. This concept is powerful in that it allows all calculations to be done without having to take into account the local conditions of the surface in which the feature has been embedded. However, it can only be used in modeling space if there is a transposition mechanism between feature space and modeling space.

In sections 4.1.2 to 4.1.5, algorithms are given for the instantiation of a feature and the transposition of a feature instance in feature space to a feature instance in modeling space. In section 4.1.2, an algorithm is given that transposes a feature from feature space to modeling space by adapting its control points and its parameter mapping to the local conditions of the embedding surface. In section 4.1.3, an algorithm is given for increasing the resolution of a feature; this algorithm is needed in the sections 4.1.4 and 4.1.5. In

section 4.1.4 and section 4.1.5, in an instantiation procedure, either the geometry of the feature is made a part of the geometry of the embedding surface, or a correspondence function between the two is constructed.

4.1.1 Feature composition

One method to create complex feature type definitions is combining existing features into a compound feature. The compound feature can then be controlled as a single entity, and will behave as a user would expect based on the individual features it is composed of. Two options exist for combining features into a compound feature: either an entirely new feature is created in which the shape representation elements, parameters and parameter mappings of the contributing features are merged, or the original feature definitions remain intact and the elements of the compound feature still refer to these definitions. The latter case is merely a case of formalizing the relation between thee two features, e.g. in defining a combination function for the region in which the two features interfere. In this case, any operation on a parent feature is rerouted to its child features (see Figure 4.1). In this section we give an algorithm for the combination of two features in one feature type. In this case, the definition of a compound feature is an internalization of the concept of feature interference, in which the shape of the interference region is not the result of two individual features, but of two parts of the same feature. In other words, the interference region can itself be parameterized on the basis of the two or more contributing features.

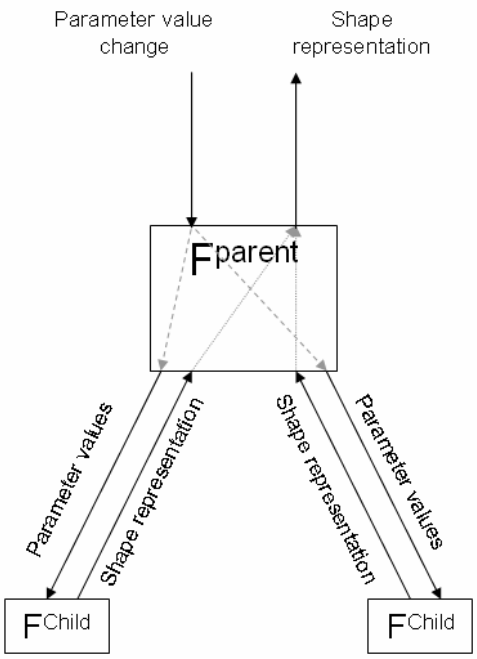


Figure 4.1: *Rerouting of the change of the parametric configuration of a parent feature*

When combining the two features $F'\left(E'^0, P', \mu'\right)$ and $F''\left(E''^0, P'', \mu''\right)$ in a new feature $F\left(E^0, P, \mu\right)$, the new feature can be constructed by separately combining E^0, P and μ. Recall that E^0 is the basic shape configuration, P is the vector of parameter values and μ is the parameter mapping of the feature.

Combining E'^0 and E''^0 into E^0 is a simple union operation. The new set of control points E^0 is a union of E'^0 and E''^0 and consists of $k^E = k^{E'} + k^{E''}$ subsets, such that $E^0_h = E'^0_h$ for $1 < h < k^{E'}$ and $E^0{}_h = E''^0_{h-E'}$ for $k^{E'} < h < k^{E'} + k^{E''}$. Hence, no new shape configuration elements have to be created.

The parameter sets P' and P'' are combined into a new parameter set P. There are two possibilities: either a parameter from P' or P'' is also contained in P, or one or more parameters $p' \in P'$ and one or more parameters $p'' \in P''$ are combined into a new parameter $p \in P$. There are many possible ways to combine parameters from both P' and P'' into a single parameter p, but not all of these combinations are meaningful in that they correspond to the intention of a user. To combine two parameter sets, user input is therefore needed. Although theoretically any algebraic combination of two parameter values is possible, we assume that the combination of two parameters is linear.

The combination of parameter mappings is paired to the combination of the parameters. When two parameters p' and p'' have been combined in the parameter p, then the parameter mapping μ^l is a combination of μ'^l and μ''^l in such a way that $\mu^l_{i,j} = \mu'^l_{i,j}$ if $e^l_{i,j} \in E'$, and $\mu^l_{i,j} = \mu''^l_{i,j}$ if $e^l_{i,j} \in E''$. However, for points of E that lie in the interference region $I\left(F', F''\right)$, a function ψ must be created that defines how the two parameter mappings are combined. This function is indicated with the same symbol as the function for feature interference because, as was mentioned earlier, the two are closely connected.

The following algorithm can be used to compute the composed shape of the feature:

Algorithm 4.1: Computing the shape of a compound feature(F', F'', ψ)

F' and F'' are the features to be combined.

ψ is the combination function.

1. The area configurations A' and A'' are computed
2. If $A' \cap A'' = \varnothing$, then $S = S' \cup S''$
3. Otherwise, for each point in $A' \cap A''$, the uv-positions (u', v') and (u'', v'') of the point in A' and A'' are determined.

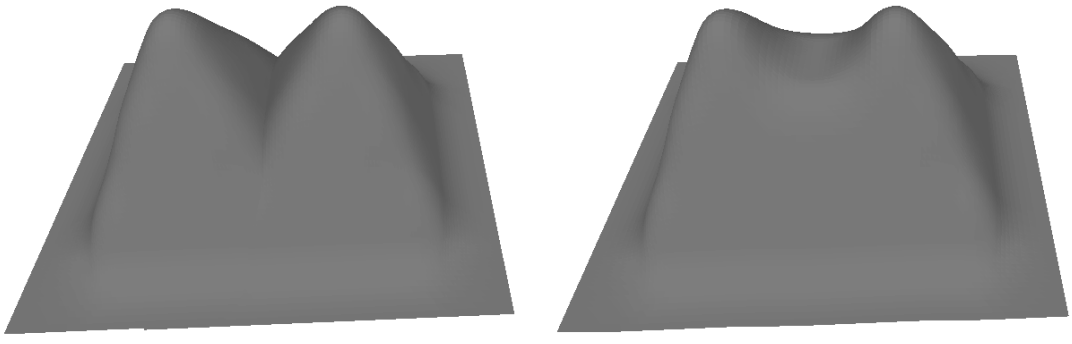
4. Denoting the final shapes of F' and F'' as respectively S' and S'', the final shape of F is defined as:

$$S = \begin{cases} S' & \text{if } A'(u',v') \in A' - A'' \\ \psi(u,v,F',F'') & \text{if } A'(u',v') = A''(u'',v'') \in A' \cap A'' \\ S'' & \text{if } A''(u'',v'') \in A'' - A' \end{cases}$$

In words, when the areas of two features overlap, then the non-overlapping parts of the shape can simply be copied to the shape of the compound feature, but for the overlapping part a new shape must be computed, making use of the combination function ψ. As an example, Figure 4.2 shows a compound feature that is composed of two bumps. The two bumps are combined using respectively the maximum function and an interpolation function. In practice we found that these two functions are the only functions that lead to useful result, but designers are able to choose any combination function they desire.

$$\psi(u,v,F',F'') = \max\left(\frac{\sum_{i=1}^{r_1}\sum_{j=1}^{c_1} N_{i,3}(u)N_{j,3}(v)w_{i,j,1}(x_{i,j,1},y_{i,j,1},z_{i,j,1})}{\sum_{i=1}^{r_1}\sum_{j=1}^{c_1} N_{i,3}(u)N_{j,3}(v)w_{i,j,1}}, \frac{\sum_{i=1}^{r_2}\sum_{j=1}^{c_2} N_{i,3}(u)N_{j,3}(v)w_{i,j,2}(x_{i,j,2},y_{i,j,2},z_{i,j,2})}{\sum_{i=1}^{r_2}\sum_{j=1}^{c_2} N_{i,3}(u)N_{j,3}(v)w_{i,j,2}} \right)$$

$$\psi(u,v,F',F'') = \frac{\sum_{i=1}^{r_1}\sum_{j=1}^{c_1} N_{i,3}(u)N_{j,3}(v)w_{i,j,1}(x_{i,j,1},y_{i,j,1},z_{i,j,1}) + \sum_{i=1}^{r_2}\sum_{j=1}^{c_2} N_{i,3}(u)N_{j,3}(v)w_{i,j,2}(x_{i,j,2},y_{i,j,2},z_{i,j,2})}{\sum_{i=1}^{r_1}\sum_{j=1}^{c_1} N_{i,3}(u)N_{j,3}(v)w_{i,j,1} + \sum_{i=1}^{r_2}\sum_{j=1}^{c_2} N_{i,3}(u)N_{j,3}(v)w_{i,j,2}}$$

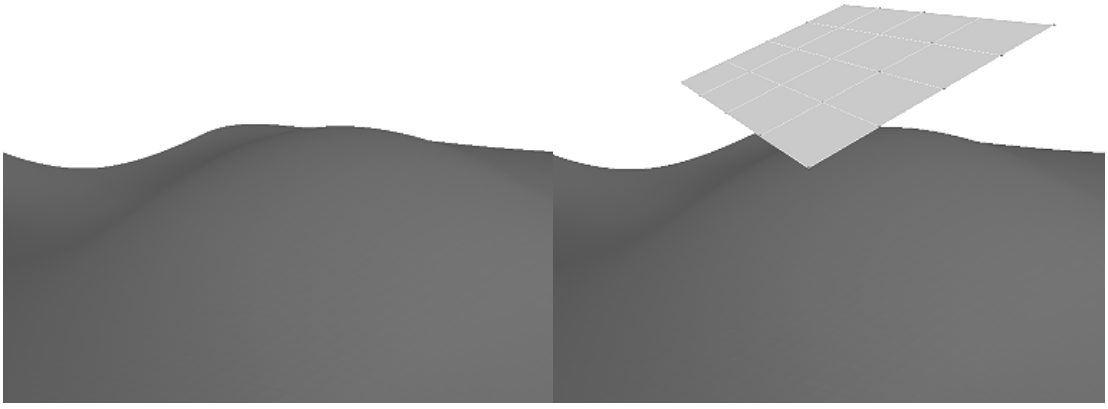


(a) *(b)*

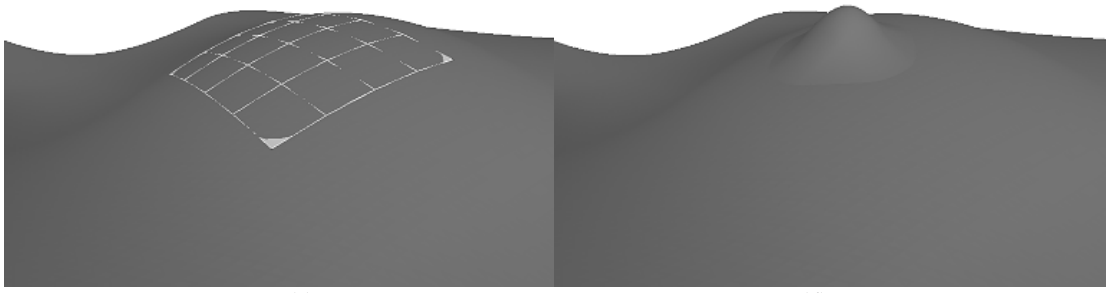
Figure 4.2*: Two bumps combined with (a) the maximum function (b) an interpolation function*

4.1.2 Transposition of a feature

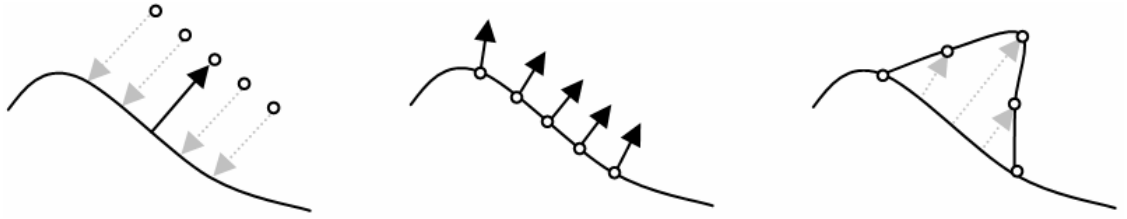
In compliance with the computational model that was presented in the previous chapter, the method for transposing a feature from feature space to modeling space consists of three steps, depicted in Figure 4.3 both as a two-dimensional and a three-dimensional example. The three-dimensional example shows how the shape configuration of the feature F can be computed. The example is that of a Bump feature, attached to a randomly generated base surface. It is assumed that the origin of the feature $a = (x_a, y_a, z_a)$ in modeling space has been given by the user. The normal of the target surface at the origin is denoted as $\overline{a}$, and it is assumed that this vector is a unit vector. The direction vector $\overline{d}$ is also a unit vector and perpendicular to $\overline{a}$.



(a) *(b)*



(c) *(d)*



(e)

Figure 4.3: *Instantiation of a bump on a surface: (a) the original surface (b) original feature area (c) projected feature area (d) final shape (e) a 2D schematics of the projection and deformation*

Algorithm 4.2: Transposing from feature space to modeling space (S, F, a, d)

S is the target surface

F is the feature

a is the origin

d is the direction vector of the feature

1. Compute the area A of the feature in feature space
2. Compute the transformation matrix M such that $M\bar{z} = \bar{a}$
3. Compute MA
4. While MA intersects S
5. Translate MA by $\bar{a}$
6. For all control points $e_{i,j}$ of A
7. Compute the projection $e^a_{i,j}$ of $e_{i,j}$ onto S in the direction of $-\bar{a}$
8. Compute the surface normal $\bar{n}_{i,j}$ of S in $e^a_{i,j}$
9. If the original direction of deformation is maintained then
10. For all $T^l_{i,j}$
11. $T^l_{i,j} = MT^l_{i,j}$
12. Else if the direction of deformation is adapted to $\bar{n}_{i,j}$ then
13. For all $T^l_{i,j}$
14. Compute the transformation matrix $M^2_{i,j}$ such that $M^2_{i,j}\bar{z} = \bar{n}_{i,j}$
15. $T^l_{i,j} = M^2_{i,j}T^l_{i,j}$
16. Compute the final state of the control points as $\prod_{p^l \notin P^{area}} p^l T^l_{i,j} e^a_{i,j}$

Expressed in terms of the three steps of the computational model, this algorithm can be explained as follows:

1. The area configuration of the feature F is instantiated in feature space such that its local coordinate system coincides with the global coordinate system. This feature instance is copied to modeling space, i.e. a shape configuration is created in modeling space that is identical (in terms of global coordinates) to the shape configuration of the feature in feature space.
2. A transformation matrix M^1 is constructed that aligns the shape configuration with the origin a. The matrix M^1 enacts a transformation between the local coordinate system of the feature in feature space and the local coordinate system that is defined by $\overline{a}$ and $\overline{d}$ and can be defined as
$$M^1 = \begin{pmatrix} \overline{a}\cdot\overline{x} & \overline{a}\cdot\overline{y} & \overline{a}\cdot\overline{z} & 0 \\ \left(\overline{a}\times\overline{d}\right)\cdot\overline{x} & \left(\overline{a}\times\overline{d}\right)\cdot\overline{y} & \left(\overline{a}\times\overline{d}\right)\cdot\overline{z} & 0 \\ \overline{d}\cdot\overline{x} & \overline{d}\cdot\overline{y} & \overline{d}\cdot\overline{z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$
It holds that $M^1\overline{z} = \overline{a}$ and $M^1\overline{x} = \overline{d}$. The matrix M^1 is then used to align the shape configuration that was computed in step 1 with the origin.
3. Each control point $e_{i,j}$ is projected onto the embedding surface in the direction of the normal vector of the origin. The projected control point is denoted $e^a_{i,j}$ and the normal vector of the target surface at the position of the projected control point is denoted as $\overline{n}_{i,j}$ (see Figure 4.3c).

Here, the user has a choice either to preserve the parameter mapping of the feature as it was defined in feature space or to adapt the deformation of the feature to the local curvature of the target surface (see Figure 4.4) .

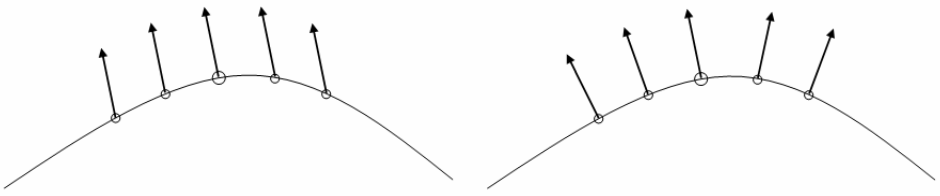


Figure 4.4: *Examples of (a) preserving the parameter mapping or (b) adapting the parameter mapping to the surface normal*

If the parameter mapping is preserved, then the relative effect of each mapping function remains the same, and the parameter mapping is in its entirety adapted to the surface

normal at the attachment point, such that each mapping function is multiplied by a transformation matrix M_2 for which it holds that $M^2\bar{z} = \bar{a}$, such that $\mu^l_{i,j}(e_{i,j}, p^l) = M^2 T^l_{i,j} e_{i,j}$.

If the parameter mapping is adapted to the embedding surface for each control point individually, then for each control point $t_{i,j}$, the parameter mapping functions defined on that control point are adapted to the normal of the base surface by multiplying them with the transformation matrix $M^2_{i,j} = \begin{pmatrix} \bar{d}\cdot\bar{x} & \bar{d}\cdot\bar{y} & \bar{d}\cdot\bar{z} & 0 \\ (\bar{d}\times\bar{n}_{i,j})\cdot\bar{x} & (\bar{d}\times\bar{n}_{i,j})\cdot\bar{y} & (\bar{d}\times\bar{n}_{i,j})\cdot\bar{z} & 0 \\ \bar{n}_{i,j}\cdot\bar{x} & \bar{n}_{i,j}\cdot\bar{y} & \bar{n}_{i,j}\cdot\bar{z} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ that aligns the vectors $\bar{z} = (0,0,1)$ and $\bar{x} = (1,0,0)$ with the vectors $\bar{n}_{i,j}$ and $\bar{d}$. In case the deformation vectors are adapted to the local curvature, then parameter mappings are defined as $\mu^l_{i,j}(e_{i,j}, p^l) = M^2_{i,j} T^l_{i,j} e_{i,j}$. Regardless of how the parameter mappings are adapted, a final position of each control point $e_{i,j}$ can be computed as $e_{i,j} = \prod_{p^l \notin P^{area}} p^l T^l_{i,j} e^0_{i,j}$ (see Figure 4.3d).

It has to be noted that the area configuration of the feature does not exactly match the embedding surface of the feature: because, in the area configuration of a feature, the control points of a feature lie on the embedding surface, the area may intersect the embedding surface (see Figure 4.5a). This is a result of the fact that the geometry of a feature is not determined until after the control points of a feature have been projected onto the target surface. The magnitude of this error is dependent on the amount of curvature of the embedding surface and the resolution of the control point grid of the feature. However, as is shown in Figure 4.5b, this error disappears once the feature is deformed. If this is not the case, then another approach to the problem is to increase the resolution of the control point grid, as is shown in Figure 4.5c. This is no solution, but it does reduce the problem. Hence, unless the deformation is very small, there is no problem, but if the deformation is small, then the problem of intersection can be reduced by increasing the resolution of feature's control point grid.

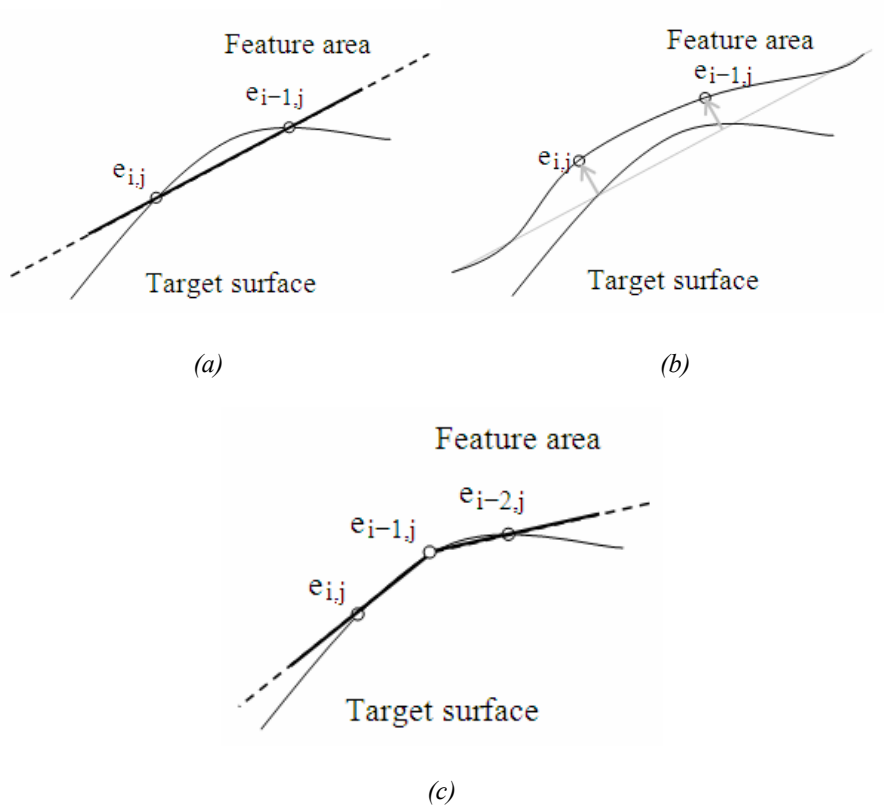


Figure 4.5: *(a) Two-dimensional diagram of the discrepancy between a feature area and a base surface, (b) disappearance of the problem due to deformation of the feature and (c) reduction of the problem through increase of the resolution*

The strength of the given algorithm is in that it makes no assumption on the type of the target surface. The target surface can be a B-spline surface, a polygon mesh or even a point cloud. The only place in the algorithm where the shape representation type plays a role is in line 7, where control points are projected onto the target surface. As long as a method is available that projects a point onto the target surface, the embedding surface of the feature to be instantiated can have any shape representation type.

However, there are also some limitations to the types of surface in which a feature can be embedded with the given algorithm. In addition, the extent to which a transposed feature resembles the original feature depends on the characteristics of the embedding surface. These limitations and their consequences will be discussed in section 5.2. However, the given algorithm suffices for most embedding surfaces. Giving improvements for algorithm 4.2 is therefore left for further research.

4.1.3 Changing the resolution of a feature

In the previous section, the increase of the resolution of the control point grid was mentioned as a possible operation on an existing feature instance. Increasing the resolution enlarges the detail of control over the feature shape, and also increases the correspondence between the control points and the shape representation of the feature. For example, when a B-spline surface is generated from the control points, then after an increase of the resolution each control point has a stronger influence on a smaller area of the feature shape.

Typically, a user will want to maintain the shape of a feature when changing the resolution of the control points. It can be guaranteed that an increase in the number of control points does not lead to a change in the shape of the feature, i.e. the shape of a feature with 25 control points should not change if its number of control points increases to 100, assuming of course that the parameter values stay the same. It is also possible to reduce the number of control points, but in this case it cannot be guaranteed that the shape of the feature stays the same. Because of the nature of B-spline surfaces, control points can only be added in rows or columns, keeping the control point grid bi-directional. Control points can be added in a process that is called *knot refinement* (Piegl and Tiller, 1995).

However, increasing the resolution of a feature is not just a matter of changing the number of control points. When control points are added to the feature's shape configuration, then basic shape configurations and parameter mappings must not only be chosen or constructed for the new control points, but they must also be adapted for the existing control points, as these will become newly positioned in the control point grid. The basic shape configuration of new control points can be easily determined, because the definition of the freeform feature requires that they are positioned in a bi-directional grid; the basic configuration of a control point can therefore be derived from the position of that of its neighbors. To adjust and augment the parameter mapping, backward computing is used: a parameter mapping is deducted from the shape of a feature after a resolution increase. This is described in the following algorithm.

Algorithm 4.3: Increasing the control point resolution (F, x, y)
F is a feature
x and y are the desired resolution in both directions of the grid

1. The original feature F is stored as F^0
2. For all parameters p^l, $l = 0,\ldots,m$:
3. Choose a parametric configuration of F such that $p^l \neq 0$ and all other parameter values are zero.
4. Add rows and columns of control points one at a time until the row and column size of the shape configuration E of F matches respectively x and y
5. Compute the mapping function for each control point by dividing the actual position of a control point $e_{i,j} = \begin{pmatrix} x_{i,j} \\ y_{i,j} \\ z_{i,j} \\ w_{i,j} \\ 1 \end{pmatrix}$ by p^l such that

$$T^l_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 & x_{i,j}/p^l \\ 0 & 1 & 0 & 0 & y_{i,j}/p^l \\ 0 & 0 & 1 & 0 & z_{i,j}/p^l \\ 0 & 0 & 0 & 1 & w_{i,j}/p^l \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

6. Set the shape configuration of the feature to that of F^0 to be able to compute the parameter mapping for p^{l+1}.

The condition in line 3 of the algorithm is necessary to prevent division by zero in line 5.

4.1.4 The instantiation of an embedded feature

To be able to use features, they need to be instantiated on an embedding surface. This operation is relevant mostly for the problem of freeform feature-based design, but also has an application in feature recognition, as will become clear in section 4.2. As was mentioned in section 3.5, there are two ways of instantiating a feature: the embedded

feature or the corresponding feature. In this section, we give an algorithm for instantiating an embedded feature, and in section 4.1.4 an algorithm is given for the instantiation of a corresponding feature.

The embedded feature is characterized by the fact that the geometry of the feature is a subset of the geometry of its embedding surface. The instantiation of a feature on its embedding surface therefore consists of three main steps: the removal of geometry from the embedding surface, the transposition of the feature from feature space and the creation of a transition between the geometry of the feature and that of its embedding surface. An algorithm that addresses these three steps can be given as follows:

Algorithm 4.4: Instantiation of an embedded feature (a, S, F, ε)
a is the point of origin of F on S

S is the target embedding surface

F is the feature to be instantiated

ε is a threshold for the accuracy of the algorithm

1. Create an area configuration A of the feature F in feature space
2. Using Algorithm 4.2, transpose A to modeling space
3. While the distance $d(A,S) < \varepsilon$
4. Using Algorithm 4.3, increase the resolution of F
5. Find the smallest region $S^a \subseteq S$ for which it holds that $d(A,S^a) \leq d(A,S)$
6. Remove S^a from S
7. Deform A to obtain a full shape configuration of F, such that $e_{i,j} = \prod_{l=0}^{m} T_{i,j}^{l} e_{i,j}^{0}$
8. Find the transition region $S^a - A$
9. Reconstruct the transition area

This algorithm follows the computational model given in section 3.5.2 with one main difference: because in Algorithm 4.2 it was specified how the parameter mapping of a feature can be adapted when it is transposed from feature space to modeling space, the area configuration of a feature can be deformed in modeling space to create a full shape configuration, instead of performing this deformation in feature space and transposing the feature a second time.

The distance measure used in step 3 of the algorithm is also known as the Directed Hausdorff Distance, which can be defined as $d_{DHD} = \sup_{s \in S} \inf_{a \in A} d_E(s,a)$, where d_E is the Euclidean distance between two points. In step 5 of the algorithm this distance measure can be used to find the subset S^a of S that corresponds to A, and in step 6, this region is removed from the embedding surface. How part of a surface can be removed depends on the shape representation type of both the feature and its embedding surface. We will not detail the removal of geometry for different shape representation types here, because this functionality is available in all existing CAD software (e.g. the trimming of nurbs surfaces, i.e. the creation of a hole in a nurbs surface is a well-supported operation).
The deformation in step 7 can be done using the adapted parameter mapping of the feature as resulted from step 2. Then, in step 8 of the algorithm, the transition region is found, i.e., that part of the embedding surface that was removed in step 6, but was not replaced by the geometry of the feature. The extent and nature of this transition region also depends on the shape representation type of the feature and the embedding surface. For example, if nurbs surfaces are used, then the region S^a can be exactly removed from S and consequently, the transition region is empty. However, if polygon meshes are used, then it is highly likely that $S^a > A$; in this case a Delaunay triangulation can be used to reconstruct the transition region (see Figure 4.6). Again, we will not give details here, because there is plenty of existing work that addresses the problem of shape reconstruction (referenced in chapter 2).

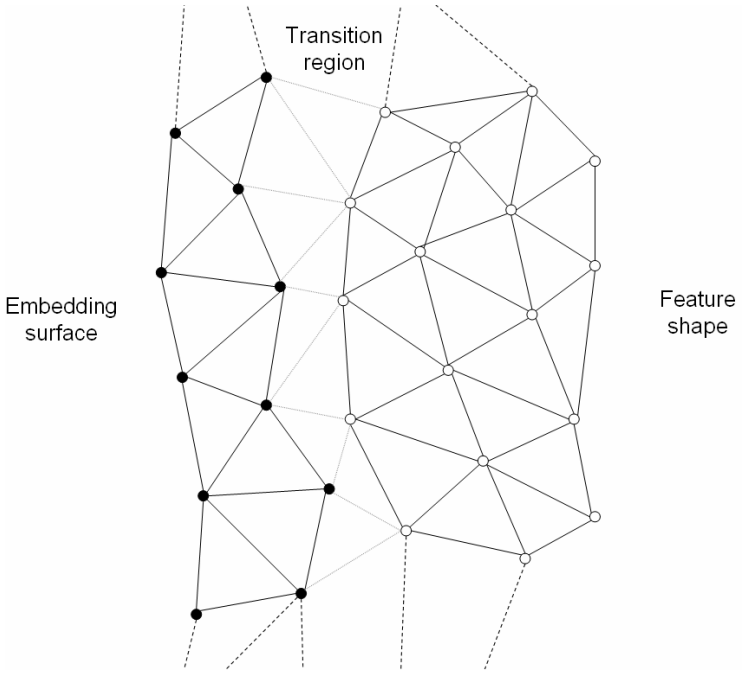


Figure 4.6*: Triangulation of the transition region of an embedded feature*

4.1.5 The instantiation of a corresponding feature

The corresponding feature is characterized by the fact that its geometry itself is not visible to the user. Rather, it has an effect on the geometry of its embedding surface through a correspondence function that defines how the geometry of the feature and that of the embedding surface are related. The idea for such a correspondence function comes from the field of shape modeling. For example, Sederberg and Parry (1986) have proposed a now well-known method of shape manipulation that is based on the correspondence between a target surface and a lattice of points that is constructed around the target surface. Shape manipulation actions are performed first on the constructed lattice and enact a deformation of the target surface through this correspondence. Song et al. (2005) show how lattice-based deformation can be used manipulate freeform feature shapes.

Algorithm 4.5: Instantiation of a corresponding feature a, S, F, ε**)**

a is the point of origin of F on S

S is the target embedding surface

F is the feature to be instantiated

ε is a threshold for the accuracy of the algorithm

1. Create an area configuration A of the feature F in feature space
2. Using Algorithm 4.2, transpose A to modeling space
3. While the distance $d(A,S) < \varepsilon$
4. Using Algorithm 4.3, increase the resolution of F
5. Find the smallest region $S^a \subseteq S$ for which it holds that $d(A,S^a) \leq d(A,S)$
6. Construct a function $\chi(F) = S^a$
7. Deform A to obtain a full shape configuration of F, such that $e_{i,j} = \prod_{l=0}^{m} T_{i,j}^{l} e_{i,j}^{0}$
8. Using χ, compute S^a

As with the computational models on which they are based, Algorithm 4.5 and Algorithm 4.4 are for a large part identical. The main difference is in step 6 of Algorithm 4.5, where the correspondence function is constructed. Different correspondence functions are possible, such as the aforementioned lattice-based correspondence. In this thesis we have used a distance-based correspondence function and constructed it as follows:

Algorithm 4.6: Construction of a distance-based correspondence function (F, S^a)
F is the feature

S^a is a region of its embedding surface

1. For each point e^0 on the area A of the feature F
2. Find the closest point s^0 on S^a
3. Construct a partial function $\chi(e) = Ts^0$, where $T = \prod_{l=0}^{m} T_{i,j}^l$
4. Combine the partial functions in a function $\chi(F, S^a)$

Algorithm 4.6 constructs a correspondence function on the basis of the idea that if a point e on the shape of a feature is deformed under the influence of the parameter mapping defined for the feature, then the point s on the embedding surface that is closest to e should be identically deformed. If this algorithm is used in the context of Algorithm 4.5, then line 3 of the algorithm guarantees that the the distance between e and s is smaller than ε.

Theoretically it is possible that a point s lies closest to more than one point e (see Figure 4.7). In this case, s will be connected to multiple of these corresponding points and the position of s becomes erratic, because it is transformed more than once. This mostly occurs in the case of a discontinuity in the embedding surface and can be solved by applying a first-come-first-serve policy in establishing a corresponding point for s or by averaging the different transformations that are performed on s. However, the effect of these solutions on the validity of s cannot be predicted without knowing the topology of S. Another solution would be that the movement of s is averaged over all its correspondence points.

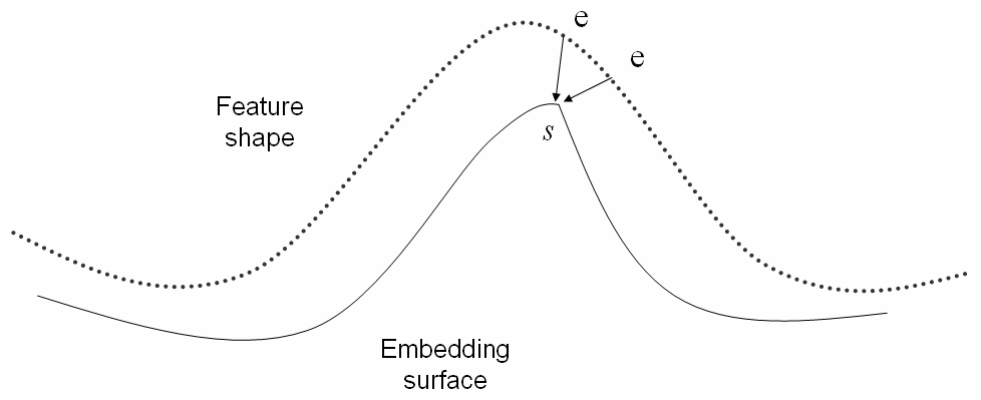


Figure 4.7*: Two-dimensional example of multiple correspondences*

4.2 Template matching

It has been shown in the previous sections how features can be instantiated on a target surface and how feature space can be a used. The given algorithms will be used in this section in the development of a freeform feature recognition algorithm.
Feature recognition can be defined as the process of attributing parametric data to geometric data by comparing it to feature type definitions that have been pre-defined and are organized in a feature library. As was reviewed in chapter 2, several techniques for the recognition of regular form features have been developed. The complexity of the problem of freeform feature recognition is higher than that of regular form feature recognition. It was hypothesized in chapter 1 that methodology for freeform feature recognition cannot be obtained by simply extending existing methods for regular feature recognition, and in chapter 2 arguments were presented to support this hypothesis. It can therefore be argued that freeform feature recognition is a new research problem, not an extension of regular feature recognition, or even that the recognition of regular features is a sub-problem of freeform feature recognition.
In this thesis we will assume that a feature library is available, in which all the relevant feature type definitions are stored. User-defined features can be added to this library at will. We assume that the freeform feature recognition methods are able to deal with any feature type that is contained in the feature library, including user-defined features.
We also assume that it is reasonable to ask for additional information from the user that can help to increase the accuracy and efficiency of the feature recognition process. When recognizing a feature $F\left(E^0, P, \mu\right) = E$, we therefore assume that the user is able to indicate a region of interest $ROI \subseteq S$, such that $E \subseteq ROI$. The region of interest can be easily selected by for example a lasso tool (which is common in most if not all existing CAD applications) or a bounding box. We also assume that the region of interest contains no other features. This imposes some limitations on the applicability of the algorithms that will be presented in this chapter. These limitations will be discussed in section 5.2.

In section 4.2.1 the template matching method is discussed, which is to the knowledge of the author the only existing method for freeform feature recognition. The method is presented in terms of the theory that was given in the previous chapter, to demonstrate the applicability of the theory to the feature recognition problem and also to be able to compare the method to the new freeform feature recognition methodology that is presented in this thesis. In a critical analysis of the template matching method, its shortcomings can be revealed and it will be argued that a new approach is needed in solving freeform feature recognition problems.

In section 4.3, such a new approach to feature recognition is presented that uses evolutionary computation not only to recognize features, but also to reconstruct the recognized features in a geometric model. We deem this method template-based evolutionary freeform feature recognition and show how it overcomes several of the disadvantages of the method.
In section 4.4 we will argue that template-based methods in general are unable to deal with multi-dimensional parameters. In response to this notion, we will give a method for the recognition of freeform features with multi-dimensional parameters which we call curve-based freeform feature recognition. This method is not directly related to template-based evolutionary freeform feature recognition, but both methods are based on the same approach to the feature recognition problem.

The concept of template matching was first introduced in the domain of regular form features (Thompson et al., 1999) and was later extended to freeform features (Vergeest et al., 2001, 2003; Song et al., 2005). Both approaches were motivated by the need of interpreting shape data obtained by a laser range scanner.
The approach of Thompson et al. operates on a point cloud model of an industrial part, which can then be subjected to feature recognition, with the goal of remanufacturing the part. Hence, the primary goal is to improve the quality and efficiency of the manufacturing process.
The contribution of Thompson et al. is limited to a general description of the feature recognition process. They do not give methods or algorithms, although they claim to have implemented a working system. The user is supposed to assist in the feature recognition process by selecting a type and an approximate location for the feature to be recognized. A feature instance of the selected type is then fitted to the point cloud model. Thompson et al. do not provide sufficient information on the algorithms used, the end results of these algorithms, how the recognized features relate to the rest of the point cloud model, and how the feature recognition contributes to the reverse design process, respectively.
A more elaborate approach to template matching is proposed by Vergeest et al. (2001, 2003) and described in more detail by Song et al. (2005). Their method is directed at freeform features, and is motivated by the desire to obtain a more efficient means to manipulate freeform shape data. In addition to feature recognition, they also propose a method to modify the object on which the feature has been recognized.
Based on the theory proposed in this thesis, the method of Song and Vergeest et al. could be redefined with the goal of higher efficiency. In addition, by presenting their methods in terms of the proposed theory, the applicability of an analysis of their methods to the new methods proposed in this thesis is increased.

4.2.1 An algorithm for template matching

It is assumed by Song and Vergeest et al. that the target shape of a freeform feature recognition procedure is given in the form of a point cloud. That is, the shape can be represented as an ordered collection of points; which can be formally defined as $S = \{s_1, \ldots, s_n\}$, where each $s_i = (x_i, y_i, z_i)$ is a point in the point cloud. We assume that there exists a subset $ROI \subseteq S$ that can be recognized as a feature. The application of the method to intersecting features and partial features will be addressed later in this section.

It is assumed that existing feature type definitions are stored in a feature library L, and that the goal of the freeform feature recognition procedure can be subdivided into three sub-goals:

1. Finding the subset $ROI \subseteq S$
2. Selecting a feature type F that best corresponds to E
3. Determining parameter values for which the similarity of the shape of an instance of F with regard to E is maximal.

The achievement of the first goal can be made easier by assuming that a region of interest is selected by the user. However, it must be assumed that this region of interest is not perfect, i.e. does not only contain the shape of the feature to be recognized, but also 'rest' data. The second goal of feature recognition can be fulfilled by having the user select a specific feature type. An instance of this feature type, called a template feature, is then generated and used to recognize a feature in E. The third goal of the feature recognition procedure is to determine what parameter values lead to a maximal similarity between the shape of the template feature, F^T, and E.

Based on the specified subgoals of feature recognition, we can formulate the essence of template-based feature recognition:

Feature recognition through template matching is the procedure of finding the template $F^T(E^0, P, \mu) = E^T$ *for which the distance* $d(E^T, ROI)$ *is minimal, given a subset* $ROI \subseteq S$.

The distance function $d(E^T, ROI)$ must be selective, such that $d(E^T, S) = d(E^T, ROI)$. In other words, the elements of S outside ROI should not contribute to the distance between S and E^T. As a distance function, the Mean Directed Hausdorff distance d_{MDHD}

can be used, which has been defined as $d_{MDHD}\left(S,E^T\right)=avg\left(d_{DHD}\left(S,E^T\right),d_{DHD}\left(E^T,S\right)\right)$, where d_{DHD} is the Directed Hausdorff Distance, which was defined earlier in this chapter. The subset s can be easily determined by examining each individual element s_i, : if $d_H\left(S,E_i^T\right)\neq d_H\left(S-\{s_i\}\right)$, then $s_i\in ROI$. In other words, the subset ROI is the smallest subset of S for which the distance to the feature shape is equal to that of S.

The search for an optimal configuration of a template feature is equivalent to a function minimization problem in multiple dimensions. The function under consideration is the Mean Directed Hausdorff distance, which can be directly derived from the parameter values and the basic shape configuration of the template feature. Because both the target shape and the basic shape configuration of the feature are fixed, the only input to the function are the m parameter values $P=\left(p^0,\ldots,p^m\right)$ and consequently the minimization takes place over *m* dimensions.

Vergeest and Song do not specify how the minimization takes place, and it is not investigated how different minimization techniques and shape similarity measures affect the efficiency and the quality of the outcome of the feature recognition procedure. The algorithm proposed here makes use of a version of the direction set method for multi-dimensional function minimization (Press et al, 2002), which is based on the notion that a multi-dimensional function minimization can be approximated by a series of one-dimensional minimizations in conjugate directions.

To find an optimal configuration for a template, the following algorithm is used:

Algorithm 4.7: Template Matching $\left(S,F,\varepsilon\right)$

S is the target surface

F is the feature type, as defined in the feature library

ε is a user-given threshold

1. Instantiate a template feature $F^{T_0}\left(E^{0T_0},P^{T_0},\mu^{T_0}\right)=E^{T_0}$ of type i from the library with a random vector of parameter values P_0.
2. Use linear optimization to determine a scalar α_0 for which $d_{MDHD}\left(\xi^{T_0}\left(\mu^{T_0}\left(\tau^{T_0},\alpha_0P^{T_0}\right)\right),S\right)$ is minimal.
3. Instantiate a second template feature $F^{T_1}\left(E^{0T_1},P^{T_1},\mu^{T_1}\right)=E^{T_1}$ of type i with a vector of parameter values P^{T_1} that is perpendicular to P^{T_0}.

4. Use a linear optimization routine to determine the scalar α_1 for which $d_{MDHD}\left(C\left(\mu^{T_1}\left(E^{0T_1}, \alpha_1\left(P^{T_1} - \alpha_0 P^{T_0}\right)\right)\right), S\right)$ is minimal, where C is a function that applies the weighted de Casteljau's algorithm to evaluate the control points of the feature.

5. Redefine F^{T_0} as $F^{T_0} = \left(\mu^{T_0}\left(E^{0T_0}, \alpha_1\left(P_1 - \alpha_0 P_0\right)\right), \alpha_1\left(P_1 - \alpha_0 P_0\right), \mu^{T_0}\right)$.

6. If $d_{MDHD}\left(C\left(\mu^{T_0}\left(E^{0T_0}, \alpha_0 P^{T_0}\right)\right), S\right) < \varepsilon$, then return F^{T_0}. Else jump to step 3.

In each iteration of the steps 3-5, this algorithm investigates a change of parameter values that is perpendicular to the vector of parameter values of the optimal configuration so far. This means that in each step, a new direction of search is tried out that is not related to the previous direction of search. This way, successive linear optimizations do not interfere with each other. An example of a template matching procedure is shown in Figure 4.8. The picture shows the target shape and the feature shape after successive iterations of Algorithm 4.7.

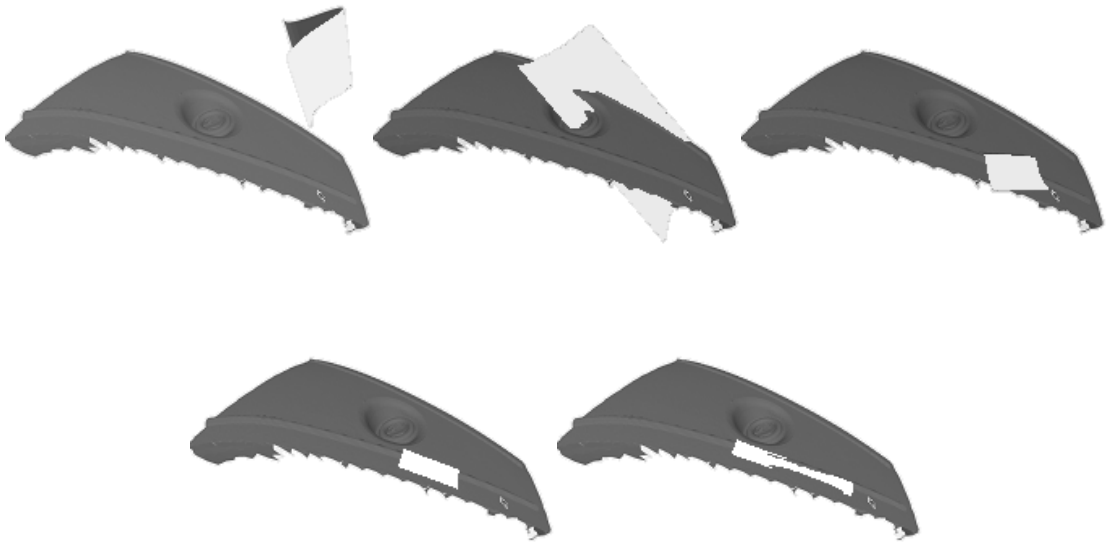


Figure 4.8: *Successive iterations of a template matching procedure, in which (from upper left to lower right) a template feature has a stepwise better correspondence with the target surface.*

Extendable templates

The template matching approach can be used to recognize pre-defined feature types. However, as was discussed in section 3.2.4, the proposed theory is based on the

assumption that parameters are one-dimensional. To also be able to recognize curve-based parameters, Song et al. (2005) introduce the concept of extendable templates. In their approach, the authors assume that parts of a feature with a two-dimensional parameter can be approximated by an ordered series of template features obtained through Algorithm 4.7. They also assume that successive template features in this series have similar parameter values. For this reason, starting values for a template matching procedure can be derived from the previous template. However, no algorithms are given by the authors, and the details of applying extendable templates are not specified. Two examples of the result of template matching with extendable templates are given in Figure 4.9.

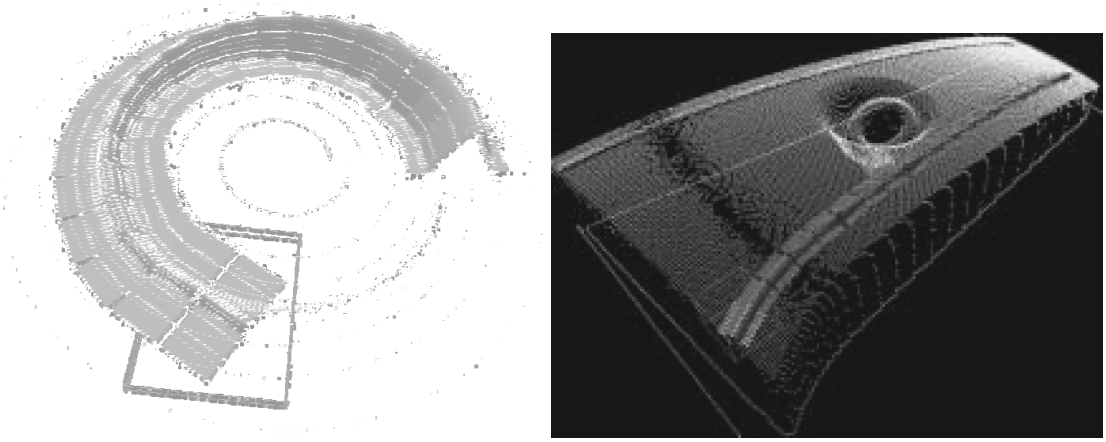


Figure 4.9*: Two examples of the result of a template matching procedure using extendable template features.*

4.2.2 Analysis of the template matching algorithm

The template matching method is a significant improvement to existing regular feature recognition methods in that it does not make use of specific assumptions regarding the geometry of the feature or the target shape. It can therefore be applied generally. As mentioned before, any valid feature type definition can be used by Algorithm 4.7 and it is possible to define new feature types. With the use of extendable templates, more complicated features can be recognized, providing that they can be decomposed into features that can be recognized using Algorithm 4.7. In some regards, the method has been poorly described in the referenced literature. For example, the method of searching (implemented here as Powell's direction set method), has not been investigated, and the use of the Hausdorff distance as a metric for shape similarity is not compared to other measures (although the authors present a case study in Vergeest et al., 2003).

Furthermore, although the authors present interesting results, the template matching method has not been extensively tested.

More importantly, there are some principle disadvantages to the template matching method. First, pre-defined template features are matched to a target surface on grounds of shape similarity. If the feature library is assumed to be a static collection of pre-defined feature types, then the applicability of the template matching method is very limited and can only be applied to specific target shapes. If user-defined features can be added to the feature library, then the method can be applied to a broader range of target shapes, but in this case the responsibility for defining a correct feature type lies with the user. If the user is responsible for defining a new feature type that matches a specific target shape, then a large part of the feature recognition is done by the user instead of by using Algorithm 4.7.

Second, the method requires a high degree of similarity between the target shape and the template feature that is used in a feature recognition procedure. The reason for this is that the template matching method adapts the parametric configuration of a feature, but not its morphological aspects. If the morphological aspects of a feature could also be adapted during the feature recognition process, then theoretically a feature type definition in the feature library initially does not have to match a target shape at all. The main reason for using a feature library in this case is that it allows the user to store other types of information (semantic, functional, etc.) with the feature. In short: because the morphological aspects of a feature cannot be adapted, the template matching method is inflexible.

Finally, although a target shape is recognized in the sense that that the parameters of the feature can be used to deform the features (Song et al. 2005), the geometric data of the region of the target shape that corresponds to the matched template feature still exists. In terms of the theory presented in this thesis, the recognized feature is a corresponding feature. The template feature has all the disadvantages of the corresponding feature, but does not benefit from the advantages, which mainly lie in the instantiation of a feature on an embedding surface. Therefore, the template matching approach cannot be used to obtain the basic shape configuration E^0 of a feature. Because E^0 functions as an 'anchor point' for the definition of the parameter mapping of a feature, even if it assumed that a perfect similarity between feature shape and target shape is possible, then the definition of the recognized feature is inaccurate. As a result, many feature-based operations cannot be performed on a recognized feature. For example, a feature that has been recognized in a template matching procedure cannot be deleted.

It can be concluded that the template matching method is a powerful tool for recognizing a small set of pre-defined features. However, even with the ability to use extendable templates, the usefulness of the template matching method is limited by its principle shortcomings. In the following sections we therefore present new feature recognition methods that improve on all three mentioned problems of template matching.

4.3 Template-based evolutionary freeform feature recognition

In this section we present a freeform feature recognition method that is template-based, but improves on the template matching method with regard to the three principle shortcomings of this method. That is, in the new method, the user is less responsible for the definition of a template feature, because the morphological aspects of the template feature are also adapted during the feature recognition process. In addition, the proposed method can be used to recognize features both as a corresponding feature and as an embedded feature. In other words, the basic shape configuration is also retrieved. As a result of these improvements, the user is to a lesser extent burdened by the need to define new feature types and can perform more advanced operations on the recognized features. An additional advantage of the proposed method is that the accuracy with which features can be recognized is significantly increased.
In the previous section, several disadvantages of the templates were discussed. However, the large advantage that additional information can be stored with a pre-defined feature motivates us to use template features despite these disadvantages. In the following sections we first outline and then detail a method that uses template features but incorporates the mentioned improvements.

4.3.1 Introduction to evolutionary computation

In section 4.2, we mentioned that Song and Vergeest et al. did not investigate different search methods or different shape similarity metrics. For the implementation of template matching as it was given in the previous section, several search methods were tried, but both brute force methods and heuristic methods proved to be slow and unpredictable, i.e. the efficiency of these search methods differed greatly for different feature types and different target surfaces. For this reason, we chose to use a probabilistic search method. In this section, we introduce the concept of evolutionary computation and show how it can be applied to feature recognition. Evolutionary computation is a well-known technique that mimics the mechanism of natural selection. There is literature on evolutionary computation in abundance. The reader is referred to Goldberg (1989) and Davis (1991), who give overviews of the concept as well as some classical issues regarding evolutionary computation. An overview of the more recent problems in the field of evolutionary computation is given by Ghosh and Tsutsui (2002).
Evolutionary computation is based on the principle of ‘survival of the fittest’, which, in nature, states that organisms with certain genetic elements have a higher chance to survive and are therefore more likely to pass on these genetic elements to a next generation. Due to this mechanism, populations adapt to their environments and ‘improve’ with each generation. In an evolutionary computation method, possible

solutions to a problem are viewed as organisms, of which the genetic elements are the variables that play a role in finding the optimal solution to the problem.
An evolutionary approach to feature recognition has two advantages:

- Our goal is to find a general approach to feature recognition, but in a general case it is impossible to predict what types of features a recognition procedure should handle. Rather than trying to prepare a specific feature recognition technique for each feature it may have to process, we want the feature recognition procedure to adapt to the target shape. The ability to adapt to a problem at hand is one of the advantages of evolutionary computation.
- We are trying to recognize freeform features, but the number of possible target feature shapes is limitless. Evolutionary computation is known to find a reasonable solution to large problems quickly, because it maintains a collection of possible solutions instead of focusing on one.

In an evolutionary computation method asset of concepts similar to that of natural evolution must be implemented. The most important of these concepts are: genetic structure, crossover, mutation, fitness and selection.

Genetic structure
In nature, the genetic structure is expressed in the DNA of an organism. The basic identity of the genetic structure is the gene, in which DNA describing certain properties of an organism is grouped. A specific occurrence of genes is collectively called a genotype. In an evolutionary computation method, the organisms are the possible solutions to the problem. The variables of the problem are the genes of the organism. In the case of feature recognition the organisms are the features. As the problem of feature recognition relates to shape, the genes of a feature are the elements that are needed to compute a shape configuration of the feature, namely the parameter values and the parameter mappings.

Crossover and mutation
Crossover and mutation determine how the genotype of an organism can be inherited by its offspring. Each new organism receives part of its genes from one parent and part from the other. Sometimes, mutation distorts the information that is stored in a gene. The result may often be that the organism becomes flawed, but it may also be possible that the organism develops a new and useful property. In nature, the concept of mutation keeps the genetic inheritance flexible, by constantly introducing new genes into the gene pool, the collection of all genes that are available in a population. In evolutionary computation, mutations are even more important, because computation procedures are limited by the

available time and memory. Populations are therefore often simulated with less-than-natural sizes. In this case the risk of inbreeding (a reduction of the size of the gene pool due to related features generating offspring) is much bigger than in nature. For feature recognition, the process of crossover amounts to a selection of a gene from either of its parents. This requires the parents to be of the same feature type, or it would be impossible to select from their respective genetic structure in a meaningful way. When the gene is copied from either of its parents, there is some possible mutation. To enable this in a practical procedure, noise is added to the value of the gene.

Fitness and selection

The fitness of an organism describes how well it adapts to its environment. The better an organism adapts to its environment, the higher the probability that it will survive and procreate. Natural selection means that only the best adapted organisms in a population have a chance of generating offspring. Because the offspring of the fittest organisms can also be expected to adapt to the environment, the population 'improves' with each generation. Because the selectivity of the procreation determines how fast and in what direction populations evolve, the selection is a vital element of an evolutionary feature recognition method. In the problem of feature recognition, the fitness expresses how well a feature matches the target surface.

Putting together all the aspects mentioned above, a typical evolutionary computation algorithm consists of the following steps:

1. Create a population of organisms.
2. Compute the fitness of each organism.
3. Generate a new population of organisms, of which the genetic structure is derived from the fittest individuals in the previous 'generation'.
4. Repeat this process until an optimal or acceptable solution to the problem has been found.

In the past, this technique has been applied to many computational problems, and in many cases with a successful, sometimes surprising outcome. Specifically, evolutionary computation has been applied to problems with a large number of variables and problems without a predictable set of correct solutions. Because of the probabilistic nature of the technique it is often difficult to predict the results, and for this reason the validity of evolutionary computation has been criticized. However, the method circumvents problems that other methods encounter and the results are at least as good in general as these other search methods.

Evolutionary algorithms have been recently applied to feature recognition by Pal et al. (2005). Their method extracts features from collections of partial feature surfaces by generating a population of individuals with a random collection of surfaces from the given collection. Consecutive generations inherit partial feature surfaces, which are thus combined to form a complete feature. This method is applicable only to polyhedral inputs and assumes that it is known what surfaces belong to a feature. In addition, it is unclear how parameters are defined for a recognized feature, and how they affect the feature shape. Finally, their method is not able to deal with feature interference, compound features or pattern features.

4.3.2 Outline of the method

The actual freeform feature recognition method we present consists of six steps. The first two steps of an evolutionary freeform feature recognition procedure can be formulated as follows:

1. Using an approach similar to Algorithm 4.7, a feature on a target shape (Figure 4.10a) is recognized in an evolutionary template matching approach. The result is a corresponding feature. (Figure 4.10b).
2. The morphological aspects of the corresponding feature found in step 1 is adapted to fit the target surface in another evolutionary procedure (Figure 4.10c).

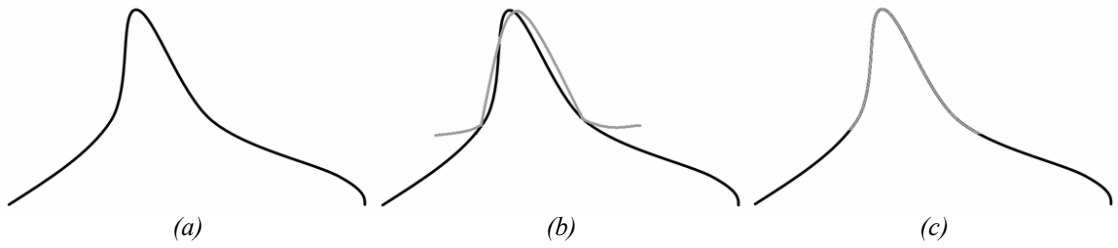
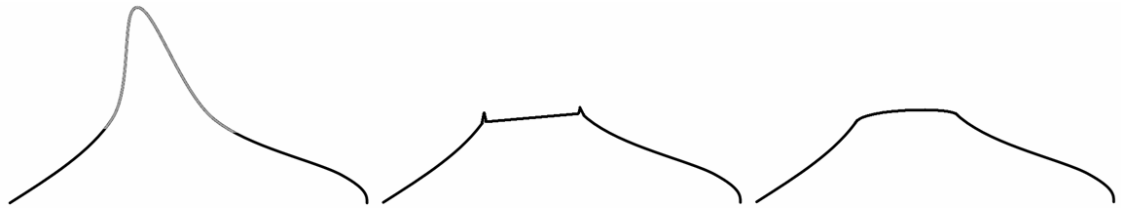


Figure 4.10*: Two-dimensional example of (a) a target surface (b) the corresponding feature that is the result of a template matching procedure and (c) the same feature after changing its morphology*

These first two steps are dealt with in section 4.3.5. The feature that results from step 2 is a feature of which the morphology is adapted to the target surface. However, it is still a corresponding feature. In the following steps, the basic shape configuration of the feature that results from step 2 is obtained:

3. The deformation parameters of the template feature are set to 0 and through a correspondence function, the target shape is manipulated correspondingly (Figure 4.11b).

4. The resulted surface is smoothed to get rid of any residual effects that are the results of an imperfect result of the feature recognition in step 1 and 2 (Figure 4.11c)



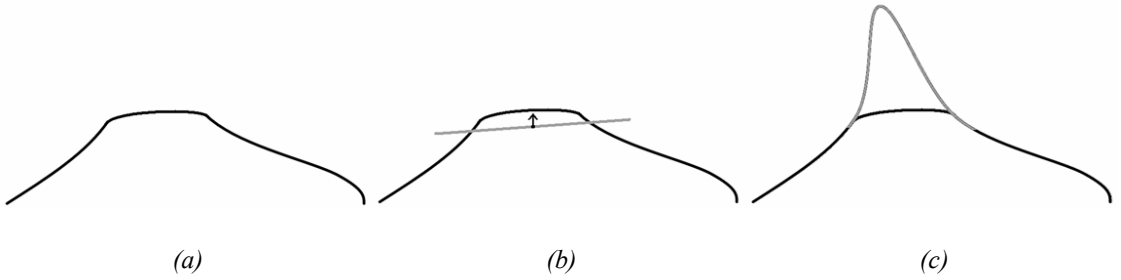
(a) *(b)* *(c)*

Figure 4.11*: Two-dimensional example of (a) a recognized corresponding feature, (b) feature-based removal of the deformation effect of the target feature and (c) smoothing of the base surface of a recognized feature*

A method for steps 3 and 4 is given in section 4.3.6. The result of steps 3 and 4 is a modified target shape that could be the embedding surface of the feature that we originally intended to recognize. Of course, this surface is one of the many possible embedding surfaces, because for most target features an original embedding surface cannot be uniquely determined.

Once the embedding surface of the feature has been reconstructed, the challenge is to find an embedded template feature on this surface, such that the resulting shape matches the original target shape. The attachment point, parameter mappings and parametric configuration that were found in step 2 are a reasonable approximation of the location and parameter values of the attached template feature. In the final steps of the procedure, an embedded feature is recognized as follows:

5. The area configuration of the feature template computed in step 1 and 2 is projected onto the target surface and a point of origin is determined (Figure 4.12b).
6. Starting from the parametric and morphological configurations that were found for the template feature in step 2, a new evolutionary feature recognition procedure is started, this time finding an optimal configuration of an embedded template feature (Figure 4.12c).



(a) (b) (c)

Figure 4.12: *Two-dimensional example of (a) a smoothed target surface, (b) the determination of an attachment point and (c) the instantiation of an attached feature*

In sections 4.3.7, the final two steps of the evolutionary freeform feature recognition method are discussed in detail.

4.3.3 A general feature-based evolutionary procedure

Before presenting dedicated methods for the identification and recognition of features, first a general evolutionary procedure for evolutionary freeform feature recognition will be presented. Based on this general procedure, specific algorithms are given for feature identification, but also for steps 1, 2 and 6 of the feature recognition procedure. The general procedure and the specific algorithms are given separately, because the kernel of the specific algorithms is the same. To prevent repeating ourselves in discussing the general properties of this kernel, we present it in a separate section.
A general evolutionary computation procedure can be given as follows:

1. An initial population $gen_1 = \{F^1, \ldots, F^\Pi\}$ is generated, where Π is the population size. The features in this population are all instances of a specific feature type that is available in the feature library.
2. The fitness of each feature is computed as the shape similarity between the feature shape and the target shape.
3. The features are ranked according to their increasing fitness value $f(\ )$, so that $f(F^1) \le f(F^2) \le \ldots \le f(F^{\Pi-1}) \le f(F^\Pi)$
4. A new population gen_{i+1} is generated as follows. For each feature instance in gen_{i+1}:
   - Two features, F^{mother} and F^{father}, are selected from gen_i with probability $P(x) = \frac{2}{\sigma\sqrt{2\pi}} e^{\frac{-x}{2\sigma^2}}$ which is a one-sided Gaussian distribution of the fitness over the domain $x \in [0, \infty)$ with standard deviation σ. With this procedure, the fittest individuals in the population are selected.

- Each gene of a new individual F^{new} is copied either from F^{mother} or F^{father}. Both have an equal chance of being selected to carry on a gene. To simulate a mutation of the genes, there is a certain chance that the gene is distorted.

5. The procedure is repeated for all feature types in the feature library

The procedure terminates when one of the following conditions is met:

- The fitness no longer increases over generations, or the increase in fitness is slow, i.e. the difference between $f(F^1)$ in gen_{i-1} and $f(F^1)$ in gen_i is smaller than a user-given threshold ε_1.
- The fittest feature in a population is a sufficient solution to the feature recognition problem, i.e. $f(F^1)$ in gen_i is smaller than a user-given threshold ε_2.

Several variables influence the efficiency, accuracy and robustness of an evolutionary procedure. The influence of the main variables is explained as follows:

- The population size Π indicates the size of the population of features. Each individual in the population signifies a single probe in the search space. Therefore, the larger the population size, the faster (in terms of number of generations needed) the procedure converges to an optimal solution to the feature recognition problem. However, a large population size also means a higher computation time. The population size can be set by the user, whose goal may be either a low computation time or a high quality of the result. Alternatively, the population can be dynamically set by the recognition method. In this case the population size will be large at the start of the method, when a large part of the search space is investigated, and can be brought back to smaller values when the recognition method gradually 'zooms in' on a specific part of the search space.
- The selection size σ determines the influence of fitness on the chance of procreation. If σ is set to a large value, then even the less successful features in a population have a chance of generating offspring in the next generation. In this case, the genetic diversity (i.e. the number of different genes) of an offspring feature population remains high, but the selective power of the evolutionary mechanism decreases. In other words, the selection size controls the speed with which the evolutionary search converges. For large values of σ, the convergence is faster, but the chance of getting stuck in a local minimum of the search space is higher. A value for σ can be set by the user

(for example to fine-tune the feature recognition to a specific type of feature), or it can be set automatically. In the latter case, the selection size is initially set to a small value and remains small unless the fitness rapidly increases from one generation to another. In this case it can be concluded that the genetic elements that contribute to a better solution are present in the fittest individuals. The selection size is increased if the fitness increases slowly, because in this case the search is likely to be stuck in a local minimum. In this case, it pays off to try different parameter values in an attempt to break free of the local minimum.

- The mutation probability χ indicates how likely it is that mutation occurs during the creation of an offspring population. Mutation increases the genetic diversity and brings new genes into the gene pool, which may or may not contribute to obtaining an optimal solution. If χ is high, then a higher number of new genes is introduced in each generation. This can be either an advantage or a disadvantage. The higher the number of new genes, the less likely it is that the feature recognition gets stuck in a local minimum. However, too many new genes may lead to an overload of new search directions to be investigated and promising search directions may therefore unjustly be discarded. If the mutation rate is automatically set, then, similarly to σ, χ is set to a small value when the increase in fitness is fast, and to a large value in case of a slowly increasing fitness in an attempt to escape a local minimum.
- The mutation rate ϕ determines the amount of mutation that occurs. The higher the mutation rate, the larger the difference between a mutated gene and the parent genes that contributed to it. The behavior of the mutation rate is similar to that of the mutation probability.

The different problems that will be targeted in the remainder of this section will differ from this general procedure in the following aspects:

- In the problem of feature identification, features in a feature population are not required to have the same feature type.
- In step 1 and step 2 of the feature recognition procedure, only part of the genes of an organism are subjected to mutation
- In step 6 of the feature recognition procedure, features are instantiated on an embedding surface before their fitness is computed

4.3.4 Feature identification

Based on the general procedure, a variant of the evolutionary computation procedure can be constructed that automatically retrieves the type of a feature on a target surface. This process is called feature identification, and has roughly the same structure as the feature recognition method, but differs in some important aspects. Most importantly, the goal of feature identification is not to find a global minimum for the fitness function, but to find a feature type that is more successful than other feature types. The search space of feature identification can be thought of as consisting of several regions, each of which represents a certain feature type. The search for the correct region, i.e. the region which represents the type of the feature on the target surface is shorter than that for the global minimum of the search function (see Figure 4.13).

The second difference between feature recognition and feature identification is that the shape of a template feature does not have to match the target surface precisely. Therefore, the parameter mapping of the feature does not need to be changed. The genetic structure of a feature during a feature identification procedure consists of genes for its parameters only, but not of genes for its parameter mappings.

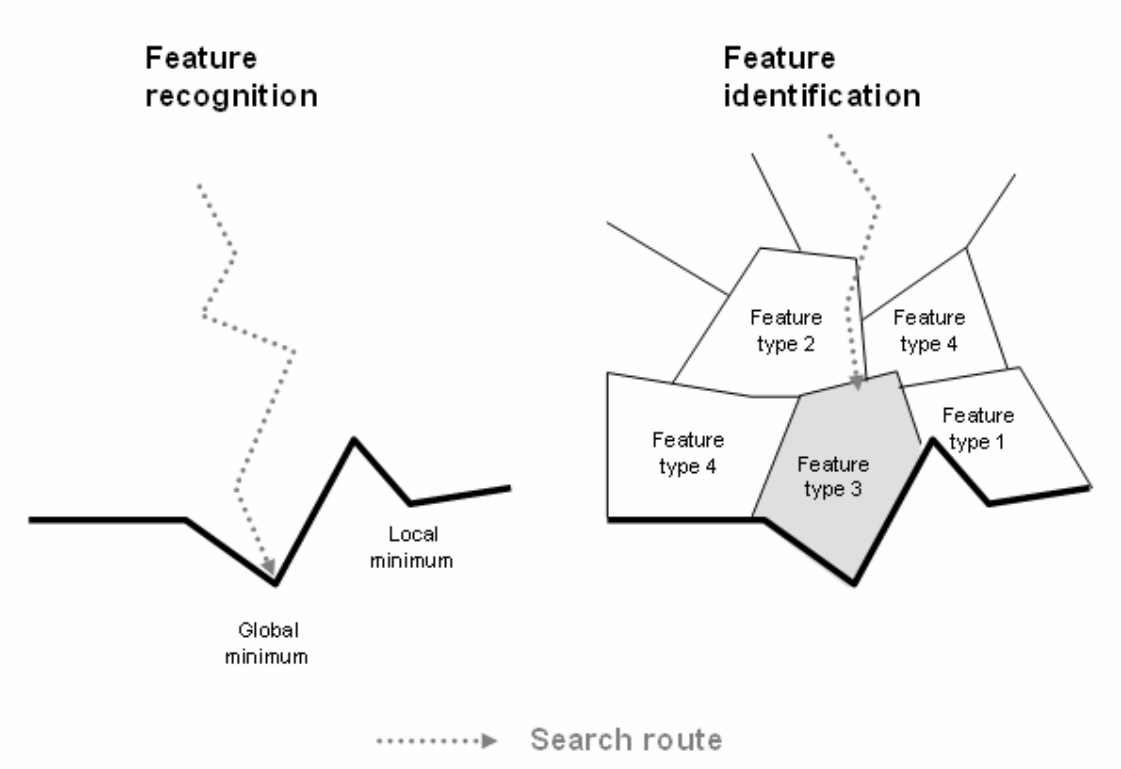


Figure 4.13*: Example of the difference between feature recognition and feature identification.*

A feature identification procedure differs from the general procedure presented in the previous sections in the following aspects:

- Only the genes that represent the parameter values of a feature are contained in the genetic structure.
- Because the dimensionality of the search space is small, a feature identification procedure requires only a small population size, and the computation time is considerably smaller than it is for a feature recognition procedure.
- Because no minimum of the search function has to be found, a fast convergence is required to quickly find a solution. The variables of the procedure can be set accordingly.
- The termination criteria for feature identification are different from that of the general procedure.

An algorithm for feature identification can be given as follows:

Algorithm 4.8: Feature identification (S, L, ε)
S is the target shape
L is the feature library
ε is a user-given threshold

1. Generate a population of features gen_1, where each feature is a copy of a random feature in the feature library L.
2. Compute the fitness and rank the features per feature type.
3. Create an offspring population gen_{i+1}, where each feature can only be coupled to a feature of the same type.
4. If all features in the offspring population have the same type, then return this type.
5. If $f(F^1)$ in gen_{i-1} minus $f(F^1)$ in gen_i is smaller than ε, then return the feature type that occurs most in gen_{i+1}.
6. Else jump to step 2.

The result of a feature identification procedure is a feature type. This feature type can then be used in an evolutionary feature recognition procedure.

4.3.5 Finding a corresponding template feature

As was discussed in section 4.3.2, in the first two steps of an evolutionary feature recognition method the parametric configuration and parameter mapping of a template feature are determined for which the template feature best matches a target surface. In the first step, an evolutionary computation procedure is used for a template matching procedure in which the optimal parametric configuration is determined. In the second step, the template that was found in step 1 is subjected to another evolutionary procedure that modifies the parameter mappings of the feature in order to better match it to the target surface.

Based on the general structure of evolutionary computation, we can develop a specific approach to the problem of freeform feature recognition. As stated before, feature recognition is initially performed in two steps.

Step 1
Given a subset *ROI* of a target shape S, an evolutionary procedure is used to find a configuration $F^{T1}\left(E^{0T1}, P^{T1}, \mu^{T1}\right) = E^{T1}$ for which the distance $d(E^{T1}, ROI)$ is minimal, similar to the template matching method. To be able to do so, a population of features is generated, which are copies of a specific pre-defined feature type in the feature library. The genes which represent the parameter values of these features are assigned a random value. These values are chosen within a range of $[0,360]$ for the basic rotational parameters and $[-1000,1000]$ for all other parameters. A range of $[0,360]$ is chosen because all meaningful rotations fall within this range of degrees in a polar system of coordinates. A range of $[-1000,1000]$ is chosen because all normal use of features is assumed to fall within this range. Without loss of generality, a larger range can be chosen if necessary.
In this step the goal is to find optimal parameter values, and therefore mutation only occurs for the relevant genes. The genes that represent the parameter mappings are identical for all features; if no mutation is applied to them, then these genes do not evolve and are identical for all feature instances at all times during the evolutionary procedure.
The first step in a feature recognition procedure differs from the general procedure presented in the previous sections only with regard to the mutation. An algorithm for this step can be given as follows:

Algorithm 4.9: Evolutionary template matching (S, F, ε_1, ε_2)
S is the target shape
F is the feature type that has been identified using Algorithm 4.8
ε_1 and ε_2 are user-given thresholds

1. Generate a population of features gen_1, where each feature is a copy of F.
2. Compute the fitness and rank the features.
3. Create an offspring population gen_{i+1}, where mutation is only applied to the relevant genes.
4. If one of the stop criteria is met, then return F^1 in gen_{i-1}
5. Else jump to step 2.

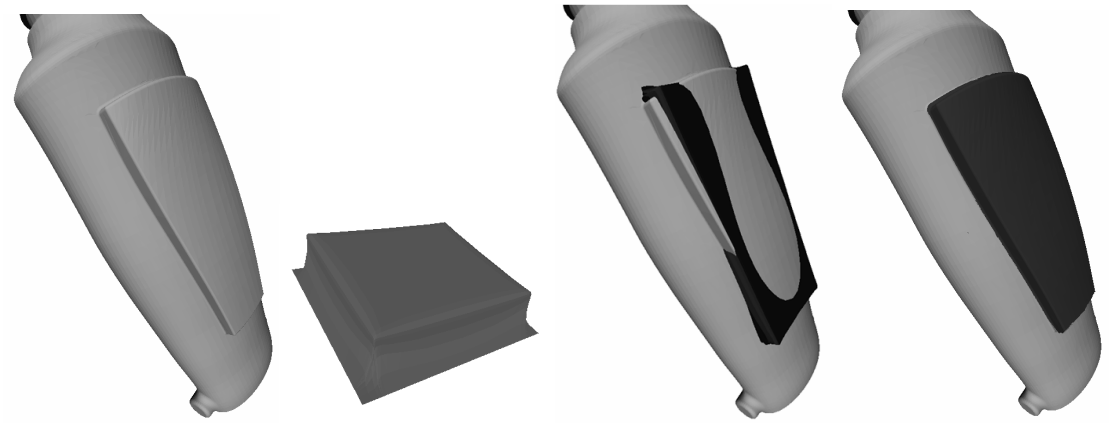
The result of the evolutionary procedure is a feature that differs from the specified predefined feature only with respect to its parametric configuration. Note that Algorithm 4.9 differs from Algorithm 4.7 only in the search strategy that is used. The disadvantages of the previously proposed template matching approach therefore also apply to this evolutionary approach.

Step 2
Starting out from the configuration that was found in step 1, a new evolutionary procedure is commenced. Again, the goal is to find a configuration $F^{T2}\left(E^{0T2}, P^{T2}, \mu^{T2}\right) = E^{T2}$. In this case, the genes that represent the parameter values are fixed copies of F^{T1}. Only the genes that represent the parameter mappings are subjected to mutation. During this second step, the parameter mappings of the feature are adapted in successive generations until a shape configuration is found which best matches the target surface. When a parameter mapping gene is subjected to a mutation rate ϕ, then the mutation has an influence on the transformation matrix that is applied by the mapping function, such that $\varphi\mu^l_{i,j}(e_{i,j}, p^l) = \varphi T^l_{i,j} e_{i,j}$. As a result, the effect of a parameter on a shape configuration element increases or decreases (depending on the sign of the mutation).
By including the parameter mappings in the evolutionary computation process, these are adapted during the recognition process. As a result, feature shapes evolve during the matching process and can be adapted to the target surface. However, although the shape of the feature changes, its position in parameter space remains constant and the parametric structure, i.e. the nature of the relation of parameters with shape remains the

same. An algorithm for this step only differs from Algorithm 4.9 in its input and in what genes are subjected to mutation, and it is therefore not given here.

Because the parameter mappings are altered, the definition of F^{T2} no longer corresponds to a predefined feature in the feature library, although it has been derived from such a definition. The feature definition is automatically adapted to the target surface. By doing so, we have overcome the main disadvantage of template matching because a strict similarity between pre-defined feature shape and target shape is no longer needed. In the next chapter, we will also show that the method presented here has a better average computation time and a better accuracy than the template matching method.

Figure 4.14 shows examples for the results of the first two steps of the evolutionary freeform feature recognition procedure. These examples show how a feature is recognized by first computing a rough match with a similar feature in the feature library (step 1), which is then adapted to the target surface to obtain a more detailed match (step 2).



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

Figure 4.14*: Example of the first two steps of a freeform feature recognition procedure: (a) the original model, (b) the feature to be recognized, (c) the result of step 1 of the recognition procedure and (d) the result of step 2 of the recognition procedure*

In step 2, the number of genes is large. Therefore, if the population size is too small, then the evolution of a feature may be biased towards a specific gene, while leaving other genes in a less-than-optimal state. This goes at the cost of the accuracy of the method. At the same time, a large population size leads to large computation times, which reduces the efficiency of the method.

To be able to increase both the efficiency and the accuracy of the method, the following two observations can be made:

- Since the parameters are independent, parameter mappings of the different parameters can be 'evolved' in separate procedures. In other words, an evolutionary recognition procedure using $n \times m$ genes can be broken down into m procedures on n of these genes. All of these procedures can be computed parallel.
- The effect of a single control point on the shape of the feature is local. To be able to compute the fitness of a feature after one of its parameter mappings has been altered, only the region of the target surface that is local to the control point for which the parameter mapping is defined has to be considered.

4.3.6 Detection of the base surface

The first two steps of the evolutionary feature recognition procedure result in an optimal configuration of a template feature. However, these two steps do not lead to a semantic understanding of a feature on the target surface, because the basic shape configuration has not been retrieved. When the basic shape configuration of a feature is known, then the efficiency and accuracy of feature-based shape manipulation or other feature-based operations can be increased. In particular, contrary to existing feature-based deformation methods, the identification of the base surface of a feature makes it possible to deform target shapes while maintaining the semantics, functional and other information stored with the feature.
In order to detect the base surface of a feature, we proceed in the following steps:

Step 3
The original target surface is stored in S^o, for it is needed to find an attached feature later in the procedure. Using parameter-driven deformation, the shape of the feature F^{T2} that resulted from step 2 can be used to manipulate the target surface such that it corresponds to the area configuration of the template, i.e. the configuration for which all deformation parameters have a zero value. However, it is unlikely that in step 2 it holds that $d(E^{T2}, ROI) = 0$. As a result, the target surface that is the result of the mentioned shape manipulation is not smooth.

Step 4
To obtain a smooth surface from the irregular surface that results from step 3, Laplacian smoothing is applied to the target surface. The result of the smoothing procedure is the surface S^{base} that is an approximation of the embedding surface of the feature that is

being recognized. It must be noted that other configurations S^{base}, obtained by different smoothing methods, can also be a reasonable approximation of the embedding surface.

In Figure 4.15, an example is shown of a feature that has been instantiated on a randomly generated embedding surface and then subjected to a feature recognition procedure. Figure 4.15a shows the original target surface, Figure 4.15b shows the surface after removal of the effect of the deformation parameters of the recognized features through a correspondence function and Figure 4.15c shows the surface after it has been smoothed.

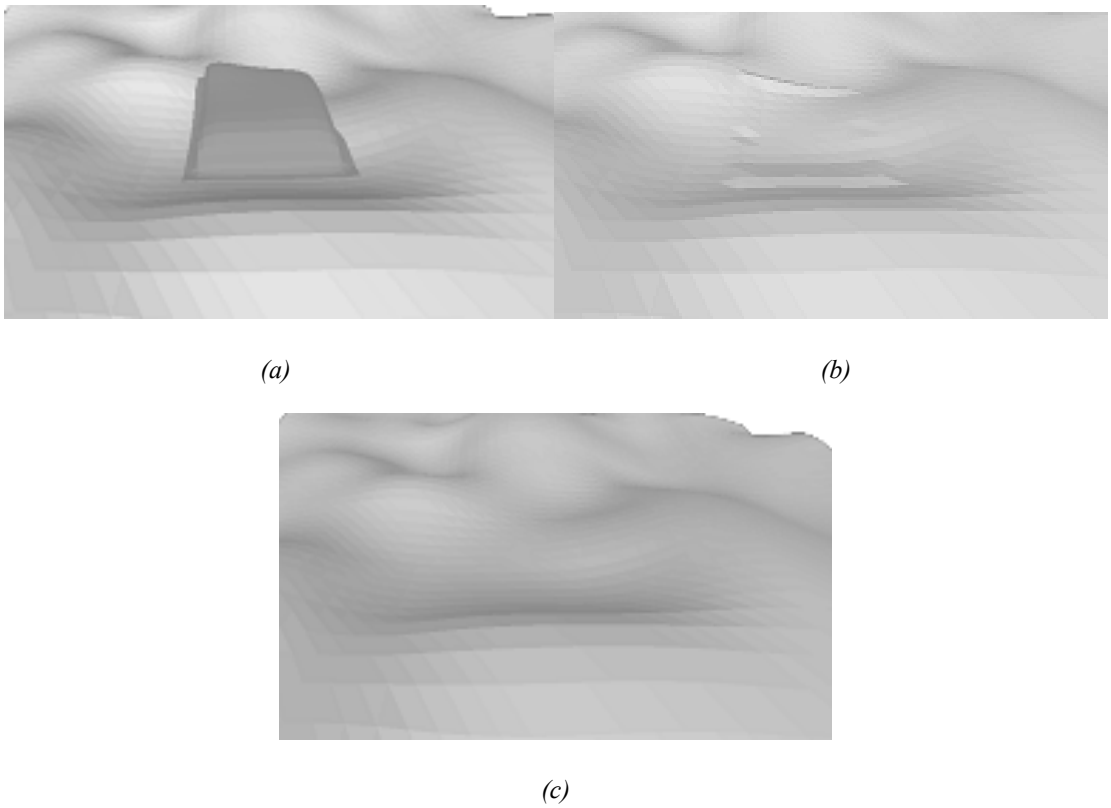


Figure 4.15*: Examples of (a) a target surface with a feature (b) the removal of the deformation effect of the recognized feature (c) the smoothed target surface*

Step 4 produces a smoothed surface S^{base} that is an approximate embedding surface of the feature that was recognized in steps 1 and 2. However, in approximating the embedding surface, it is assumed that the embedding surface is smooth at the position of the feature. Although it is merely an approximation and theoretically any embedding surface suffices as an approximation, the local environment is not taken into account. When, for example, the feature lies on an embedding surface with a relief or regular pattern, then the approximated embedding surface will not exhibit this pattern. This is a complicated, open issue, that will not be discussed in this thesis.

In steps 5 and 6, a feature template will be instantiated on S^{base} and a configuration of the feature will be found such that its shape best resembles S^o.

4.3.7 Finding an attached template feature

Once S^{base} has been found, another evolutionary feature recognition procedure can be used to find an optimal configuration of a feature template that is embedded in S^{base}. The starting values for the parameters of this attached feature can be derived from the feature template that was found in step 2. The attachment point of the feature can also be derived from the position of the result of step 2 as follows:

Step 5
Because in step 2, a corresponding feature was found, but not an embedding surface, no point of origin for the recognized feature on the embedding surface is available. However, F^{T2} does have a local coordinate system, of which the origin can be projected onto the embedding surface S^{base} that is the result of step 4 in the direction of the normal of the template feature that was found in step 2.
Once the attachment point has been determined, F^{T2} can be instantiated on S^{base} as an embedded feature. The resulting feature F^{T3} has the same parameter values and the same parameter mapping as F^{T2}.

Step 6
Because the shape configuration of F^{T3} has been adapted to S^{base} it is different from that of F^{T2}. In general it holds that $d(E^{T3}, S^o) > d(E^{T2}, S^o)$. An optimal configuration of F^{T3} therefore has to be found, such that $d(E^{T3}, S^o)$ is minimal. This process is almost identical to that of step 1 and step 2 of the feature recognition procedure, the only difference being that the subject of the search is an embedded feature rather than a corresponding feature. First F^{T3} is subjected to an evolutionary procedure similar to that used in step 1, i.e. only the genes that represent the parameter values are subjected to mutation. Once an optimal configuration, F^{T4}, has been found, this feature is subjected to another evolutionary procedure similar to that used in step 2, i.e. only the genes that represent the parameter mappings are subjected to mutation. This leads to a template feature F^{T5}, which is the final result of the feature recognition procedure.

4.4 Curve-based feature recognition

As was discussed earlier, the concept of template features has several disadvantages, one of which is that it cannot be used to recognize features with multi-dimensional parameters. Unfortunately, this disadvantage of template matching also applies to the evolutionary method presented in the previous section. In this section we therefore propose an additional feature recognition method that specifically targets a category of features with one two-dimensional parameter. This method is not exhaustive in the sense that it is able to recognize a feature with any number of multi-dimensional parameters, but it shows that the improvements that are offered by the evolutionary feature recognition method can also be applied to the specified category of features. That is, the method is able to adapt to specific target surfaces and therefore does not require an active role of the user in the definition of new feature types. In addition, it recognizes embedding features as well as corresponding features. Finally, it also retrieves the embedding surface of the feature. The specific type of feature that is targeted is called a curve-based feature and will be discussed in section 4.4.1. In section 4.4.2, a method to recognize this type of feature will be given.

4.4.1 Definition of a curve-based feature

Multi-dimensional parameters are parameters to which values can be assigned in different dimensions. In this section we limit ourselves to two-dimensional parameters, which can be thought of as curve-based parameters. An example of a feature with a two-dimensional parameter is the Ridge feature, the shape of which can be considered to be a profile that is swept along a two-dimensional curve. In this case, the profile of the ridge is determined by parameters such as height or width, while the curve along which the profile has been swept can be seen as a two-dimensional parameter. Figure 4.16 shows some examples of a ridge feature. The problem with two-dimensional parameters is that they cannot be pre-defined. For example, the features shown in Figure 4.16 are all Ridge features, i.e. can be considered to be instances of the same feature type, yet the relation between the curves of the features cannot be specified other than by the vague notion of similarity: there is an infinite amount of curves with which a Ridge feature can be defined.

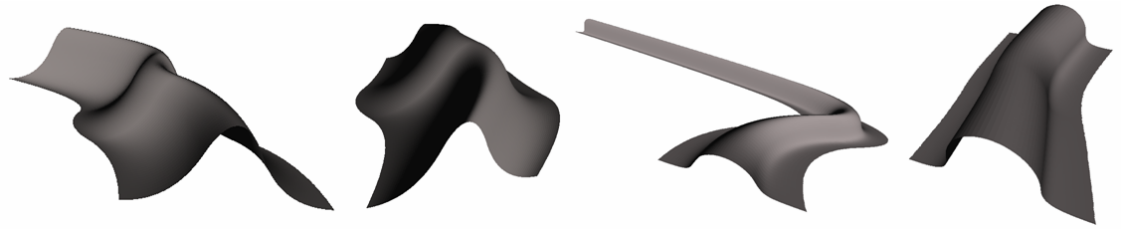


Figure 4.16*: Examples of Ridge features with different curves*

In this section we formalize features with a two-dimensional parameter in the form of a curve-based feature, which can be specified by giving an alternative implementation of Definition 3.7. For each curve-based feature, a two-dimensional parameter is given in the form of a *trajectory curve* $\Theta(u) = \sum_{i=0}^{n} N_{i,3}(u)\theta_i$, where θ_i are the control points of the trajectory curve. In addition, a set of h profile curves $\Upsilon_{0<j<h}(u) = \sum_{i=0}^{n} N_{i,3}(u)\rho_i^j$ is given, where ρ_i are the control points of a profile curve. The profile curves are perpendicular to the trajectory curve. Because of this perpendicularity, the trajectory curve and the profile curves can be organized in a bi-directional grid, such that $E^0 = \begin{pmatrix} \rho_0^h + \theta_{r-1} & \cdots & \rho_{c-1}^h + \theta_{r-1} \\ \vdots & \ddots & \vdots \\ \rho_0^0 + \theta_0 & \cdots & \rho_{c-1}^0 + \theta_0 \end{pmatrix}$. In other words: although the profile curves and the trajectory curve are stored as variables in their own right, they also become apparent in the shape configuration of the feature. Because curve-based features can be defined in terms of definition 3.7, the theory that was presented in chapter 3 can be applied to curve-based features.

The only difference with features as they were defined before is that parameters are assumed to have an influence either on the profiles or the trajectory. The parameters can be subdivided into two types, *profile parameters* and *trajectory parameters*. Profile parameters only influence columns of the control point grid; the trajectory parameters only influence rows of the control point grid. That is, the rows and columns of the direction and function mask matrix are identical. For example, for a 5×5 control point grid it holds that: $\delta_{i1} = \delta_{i2} = \delta_{i3} = \delta_{i4} = \delta_{i5}$ and $\varphi_{i1} = \varphi_{i2} = \varphi_{i3} = \varphi_{i4} = \varphi_{i5}$ for profile parameters and $\delta_{1j} = \delta_{2j} = \delta_{3j} = \delta_{4j} = \delta_{5j}$ and $\varphi_{1j} = \varphi_{2j} = \varphi_{3j} = \varphi_{4j} = \varphi_{5j}$ for trajectory parameters.

In the previous sections, arguments were given for the use of a feature library. However, if features can no longer be pre-defined because of the dimensionality of their parameters, then a library can not be maintained. As a solution to this problem, instead of storing an entire feature type definition, we store one characteristic profile for each feature type. Parametric information (but also semantic, functional or other information) can be stored with this profile.

4.4.2 An algorithm for curve-based feature recognition

If only the profile curves are stored in the library, then prior to a feature recognition procedure, a template feature must first be constructed. To be able to construct a template

feature, first the trajectory curve and the profile curves of a feature must be retrieved from a target shape.

To be able to follow the trajectory of a feature, we assume that in each profile of the feature, one or more characteristic singularities can be identified. These singularities occur in successive profiles along a trajectory and can be used to identify the trajectory of the feature. The singularities are operationalized as a discontinuity in one of the derivatives of the target surface. Although discontinuities in the surface are most obvious when detected using the first derivative, they can just as well be identified in higher derivatives. Figure 4.17 shows three curves (top row) and their first, second and third derivatives below. In each of the examples, a discontinuity in the surface can be identified by determining the highest derivative that crosses the x-axis, i.e. becomes zero at some point. This point corresponds to a discontinuity in the original curve. We say that the *order* of a discontinuity is the lowest derivative for which such a point occurs.

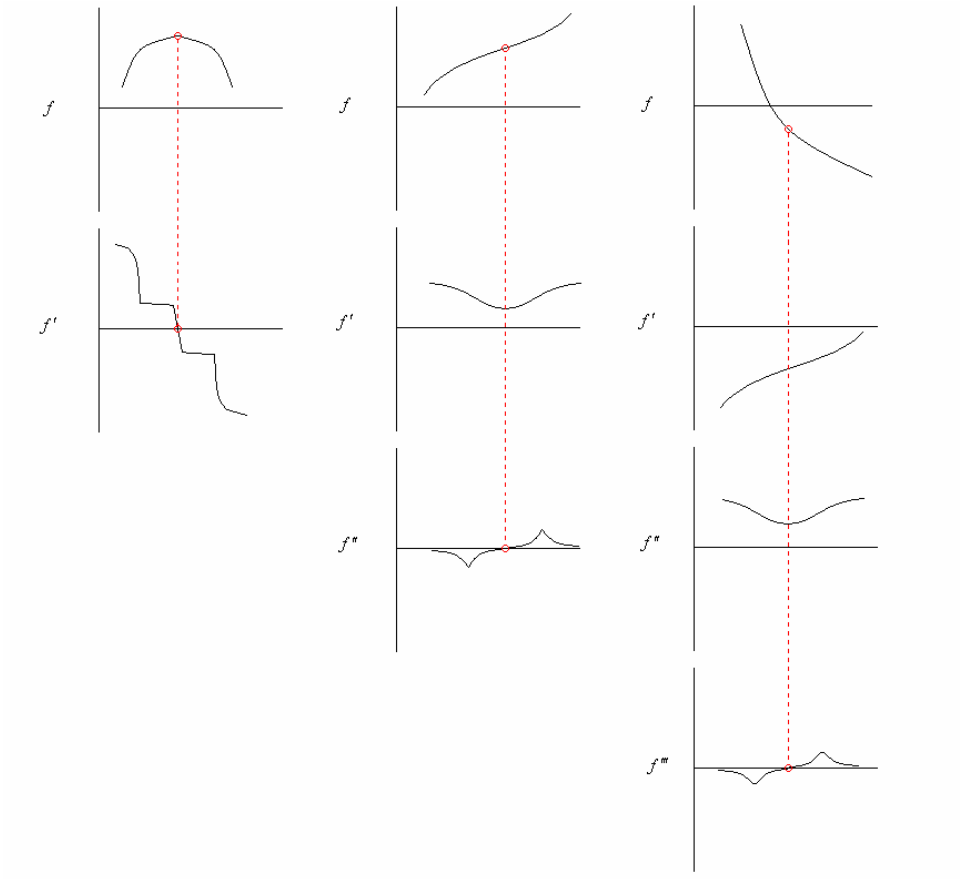


Figure 4.17: *Examples of discontinuities that become apparent in (a) the first, (b) the second and (c) the third derivative of a curve.*

Because the examples given in figure 4.17 are two-dimensional, it may not be apparent how this can be applied to determining the feature trajectory. According to definition 6.2, profile curves are perpendicular to the trajectory curve; all curves $\Upsilon_i(u)$ can therefore be defined to lie in a two-dimensional cross-section of the surface, perpendicular to $\Theta(u)$ in the point θ_i. As a result, profile curves can be assumed to be two-dimensional curves.

The feature trajectory can be found by analyzing cross-sections of the target surface. To obtain these cross-sections in an organized and efficient way, the trajectory recognition method proposed in this section sweeps a plane over the target surface and analyses the resulting cross-sections.

In this section, it is assumed that instead of indicating a region of interest, the user indicates a cross-section of the curve-based feature to be recognized. That is, the user is asked to position an initial location of an intersection plane *I* that intersects the trajectory to be recognized. The centre of the intersection plane is denoted O^I and its normal vector n^I. The normal vector of the plane approximately points in the direction of the supposed feature trajectory. This initial configuration of the intersection plane is denoted I_0, with origin O_0^I and normal n_0^I.

An algorithm for trajectory recognition can be given as follows.

Algorithm 4.10: Curve-based feature recognition (S, L, I, δ, λ)
S is the target shape
L is the library of profiles
I is a plane that indicates a cross-section of the trajectory
δ and λ are user-given constants

1. Compute a cross-section of the target shape S through I_0.

2. Interpolate a B-spline curve with λ control points $\Upsilon_0(u) = \sum_{i=0}^{n} N_{i,3}(u)\rho_i^0$ through this cross-section.

3. Align the curve to the xy-plane by the transformation matrix

$$M_0^{ItoXY} = \begin{pmatrix} (1-\cos(\gamma))x^2+\cos(\gamma) & (1-\cos(\gamma))yx-z\sin(\gamma) & (1-\cos(\gamma))zx+y\sin(\gamma) & -Ox \\ (1-\cos(\gamma))xy+z\sin(\gamma) & (1-\cos(\gamma))y^2+\cos(\gamma) & (1-\cos(\gamma))zy-x\sin(\gamma) & -Oy \\ (1-\cos(\gamma))xz-y\sin(\gamma) & (1-\cos(\gamma))yz+x\sin(\gamma) & (1-\cos(\gamma))z^2+\cos(\gamma) & -Oz \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where $\gamma = \arccos\left(n^I \cdot \bar{z}\right)$, such that $M_0^{ItoXY} O^I = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ and $M_0^{ItoXY} n^I = \bar{z}$.

4. Compute the curve $\Upsilon_0^{XY}(u) = M_0^{ItoXY}\Upsilon_0(u)$, each control point of which has a z-coordinate that equals 0.

5. Compute the derivative $\Upsilon_0'^{XY}(u) = \sum_{i=0}^{n} N_{i,3}(u)\rho_i'^{XY}$ by computing each control point $\rho_i'^{XY}$ as $\rho_i'^{XY} = \begin{pmatrix} \rho_i^{XY}(u)x \\ \sum_{i=0}^{n-1} N_{i,2}(u)\dfrac{3}{u_{i+3+1}-u_{i+1}}(\rho_{i+1}^{XY}(u)y - \rho_i^{XY}(u)y) \\ 0 \\ 1 \end{pmatrix}$, where

$\rho_i^{XY}(u)x$ and $\rho_i^{XY}(u)y$ are respectively the x- and y-component of $\rho_i^{XY}(u)$.

6. Compute the set of values $u^{Y=0}$ for which it holds that $\Upsilon_0'^{XY}\left(u^{Y=0}\right) = 0$. If no such value exist, then $\Upsilon_i''^{XY}$ is computed and so on. From this set, the value u^t is chosen as the point for which the distance $dist(\Upsilon_i^{XY}\left(u^t\right), O_0)$ is minimal. The point $\theta_0 = \Upsilon_0(u^t)$ is the first element of the feature trajectory.

7. The curve Υ_0^{XY} is translated by the transformation matrix

$$M_0^{XYtoO} = \begin{pmatrix} 1 & 0 & 0 & \Upsilon_0^{XY}\left(u^t\right)x \\ 0 & 1 & 0 & \Upsilon_0^{XY}\left(u^t\right)y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \text{ such that } M_0^{XYtoO}\Upsilon_o^{XY}\left(u^t\right) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$ This centers the profile around θ_0.

8. The intersection plane is moved over a distance δ in the direction of n_0^I to the configuration I_1. Steps 1-7 are repeated for the new configuration of the

intersection plane. This time, if more than one value $u^{Y=0}$ exist, then the value for which $\Upsilon_1(u^t)$ has the smallest distance to θ_0 is chosen to compute the feature trajectory point θ_1.

9. n_1^I is set to $\theta_1 - \theta_0$ and the origin of the intersection plane is set to θ_1. Then the plane is moved over a distance δ in the direction of its new normal.

10. For $i \geq 2$, steps 8 and 9 are repeated (a) for a fixed number of steps, (b) until no value u^t can be found in the derivative in the degree of the trajectory or (c) the intersection of I and S is empty.

11. The intersection plane is reset, so that $O^I = O_0^I$ and $n^I = -n_0^I$, and the steps 1-10 are repeated.

12. Each profile $P_i(u)$ is transformed back to its original location by multiplying it with $M_i^{ItoXY^{-1}} M_i^{XYtoO^{-1}}$.

In step 1 of the algorithm, the cross section of the target shape S through the initial intersection plane I is computed. In steps 2-5, a (two-dimensional) curve through this cross-section is transformed so that it comes to lie in the xy-plane. The xy-plane is chosen as a common point of reference for all curves, because computations can be done more efficient when the z-coordinate does not have to be taken into account. In step 6, a singularity on the curve is identified, and in step 7 the curve is centered at the identified singularity. In step 8-12, the intersection is swept step-by-step over the target surface, and in each step a profile curve is constructed.

Figure 4.18 shows the application of Algorithm 4.10 to a digitized model of a computer mouse. In this example, the algorithm was used to find a surface discontinuity running down from the lower corner of the left mouse button past the right side of the trackball-hole. Figure 4.18a shows the initial placement of the intersection plane and Figure 4.18b shows the result after several iterations of the curve-based feature recognition.

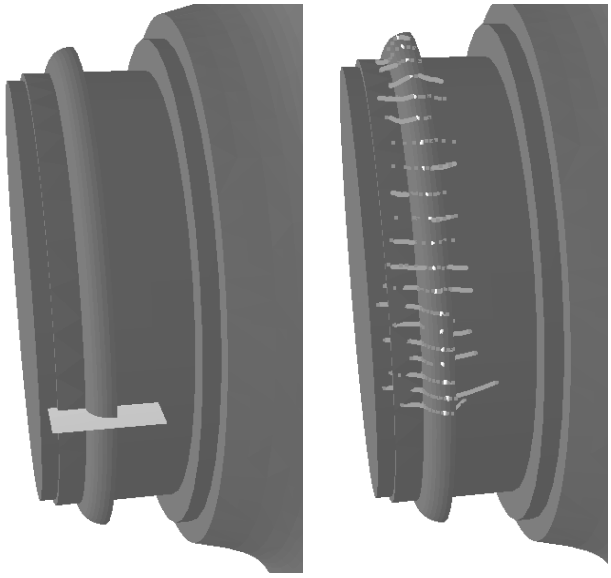


Figure 4.18: *Stages of a curve-based feature recognition procedure: (a) the initial intersection plane and (b) constructed profile curves after a few steps of the curve-based feature recognition*

A user has, to a certain extent, control over the trajectory recognition procedure through the variables δ and λ, which control respectively the row size and the column size of the shape configuration. During the recognition procedure, the variable δ determines the distance between the origins of two successive intersection planes. If the value of δ is large, then the distance between intersection planes is large and as a consequence the computation time of the method is low (as fewer intersection planes are needed to traverse a trajectory). However, there is a risk that the method goes 'off the road' and fails as it follows a feature trajectory with high curvature; even if this does not happen, then the accuracy in areas of high curvature may be low (see Figure 4.19). A large value for δ should be used when the trajectory of the feature to be recognized has a low curvature, and a small value should be used for trajectories with high curvature or if high accuracy is needed.

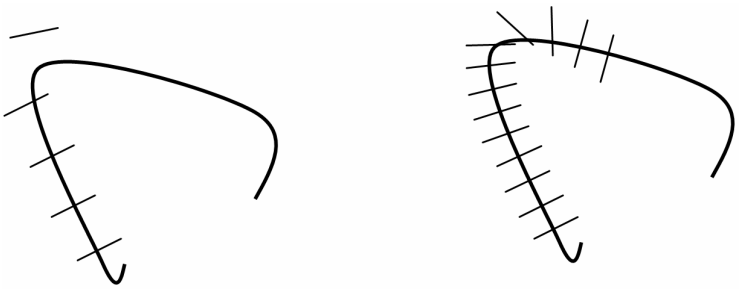


Figure 4.19*: Two-dimensional example of the effect of the variable*

The variable λ indicates the amount of control points that are used in the interpolation of the points on the cross-sections of the target surface. For a small value of λ, only a rough estimation of the feature profile is computed, making it less likely that profile similarity is maintained during the traversal of the trajectory. A large value of λ leads to a large computation time.
Basically, in determining optimal values for δ and λ, there is a trade-off between computation time and accuracy. A large value of λ and a small value of δ lead to a large computation time and large accuracy. A small value of λ and a large value of δ lead to a small(er) computation time and small(er) accuracy.

Once a basic shape configuration has been determined, all that remains is to construct parameters and parameter mappings for the different profile curves. Parameters and parameter mappings have been defined for profiles in the library, but before these can be used, a correspondence must be found between the profiles $\Upsilon_i(u)$ that were found in Algorithm 4.10 and the profiles stored in the library. This process is equivalent to the feature identification problem that was addressed in the previous chapter. However, because the problem of 'profile identification' is two-dimensional rather than three-dimensional, its complexity is much lower.
Once a profile in the feature library has been found, the only relevant difference between the profiles $\Upsilon_i(u)$ and the identified profile is in the number of control points of both curves. Control points can be added to a curve without modifying the shape. Therefore if the profile identified in the library has a smaller number of control points than the profiles $\Upsilon_i(u)$, control points are added to this profile until they have an equal number of control points, and vice versa. Once an equal number of control points has been obtained, then both the parameters and the parameter mappings can simply be copied. If $\Omega_0(u) = \sum_{i=0}^{n} N_{i,3}(u)\omega_i^0$ is the profile from the library, then a parameter mapping $\mu_{i,j}^l(\omega_{i,j}, p^l) = T_{i,j}^l \varpi_{i,j}$ can be copied, such that $\mu_{i,j}^l(\rho_i^j, p^l) = T_{i,j}^l \rho_i^j$ for any control point on any profile $\Upsilon_i^j(u)$.

Corresponding to the reimplementation of definition 3.7 that was proposed at the start of this section, in Algorithm 4.10 and in the above paragraph, a custom feature template is constructed. Once this template has been constructed, it can be subjected to an evolutionary feature recognition method as was proposed in the previous section. There are several reasons for doing so:

- In Algorithm 4.10, some inaccuracy may have been introduced that make that the constructed curve-based feature is not an optimal match to the target surface. This is mainly a consequence of computing the curve-based feature step-by-step. Even if the individual profiles are an optimal match to the target surface, this does not have to hold for the entire feature. By subjecting the curve-based feature to an evolutionary feature recognition procedure, the quality of the match can be improved.
- The curve-based feature that is constructed in Algorithm 4.10 is a corresponding feature. By subjecting it to an evolutionary feature recognition procedure, the curve-based feature can also be recognized as an embedding feature.

Conclusion

The curve-based feature recognition method was presented as a method to deal with a specific subcategory of features with a single two-dimensional parameter. This method cannot be generalized to features with multi-dimensional parameters, or at the least we cannot claim this without having investigated multi-dimensional parameters in more detail. However, it has been shown that the principles of the feature recognition approach that is suggested in this thesis can be applied to cases other than those that have been traditionally investigated within the topic of feature recognition.

Curve-based feature recognition differs from traditional template matching in the sense that less information is stored in the feature library. A feature template is constructed rather than derived from the feature library, and the (parametric) information that is stored in the feature library is generalized over the constructed feature template. It must be noted that this generalization is valid only under the assumption that the profiles in the constructed template feature are topologically related to the stored profile.

5 Application examples and validation of freeform feature recognition

At the start of this thesis, a problem description was given. To solve this problem, a theory was presented on the basis of which a methodology was developed that addresses the given problem. To complete the research and to support the contribution of this research to technological and scientific progress, the proposed work must be verified and validated. In section 5.1, application examples are given, both for evolutionary feature recognition and for curve-based feature recognition. To verify the theory, we will critically review the different assumptions on which the proposed theory is based in section 5.2.
In addition to the verification of the theory, the proposed work must also be validated. To evaluate the internal validity of the research, the consistency of the different steps in the research is discussed in section 5.3. In addition, the pilot implementation that was developed is tested on a large number of automatically generated test cases, to identify any systematic shortcomings of the methods. The results of these tests are evaluated with regard to correctness, efficiency and accuracy and given in section 5.4. In addition, in this section an analysis of the time complexity of the algorithms is given. Finally, in section 5.5 we discuss experiments that were conducted to test the accuracy of the input given by the user.

5.1 Application examples

In this section, two examples will be given of an application of the developed algorithms to two cases from the practice of industrial design. The CAD models that are used were provided by industrial designers, who indicated that the features that are targeted on these models are representative for the general case of features as they occur in CAD models in the industrial design practice. Both models were provided in the form of polygon meshes, respectively with 24650 and 48964 polygons. In section 5.1.1, the application of evolutionary feature recognition is demonstrated and section 5.1.2 deals with the application of the curve-based feature recognition method.

5.1.1 Application of evolutionary feature recognition

The first application example that is presented here deals with a model of a soap dispenser bottle, such as is typically used in for example bathrooms. In a practical usage situation, this plastic bottle contains soap and can be inserted into a casing which has the functionality of dispensing measured quantities of soap when the user pushes a button or

a movable part of the casing. When the bottle is empty, it can be removed from its casing and a replacement bottle can be inserted. The model is shown in Figure 5.1a.

The design company that provided this example had been asked by the producer of the bottle to redesign it due to changes in the requirements of manufacturing and usage of the bottle. Unfortunately, the CAD-model that was originally used to produce the bottle was not available; only a physical specimen of the bottle was available. In order to quickly create a new CAD model, a digital model of the bottle was obtained through laser range scanning, and freeform feature recognition was applied to this digital model. By using freeform feature recognition, the features on the original object could be reconstructed. As a result, the required changes to the model could be obtained by changing the parameter values of the reconstructed parameters.

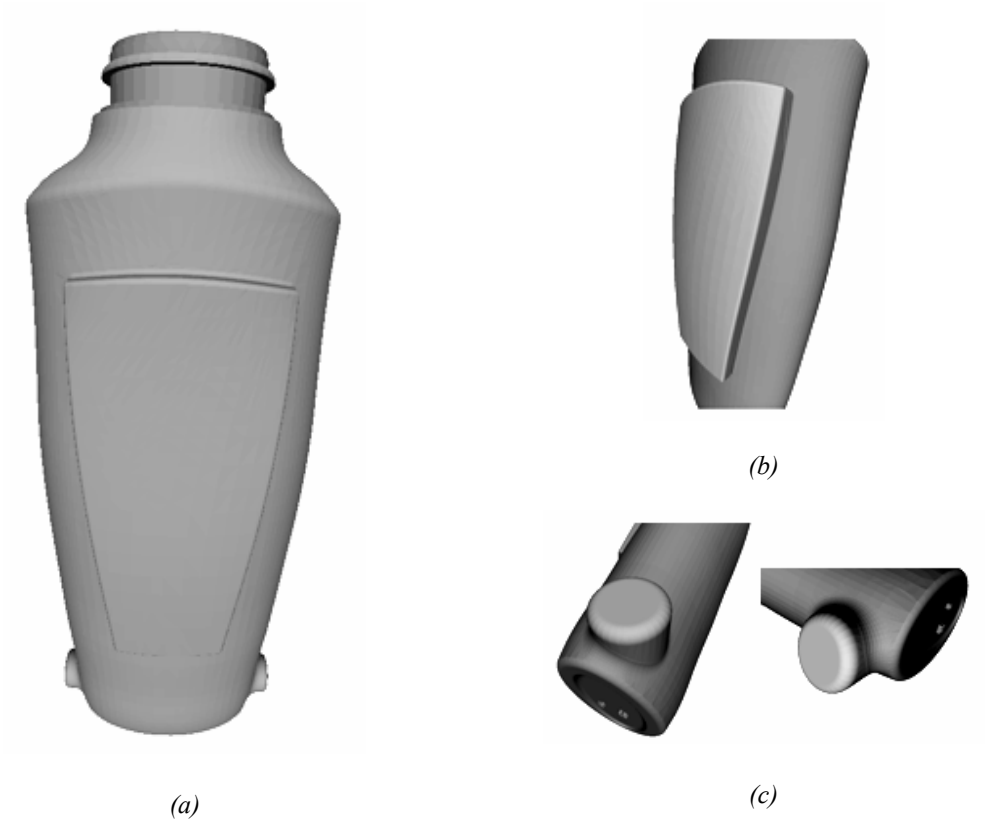


Figure 5.1*: (a) A reconstructed model of a soap dispenser bottle and (b) and (c) a close up of two freeform features on this bottle*

In this section, we focus on the recognition of two specific features. On the front of the bottle is a displaced area that is, in practice, used to attach a label sticker to the bottle (see Figure 5.1b). This feature is targeted because it is fully freeform in the sense that its shape can be defined using curved surfaces. Its embedding surface is slightly but

regularly curved. The other feature is a circular extrusion on the bottom of the bottle (see Figure 5.1c). In fact there are two of these features, but because both the features and their attachment to the embedding surface are identical we treat them as one example. In practice, this feature is used to attach the bottle in its encasing. This feature is targeted because although it is not fully freeform (it is rotationally symmetric), it is embedded in a region of the embedded surface that is highly curved. The main challenge therefore is not to recognize the feature itself, but its embedding surface.

To be able to recognize the targeted features, two feature type definitions were created (Figure 5.2). Both feature types were defined using a 5×5 control point grid. Both features were defined with two area parameters that control the width and length of the feature shapes, a deformation parameter that controls the height of the feature shapes and two weight parameters that control the roundness of the feature shapes.

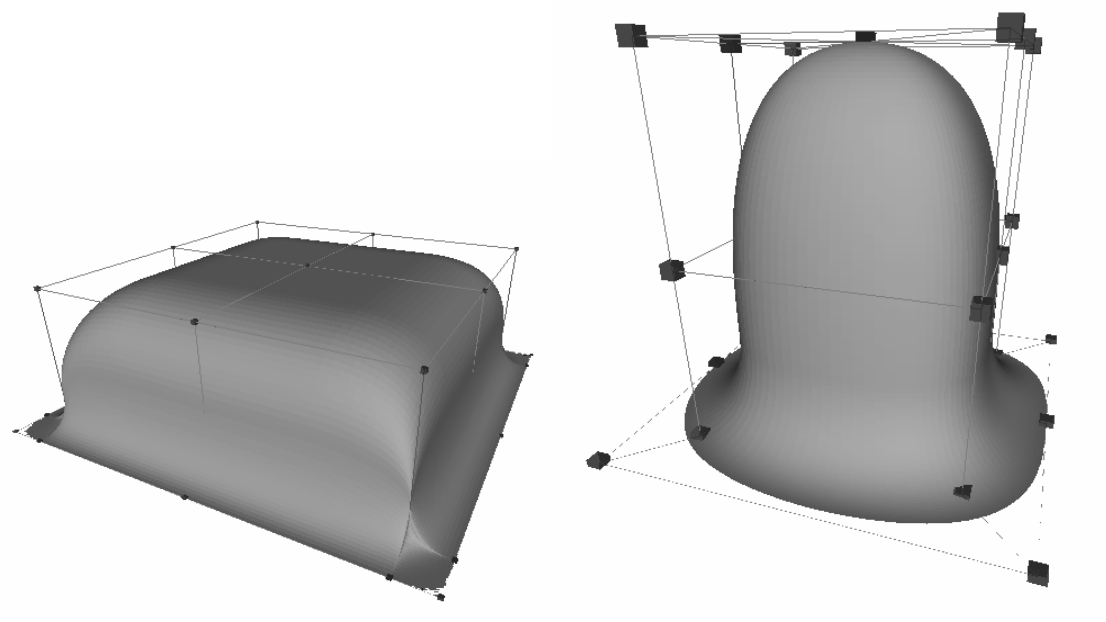


Figure 5.2*: Feature type definitions that match the features shown in Figure 5.1*

To recognize the two defined features, evolutionary feature recognition was applied to the target model in two separate procedures. The results of these two procedures will be given per step.

Step 1: Finding a corresponding feature

In the first step of evolutionary feature recognition, the defined feature types were instantiated as a template feature and matched to the target model. First, regions of interest were selected around the target features. These regions of interest contained

respectively 1208 (for the feature shown in Figure 5.1c) and 3615 (for the feature shown in Figure 5.1b) polygons. The result of this first step is shown in Figure 5.3a. Corresponding features were found in respectively 112 seconds and 127 seconds. The distance to the target model, in terms of the Mean Directed Hausdorff Distance, was respectively 3.12 mm. and 5.22 mm.

Step 2: Changing the morphology

In the second step, the features that resulted from step 1 are subjected to another evolutionary procedure, in which the morphology of the feature is adapted. The result of this step is shown in Figure 5.3b. The resulting feature configuration was found in respectively 194 seconds and 241 seconds and the distance to the target model was respectively 0.86 mm. and 1.07 mm.

(a) *(b)*

Figure 5.3*: Results of the first step (a) and the second step (b) of the feature recognition procedure*

Step 3 and step 4: Reducing the deformation and smoothing the embedding surface

In the third and fourth step of the procedure, the deformation of the feature on the target model is reduced, so that the resulting model corresponds to the area configuration of the feature that was recognized in step 1 and 2. The resulting target model is then smoothed

to obtain the embedding surface of the feature to be recognized. The result of step 3 and 4 is shown in Figure 5.4a and b.

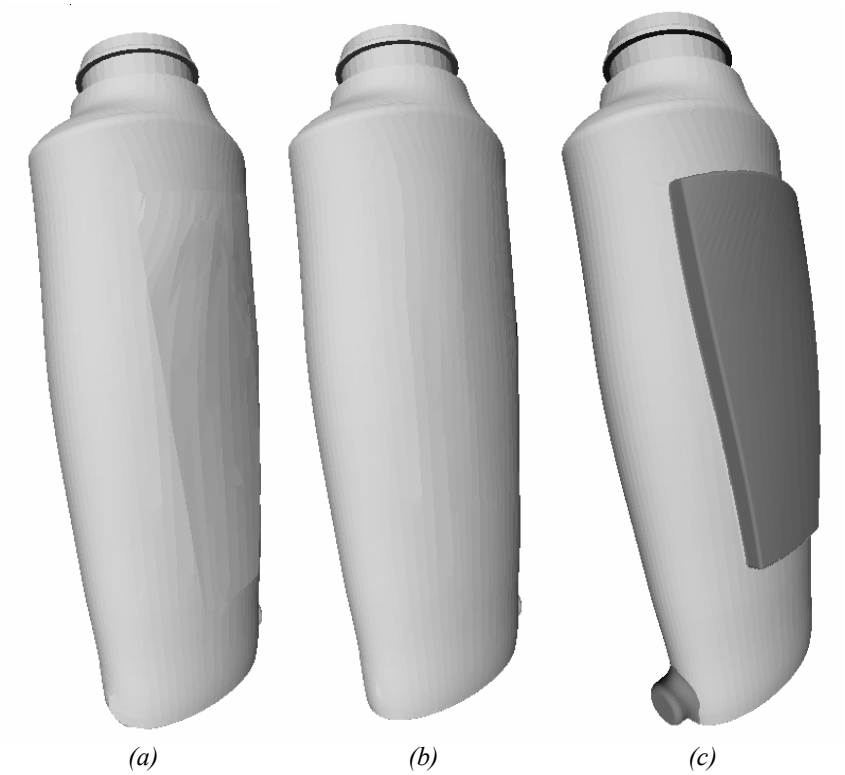


Figure 5.4*: Results of the third step (a), the fourth step (b) and the fifth and sixth step (c) of the feature recognition procedure*

Step 5 and step 6: Finding an embedded feature

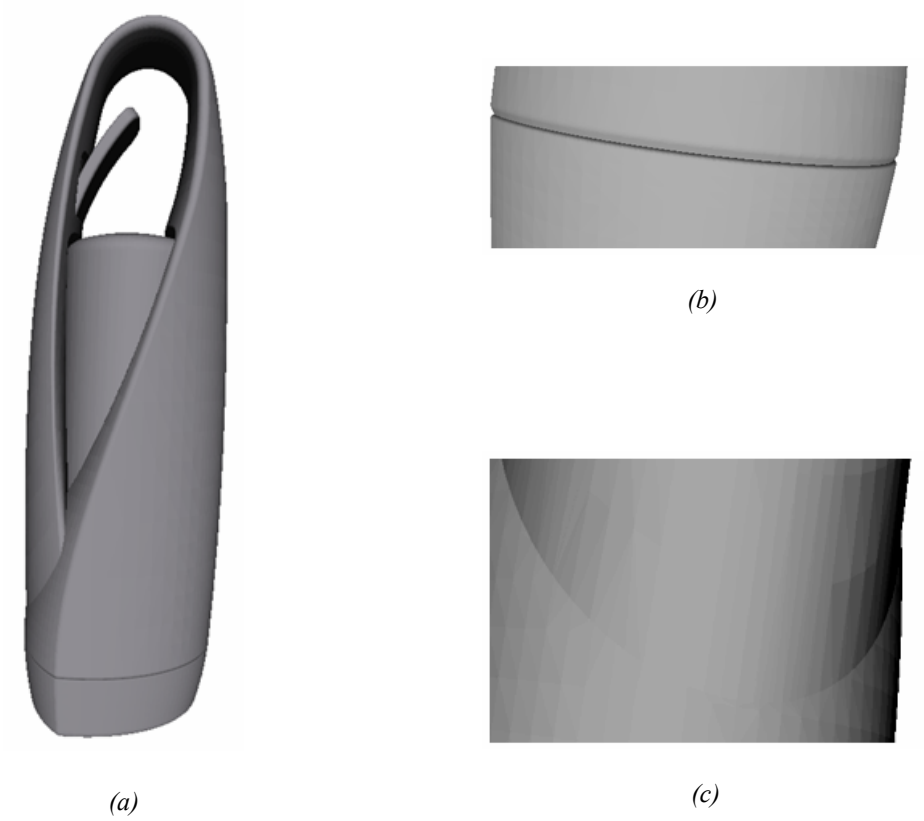
In the final steps of the procedure, an embedded feature is found on the embedding surface that results from step 4, starting from the configuration of the feature as it was found in step 2. The result of these steps is shown in Figure 5.5. Note that these results are visually almost indistinguishable from those of step 2, but that the features shown in Figure 5.5 are embedded features, whereas the features shown in Figure 5.4c are corresponding features. An embedded feature was found in respectively 221 seconds and 253 seconds, and the distance of the shape of these embedded features to the original target surface is 0.28 mm. and 0.64 mm., respectively.

With an accuracy of 0.28 mm. and 0.64 mm., the result of the feature recognition procedure is much more accurate then when only a corresponding feature was recognized (comparable to the 3.12 mm. and 5.22 mm. that resulted from step 1). In total, it took 557 seconds and 621 seconds to find an embedding feature.

A computation time of around 10 minutes is too much to be able to use the procedure in an actual design process, in particular when taking into account the fact that this is the computation time needed per feature. A model that contains many features will require a large computation time, but, as will be discussed in section 5.4 it must be noted that our pilot implementation was not optimized. In section 5.5 we will further discuss the performance and complexity of the algorithms.

5.1.2 Application of curve-based feature recognition

Another model that was obtained from the practice of industrial design is a model of a fire extinguisher. This model is shown in Figure 5.5a. On the model, two features were selected as a test case for the proposed feature recognition methods. The first feature (see Figure 5.5b) is a depression of the surface that runs along the bottom of the model in a closed curve. The second feature (see Figure 5.5c) is a transition feature between two distinct regions of the model.



(a) *(b)* *(c)*

Figure 5.5*: (a) A model of a fire extinguisher and (b) and (c) two close-ups of freeform features on this model*

Both selected features cannot be recognized by an evolutionary procedure. Although feature definitions can be created for this example, the selected features can generally not be recognized using pre-defined features. Instead, we apply a curve-based feature recognition procedure, in which a template feature is constructed specifically for these cases. First, initial intersection planes are positioned as shown in Figure 5.6.

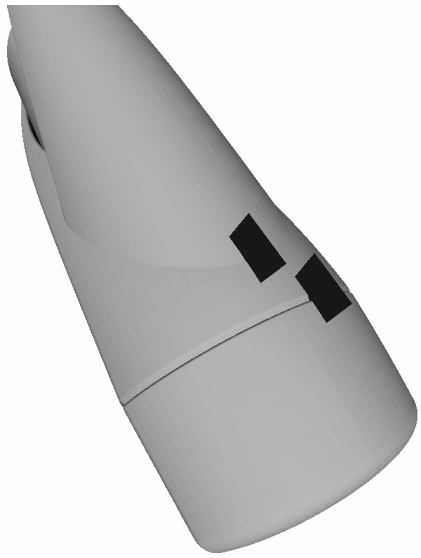


Figure 5.6*: Initial intersection planes, positioned with respect to the selected features*

Once the initial positions of the intersection planes have been determined, a curve-based feature recognition procedure can be started. The result of this recognition procedure is shown in Figure 5.7. To recognize the depression feature, several values for δ and λ had to be tried out, because in the initial runs of the algorithm, it was unable to deal with the sharp corners of the feature on the target model.

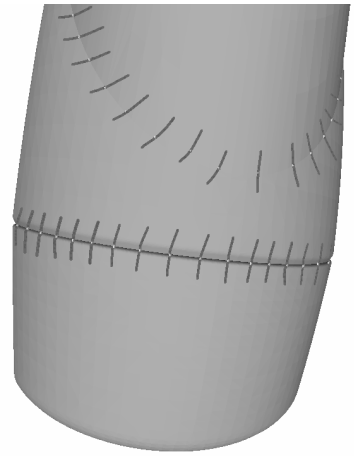


Figure 5.7*: Profile curves that were reconstructed in the curve-based feature recognition process*

Once a curve-based feature was constructed, the resulting feature could be subjected to an evolutionary feature recognition procedure. The constructed curve-based features were recognized in respectively 2 hour and 54 minutes (for the depression feature) and 1 hour and 38 minutes (for the transition feature). The reason for the fact that these computation times are larger than the computation times that were found in the previous application example is that the number of control points in the control point grids of the curve-based features is large (41×12 and 19×12, respectively). The distance between the recognized features and the target shape was respectively 1.88 mm. and 0.24 mm.

5.1.3 Conclusions

Both feature recognition methods that were proposed in this thesis were applied to two test cases. In both tests, the selected features were successfully recognized. However, in several cases, we had to experiment with the variables of the feature recognition procedure (the evolutionary variables Π, σ, χ and φ and the trajectory variables δ and λ). After several tries, we decided to implement a mechanism for the automatic setting and adapting of the value of these variables during the feature recognition procedure. In the previous chapter, it was discussed how this can be implemented. The results that are shown in this chapter were obtained with an implementation of the feature recognition procedures that included this mechanism.
The computation times that were found were too large for an application of the feature recognition methods in the practice of industrial design, particularly when considering that the computation times were given per feature. However, it must be taken into account that the pilot implementations are unoptimized. In addition, the tests were conducted on a normal desktop computer, while industrial design professionals can be assumed to work with more advanced hardware. In particular, parallel processing technology can significantly increase the efficiency of the feature recognition procedure.

5.2 Verification of the theory

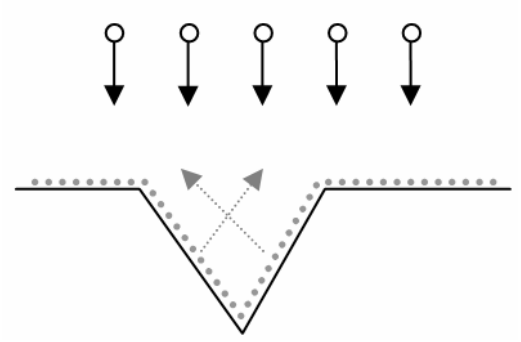
A verification of the proposed theory has several aspects, which are contained in the questions that were posed in the previous section: the theory must be logically true, consistent, applicable to the relevant problems and feasible. To verify the theory, we discuss some of the assumptions that were made with regard to the proposed theory. The assumptions can be checked on the basis of the following criteria:

- What is the motivation for these assumptions, i.e. why are they needed?
- Are the assumptions consistent? If not, under what conditions are they consistent or not?
- Do these assumptions limit the applicability of the proposed theory?

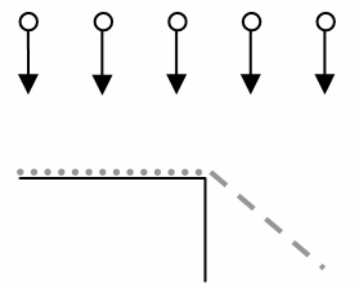
In section 3.2.1, the implications were discussed of the notion that features are described on the level of geometry and on the level of morphology. This distinction is relevant because although in the end mainly the geometry of a feature is used in an industrial design process, we presented a theory that considers features on the level of morphology and is independent of the shape representation type of the geometry of a feature. As a result, although we had to choose a specific shape representation type to be able to implement the proposed methods, the neutrality of the theory makes it possible to also implement the theory for other shape representation types.
This reasoning led to the assumption that the morphology of a feature is also independent of the geometry of the surface onto which a feature is embedded. This assumption allowed us to simulate the morphology of a feature in feature space. However, it must be noted that this assumption only holds if there is no other morphological structure defined on the embedding surface. In section 3.2.1 we assumed that no other morphological information is defined for an embedding surface, but the problem also occurs when features are nested, i.e. one feature is (partly) imposed on top of another feature. It can therefore not be proven that the proposed methods work for nested features; however this problem can be partly remedied by considering a nested feature to be noise when recognizing the nesting feature and vice versa.

In section 4.1.2, it was mentioned that there are limitation to the embedding surface onto which a feature can be instantiated using the given algorithms. Figure 5.8 shows a number of situations in which the proposed instantiation method leads to incorrect or at best unpredictable results. Figure 5.8a shows a situation in which the embedding surface is convex. If the embedding surface is convex, then there is a parametric configuration for which the shape of the feature self-intersects. The parameter values for which this is the case depend on the degree of convexity of the embedding surface. To prevent these parametric configurations from occurring, constraints must be used. However, these constraints must be directly coupled to the parameter mapping defined for the feature, and a discussion in such a constraint management system therefore falls outside the discussion that can be found in the existing literature. This topic must therefore be left for further research. Figure 5.8b shows a situation in which the embedding surface has a corner of equal to or more than 90 degrees. In this case the control points of a feature can not be projected onto the embedding surface and even if they can, then the definition of area and deformation parameters as it has been given in chapter 3 must be revised. This problem is also left for further research. Finally, Figure 5.8c shows a situation in which part of the embedding surface is obscured by another part of the embedding surface (note that it is also a problem if another shape obscures the embedding surface). This is not only the case for a fold such as is shown in Figure 5.8, but also if the maximum difference in surface normal vector of the embedding surface is larger than 180 degrees

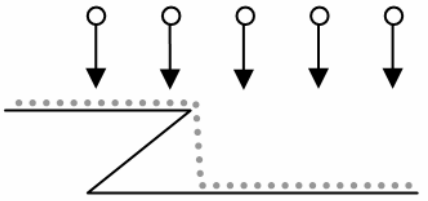
(e.g when the embedding surface is a sphere). In some cases the problem can be solved by adapting the direction of projection to the local condition of the embedding surface (instead of using the reverse surface normal vector at the point of origin of the feature). However, for the fold shown in Figure 5.8 this is not a solution.



(a)



(b)



(c)

Figure 5.8*: Three examples of embedding surfaces where the feature instantiation method fails (a) a convex embedding surface (b) sharp corners or highly curved surface (c) folded surface*

Finally, the assumption was made that a region of interest contains only one feature. There are two cases in which this assumption does not hold: if a target feature is nested and if a target feature intersects with another feature. If a target feature is nested, then the feature on which it is nested cannot be seen separately from the target feature, because it forms the embedding surface of the feature. In this case the region of interest by

definition contains two features. However, it can be assumed that when the region of interest is carefully selected, only part of the feature on which the target feature is nested is contained in the region of interest. In this case there is no problem.
Intersecting features cannot be recognized by the proposed feature recognition method, unless the interfering feature is considered to be noise. Feature interference is a well-known problem in feature recognition that has not even been fully solved in the domain of regular form features and that is therefore left as an open issue.

5.3 Evaluation of the research methods

In chapter 1, we stated that the type of research that is reported on in this thesis, namely foundational research with a strong practical component, consists of three main phases: exploration, creation and evaluation. In the exploration phase, first a literature study was done to investigate the state-of-the-art theory and methods in feature research. A difficulty that was encountered during this study was that the term 'feature' is overloaded. Even within the field of computer-aided design, the term feature is used for almost any anomaly that can be discerned during a design process. Few authors use the term 'form feature' and a selection therefore had to be made in what features were considered in the literature study. Only work on the feature concept was considered that:

- assumes that features incorporate a sort of parameterization. This excludes work that considers features purely on a geometric level.
- applies features in a feature-based operation. This excludes work in which features are a by-product or supporting concept of other operations.

The main result of the literature study was the identification of three different views on the freeform feature concept. Based on the results of this study, hypotheses were formulated that form the basis of the developed theory. The hypotheses and the theory are linked through a set of assumptions: these are based on the hypotheses and in turn support the development of the theory. The foundational aspects of the theory on freeform feature recognition were developed, and an implementation study was done, on the basis of which the foundational theory was implemented. To conceptualize a methodology on the basis of the implemented theory, a computational model was given. On the basis of this computational model, algorithms were developed for the recognition of freeform features.
In the implementation study we chose not to base our implementation on any existing software platform. Although this forced us to spend time on implementing some basic aspects in handling freeform features, it allowed us to gain insight in operational difficulties that occurred when implementing these basic aspects. The main disadvantage of this approach is that the pilot implementation that was created is not optimized.

However, the goal of the research was not to create a commercially viable software tool, but to demonstrate the developed methodology. The main advantage of not basing the implementation on an existing software platform is that we are better able to generalize the developed theory and methodology.
Finally, in an application study, the pilot implementation was tested. In this application study we make use of automatically generated test cases, because time-wise it is not feasible to evaluate the developed methods on a large number of test. Because the goal of the implementation was not to create a tool that will be actually used in the practice of industrial design (or any other domain of application), we did not test the usability of the pilot implementation. Instead, the application study focused on aspects of the user input on which assumptions were made for a feature recognition procedure.

5.4 Complexity and performance of the algorithms

The time complexity of an algorithm can be defined as the relation between the size of its input and the computation time that is needed by the algorithm. By determining the complexity of an algorithm, the performance of the algorithm can be generalized to larger inputs. The time complexity is the main indicator for the efficiency of an algorithm, rather than the computation time which, as is the case for our implementation, depends on the extent to which the implementation of the algorithm has been optimized. In this section, first the time complexity of the given algorithms will be discussed. Then, the performance of the implemented algorithms is tested by subjecting it to a large number of automatically generated test cases.

5.4.1 Analysis of the complexity of the algorithms

First we discuss the complexity of the template matching approach. The evolutionary feature recognition approach and the curve-based feature were presented as methods that overcome the main disadvantage of the template matching approach and it must be investigated if this goes at the cost of a higher complexity. The complexity mentioned here is that of the re-implemented algorithm given in this thesis.

Complexity of the template matching algorithm
The time complexity of the template matching algorithm depends on the number of parameters, m, the size of the region of interest $|ROI|$ and the size of the shape representation of the feature. Although we do not make any assumptions on the shape representation type of the region of interest or the feature shape, to be able to compute the distance between the two, both shapes are approximated by point samples. The resolution of these samples determines how well the point sample approximates the actual surface

and the size of the point samples is denoted $|ROI|$ and $|E|$, respectively. If a region of interest can be selected with sufficient accuracy (see section 5.6), then $|ROI|$ is of the order $O(|E|)$. The time complexity is further dependent on the number of iterations of the algorithm. The number of iterations needed for multi-dimensional function minimization to find a global minimum is exponential in the number of dimensions. The complexity can be expressed as $O(m^m |ROI||E|)$, which can be simplified to $O(m^m |E|^2)$ because $|ROI|$ is of the order $O(|E|)$. The complexity of template matching is very high and it is only feasible for very small numbers of parameters.

Complexity of the evolutionary freeform feature algorithm
The time complexity of the evolutionary freeform feature recognition algorithm can be given per step of the algorithm:

Step 1
The complexity of the first step of evolutionary feature recognition is in principle identical to that of template matching. However, the result of step 1 does not have to be an optimal result, but rather it has to be an acceptable approximation for step 2 of the procedure. Based on empirical results we conclude that in the average case the number of iterations needed to find a sub-optimal solution for the feature recognition problem is of the order $O(m^2)$. These results could not be supported by any theoretical argument, but were concluded after a large number of tests (more than 25000). The most feasible explanation for the quadratic complexity is that each combination of parameters in this case investigated. The time complexity can then be given as $O((m|E|)^2 \Pi)$. Recall that Π is the population size.

Step 2
In step 2, the number of genes is considered to be of the order $O(mn)$. Again, it can be assumed that the average case of iterations is needed, because step 2 starts with the approximation that resulted from step 1. Taking into account the independence of parameters and the local influence of control points, the complexity can be given as $O((n|E|)^2 \Pi)$.

Step 3

The complexity of this step depends on the size $|E|$ of the template feature and the size $|ROI|$ of the target feature. As we assumed that $|ROI|$ is of the order $O(|E|)$, and the complexity can therefore be given as $O(|E|)$.

Step 4

The complexity of a smoothing procedure depends on the procedure used. The complexity of Laplacian smoothing, which was used in our implementation, is of the order $O(|ROI|)$.

Step 5

In this step, the origin of the local coordinate system of the feature found in step 2 is projected onto the target surface. The complexity of this step is of the order $O(|ROI|)$.

Step 6

Because in this step, steps 1 and 2 are repeated, the complexity of step 6 is a combination of the previously given complexities and can be given as $O\left((m+n)|E|^2\,\Pi\right)$.

In the entire evolutionary feature recognition procedure, step 6 dominates. The total complexity of evolutionary feature recognition can be given as $O\left((m+n)|E|^2\,\Pi\right)$.

Complexity of curve-based feature recognition

The time complexity of the curve-based feature recognition algorithm depends on the variables λ and δ, as well as on the size $|ROI|$ of the region of interest, and can be given as $O(|ROI|\lambda\delta)$. As $\lambda\delta = O(n)$, this can also be written as $O(|ROI|n)$.

5.4.2 Evaluation of the performance of the algorithms

In chapter 4, several algorithms were presented, both for feature recognition and for the operations that support feature recognition. In this section we will evaluate the performance of the implementation of feature identification and that of evolutionary feature recognition. The evaluation was done using a computer with a 3 GHz processor and 1 GB of RAM memory.

Evaluation criteria

In a feature identification procedure, given a target shape S and a library of pre-defined feature types $L = \left\{F^1, \ldots, F^{|L|}\right\}$, it is determined which of the features in the library best corresponds to the target shape. The criteria on which feature identification is evaluated are *correctness, efficiency* and *sensitivity*. A feature identification that results in a feature type F^i is correct if, denoting a feature recognition procedure as $\wp\left(F, S\right)$, it holds that $d\left(\wp\left(F^i, S\right), S\right) < d\left(\wp\left(F^{j\neq i}, S\right), S\right)$. In words, a feature identification procedure is correct if the identified feature can be better recognized on the target shape than the other features in the library. The efficiency of a feature identification procedure can be measured as the amount of time that is needed to correctly identify a feature. Finally, the sensitivity of a feature identification procedure can be measure by how much a target shape can be distorted before a feature on the target shape can no longer be successfully identified

In an evolutionary feature recognition procedure, a part of the target shape is replaced by a feature shape. The criteria for the evaluation of feature recognition are *efficiency, sensitivity* and *accuracy*. Again, the efficiency is measured by the amount of time that is needed to recognize a feature, and the sensitivity is measured by comparing the computation time for different levels of distortion of the target shape. In addition, the accuracy of both methods is analyzed. The accuracy of a feature recognition procedure can be measured by the distance between the shape of the template feature after the recognition and the target shape before the recognition. The efficiency, the accuracy and the sensitivity of evolutionary feature recognition are compared to that of template matching.

Experiment setup

In order to evaluate the mentioned criteria, the implementation of the feature identification method and that of the evolutionary feature recognition method must be tested on a large number of target shapes, as an unlimited amount of different freeform shapes is possible. However, applying both procedures to a target shape is time-consuming: to be able to identify and recognize a feature on a target shape, possibly a new feature type must be defined and a region of interest must be selected on the target shape. We therefore make use of artificial target shapes, i.e. target shapes that are automatically generated. This way, we are able to simulate a large variety of target surfaces in multitudes that would otherwise not be possible.

As a basis for the generation of artificial target surfaces, a feature library was created that contained 10 different feature type definitions. We tried defining a set of feature types that we believed up front would pose a variety of challenges to the feature recognition algorithm. Some of the 10 pre-defined feature types are depicted in Figure 5.9.

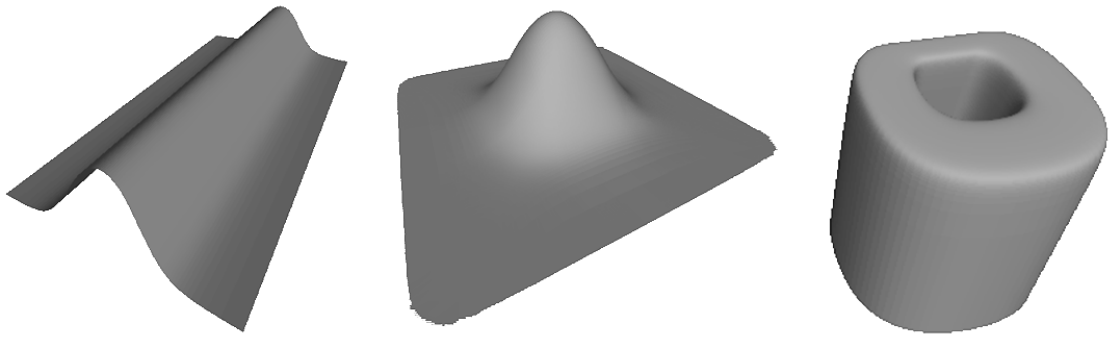


Figure 5.9*: Some of the feature types defined in the feature library*

Artificial target shapes were created by first generating an embedding surface as follows: A patch of 20x20 control points was generated; the control points were horizontally spaced in a bi-directional grid, each at a random height. This creates a terrain-like surface as the one depicted in Figure 5.10a. In a second step, a new feature type is created by starting from a feature type definition that is randomly chosen from the feature library. The chosen feature type was recorded, because it is later used to determine if the feature has been correctly recognized. This feature type is modified by multiplying its parameter mappings with a Gaussian value in the domain $\left|1,\Delta\right\rangle$, where Δ is the amount of distortion of the parameter mapping (see Figure 5.10b). The parameter values for the feature were chosen randomly. The resulting feature type is then instantiated on a random location on the generated base surface (see Figure 5.10c). Finally, a region of interest is generated by computing the area of the feature and multiplying its size by 1.25. As the target surface is a landscape, the height of the bounding box that determines the region of interest can simply be set to infinity.

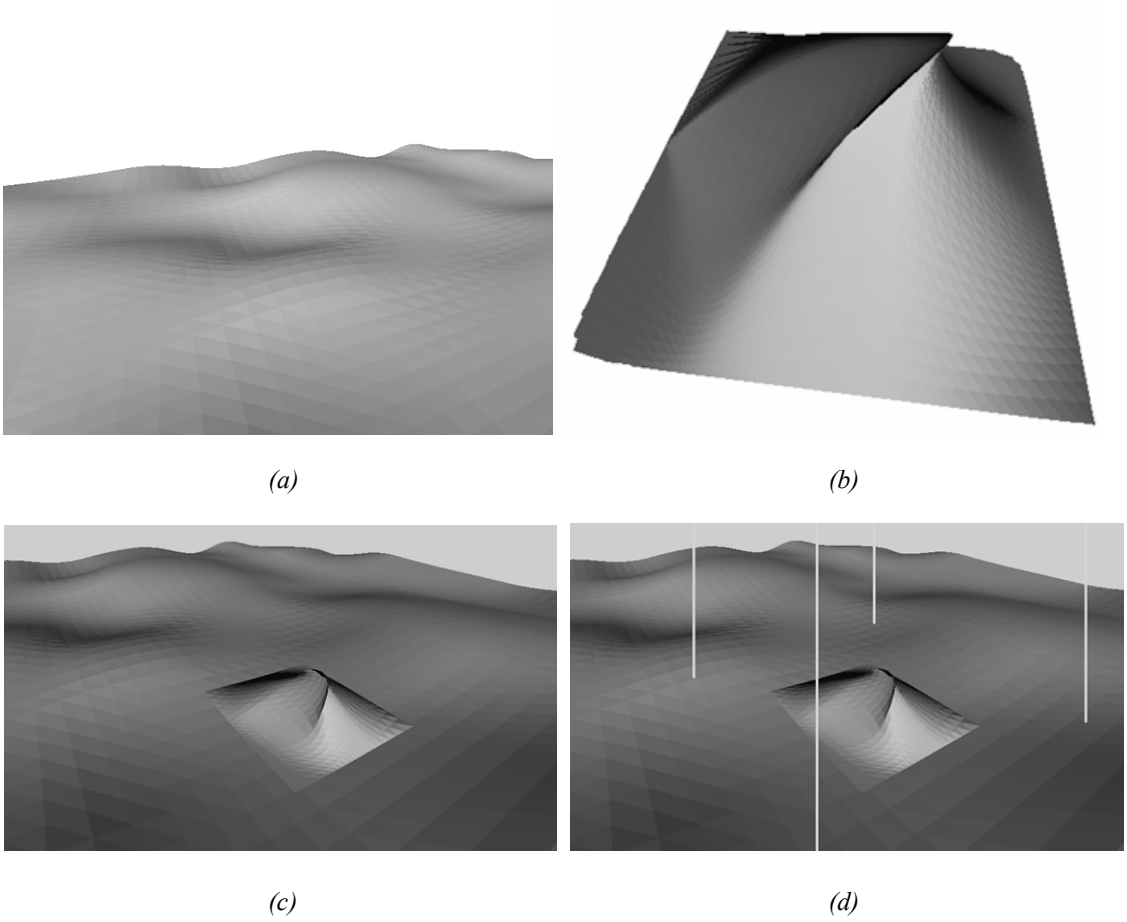


Figure 5.10: *Creation of an artificial target shape: (a) a generated target shape, (b) a generated feature type, (c) the feature instantiated on a random location on the base surface and (d) the automatically generated region of interest*

In a first test, the algorithm for feature identification was evaluated. A series of 2000 target surfaces was generated and to each of these target surfaces, the feature identification algorithm was applied. The cases were generated with different levels of distortion, gradually increasing from 1 to 2. That is, parameter mappings of a feature remain the same for a distortion of 1, and are at most twice as large as the original parameter mapping for a distortion of 2. The feature type that results from the feature identification procedure was compared to the recorded feature type.

In a second test, the implementation of the evolutionary feature recognition algorithm was tested. Again, 2000 target surfaces were generated. For each of these target surfaces, the feature type was given, i.e. it was assumed that the feature was correctly recognized prior to a procedure of feature recognition. To be able to compare evolutionary feature recognition to template matching, each target shape was subjected to both methods.

Results

Table 5.1 shows the results of the tests of the implementation of the feature identification method.

Amount of distortion	*Nr. of correctly identified features*	*Nr. of incorrectly identified features*	*Average computation time*
1.0-1.1	200	0	109 sec.
1.1-1.2	200	0	108 sec.
1.2-1.3	200	0	109 sec.
1.3-1.4	199	1	109 sec.
1.4-1.5	196	4	111 sec.
1.5-1.6	191	9	108 sec.
1.6-1.7	184	16	109 sec.
1.7-1.8	165	35	110 sec.
1.8-1.9	138	62	109 sec.
1.9-2.0	119	81	109 sec.

Table 5.1*: Comparison of the results for different amounts of distortion*

Table 5.1 shows that on average, the feature identification algorithm terminates in approximately 109 seconds, or a little less than two minutes. If the distortion that is applied when generating the test cases is maximally 30%, then all features can be correctly recognized. For a distortion above 60%, the number of features that can be correctly recognized rapidly decreases.

Table 5.2 shows the results of the experiments on the implementation of the freeform feature recognition algorithm. The table shows the computation time (T) and the distance between the result of the recognition procedure and the target shape (δ). It is shown that for all levels of distortion δ is much smaller for evolutionary feature recognition than for template matching. The computation time for evolutionary feature recognition is slightly higher than that for template matching. For higher levels of distortion, δ and T increase relatively faster in the case of evolutionary feature recognition than for template matching.

Amount of distortion	*Average T template matching*	*Average T evolutionary feature recognition*	*Average δ template matching*	*Average δ evolutionary feature recognition*
1.0-1.1	1571 sec.	1635 sec.	3.082 mm.	0.027 mm.
1.1-1.2	1598 sec.	1661 sec.	3.167 mm.	0.033 mm.
1.2-1.3	1631 sec.	1711 sec.	3.241 mm.	0.047 mm.
1.3-1.4	1647 sec.	1760 sec.	3.286 mm.	0.051 mm.
1.4-1.5	1656 sec.	1843 sec.	3.413 mm.	0.058 mm.
1.5-1.6	1662 sec.	1887 sec.	3.877 mm.	0.072 mm.
1.6-1.7	1681 sec.	1958 sec.	4.142 mm.	0.084 mm.
1.7-1.8	1690 sec.	2019 sec.	4.780 mm.	0.090 mm.
1.8-1.9	1705 sec.	2073 sec.	4.989 mm.	0.091 mm.
1.9-2.0	1718 sec.	2149 sec.	5.484 mm.	0.102 mm.

Table 5.2*: Comparison between evolutionary feature recognition and template matching*

Conclusions

From Table 5.2, it can be concluded that although the implementation of the feature identification method is sensitive to the input data, this holds mainly for high levels of distortion. For low and moderate levels of distortion, the correctness of feature identification is large.

Our implementations of both the feature identification algorithm and the feature recognition are slow. The efficiency of both implementations is not sufficiently high to be able to use them in practice in a design process. However, as was discussed before, this can partly be attributed to the fact that both implementations were not optimized and the fact that the tests were done on relatively slow hardware. Therefore, the performance of the implementation of evolutionary feature recognition was compared to the re-implementation of the template matching method. Likewise, this implementation was not optimized and the results are therefore comparable. As can be seen in Table 5.2, the computation time of evolutionary feature recognition is slightly higher than that of template matching. However, as is also shown in Table 5.2, this is coupled to a large increase in the accuracy of feature recognition. If the automatically generated test cases are not distorted, then the result of evolutionary feature recognition is more than 100 times more accurate than a template matching procedure. The evolutionary feature recognition procedure is accurate enough to be used in the design process.

5.5 Analysis of accuracy and correctness of the user input

In the freeform feature recognition methods that were proposed in the previous chapter, input from the user was assumed to be available for:

- Location of the target feature on the target shape by selecting a region of interest.
- Definition of new feature types .

The accuracy and correctness of these two types of input are an important factor in the performance of the developed feature recognition method. In this chapter we report on experiments that were conducted to evaluate the accuracy and correctness of these two types of input. In section 5.6.1, we will give criteria for the evaluation of the selection of a region of interest and report on the experiments in which these criteria were measured. Likewise, in section 5.6.2, an evaluation of feature type definition is discussed.

5.5.1 Evaluation of the selection of a region of interest

When the task is to locate a feature on a target surface in order to reduce the amount of data that needs to be processed by the feature recognition procedure, users should select a region of interest. A region of interest can be selected by positioning a bounding box around a region of the target surface. All the shape data that is contained in the bounding box is taken to be the region of interest. If part of the feature falls outside the region of interest, then it is less likely that the feature can be correctly recognized. On the other hand, if the region of interest contains, apart from the feature, a large amount of additional shape data, then the feature recognition procedure is less efficient. The selection of a region of interest can be evaluated on the basis of two criteria: a region of interest is *correct* if it contains the entire shape of the target feature. The selection of a region of interest should in all cases provide a correct result to be able to support a feature recognition procedure. The region of interest is *accurate* if, apart from the shape of the feature, the amount of shape data that is contained in the region of interest is minimal. A region of interest is sufficiently accurate when it contains no more than 156 % of the shape of the feature. This percentage is chosen because it corresponds to the percentage of excess data that was used in the automatically generated test cases in section 5.5 (note that $1.25^2 \approx 1.56$).

Experiment setup
In order to evaluate the correctness and accuracy of a bounding box selection tool, an experiment was set up in which 25 test subjects were asked to select a region of interest

on a series of 4 polygon meshes. Of the test subjects, 10 were novices without any experience with 3D modeling environments, 10 were design students, and 5 were experienced designers.

Prior to showing the shape models to the user, in each model a form feature was identified. Of the many features that were available on each model, a single feature was selected that is related to the normal use of the product, in order to increase the probability that test subjects recognized it as being a feature. For each feature, it was determined what part of the shape can be considered to belong to the shape of the feature. Figure 5.11 shows the shape models for which the user had to perform the selection task; in the figure, the target form feature is indicated by a box. Each of the shape models were shown to the test subjects, and on each model, test subjects were asked to select the pre-determined feature by indicating a region of interest that contained the complete shape of the feature. Once they were satisfied with their selection, the number of polygons in the selected region of interest was recorded.

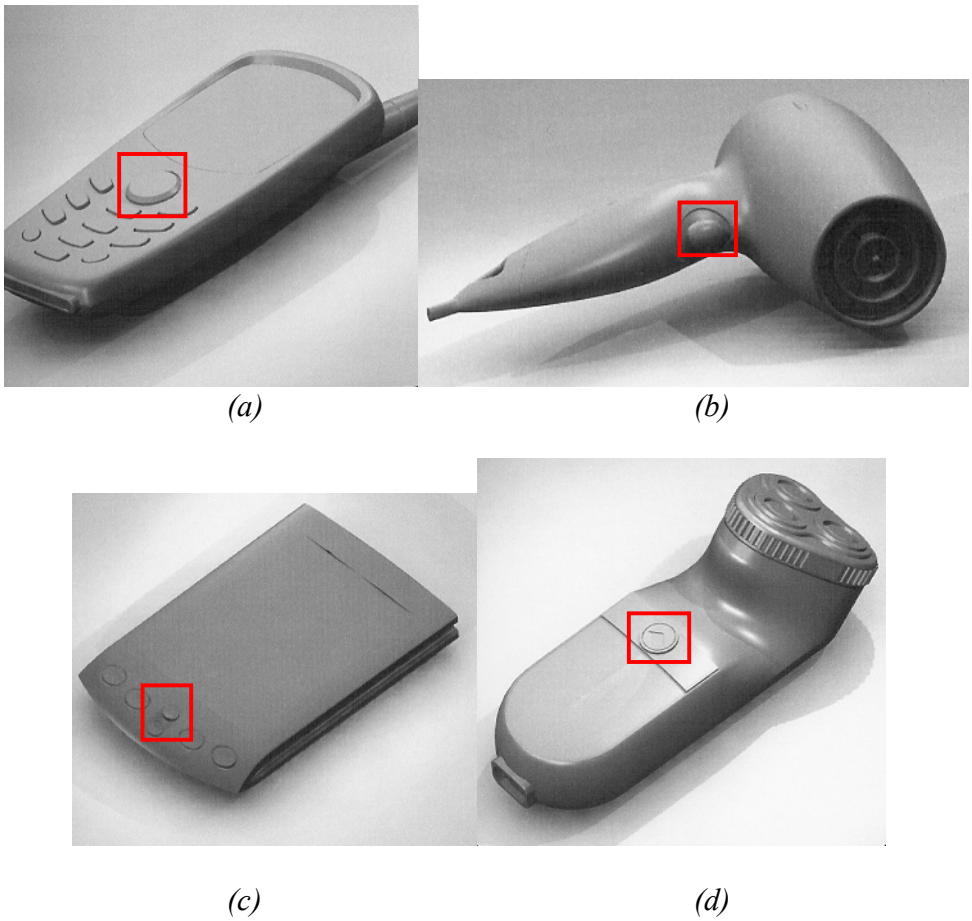


Figure 5.11: *The cases that were presented to the user: (a) a cell phone (b) a hairdryer, (c) a palmtop and (d) an electric razor.*

Results

In all cases, the region of interest was correctly selected, i.e. in each test case the entire feature shape was contained in the region of interest. Figure 5.12 shows the results in terms of the accuracy of the selection. The figure shows the percentage of the amount of shape data that was selected, where an accuracy of 100% corresponds to a perfect selection, i.e. a region of interest in which only the shape of the feature is contained.

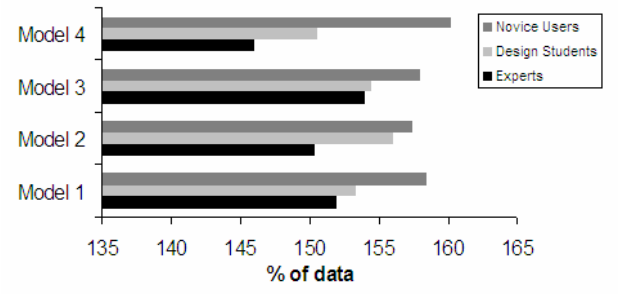


Figure 5.12*: Accuracy of the selection of a region of interest*

Conclusion

In all cases, the selection of a region of interest was correct. As is shown in Figure 5.12, the accuracy of the region of interest selection lies below the given threshold of 156% for the expert group and the group of design students, but not for novice users. However, as the target application of the feature recognition method lies in computer-aided design, we conclude that the selection of a region of interest is sufficiently accurate.

5.5.2 Accuracy of the feature type definition

An essential part of a feature recognition procedure that is controlled by the user is the definition of a new feature type. Although the presented feature recognition method is flexible and able to adapt feature definitions to the target shape, the resemblance of a feature shape to the target surface must be large enough to facilitate an efficient feature recognition process. In addition, the resemblance must be large enough to guarantee that the correct feature is recognized.

The accuracy of a feature type definition can be measured as the distance of the result of a feature recognition method that is applied on the shape of the target feature that the user intends to define.

Experiment setup

To evaluate the accuracy of the feature type definition process, an experiment was conducted that consisted of the following steps:

- Test subjects were given a lump of clay and were asked to create one or more physical examples of a feature shape. The users were allowed to create any feature type they wanted to create. The physical examples were scanned by a 3D laser range scanner to obtain a digital model.
- The test subjects were asked to define a new feature type with the goal of using this feature type definition in a feature recognition process that targets the digitized model.
- The defined feature type was applied in a feature recognition procedure.

Photos of a selection of the created clay models are shown in Figure 5.13.

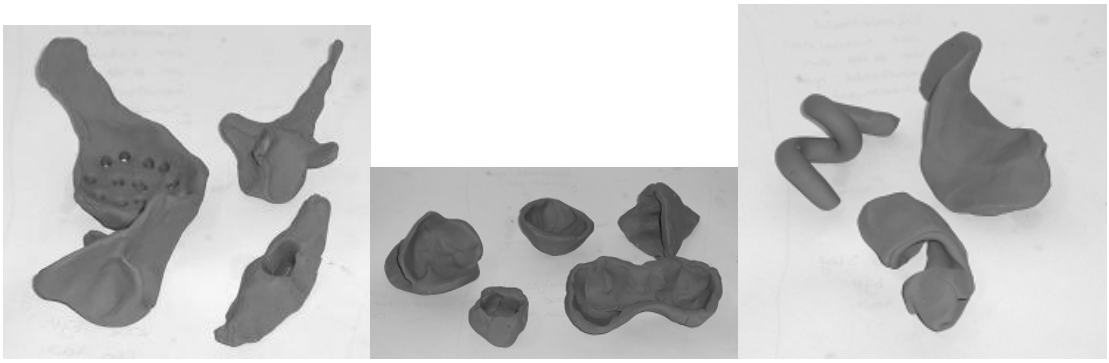


Figure 5.13*: A selection of the clay models created by the users*

Results

In total, 10 test subjects were involved in the experiment, who created 23 clay models. For 19 of these clay models, feature type definitions could be created (as one test subject had to leave before the experiment was finished). In Figure 5.14, examples are shown of the digitized clay models and the corresponding feature type definitions. The feature type definitions were used in the application of our implementation of the feature recognition method to the digitized models. In Table 5.3, the results (measured in mm.) of these feature recognition procedures are given.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
δ	0.046	0.072	0.083	0.169	0.060	0.058	0.049	0.065	0.087

	10	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*
δ	0.052	0.028	0.074	0.040	0.066	0.039	0.073	0.122	0.059	0.047

Table 5.3*: Results of the feature recognition procedure applied on the digitized clay models*

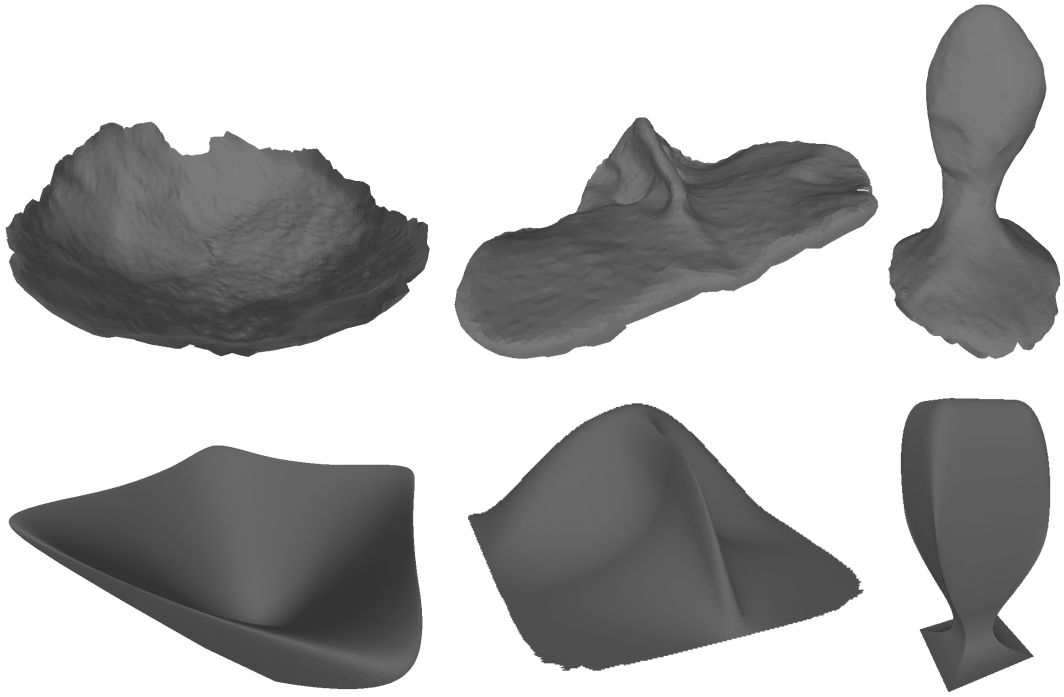


Figure 5.14*: Examples of scanned features and the corresponding feature type definitions*

Conclusion

The average distance of the result of the application of a feature recognition procedure to the digitized models was 0.068, which corresponds to the result of the recognition of automatically created test cases that was found in section 5.5 for a level of distortion of 60%. This can be explained by the fact that the digitized models contained a certain level of noise due to the limited resolution of the digitization process. The amount of noise that is introduced in the digitization process cannot be generally determined, but we conclude that new feature types can be defined with enough accuracy to serve as an input to a feature recognition procedure.

6 Conclusions and future research

In this chapter, we revisit the findings of the promotion research. This thesis was built up out of several components, of which the implications were discussed in the corresponding chapters, but until now no conclusions were drawn on the research in its entirety. In section 6.1, we discuss the results and the implications of the research. In section 6.2, we discuss the utilization of the research in the practice of design and the contribution of the research to existing knowledge. In section 6.3, we review the research hypotheses as they were posed in the introductory chapter of this thesis. Finally, in section 6.4 we conclude this thesis by looking ahead at the directions for future research.

6.1 Results and implications of the research

Throughout this thesis, the findings of the different chapters accumulated to the implementation of a new freeform feature recognition method. However, the center of gravity of the research does not lie with this pilot implementation. The first achievement of the research is the development of a foundational theory on freeform features that extends beyond an application to feature recognition. The development of such a theory was motivated by the following issues:

- There is a lack of comprehensive theory on freeform features in the existing literature. Although there is some pioneering work on the topic of freeform features, this work is fragmented and based on assumptions that are in some cases conflicting. It can therefore not be extended to a general theory for the support of freeform features.
- Existing work on freeform features leans on the extensive theory that is available on the topic of regular form features. However, one of the conclusions of the literature study that was done was that the assumptions on which the theory on regular form features is based do not hold in the freeform domain. For this reason, the applicability of much of the existing theory on freeform features is limited.
- Existing theory on freeform features is based on a geometry-centered view on the freeform feature concept. Theory that is based on a geometry-centered view does not fully utilize the potential of the freeform feature concept. Freeform features are in fact morphological structures and by regarding features on the level of morphology, one can not only use the feature as a parametric modeling entity, but also reason about features on a higher theoretical level.

The foundational theory that was proposed in this thesis addresses these three shortcomings of the existing research. A comprehensive theory was presented that specifically supports the development of methodology for the problem of feature recognition, but that can be extended to the freeform concept in general. The theory specifically deals with features on the level of morphology and is not an extension of the regular form feature, i.e. the theory was implemented for freeform shapes.

The theory was not directly implemented in the form of methods that could be applied directly to the feature recognition problem. Instead, a methodology was developed based on computational models that were proposed and discussed on the basis of the theory. To support the development of methodology, we introduced the concept of feature space. The concept of feature space is based on the assumption that the morphological aspects of a feature are independent of its geometry. This assumption was realized by defining the freeform feature to be composed out of components that relate to the geometry and that are connected through the parameter mappings. The alternative environment of feature space does not only allow complicated computations to be done more efficiently, but it also enables us to formulate computational processes more concisely. The concept of feature space has the potential to be applied in a much broader context, for example to the problem of feature interference.
Computational models were given that indicate the computational steps and information flow for feature recognition as well as for several methods that support feature recognition. On the basis of these models, methods were developed and algorithms were given. The problem of feature recognition was addressed by first re-implementing and analyzing an existing freeform feature recognition method, namely the template matching method. To our knowledge this is the only freeform feature recognition method available in existing literature. By re-implementing it and evaluating its performance and shortcomings, we were able to formulate requirements for a new feature recognition method. We then proposed an evolutionary feature recognition method that is also based on the principle of template features, but improves on the template matching method in the following aspects:

- The morphological structure of the feature is included in the recognition process, meaning that the morphology of the feature can also be adapted in a feature recognition procedure. As a result, the user is no longer responsible for the definition of a feature type specifically for a certain target shape.
- The evolutionary feature recognition method does not only recognize a feature, but also retrieves its embedding surface. By doing so, advanced operations can be applied to the feature, such as deletion or copying of the feature. In

addition, if the base surface of a feature is known, operations on the feature can be applied with more accuracy and efficiency.

- The result of an evolutionary feature recognition method is an embedded feature, meaning that the geometry of the feature is a subset of that of the target shape. As a result, the accuracy with which features can be recognized is considerably better than that of the template matching approach.

Although the evolutionary feature recognition method can be applied to a broad range of features, there are categories of features to which it cannot be applied. Among others, it cannot be applied to some features that change the topology of their embedding surface, such as holes, and to features with multi-dimensional (e.g. curve-based) parameters. The development of a feature recognition method that can be applied to any conceivable feature type cannot be contained in a four year promotion research, but to demonstrate that the principle of the developed theory can be extended to other feature types, a curve-based feature recognition method was developed, which can be applied to a features with one curve-based parameter. Likewise, the proposed theory can be extended to support the recognition of other feature types.

6.2 Utilization and added value of the research

To demonstrate how the developed theory and methodology can be used in the practice of design, we demonstrated the use of the proposed method on some example from the practice of industrial design in section 5.1. The time needed to recognize the features in the example of the soap dispenser bottle was reported to be around 10 minutes. Another 15-30 minutes must be assumed to be needed for the user to give accurate input (i.e. give a region of interest and define a feature type that corresponds to the design intent of the designer). Adding up these times, we can conclude that a parameterized interpretation of the shape data that was obtained through scanning could be constructed in roughly 2 hours.

The contribution of the promotion research to existing knowledge is twofold. First, a comprehensive theory was developed that supports methodology for freeform feature recognition but can be extended to support other freeform feature-based operations:

- The result of a freeform feature recognition procedure can be successively used in a feature-based design process.
- It has been discussed that the open issues such as feature interference can be addressed within the proposed theory.

Second, a method was developed that improves on existing methods for freeform feature recognition.

- The method recognizes a feature while taking into account that the embedding surface of a feature is an inherent part of the feature concept.
- The method allows the morphological aspects of a feature to change, while maintaining the same parameters for a feature, thus leading to a more accurate recognition.

6.3 Review of the research hypothesis

In the early phases of the research, hypotheses were formulated which embodied the expectations regarding some of the aspects of the freeform feature concept and its application to freeform feature recognition. In this section, we review these hypotheses and discuss if they hold for the work that was done.

Hypothesis 1 stated that a morphology-centered view on the freeform feature concept is better able to support freeform feature-based methodology than the geometry-centered view of existing literature on freeform features. In chapter 2, a historical overview of feature research was given and it was found that the theory on regular form features gradually developed from a geometry-centered view to a morphology-centered view. When the feature concept made the shift to the freeform domain, the theory fell back to a geometry-centered view. In the definition of the freeform feature that was given in chapter 3, a morphology-centered view was employed. In the development of a new method for freeform feature recognition, it was shown how this morphology-centered definition of a feature is instrumental in improving on the shortcomings of existing freeform feature recognition methodology.

Hypothesis 2 stated that the definition of the regular form feature concept cannot be extended to the freeform feature concept. In chapter 2, three different approaches to the freeform feature concept were identified. Two of these approaches were an extension of approaches to the freeform feature concept. They were found not to be applicable to the freeform feature recognition problem. The third approach was taken as a starting point for the development of new theory.

In hypothesis 3, it was argued that dedicated theories for specific applications of the freeform feature concept can be combined into a single comprehensive theory. In chapter 3 we showed that the proposed theory on the freeform feature does not only support the development of a methodology for freeform feature recognition, but can also be the basis of methodology for related operations, such as the instantiation of features on a target surface. In the related hypothesis 4, it was stated that a comprehensive theory is not only possible, but is essential in validating and generalizing freeform feature-based methods. In chapter 4, a method was developed for the recognition of freeform features, which was

supported by methods for related operations. To be able to validate and generalize the developed method for freeform feature recognition, it is therefore essential to be able to relate the method to the supporting methods through the supporting comprehensive theory.

Hypothesis 5 emphasized the need for freeform feature-based operations to be able to deal with user-defined features. One of the main advantages of the developed freeform feature recognition method is that it no longer requires users to define a new feature type for each target shape. This means that users can freely define new feature types without this having a negative effect on the accuracy and the efficiency of the freeform feature recognition method. Instead of being able to handle user-defined features, it can therefore be argued that the developed method takes away the need for user-defined features.

Hypothesis 6 stated that operations on a feature should be regarded separately from the effect that these operations have on the embedding surface of the feature and on the relation between feature and embedding surface. In chapters 3 and 4 we proposed and implemented the concept of feature space, which supports this hypothesis. By providing a method to transpose and instantiate a features from feature space into modeling space, we separated the effect of a feature-based operation on the feature (which can be computed in feature space) and the effect on the embedding surface (which can be determined through algorithms 4.2, 4.4 and 4.5).

Finally, hypothesis 7 stated that a brute force or probabilistic method is needed for the development of a new freeform feature recognition method. The method we developed uses an evolutionary approach. In chapter 4, it was argued why such an approach is a logical way to target the freeform feature recognition problem.

6.4 Directions for future research

In this section we look into the future research on freeform feature recognition and freeform features in general. Partly, the directions for future research are based on problems that we could not or chose not to address in the promotion research. Also, the assumptions on which the research was based may start discussions which in turn may lead to new directions of research. In this case, the challenge is to define how these topics relate to the research that was addressed in the promotion research. Another part of the topics for future research is directly implied by the research that was done. The research brought up new questions and clarified or completed existing theory. Finally, there may be alternatives to the proposed research, i.e. other solutions to the same research problem. Not only are these alternatives possible improvements of the proposed research, they also provide a frame of reference that enables us to better judge the research presented in this thesis.

Extension of the theory and methodology

During the development of the theory and methodology that are presented in this thesis, we were limited by the problem description, which steered the research towards a feature recognition method. However, as we argued earlier, the developed theory can be used to support other feature-based operations, the most important of which is feature-based design. Future research can target the extension of the theory and the development of new methods that are dedicated to these operations.

Application to other domains

The theory and methodology for the support of freeform features were developed with the intention of applying them in the context of a computer-aided design process. However, the use of the feature concept in other domains of applications can be considered, such as for example face recognition or robotics.

A theory for hole features

As was mentioned earlier, the proposed theory does not support some features that change the topology of their embedding surface, e.g. hole features. In existing literature on both regular form features and freeform features, such features are categorized as eliminative features, which differ only from other feature types in that they remove geometry instead of adding or deforming it. However, in our opinion, the hole feature is a fundamentally different type of feature, because the morphological structure of the feature relates only to its boundary (as the inner portion of the feature is 'empty'). To support the hole feature, new theory must be developed that we believe can be an alternative implementation of the definition of the freeform feature concept given in section 3.3. Note that there are other feature types that change the morphology, e.g. features with multiple attachment points, such as for example a handle that is attached to its embedding surface at two locations, but these can be addressed by the proposed methods or a slight modification thereof.

Multi-dimensional parameters

Multi-dimensional parameters were discussed throughout the thesis, and although a feature recognition method was developed that targets a specific feature type with a two-dimensional parameter, more extensive research is needed. The relevance in multi-dimensional parameters is that they can be used to facilitate a more intuitive and efficient definition of feature types. For example, a two-dimensional parameter can be defined with a single sketch rather than with a number of variable assignments. Especially when applied to the domain of industrial design this is a significant improvement on the method of feature definition that was presented in this thesis, but also on the methods that are used in current commercial CAD software.

4-dimensional feature recognition

In the proposed methodology, the morphological structure of a feature on a target surface is reconstructed with reference to pre-defined feature types in the feature library. An alternative approach to feature recognition can be taken when it can be assumed that there are two or more target shapes, which represent different states of the same feature. In this case, the morphological aspects of a feature can be derived from the difference between the two or more states of the feature shape. This concept is not applicable to the reuse of existing shapes, because in general no two states of an existing shape are available, but it can be used for example during a process of hand-based clay modeling.

Own papers

Langerak, T.R., Vergeest, J.S.M, Song, Y., *Recognising and editing styling lines in free form shapes*. In Pan, Y., Vergeest, J.S.M., Lin, Z., Wang, C., Sun, S., Hu, Z., Tang, Y., Zhou, L., Editors, Applications of digital techniques in industrial design engineering - CAID&CD'2005, Beijing: IAP – WPC, 2005.

Langerak, T.R., Vergeest, J.S.M., Song, Y, *Parameterising styling lines for reverse design using free form shape analysis*, Proceedings of IDECT/CIE 2005, New York: ASME, 2005.

Langerak, T.R., Vergeest, J.S.M., *A new framework for the definition and recognition of free form features*, Horvath, I.. Duhovnik, J., Editors, Proceedings of the sixth International Symposium on Tools and Methods of Competitive Engineering, Delft - Ljubljana: Delft University of Technology - University of Ljubljana Slovenia, 2006.

Langerak, T.R., Vergeest, J.S.M., *A new framework for the definition and recognition of free form features*, Journal of Engineering Design, Vol. 18, No. 5, pp. 489-504, 2007.

Langerak, T.R., Vergeest, J.S.M., *A New Method for Defining and Composing Free Form Features*, Journal of computer-aided design and applications, Vol. 4, No. 1-4, 2007

Langerak, T.R., Vergeest, J.S.M, *A dual environment for 3D modeling with user-defined free form features*, ASME/DETC conference, Las Vegas, 3-7 September 2007

Langerak, T.R., Vergeest, J.S.M, *An Evolutionary Strategy for Free Form Feature Identification in 3D CAD Models*, Proceedings of the WSCG conference, January 29 - February 2 2007, Plzen

Langerak, T.R, *Parameter reconstruction of freeform shapes for improved product modeling*, I. Horváth and Z. Rusák, Editors, Proceedings of the seventh International Symposium on Tools and Methods for Competitive Engineering, April 21–25, Izmir, Turkey, 2008.

Langerak, T.R., *Geometric Feature Deletion Through Freeform Feature Recognition*, Full paper proceedings of the WSCG conference, February 4-7, 2008.

Langerak, T.R., *Instantiation and manipulation of user-defined freeform features*, accepted for the Proceedings of the ASME/DETC conference, New York, August 3-6, 2008.

Other references

Au, C., Yuen, M., *A semantic feature language for sculpted object modeling*, Computer-Aided Design, Vol. 22, No. 1, pp. 63-74, 2000.

van den Berg, E., van der Meiden, H.A., Bronsvoort, W.F., *Specification of freeform features*, Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications, pp. 56–64, 2003.

Van den Berg, E., *Freeform feature modeling and validity maintenance*, PhD thesis, Delft University of Technology, 2007.

Besl, P., McKay, N.D., *A method for registration of 3-D shapes*, IEEE Transactions on pattern analysis and machine intelligence, Vol. 14, No. 2, pp. 239-256, 1992.

Bidarra, R., Bronsvoort, W.F., *Semantic feature modeling*, Computer-Aided Design, Vol. 32, No. 3, pp. 201-225, 2000.

Biederman, I., *Recognition-by-components: A theory of human image understanding*, Psychological Review, Vol. 94, No. 2, pp. 115-147, 1987.

Bronsvoort, W.F., Jansen, F.W., *Feature modelling and conversion - Key concepts to concurrent engineering*, Computers in Industry, Vol. 21, No. 1, 1993.

Cardone, A., *A feature-based shape similarity assessment framework*, Ph.D. Thesis, University of Maryland, 2005.

Cavendish, J.C., Marin, S.P., *Feature-based surface design and manufacturing*, IEEE computer Graphics and Applications, Vol. 12, No. 5, pp. 61-68, 1992.

Cavendish, J.C., *Integrating feature-based surface design with freeform deformation*, Computer-Aided Design, Vol. 27, No. 9, pp. 703-711, 1995.

Chuang, S. H., Henderson, M.R., *Three-dimensional shape pattern recognition using vertex classification and vertex-edge graph*, Computer-Aided Design, Vol. 22, No. 6, pp. 377-387, 1990.

Cunningham, J. J., Dixon, J. R., *Designing with Features: The Origin of Features*, Proceedings of the ASME International Computers in Engineering Conference, Vol. 1, San Diego, pp. 237-243, 1988.

Davis, L, editor, *Handbook of genetic algorithms*, New York, Van Nostrand Reuinhold, 1991

De Floriani, L., *A graph based approach to object feature recognition*, Proc. 3rd Annual ACM Symposium on Computational Geometry, pp. 100-109, 1987.

De Floriani, L., *Feature Extraction from Boundary Models of Three-Dimensional Objects*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 8, pp.785-798, 1989.

De Martino, T., Falcidieno, B., Giannini, F., Hassinger, S., Ovtcharova, J., *Feature-based modelling by integrating design and recognition approaches*, Computer-Aided Design, Vol. 26, pp. 646-653, 1994.

Desouza, G.N., Kak, A.C., *Vision for mobile robot navigation: a survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 2, pp. 237 – 267, 2002.

Dong, X., Wozny, M.J., *Instantiation of user defined features on a geometric model*, Product modeling for computer-aided design and manufacturing, Turner, J., Pegna, J., Wozny, M.J., Editors, 1991.

Duffy, A.H.B., Ferns, A.F., *An analysis of design reuse benefits*, Proceedings of the ICED99 conference, pp. 799-804, 1999.

van Elsas, P.V., Vergeest, J.S.M., *Displacement feature modelling for conceptual design*, Computer-Aided Design, Vol. 30, No. 1, pp. 19-27, 1998.

Falcidieno, B., Giannini, F., *Neutral format representation of feature-based models in multiple viewpoints context*, Product modeling for computer-Aided Design and manufacturing, Turner, J., Pegna, J., Wozny, M., Editors, 1991.

Fisher, R.B., *Applying Knowledge to Reverse Engineering Problems*, Computer-Aided Design, Vol. 36, No. 6, pp. 501-510, 2004.

Fontana, M, Giannini, F., Meirana, M., *A free form feature taxonomy*, Eurographics 1999, Vol. 18, No. 3, 1999.

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D., *Modeling by Example*, ACM Transactions on Graphics (SIGGRAPH 2004), Los Angeles, CA, August 2004

Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., Jacobs, D., *A Search Engine for 3D Models*, ACM Transactions on Graphics, Vol. 22, No. 1, pp. 83-105, 2003.

Guillet, S. and Léon, J.C., *Parametrically deformed freeform surface as part of a variational model*, Computer-Aided Design, Vol. 30, No. 8, pp. 621–630, 1998.

Gao S., Shah J. J., *Automatic recognition of interacting machining features based on minimal condition subgraph*, Computer-Aided Design, Vol. 30, No. 9, pp. 727-739, 1998.

Ghosh, A. and Tsutsui, S., Editors, *Advances in Evolutionary Computing*, Natural Computing Series, Springer, 2002.

Grabowksi, H., Braun, S., Suhm, A., *User-defined feature libraries for maintenance of recurrent solutions*, Proceedings of the 6th international conference on CAD/CAM, Robotics and factories of the future, Springer-Verlag, Berlin, 1991.

Grayer, A., *Recognition of machinable volumes*, Ph.D. dissertation, Cambridge University, 1975.

Gindy, N.N.Z., *A hierarchical structure for form features*, International Journal of Production Research, Vol. 27, No. 12, pp. 2089-2103, 1989.

Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, Massachusetts, 1989.

Han, J, Requicha, A.A.G., *Geometric reasoning for feature recognition*, Technical report IRIS-95-343, Insititute for Robotics and Intelligent Systems, USC, USA, 1995

Han, J., Requicha, A.A.G., *Integration of feature-based design and feature recognition*, Computer-Aided Design, Vol. 29, No. 5, pp. 393-403, 1997.

Han, J., Pratt, M., Regli, W.C., *Manufacturing feature recognition from solid models: a status report*, IEEE Transactions on robotics and automation, Vol. 16, No. 6, 2000.

Higuchi, K., Hebert, M., Ikeuchi, K., *Building 3-d models from unregistered range images*, Graphical models and image processing, Vol. 57, No. 4, pp. 315-333, 1995.

Hoffmann, C.M., Joan-Arinyo, R., *On user-defined features*, Computer-aided Design, Vol. 30, No. 5, pp. 321-332, 1998.

Horváth, I., *A workbench architecture for object oriented handling of features*, Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, August 18-22, 1996, Irvine, California.

Iyer, N., Jayanti, S., Lou, K., Kalyanaraman, Y., Ramani, K., *Three-dimensional shape searching: state-of-the-art review and future trends*, Computer-Aided Design, Vol. 37, No. 5, pp. 509-530, 2004.

Jiantao P., Ramani K., *A 2D Sketch Based User Interface for 3D CAD Model Retrieval,* Journal of Computer Aided Design and Application, Vol. 2, No. 6, pp.717-727, 2005.

Joshi, S., Chang, T.C., *Graph-based heuristics for recognition of machined features*, Computer-Aided Design, Vol. 20, No.2, pp.58-66, 1988.

Kazhdan, M., *Shape Representations and Algorithms for 3D Model Retrieval*, PhD Thesis, Princeton University, 2004.

Kim, Y. S., *Convex decomposition and solid geometric modelling*, PhD Thesis, Department of Mechanical Engineering, Stanford University, USA, 1990.

Kim, Y. S., *Recognition of form features using convex decomposition*, Computer-Aided Design, Vol. 24, No. 9, pp. 461-476, 1992.

Ko, H., Park, M., *Integration methodology for feature-based modeling and recognition*, Advances in Engineering Software, Vol. 20, No. 2-3, pp. 75-89, 1994.

Kyprianou, L., *Shape classification in Computer-Aided Design*, Ph.D. Thesis, Cambridge university, 1980.

Laakko, T., Mäntylä, M., *Feature modeling by incremental feature recognition*, Computer-Aided Design, Vol. 25, No.8, pp. 393-403, 1993.

Luby, S. C., Dixon, J. R., and Simmons, M. K., *Creating and Using a Features Database*, Comput. Mech. Eng., Vol. 5, No. 3, pp. 285-292, 1986.

Maintz, J.B.A., Viergever, M.A., *A survey of medical image registration*, Medical Image Analysis, Vol. 2, No. 1, pp. 1-36, 1998.

Mandorli, F., Cugini, U., Otto, H.E., Kimura, F., *Modeling with self-validation features*, Proceedings of the Fourth Symposium on Solid Modeling and Applications, 1997.

Marefat, M., and Kashyap, R.L., *Geometric reasoning for recognition of three dimensional object features*, IEEE Transactions on Pattern Analysis and Machine lntelligence, Vol. 12, No. 10, 949-965, 1990.

Masuda, T., *Registration and integration of multiple range images by matching signed distance fields for object shape modelling*, Computer Vision and Image Understanding, Vol. 87, No. 1-3, pp. 51-65, 2002.

Menon, S., Kim, Y. S., *Handling Blending Features in Form Feature Recognition Using Convex Decomposition*, Proceedings of the ASME Computers in Engineering Conference, 1994.

Menon, S., Kim, Y. S., *Cylindrical Features in Form Feature Recognition Using Convex Decomposition*, Proceedings of the IFIP Conference on Feature Modeling and Recognition in Advanced CAD/CAM Systems, 1994.

Mitchell, S.R., Jones, R., Catchpole, G., *Modelling a thin-section sculptured product using extended form feature methods*, Journal of Engineering Design, Vol. 11, No. 4, pp 331-346, 2000.

Nyirenda, P.J., Mulbagal, M., Bronsvoort, W.F., *Definition of freeform surface feature classes*, Journal of computer-aided design and applications, Vol. 3, No. 5, 665-674, 2006.

Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D., *Matching models with shape distributions*, Proceedings of the international conference on shape modelling and applications, pp. 154-166, Genova, Italy, 2001.

Ovtcharova, J., Pahl, G., Rix, J., *A Proposal for Feature Classification in Feature-Based Design*, Computers & Graphics, Vol. 16, No. 2, pp. 187-195, 1992.

Pal, P., Tigga, A.M., Kumar, A., *Feature extraction from large CAD databases using genetic algorithm*, Computer-Aided Design, Vol. 37, No. 5, pp 545-558, 2005.

Pernot, J.-P., *Fully free form deformation features for aesthetic and engineering designs*, Ph.D. thesis, INP-Grenoble, 2004.

Pernot, J.-P., Falcidieno, B., Giannini, F., Leon, J.-C., *Fully free form deformation features for aesthetic shape design*, Journal of Engineering Design, Vol. 16, No. 2, pp 115-133, 2005.

Piegl, L.; Tiller, W., *The NURBS book*, Springer-Verlag, Berlin, 1996.

Poldermann, Horváth, I., *Surface design based on parameterized surface features*, Proceedings of the TMCE conference, 1995.

Pratt, M.J., Wilson, P.R., *Requirements for support of form features in a solid modeling system*, Report R-85-ASPP-01, CAM-1, 1985

Pratt, M.J., *Synthesis of an optimal approach to form feature modelling*, Proceedings of the ASME Conference on Computers in Engineering, San Francisco, 1988.

Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P., *Numerical Recipes in C++: the art of scientific computing*, Cambridge University Press, 2002.

Regli, W., Gupta, S, Nau, D., *Extracting alternative machining features: an algorithmical approach*, Research in Engineering Design, Vol. 7, No. 3, pp. 173-192, 1995.

Regli, W.C, Pratt, M.J., *What are feature interactions?*, Proceedings of the ASME Design Engineering Technical Conference and Computers in Engineering Conference, 1996.

Ribelles, J., Heckbert, P.S., Garland, M., Stahovich, T., Srivastava, V., *Finding and removing features from polyhedra*, Proceedings of the ASME Design Engineering Technical Conferences, 2001.

Rossignac, J.R., *Issues on feature based editing and interrogation of solid models*, Computer & Graphics, Vol. 14, No. 2, pp 149-172, 1990.

Sakurai, H., Gossard, D.C., *Recognizing Shape Features in Solid Models*, IEEE Computer Graphics and Applications, Vol. 10, No. 5, pp. 22-32, 1990.

Salomons, O., van Houten, F. J., Kals, H. J., *Review of Research in Feature-Based Design*, Journal of Manufacturing Systems, Vol. 12, No. 2, pp. 113-132, 1993.

Scheenstra, A., Ruifrok, A., Veltkamp, R., *A survey of 3D face recognition methods*, Proceedings of the AVBPA meeting, pp. 891-899, 2005.

Sederberg, T. W., Parry, S. R., *Freeform Deformations of Solid Geometric Models*, Computer Graphics, Vol. 20, No. 4, pp. 151–160, 1986

Shah, J.J., *Feature transformations between applications-specific feature spaces*, Computer-Aided Engineering Journal, Vol. 5, No. 6, pp. 247-255, 1984.

Shah, J.J., Rogers, M.T., *Expert form feature modeling shell*, Computer-Aided Design, Vol. 20, No. 9, pp. 515-524, 1988.

Shah, J.J., Hsiao, D., Leonard, J., *A systematic approach for design-manufacturing feature mapping*, Geometric modeling for product realization, Wilson, P.R., Wozny, M.J., Pratt, M.J., Editors, 1993.

Shah J.J., Ali, A., Rogers M.T., *Investigation of Declarative Feature Modeling*, Proceedings of the ASME '94 Computers in Engineering, pp. 1 - 11, 1994.

Shah J.J., Mäntylä M., *Parametric and feature based CAD/CAM*, Wiley-Interscience Publication, John Wiley Sons Inc. 1995.

Shah, J.J., Anderson, D., Kim, Y.S., Joshi, S., *A discourse on geometric feature recognition from CAD models*, Journal of computing and information science in engineering, Vol 1, pp. 41-51, 2001.

Song, Y., Vergeest, J.S.M., Bronsvoort, W., *Fitting and manipulating freeform shapes using templates*, Journal of Computing and Information Science in Engineering, Vol. 5, No. 2, pp. 86-94, 2005.

Song, Y., Vergeest, J.S.M, Langerak, T.R., *Selective clay milling for interactive prototyping*, Proceedings of IDETC/CIE 2005, New York, ASME, 2005.

Song, Y., Vergeest, J.S.M., Horvath, I., *Feature interference in free form template matching*, In: I Navazo Alvaro, Ph Slusallek, Editors, Eurographics 2002, pp. 9-18, Eurographics Association, 2002.

Sonthi, R. Kunjur, G., Radh, R., *Shape Feature Determination using the Curvature Region Representation*, Proceedings of the fourth symposium on Solid Modeling and applications, pp. 285-296, 1997.

Subrahmanyam, S., Wozny, M., *An overview of automatic feature recognition techniques for computer-aided process planning*, Computers in industry, Vol. 26, pp. 1-21, 1995.

Tang, K., Woo, T., *Algorithmic aspects of alternating sum of volumes. Part 1: Data structure and difference operation*, Computer-Aided Design, Vol. 23, No. 5, pp. 357-366, 1991.

Tang, K., Woo, T., *Algorithmic aspects of alternating sum of volumes. Part 2: Nonconvergence and its remedy*, Computer-Aided Design, Vol 23, No 6, pp. 435-443, 1991.

Thompson, W.B., Owen, J.C., de St. Germain, H.J., Stark, S.R., Henderson, T.C., *Feature-based reverse engineering of mechanical parts*, IEEE Transactions on robotics and automation, Vol. 15, No. 1, 1999.

Trika, S. N., and Kashyap, R. L., *Geometric reasoning for extracting manufacturing features in iso-oriented polyhedrons*, IEEE Transactions on Pattern Analvsis and Machine Intelligence. 1993. Vol 16, No. 11, 1087-1100.

Vandenbrande, J., Requicha, A.A.G., *Spatial reasoning for the automatic recognition of machinable features in solid models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 12, pp 1269-1285, 1993

Vandenbrande, J, Requicha, A.A.G., *Geometric computation for the recognition of spatially interacting machining features*, In: Advances in feature based manufacturing, Shah, J., Mäntylä, M, Nau, D., eds. Elsevier Science, New York, pp. 39-63, 1994

Varady, T., Martin, R., Cox., J., *Reverse Engineering Of Geometric Models - An Introduction*, Computer-Aided Design, Vol. 29, No. 4, pp. 255-268, 1997.

Varady, T., Facello, M.A., *New trends in digital shape reconstruction*, In: Martin, R.R., Bez, H., Sabin, M., Editors., The mathematics of surfaces XI, Springer, pp. 395-412, 2005.

Veltkamp, R.C., Hagedoorn, M., *Shape Similarity Measures, Properties, and Constructions*, Lecture Notes In Computer Science, Vol. 1929, pp. 467-476, 2000.

Veltkamp, R.C., Tanase, M., Sent, D., *Features in Content-Based Image Retrieval Systems: A Survey*. In Veltkamp, R.C., Burkhardt, H., Krieger, H.-P., Editors, State-of-Art in Content-Based Image and Video Retrieval, pp. 97-124, 2001.

Veltkamp, R.C., Hagendoorn, M., *State-of-the-art in shape matching*, pp. 87-119, Principles of visual information retrieval, Springer, 2001

Venkataraman, S., Sohoni, M., Kulkarni, V., *A graph-based framework for feature recognition*, Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 194-205, 2001.

Vergeest, J.S.M., Song, Y., Broek, J.J., *Integrating traditional and digital modeling of freeform product concepts using 3d scanning technology*, Proceedings of the TMCE conference, 2004.

Vergeest, J.S.M., Spanjaard, S., Horváth, I, Jelier, J.J.O., *Fitting Freeform Shape Patterns to Scanned 3D Objects*, Journal of Computing and Information Science in Engineering, Vol. 1, No. 3, pp. 218-224, 2001.

Vergeest, J.S.M., Spanjaard, S., Wang, C., Song, Y., *Complex 3D feature recognition using a marching template*, Full paper proceedings of the WSCG conference, pp. 243-250, 2003.

Vergeest, J.S.M., Horváth, I., *Parameterization of freeform features*, Proceedings of the International Conference on Shape Modeling & Applications, pp. 20-28, 2001.

Vergeest, J.S.M, Spanjaard, S., Song, Y., *Directed Mean Hausdorff Distance of parameterized freeform shapes in 3D: a case study*, The visual computer, Vol. 19, No. 7-8, pp. 480-492, 2003.

Vergeest, J.S.M., Horvath, I., Spanjaard, S., *A methodology for reusing freeform shape content*, Proceedings of the 2001 Design Theory and Methodology Conference, DETC'01/DTM-21708, ASME, New York, 2001.

Vergeest, J.S.M., Song, Y., Langerak, T.R., *Design intent management for design reuse*, In Rohatynski, R., Poslednik, P., Editors, Proceedings of the Design Methods for Practice, pp. 163-170, Zielona Gora: University of Zielona Gora, 2006.

Vosniakos, G., *Investigation of feature-based shape modeling for mechanical parts with free form features*, International Journal of Advanced Manufacturing Technology, Vol. 15, No. 3, pp. 188-199, 1999.

Woo, T., *Feature Extraction by Volume Decomposition*, Proceedings of the Conference on CAD/CAM Technology in Mechanical Engineering, pp. 76–94, 1982.

Wilson, P.R., Pratt, M.J., *A taxonomy of features for solid modeling*, Geometric Modeling for CAD applications, Wozny, M.J., McLaughlin, H.W., Encarnacao, J.L., (eds.), Elsevier Science Publishers, 1988

Wu, M.C., and C.R. Liu, *Analysis on Machined Feature Recognition Techniques Based on B-Rep*, Computer-Aided Design, Vol. 28, No. 8, pp. 603-616, 1996.

Summary

Form features are characteristic shapes or parts of shapes to which parametric data can be attributed. This data makes possible a parameter-based deformation of the shape. Parameter data can be attributed to shape data by reference to existing feature definitions. In these features definitions, the relation between parameters and shape can be pre-defined. In the past decades, features have played an important role in the automation of the design process. By using features in a computer-aided design (CAD) process, a designer can efficiently access and manipulate shape data. In addition, form features play a role in the validity maintenance of shapes.
In most computer-aided design processes, form features are included in a design during the construction of a design. However, if one wants to introduce physical objects into a design or use other sources of non-parameterized shape data, then a process of feature recognition is needed to interpret parts of this data as a form feature. A large part of the existing literature on form feature recognition deals with so-called regular form features, which are features of which the shape is derived from geometric primitives. Nowadays, freeform shape has become the standard in the domain of industrial design and regular form features have become outdated. To be able to apply the feature concept in the freeform domain, new theory and methodology is needed.
However, very little theory currently exists on the topic of freeform features. Most of the existing theory is based on the theory for regular form features, and therefore falls short of supporting the freeform feature concept. The freeform feature-based methodology that has been developed is based on this insufficient theory and is therefore limited in terms of correctness, completeness, validity and generalizability. In addition, the existing methodology is based on application-specific assumptions and can not be combined into a comprehensive methodology. To overcome these problems, a new theory must be developed that is based on the freeform feature paradigm.

In a Ph.D. research project, the development of new freeform feature recognition techniques has been investigated. The goal of the research was to develop techniques that support the reuse of shape in the design process by recognizing the features in the shape data. However, in an early stage of research the above shortcoming of theory and methodology for freeform features has been recognized. For this reason, the problems that were targeted in the research came to include the development of a foundational theory for freeform features on the basis of which a methodology could be developed and implemented.
The structure of the thesis is as follows: First, a literature study is presented and discussed, on the basis of which hypotheses were formulated. On the basis of the assumptions that

could be derived from these hypotheses, an exploration of the freeform feature concept is presented and a foundational theory for freeform features is given. On the basis of this theory, a methodology is proposed that is directed at the problem of freeform feature recognition but also targets related and supporting freeform feature-based operations. Finally, algorithms are given for the application of the developed theory on the problem of freeform feature recognition.

Literature study

To analyze the state of the art of research on freeform features, to establish the position of the freeform feature concept with regard to related research problems and to identify the results of feature recognition techniques in the domain of regular form features, a literature study has been conducted. The literature study has two important results: first, it can be concluded that many regular feature recognition methods are based on assumptions that do not hold in the domain of freeform shapes and freeform features. Second, three approaches to the freeform feature concept could be identified in the existing literature on freeform features. Only one of these approaches, in which features are conceptualized as entities that are instantiated on a target surface, was found to be suitable for application to the problem of freeform feature recognition.

Development of a foundational theory

In the main body of the research, a foundational theory has been developed that is based on the concept that features must be described on the level of geometric and the level of morphology. On the level of geometry, the shape of a feature is described, optionally with regard to an embedding surface. On the level of morphology, a feature is described in terms of functions that determine how the shape of a feature can be manipulated. Several theoretical aspects of freeform features are described in this thesis, and the freeform feature itself is defined as a construct that contains a basic shape configuration of the feature shape, a parametric configuration and a description of the morphology of the feature.

An implementation of the theory was given, in which the geometry of a feature is represented by a set of control points and in which the morphology is operationalized as transformation matrices that operate on these control points. The configuration of these transformation matrices is based on the actual parametric configuration of a feature. On the basis of the foundational theory, computational models are given for the recognition of features, as well as for the operations that play a role prior to or during a feature recognition process.

Methodology for freeform-feature based operations

Corresponding to these computational models, algorithms are given for the definition and instantiation of features and for feature recognition. For the recognition of features, first a re-implementation of an existing method, the template matching method, is given. This method matches a template feature onto a target shape, and so recognizes simple features on a target shape with an accuracy that is large enough to be able to manipulate the target shape on the basis of the parameters of the template feature. However, because the morphology of a template feature is not changed, the quality of the match between template feature and target surface is limited. In addition, the recognized feature is incomplete, as it does not include a basic shape configuration for the feature. Finally, the template feature cannot be easily used for other purposes than shape manipulation, as it is not embedded in the target surface.

Development of algorithms for freeform feature recognition

To improve on the template matching approach, an evolutionary feature recognition method was developed, that consists of three stages. First, a template feature is matched to a target surface by changing its parametric configuration as well as its morphology. Then, the region of the target surface that corresponds to the matched template feature is removed. Finally, the template feature that was found in the first stage is instantiated on the modified target surface, and in two more evolutionary procedures this instance is adapted to match the original target surface.

The evolutionary feature recognition method can only be applied to features with scalar parameters. To demonstrate that it can easily be adapted to also recognize features with curve-based parameters, this is demonstrated for a specific type of feature: the curve-based feature. In a method called the curve-based feature recognition method, first a template feature is constructed based on an analysis of the curvature of a target surface. This constructed template feature can then be recognized in an evolutionary feature recognition procedure.

The evolutionary feature recognition method is an improvement on the template matching method, because it enables the recognition of freeform features without the mentioned shortcomings. In addition, it improves the accuracy with which freeform features can be recognized.

Validation and verification

A pilot implementation of the developed methods has been developed and validated in an application study. In this study, the accuracy and computation time of the methods are compared to those of the re-implemented template matching method. This study shows that evolutionary feature recognition method is slightly slower than the template matching method, but its accuracy is considerably higher. Finally, the proposed theory is

verified, and the computational complexity of the developed algorithms is analyzed and tested.

The achievements of the research work can be summarized as follows:

- A comprehensive theory has been developed for freeform features in general and for freeform feature recognition in particular. Such a theory was previously unavailable and supports the development of freeform feature-based methodology. Although in this thesis the theory is mainly applied to freeform feature recognition, it has the potential to be extended to other domains of application.
- Methods have been developed that address most of the shortcomings of existing methods for freeform feature recognition. In addition, the developed methods can be used to recognize features with a considerably higher accuracy than existing methods.

To be able to complete the research in four year, assumptions had to be made that limit the applicability of the given algorithms. Specifically, two categories of features were identified to which the algorithms cannot be applied, namely feature that change the topology of their embedding surface (e.g. holes) and features with multi-dimensional parameters. By developing the curve-based feature recognition algorithm, we demonstrated that the developed theory and methodology can be extended to also support these categories of features, but the actual implementation of this extension is a topic for future research.

Thomas Robin Langerak

Samenvatting

Vormfeatures zijn karakteristieke vormen of delen daarvan waaraan parameters toegekend kunnen worden. Deze parameters maken het mogelijk om de vorm in kwestie parametrisch te vervormen. Het toekennen van parameters gebeurd door de betreffende vorm te vergelijken met bestaande definities van features. In deze definities is de relatie tussen parameters en vorm al vastgelegd.
In de afgelopen decennia hebben vormfeatures een belangrijke rol gespeeld in het automatiseren van het ontwerpproces. Door vormfeatures te gebruiken in een computer-ondersteund ontwerpproces kan een ontwerper snel vormen manipuleren. Bovendien spelen vormfeatures een rol het bewaken van de validiteit van het ontwerp.
In de meeste computer-ondersteunde ontwerpprocessen worden vormfeatures gedurende het ontwerpproces aan een ontwerpmodel toegevoegd. Echter, wanneer men gebruik wil maken van fysieke objecten in het ontwerp of van andere bronnen van niet-geparameteriseerde data, dan is een proces van featureherkenning nodig om de betreffende data te kunnen interpreteren als parametrisch. Een groot deel van de bestaande literatuur over het probleem van featureherkenning betreft zogenaamde regelmatige vormfeatures. Dit zijn features waarvan de vorm is afgeleid van geometrische basisvormen. Tegenwoordig zijn vrije vormen de standaard in het ontwerpproces en zijn regelmatige vormen gedateerd geraakt. Om het concept van features ook in het domein van vrije vormen toe te kunnen passen is nieuwe theorie en methodologie nodig.
Er is echter weinig theorie beschikbaar op het gebied van vrijgevormde features. Het grootste deel van de bestaande theorie is gebaseerd op de theorie voor regelmatige vormfeatures en schiet tekort bij het ondersteunen van vrijgevormde features. De methodologie die is gebaseerd op deze theorie is beperkt wat betreft aspecten als juistheid, volledigheid, validiteit en generaliseerbaarheid. Bovendien is de bestaande methodologie voor vrijgevormde features gebaseerd op situatie-specifieke aannames en kan om deze reden niet gecombineerd worden in één samenhangende methodologie. Om deze problemen op te lossen moet nieuwe theorie specifiek voor vrijgevormde features ontwikkeld worden.

In dit promotietraject lag de focus oorspronkelijk op de ontwikkeling van nieuwe technieken die het hergebruik van vormen ondersteunen door vrijgevormde features te herkennen. In een vroeg stadium is echter het probleem van het eerder genoemde ontbreken van theorie en methodologie voor vrijgevormde features gesignaleerd. Om deze reden verschoof de focus van het onderzoek naar het creëeren van nieuwe theorie en, op basis van deze theorie, het ontwikkelen van nieuwe methodologie.

De opbouw van het proefschrift is als volgt: als eerste is een literatuurstudie uitgevoerd, op basis waarvan hypotheses zijn geformuleerd. Met behulp van de assumpties die uit deze hypotheses afgeleid kunnen worden is het concept van de vrijgevormde feature verkend en is een fundamentele theorie ontwikkeld. Aan de hand van deze theorie is een methodologie ontwikkeld voor het herkennen van vrijgevormde features, maar ook voor de gerelateerde en ondersteunende aspecten van vrijegvormde features. Tenslotte is de methodologie geïmplementeerd in de vorm van algoritmen voor de herkenning van vrijgevormde features.

Literatuurstudie
Om de nieuwste wetenschap op het gebied van vrijgevormde features te kunnen analyseren, om de positie van wetenschap op het gebied van vrijgevormde features ten opzichte van gerelateerde onderwerpen te kunnen bepalen en om de resultaten van onderzoek op het gebied van regelmatige features te kunnen identificeren is een literatuurstudie verricht. De literatuurstudie had twee belangrijke resultaten: ten eerste kon worden geconcludeerd dat veel methoden voor de herkenning van regelmatige features gebaseerd zijn op aannames die niet gedaan kunnen worden voor vrijgevormde features. Ten tweede konden drie benaderingen van features worden geïdentificeerd. Slechts één van die benaderingen, waarin features worden opgevat als zijnde gevat in een onderliggend oppervlak, werd geschikt geacht voor een toepassing op feature-herkenning.

Ontwikkeling van een fundamentele theorie
Het zwaartepunt van het promotieonderzoek ligt bij het ontwikkelen van een fundamentele theorie, die gebaseerd is op het uitgangspunt dat zowel de geometrische als de morfologische aspecten van een feature in aanmerking moeten worden genomen.
Op het niveau van geometrie moet de vorm van een feature worden beschreven, al dan niet met betrekking tot een onderliggend oppervlak. Op het niveau van morfologie moet een feature beschreven worden in termen van functies die bepalen hoe de vorm van een feature gemanipuleerd kan worden. In de theorie zijn verschillende aspecten van het feature concept beschreven en is de vrijgevormde feature omschreven als een geheel dat een beschrijving van de basisconfiguratie van de feature, parameters en een zogenoemde ‘parameter mapping’ omvat. De parameter mapping is een beschrijving van de relatie tussen parameters en vorm.
In dit proefschrift wordt een implementatie van de theorie gegeven, waarin de geometrie voorgesteld wordt als bepaald door een set control points, en wordt de parameter mapping geoperationaliseerd door middel van transformatiematrices die kunnen worden toegepast op de control points. De configuratie van de transformatiematrices wordt bepaald door de parameterwaarden van de feature. Op basis van de fundamentele theorie

zijn rekenkundige modellen gegeven voor de herkenning van features, alsmede voor de andere toepassingen van features die een rol spelen in het herkenningsproces.

Een methodologie voor op features gebaseerde toepassingen
Corresponderend met de computationele modellen zijn algoritmen gegeven voor definitie en de instantiatie van vrijgevormde features. Voor het herkennen van features is eerst een implementatie gegeven van een bestaande methode, de 'template matching' methode. Deze methode vergelijkt een sjabloon van een feature met de vorm die men als feature wil herkennen (de doelvorm). De nauwkeurigheid van deze methode is groot genoeg om deze vorm vervolgens te kunnen manipuleren met behulp van de parameters die voor de sjabloon gedefinieerd zijn. Echter, omdat de morfologie van de sjabloon feature niet aangepast wordt, is de nauwkeurigheid van deze methode niet optimaal. Bovendien is de herkenning van de feature niet volledig, omdat het basisoppervlakte van de feature niet herkend kan worden. Tenslotte is de nauwkeurigheid weliswaar groot genoeg voor manipulatie, maar niet voor andere toepassingen.

De ontwikkeling van algoritmen voor de herkenning van vrijgevormde features.
Om de template matching methode te verbeteren is een evolutionaire methode voor feature herkenning ontwikkeld, die bestaat uit drie delen. Eerst wordt een sjabloon feature vergeleken met de doelvorm, zoals dat ook in template matching gebeurd. Vervolgens wordt de herkende feature verwijderd van de doelvorm en de vorm die zo ontstaat wordt opgevat als de basisconfiguratie van de te herkennen feature. Tenslotte wordt op de nieuw ontstane doelvorm een feature geïnstantieerd; de ontstane vorm wordt vergeleken met de oorspronkelijke doelvorm.
De methode voor evolutionaire feature herkenning kan alleen toegepast worden op features die getalsmatige parameters hebben. Om te demonstreren dat de methode ook toegepast kan worden op curve-parameters wordt hiervoor tevens een methode gegeven. In deze methode, die aangeduid wordt als 'curve-based feature recognition', wordt een sjabloon stap voor stap opgebouwd door het analyseren van de doelvorm. De zo opgebouwde sjabloon kan vervolgens onderworpen worden aan een evolutionaire feature herkenning.
De methode voor evolutionaire feature herkenning is een verbetering van de template matching methode omdat het features met meer nauwkeurigheid herkent en ook de basisconfiguratie herkent.

Validatie en verificatie
De ontwikkelde methodes zijn geïmplementeerd in de vorm van verschillende algoritmen en deze zijn in een validatiestudie onderzocht. In deze studie zijn de nauwkeurigheid en de benodigde rekentijd vergeleken met die voor een herimplementatie van de template

matching methode. Deze studie wees uit dat hoewel de evolutionaire featureherkenning methode iets meer rekentijd nodig heeft, de nauwkeurigheid vele malen groter is.
Vervolgens is de consistentie van de ontwikkelde methode onderzocht en is de rekenkundige complexiteit geanalyseerd.

De toegevoegde waarde van het onderzoek in dit proefschrift kan als volgt worden samengevat:

- Een samenhangende theorie is ontwikkeld, voor vrijgevormde feature in het algemeen en specifiek voor de herkenning van deze features. Deze theorie was tot nu toe niet beschikbaar. Alhoewel de theorie in dit proefschrift voornamelijk toegepast wordt op de herkenning van features kan hij potentieel uitgebreid worden tot andere toepassingen van features.
- Er zijn methoden ontwikkeld die de meeste tekortkomingen van bestaande methoden vermijden. Bovendien kunnen de ontwikkelde methoden features met een grotere nauwkeurigheid herkennen.

Om het promotieonderzoek in vier jaar af te kunnen ronden zijn bepaalde aannames gemaakt die de toepasbaarheid van de theorie en methodologie deels beperken. Specifiek kunnen twee categorieën features genoemd worden waarop de ontwikkelde theorieën niet toegepast kunnen worden, namelijk features die de topologie van het onderliggend oppervlak veranderen (zoals gaten) en features met meerdimensionale parameters. Door de curve-based feature herkenning methode te ontwikkelen is gedemonstreerd dat de ontwikkelde technieken makkelijk kunnen worden aangepast voor meerdimensionale parameters, maar de daadwerkelijke aanpak van dergelijke features blijft een uitdaging voor de toekomst.

Thomas Robin Langerak

Index

Curriculum vitae

Thomas Robin Langerak was born in Tilburg in 1978. He attended the grammar school Johan van Oldebarneveldt in Amersfoort, where he obtained a VWO diploma in 1995.
In the same year he enrolled in the faculty of Mathematics and Computer Science at Utrecht University, where he obtained a MSc degree in 2003.
In March 2004 he started on a promotion research project at the Delft University of Technology, faculty of Industrial Design Engineering, in the section of Computer Aided Design Engineering. This thesis is the result of this promotion research.