Feature Recognition of User-Defined Freeform Features

Thomas R. Langerak and Joris S. M. Vergeest

Delft University of Technology, {T.R.Langerak, J.S.M.Vergeest}@tudelft.nl

ABSTRACT

In the past, several methods for feature definition have been proposed, most recently for freeform features. Among these methods are several examples of user-driven feature definition, in which users can define their own feature types. The motivation behind user-defined features is that they can consequently be used in a feature-oriented application. One of these applications is feature recognition, in which parts of an existing shape model are recognized as an instance of a predefined feature type. However, little attention has been paid to making user-defined feature types available for application in feature recognition. This paper proposes a basic user-driven feature definition method and shows how features defined using this method can be used in a feature recognition procedure. This enables users to recognize any features on any target shape, providing that they can be defined using the proposed method. At the same time, the link between user-defined features and feature recognition helps to keep the set of predefined features, the feature library, small.

Keywords: Freeform feature, user-defined features, feature recognition, feature-based modeling, feature library, feature taxonomy.

1. INTRODUCTION

In the second half of the 20th century, Computer-Aided Design (CAD) was introduced into the world of designing, and in the decades after its first appearance, CAD went through a rapid development. The possibility to virtually construct and visualize designs where this was traditionally done with sketches was a huge asset to the design process. But as the complexity of CAD models increased, so did the amount of effort that was required from the designer. As a result, many CAD applications today automate (parts of) the modeling process, and offer designers a high-level interface to the shape data. This high-level interface allows designers to modify a large part of the model with only simple actions. One of the possible approaches to offering designers a high-level interface is through the use of form features, which can be defined as parametric descriptions of shape data. Using form features, a designer can modify part of a CAD model by setting one or more parameter values. Basic geometric shapes such as a sphere or a cube are simple examples of features: their shape is driven by parameters such as radius or width, height and length. The power of the notion of features increases when more complex features are considered. For readability purposes, in this paper the term 'feature' will be used short for 'form feature'.

Apart from the high-level interface to shape data, features offer some additional benefits. With the use of features a direct translation can be made between parameterized shape data in a CAD system and manufacturing routines in a Computer-Aided Manufacturing (CAM) system. In addition, features are useful in the geometric and semantic reasoning about CAD-models, leading to advances in the validity management of shape models and the support of designers.

Although many theories on features have been developed and published, several topics in feature research are still open. The fast pace of technological progress in the area of CAD has led researchers to focus on developing new methods and systems, instead of establishing a widely accepted theoretical basis for the different aspects of feature research. Regli and Pratt[10], for example, state that "…there has not emerged any general consensus as to what features are, how they are defined and what it means when one says that features interact". Although this quote dates back to 1996, to our opinion no developments have taken place since then that make it less appropriate. Strangely, although many open research questions exist, the interest for features has decreased over the past years, assumedly because it is no longer driven by the need to bridge the gap between CAD and CAM.

The research on features can be roughly subdivided into two main topics: feature recognition and feature-based design. In this paper we focus on the topic of feature recognition, which deals with retrieving parametric information from existing shape data. The main application of feature recognition lies in linking CAD to CAM [4,16,19]. Another, less fervently researched application lies in the translation from one CAD application to the other: when feature definitions are not shared between the two applications, then parametric information needs to be reconstructed [13]. Finally, feature recognition is required when shape data is acquired from physical shapes, for example by laser range scanning [17]. In this case, parts of the acquired model can be perceived by the designer as parametrically deformable, but the associated manipulation handles are only available to the designer once feature recognition has been applied to the model.

With the introduction and general acceptance of NURBS, feature theory moved into the freeform domain and the need for a theoretical basis of feature research was renewed. Many feature recognition techniques that have been developed for regular features appear not to be applicable to the freeform domain. Efforts are being made to develop new techniques for feature recognition in the freeform domain, but the results are very preliminary [7, 15].

Elevating feature recognition to the freeform domain poses a lot of challenges: apart from the fact that new techniques are needed to deal with a larger variety of features, these techniques also need to be able to deal with the increased complexity of freeform shape models. As the creation of one all-enveloping freeform feature recognition technique is not feasible at this moment, a more logical approach would be to perform feature recognition by comparing (part of) a target shape to one or more features from a collection of predefined feature templates, typically referred to by different authors as a *feature library*, a *feature database*, or a *feature taxonomy*.

In the domain of regular features, several feature taxonomies have been proposed such as that by Wilson and Pratt [18]. They mention through-holes, depressions, protrusions and areas as basic elements of a feature taxonomy, but admit that more classes are needed to capture every possible feature type. Ovtcharova et al. [10] also mention pattern features and compound features as significant elements of a feature taxonomy. In the freeform domain, a feature taxonomy has been proposed by Fontana et al. [3]. They define features as being either a modification or an elimination of an existing surface, and further split up these two categories to obtain a feature taxonomy. A generic freeform feature taxonomy has been given by Nyirenda et al. [8], who propose a set of basic feature classes and a mechanism to add new features to the taxonomy by deriving them from features that are already in the taxonomy.

Other authors have argued that a feature library should be an interactive, dynamic set of features, instead of the static feature taxonomies mentioned above, for example because the set of features that a user wants to have access to depends highly on the domain in which the user operates. Dong and Wozny [2] present a method to define customized features in either a B-rep or a CSG representation and show how such a user-defined feature can be instantiated on an existing geometric model. Hoffmann and Joan-Arinyo [5] propose a procedural mechanism for generating features and instantiating them on existing models. Van den Berg et al. [1] describe a declarative definition method for freeform feature definition, in which features are defined by using characteristic points which they call Freeform Feature Definition Points. Relationships between these points can be created, such as distance and angle, and these can be used for both geometry generation and constraint solving. This method has been extended to the freeform domain by Nyirenda et al. [9].

However, the literature usually stops with feature taxonomies and methods of user-driven feature recognition. To our knowledge, there is no existing work on how these taxonomies or user-defined features can consequently be used for feature-based design or feature recognition. In reverse, the known feature recognition methods all make their own assumptions on the features involved in the recognition process. These methods are therefore unable to handle customized features, or at least it has been undefined how they do so. With the absence of a link between feature definitions and the application of these definitions in different applications, the practical use of both pre-defined feature taxonomies and user-driven feature definitions is severely limited.

This paper proposes a new method for user-driven definition of freeform features and shows how any user-defined feature can be used in a feature recognition procedure that was developed by the authors earlier. In section 2, a definition method for features is proposed and it is shown how the method can be used to easily create new features. In section 3 the feature recognition method that was proposed in [7] will be reviewed and it will be shown how the method is able to deal with any feature definition generated by the definition method proposed in section 2. Section 4 gives an application example and section 5 contains a discussion of the proposed method and conclusions.

2. THE DEFINITION OF A FREEFORM FEATURE

Many definitions of a feature have been given in the regular feature domain. Although no general consensus has been achieved with regard to a single feature definition, many aspects of a feature definition have been mentioned by different authors. In the freeform domain, however, it is less clear what elements a complete feature definition should contain. The feature definition presented in this chapter is a simple approach that is based on some of the definition elements that have been mentioned in previous literature. It is shown in section 3 that the definition is sufficient for the purpose of feature recognition, but we do not claim that the definition is complete in that it incorporates all aspects that a freeform feature should contain. Apart from the fact that a complete feature definition falls outside the scope of this paper, the research of freeform features is too immature to draw conclusions on such a feature definition.

2.1 A Theoretical Basis for Feature Definition

From existing definitions of form features, it can be noted that a feature definition includes, naturally, a shape representation and a parametric representation. In addition to these two elements, some mechanism is needed to connect the shape representation to the parametric representation.

One of the goals of using features is to give users a parametric interface to shape data and therefore the shape description cannot be abstract. It has to be presented to the user in the form of a set of descriptive elements. This set can consist of vertices, edges and faces (a polygonal mesh), control points (a NURBS patch) or even variables and constants in the case of an implicit shape. In this paper, the focus lies on freeform shapes and we therefore assume that features are defined using B-spline control points as descriptive elements.

The shape is controlled by the parameters defined on the feature: each combination of parameter values indicates a specific configuration of the shape. In other words, if a feature has *m* parameters, then a configuration of the shape corresponds to a position in the *m*-dimensional parametric space of the feature. The relation between the parameters and the elements that describe the feature shape is expressed in a *parameter mapping*, which is a set of functions on the injective relation between the set of geometric elements and the set of parameters. It is assumed that parameter influence is linear; although non-linear parameter influence can be defined as easily, it is our opinion that a non-linear interface to shape data is unintuitive and should therefore be avoided.

All the aforementioned considerations lead to the following formal definition of a feature:

A freeform feature $F = \{E, P, \mu\}$ is a shape that is described by a set of shape descriptors E that are parametrically deformable by the parameter mapping μ, and of which the actual state depends on the set of parameter values P.

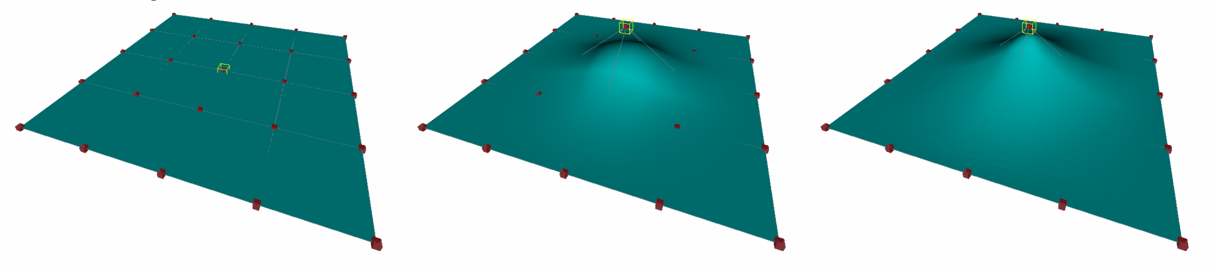
In this definition, E indicates the shape representation of the feature, expressed in k^E sets of control points $E = \bigcup_{k^E} E_k$ with subsets $E_k = \{e_{0,0,k}, \ldots, e_{r_k,c_k,k}\}$ that can be organized in bidirectional sets $E_k = \begin{pmatrix} e_{0,0,k} & \cdots & e_{0,c_k,k} \\ \vdots & \ddots & \vdots \\ e_{r_k,0,k} & \cdots & e_{r_k,c_k,k} \end{pmatrix}$, r_k and c_k being respectively the row and column size of E_k. All control points are given as $e_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j}, w_{i,j})$, where x, y and z are the three-dimensional Cartesian coordinates of the control point and w is its weight. Organizing the control points in subsets is necessary to be able to define *pattern features*, which are topologically unconnected features that nonetheless respond to the same parameters. However, unless specifically stated otherwise, in the remainder of this paper it is assumed that $k = 1$. For the sake of clarity, if $k = 1$, it will be omitted, so E will be used short for E_1, $e_{i,j}$ short for $e_{i,j,k}$. The parameter set P is a set of m real numbers $P = \{p^1, \ldots, p^m\}$ and the parameter mapping is a set of functions $\mu = \{\mu^l_{i,j} : E \times P \to E \mid e_{i,j} \in E, p^l \in P\}$. Each parameter mapping $\mu^l_{i,j}$ is a function of the type $(e, p) \to e$ that gives the effect of a parameter p^l on the control point $e_{i,j}$. In this paper, we assume that each function $\mu^l_{i,j}$ affects the position and weight of a control point by means of a translation vector $T^l_{i,j} = (\Delta^l_{i,j}x, \Delta^l_{i,j}y, \Delta^l_{i,j}z, \Delta^l_{i,j}w)$. We define $\mu^l_{i,j}(e_{i,j}, p^l) = e_{i,j} + p^l T^l_{i,j}$, and the actual state of a control point $e_{i,j}$ can be determined by computing the accumulated effect of all translation vectors for that control point. Denoting the initial state (in which there is no parameter influence) of an element as $e^0_{i,j}$, the actual state of a control point can be defined as $e_{i,j} = (x^0_{i,j}, y^0_{i,j}, z^0_{i,j}, w^0_{i,j}) + \sum_{l=1}^{m} p^l T^l_{i,j}$.

Once the final position and weight of the control points has been determined, a shape S can be determined using the weighted de Casteljau's algorithm [11]. In this paper, we assume that all B-splines are uniform and of degree 3.

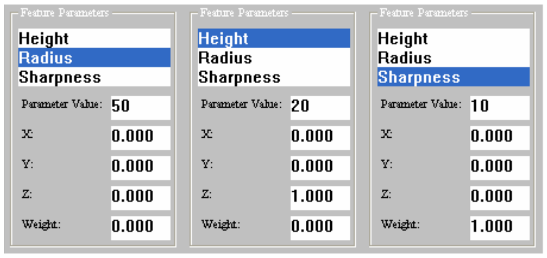
It must be noted that the definition given above can not simply be used to define elimination features. However, although the elimination feature has been mentioned by both Fontana et al. [3] and Nyirenda et al. [8], in the freeform domain (where the CSG representation is not easily applicable) no implementation or definition of elimination features has so far been given. Dealing with elimination features seems a broader problem, which perhaps requires a separate theoretical model.

2.2 A User Interface to Feature Definition

The definition of a feature F requires the user to specify the basic positions of the control points $e_{i,j}$ of the feature, as well as, for each control point, the parameter mappings of the m parameters defined on the feature. However, this requires a large amount of effort from the user. For example, a feature that is defined using a 5 by 5 control point grid and 3 parameters requires the user to enter 400 numerical values to specify the parametric behavior of the feature.
The parametric behavior of a feature is usually determined by only a subset of the control points. Fig. 1 shows the polygonal mesh representation of a 'Bump' feature, defined on a grid of 5 by 5 control points with 3 parameters. Of all the numerical values that determine the behavior of the feature, only 52 numerical values are non-zero. By using default values for certain parameters, this has been further brought back to only 2 values that are requested from the user. For example, by default, $e_{i,j}^0 = (0,0,0,0)$, so the initial state of a feature is a point. Figures 1a-c show the behavior of the defined parameters.



(a) (b) (c)



Feature Parameters	Feature Parameters	Feature Parameters
Height	Height	Height
Radius	Radius	Radius
Sharpness	Sharpness	Sharpness
Parameter Value: 50	Parameter Value: 20	Parameter Value: 10
X: 0.000	X: 0.000	X: 0.000
Y: 0.000	Y: 0.000	Y: 0.000
Z: 0.000	Z: 1.000	Z: 0.000
Weight: 0.000	Weight: 0.000	Weight: 1.000

(d)

Fig. 1(a)-(c): Three polygonal mesh representations of instances of the bump feature. (d) The parameter values for fig. (a)-(c) and the parameter mappings for the selected point (indicated with a bounding box).

Fig. 1a shows the default behavior of the radius parameter, which is defined as $T_{i,j}^{radius} = (2-0.5i, 2-0.5j, 0, 0)$. Fig. 1b shows the behavior of the height parameter, for which the default translation vector is $(0,0,0,0)$; only for the central point it is defined as $(0,0,1,0)$. The same holds good for the sharpness parameter, which influences only the weight of the central control point with a translation vector of $(0,0,0,1)$. The behavior of the sharpness parameter is shown in fig. 1c. Fig. 1d shows the interface for the parameter values and the parameter mappings for the central point.

When creating a new feature definition, a user can assign a set of predefined values to a single parameter mapping or to the patch of control points as a whole. Of course, a user is also able to define his/her own default values.

2.3 Feature Space

The feature definition proposed in the previous section enables the user to create new feature types in a feature library. These features must then be applied in processes of feature-based design and feature recognition, and the way their shape is computed may have to be adapted to the local curvature of the surface the features are instantiated on. However, while defining a feature, a user should not be bothered by the eventual local surface. We therefore introduce the concept of *feature space*. Feature space is a three-dimensional environment in which all features are defined with the xy-plane as a base surface. In this section we will show how features defined in feature space can be mapped to any surface. Feature space and modeling space form a dual environment: for each feature, two corresponding instances are maintained - one in feature space, the other in modeling space.

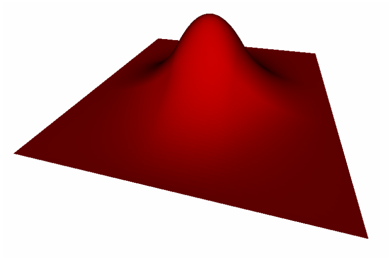
To be able to translate features from feature space to modeling space, we distinguish two types of parameters: *area* parameters and *deform* parameters. Area parameters determine the area of the base surface in which the feature affects a certain deformation; in feature space, area parameters indicate a displacement parallel to the base surface. Deform parameters determine the extent of the deformation and indicate a displacement that has a non-zero z-component. The subsets of respectively area and deform parameters are denoted p^{area} and p^{deform} .

A feature for which all deform parameters have zero influence is a subsection of its base surface. This subsection is called the *area* $A(u,v)=\sum_{i=0}^{r}\sum_{j=0}^{c}N_{i,3}(u)N_{j,3}(v)\left(e_{ij}^{0}+\sum_{p\in p^{area}}p\mu_{j}^{i}\right)$ of the feature, with N the Cox-de-Boor recursion formula for degree 3. Using the two types of parameters, the following method is used to map a feature F to modeling space:

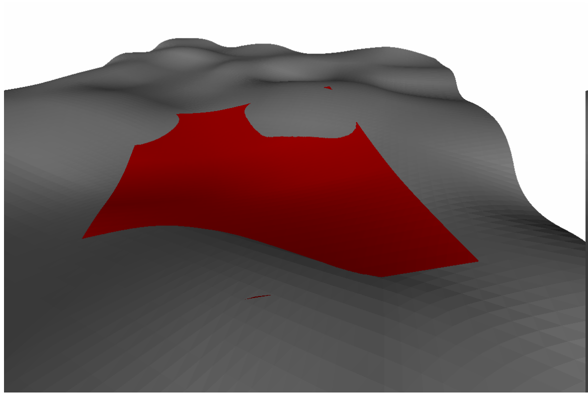
1. A target surface is selected and a target point chosen for the feature to be instantiated. This point is denoted as the *attachment point*. The attachment point corresponds to the origin of a feature in feature space.

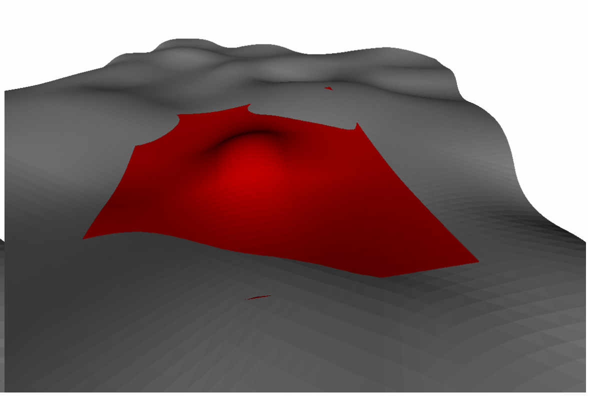
   The normal of the target surface at the attachment point is denoted as $\overline{a}$, and it is assumed that this vector is a unit vector. A direction vector $\overline{d}$, a unit vector as well, is chosen perpendicular to $\overline{a}$. The direction vector indicates the orientation of the feature and can later be used to rotate it on its base surface.
2. A feature area is instantiated on a plane that is perpendicular to $\overline{a}$. This is done in two steps; first, a copy of the area of F is instantiated on the xy-plane. This area is then translated to the target location.
3. Each control point $e_{i,j}$ is projected onto the target surface in the reverse direction of the normal of the attachment point. The normal of the target surface in the projected control point is denoted as $\overline{n}_{i,j}$.
4. For each control point $e_{i,j}$, the deformation vectors defined on that control point are adapted to the normal of the base surface by multiplying them with the transformation matrix *M* that aligns the vectors $\overline{z}=(0,0,1)$ and $\overline{x}=(1,0,0)$ (the normal of the control point and the direction of the feature in feature space) with the vectors $\overline{n}_{i,j}$ and $\overline{d}$.
5. A final position of each control point $e_{i,j}$ is determined by displacing it with $\sum_{p^k\notin P^{deform}}p^kT_{i,j}^k$ and the final state of the feature is computed using the de Casteljau's algorithm [11].

The result of this translation is shown in Fig. 2. Fig. 2a shows the feature in feature space, and Fig. 2b and 2c show the same feature, instantiated on a randomly generated terrain.



(a)





(b) (c)

Fig. 2: A bump feature (a) in feature space (b) as an area on a target surface in modeling space (c) as a deformed feature in modeling space.

2.4 Feature Composition

One of the methods to create more complex features is by combining existing features into a compound feature, of which the shape as well as the parametric behavior is derived from the individual features it is composed of. Although the constitution of the individual features is maintained in the composition process, the combination of parameters and parameter mappings allows the user to manipulate the feature as if it were a single feature.
In section 3 it will be shown how feature composition can be used to be able to recognize interfering features.

Two features can be combined in three different ways:

- One feature sits on top of another feature. In this case the latter feature is the base surface of the first feature. This is not a case of feature composition, but of serial feature instantiation.
- Both features have the same base surface and their areas overlap. In this case, the part of the base surface where the areas intersect is influenced by parameters from both features.
- Both features have the same base surface but their areas do not overlap. In this case a combination of the two features results in a *pattern feature.*

Note that there are several additional ways in which features can interfere. However, in these cases no meaningful compound feature can be created. In this section, we deal with the second and third case mentioned. When combining the two features $F' = \{E', P', \mu'\}$ and $F'' = \{E'', P'', \mu''\}$ in a new feature $F = \{E, P, \mu\}$, the new feature can be constructed by separately combining E, P and μ.

Combining E' and E'' into E is a simple union operation. The new set of control points E consist of $k^E = k^{E'} + k^{E''}$ subsets, so that $E_h = E'_h$ for $1 < h < k^{E'}$ and $E_h = E''_{h-E'}$ for $k^{E'} < h < k^{E'} + k^{E''}$. The combination of parameters depends on the user's wishes: parameters from P' can be combined with parameters from P'' into one parameter in P, so that $\max(|P'|, |P''|) \leq |P| \leq |P'| + |P''|$. The way parameters are combined is completely determined by the users. Parameter mappings are combined in correspondence with the parameter combination.

The computation of the final shape has to take into account overlapping regions. In these regions, there are deformation influences of two features and these can be combined in several ways. How the two influences can be combined is determined by the user, who chooses a certain combination function f. Using this function, the final shape of a compound feature is computed using the following procedure:

1. The areas 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. It holds that $A'(u', v') = A''(u'', v'')$.
4. Denoting the final shapes of F' and F'' as respectively S' and S'', the final shape of F is defined as:

$$S = \begin{cases} S' & \textbf{\textit{if}}\ A'(u',v') \in A' - A'' \\ f(S', S'') & \textbf{\textit{if}}\ A'(u',v') = A''(u'',v'') \in A' \cap A'' \\ S'' & \textbf{\textit{if}}\ A''(u'',v'') \in A'' - A' \end{cases}$$

By choosing an appropriate function f, the user can determine the way two feature shapes are combined. Figure 3 shows a composition of two bump features for two different combination functions.

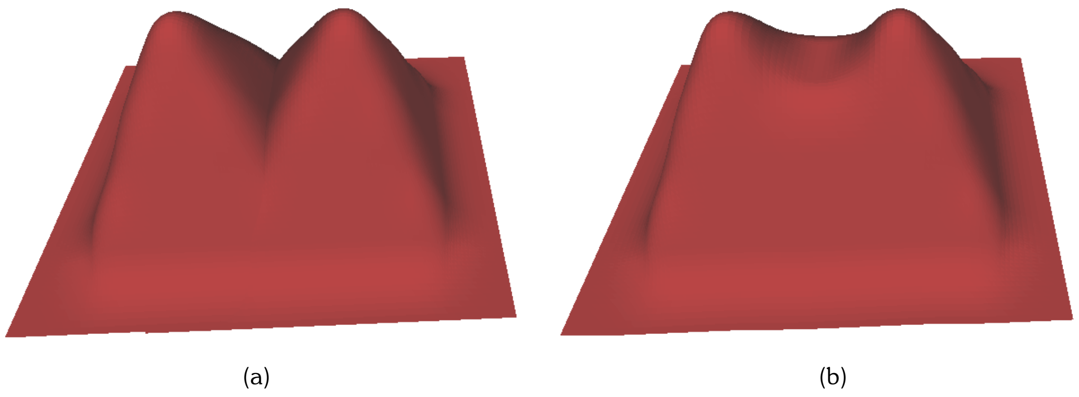


Fig. 3: Two bumps combined with (a) the maximum function (b) an interpolation function.

3. USER-DEFINED FEATURES IN FEATURE RECOGNITION

In an earlier publication [7], we proposed an evolutionary method for freeform feature recognition. In section 3.1, a short overview of the method will be given, followed by an application example that deals with a case where a user-defined feature is needed. Section 3.2 shows how the ability to create new feature definitions can be useful in dealing with feature interference.

3.1 A feature Recognition Method

The feature recognition method that was proposed in [7] makes use of the well known principle of evolutionary computation. The idea behind this is that freeform features cannot be exhaustively classified and therefore a method that recognizes these features should be adaptive, at least to a certain degree. The idea behind evolutionary computation enables users to recognize a certain target feature by defining a roughly similar feature and having the feature recognition 'grow' this feature into resembling the target feature.

Given a target shape that contains the feature to be recognized, the method works as follows:

1. A region of interest is selected by the user and a feature type is selected from the feature library.
2. A population of feature instances is generated. The feature instances are viewed as 'organisms', of which the shape is determined by its 'genes'. For features, the genes are those elements that are required of a user to fill

in during a feature definition process: the basic positions of the control points, the parameter values and the parameter mappings
3. For each individual feature, its 'fitness' is computed as the distance of its shape to the region of interest.
4. The fittest features in the feature population are used to 'breed' and produce offspring in the next generation of feature instances. Mechanisms from natural evolution such as cross-over and genetic mutation are used to guarantee that the next generation of features has a different genetic constitution than its parent generation.
5. The process is repeated until the fitness of one or more features in a generation are below a user-given threshold.

By allowing the parameter mappings to evolve, features are created that are no longer of a predefined feature class. This process depends on the amount of genetic mutation: a large amount of mutation increases the extent to which existing feature definitions are modified, but decreases the chance that a mutated feature is feasible and resembles the target feature. A balanced choice in the amount of mutation determines the success of the feature recognition method.

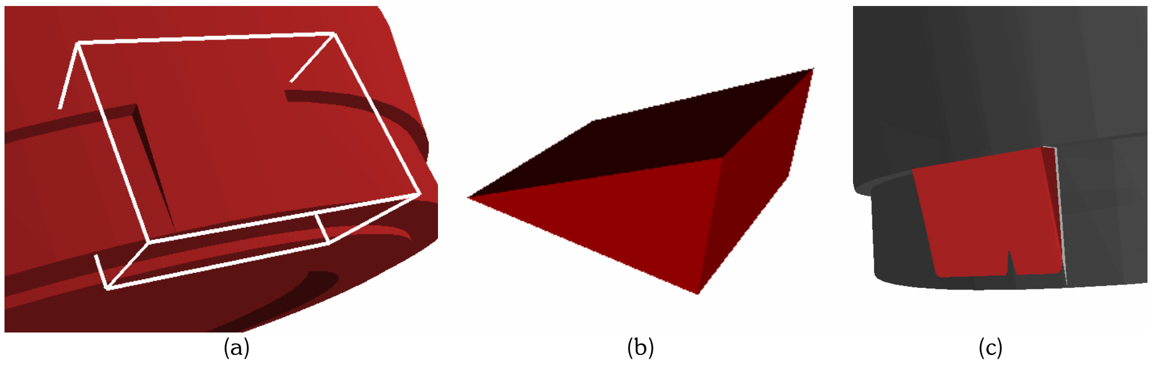


Fig. 4: (a) a region of interest on a target model (b) the user-defined Wedge feature (c) the wedge feature recognized on the target model.

Fig. 4 shows an application example of the feature recognition method. Fig. 4a shows a boxed selection of a region of interest on a CAD model (the depicted surface is a part of the bottom of a plastic coffee cup). To be able to recognize the feature that is present in the region of interest, a new feature is defined. The 'Wedge' feature is shown in Fig. 4b, of which the sharp edge is defined using multiple control points. Fig. 4c shows the result of a feature recognition procedure, in which the wedge feature shown in Fig. 4b is recognized on the model depicted in Fig. 4a.

The feature recognition method makes use of the properties of the feature definition that were mentioned in section 2.1. This guarantees that any valid user-defined feature can be used as an input to the feature recognition method. All the computations done on features, such as the 'breeding' of two features, are done in feature space, where they can be done efficiently. The mapping presented in section 2.3 guarantees that any change in feature space is translated to modeling space.

3.2 Dealing with Feature Interference

One of the most reported problems in feature recognition is the occurrence of feature interference, both in the regular domain and in the freeform domain. Regli and Pratt [12] give an overview of the different types of feature interaction, and most of the analysis they give can be translated to the freeform domain. Song et al. [15] report on the behavior of the freeform feature recognition method proposed in [14] in the case of feature interference. They show how their method is able to recognize individual features while they interfere with other features, but their method does not identify the interference as such and no general methodology for dealing with feature interference is proposed.
In this section we propose a method to deal with feature interference that circumvents the actual problem. Instead of dealing with the interference, we propose to view a case of two (or more) interfering features as a compound feature type that is composed out of the two interfering features. Imagine that a target surface has two interfering features, F' and F''. Assuming that these features can be individually recognized (as is shown in [15]):, the following procedure can be followed:

1. Select a region of interest around F' and recognize F'
2. Select a region of interest around F'' and recognize F''
3. In feature space, combine the two recognized features to a compound feature, in which the original parameters of the individual features are kept. Use the maximum function for the combination.
4. Unify the regions of interest from 1. and 2. and recognize the compound feature.
5. Optionally repeat for additional interfering features

Once the compound feature has been recognized, the user has control over the parameters of the individual features, as well as over the interfering region of the two features (through the combination function f). It is important to attempt to recognize the compound feature again in step 4, otherwise the inevitable inaccuracy of the recognition of F' and F'' adds up.

4. CONCLUSIONS

In this paper, a basic definition for freeform features was proposed. The definition itself is not particularly innovative, but we have shown how it can be used to create user-driven feature definitions, which can then be stored in a feature library. Furthermore, we have introduced the concept of feature space, a dual environment to link the definition environment for features to its application in feature-based design and feature recognition.
The proposed methods contribute to the investigations in the field of freeform feature research in that it shows how a link between user-driven feature definition and applications of freeform features can be established. In particular, this paper shows how such a link can contribute to a more efficient implementation of these applications.

It is the expectation of the authors that once the research community has agreed on a general definition of form features, developing methods for feature-based design, feature recognition and related problems will be much easier. Most of these problems are well-known and although they have not been defined for the freeform domain, a clear feature definition will certainly add to solving them.

5. ACKNOWLEDGEMENTS

The research project DIT.6240 is supported by the Technology Foundation STW, applied science division of NWO and the technology program of the Ministry of Economic Affairs, The Netherlands.

6. REFERENCES

[1] 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, 2003, 56–64.
[2] 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. (eds), Elsevier Science Publishers B.V., 1991
[3] Fontana, M.; Giannini, F.; Meirana, M.: A free-form feature taxonomy, Computer Graphics Forum, 18(3), 1999, 107-118.
[4] Han, J.; Pratt, M.; Regli, W. C.: Manufacturing Feature Recognition from Solid Models: A Status Report, IEEE Transactions on Robotics and Automation, 16(6), 2000.
[5] Hoffmann, C. M.; Joan-Arinyo, R.: On user-defined features, Computer-Aided Design, 30(5), 1998, 321-332.
[6] Langerak, T. R.; Vergeest, J. S. M.: A new framework for the definition and recognition of free form features, Journal of Engineering Design, to be published.
[7] Langerak, T. R.; Vergeest, J. S. M.; Wang, H; Song, Y.: An evolutionary strategy for free form feature identification in 3D CAD models, Proceedings of the WSCG conference,
[8] Nyirenda, P. J.; Bronsvoort, W. F.; Langerak, T. R.; Song, Y.; Vergeest, J. S. M.: A generic free form feature taxonomy for defining new classes, Computer-Aided Design and Applications, 2(1-4), 2005, 497-506.
[9] Nyirenda, P. J.; Mulbagal, M; Bronsvoort, W. F.: Definition of freeform surface feature classes, Computer-Aided Design and Applications, 3(5), 2005, 665-674.
[10] Ovtcharova, J.; Pahl, G.; Rix, J.: A Proposal for Feature Classification in Feature-Based Design, Computers & Graphics, 16(2), 1992, 187-195.
[11] Piegl, L.; Tiller, W.: The NURBS Book, Springer-Verlag, Berlin, 1997.

[12] Regli, W. C.; Pratt, M. J.: What are feature interactions?, Proceedings of the ASME Design Engineering Technical Conference and Computers in Engineering Conference, 1996
[13] Shah, J. J.: Feature transformations between applications-specific feature spaces, Computer-Aided Engineering Journal, 5(6), 1984, 247-255.
[14] Song, Y.; Vergeest, J. S. M.; Horvath, I.: Fitting and manipulating freeform shapes using templates, Journal of computing and information science in engineering, 5(2), 2005, 86-95.
[15] Song, Y; Vergeest, J. S. M; Horvath, I.: Feature interference in free form template matching. In: Navazo Alvaro, I., Slusallek, Ph (Ed.), Eurographics 2002, 9-18, Eurographics Association, 2002.
[16] Subrahmanyam, S.; Wozny, M.: An overview of automatic feature recognition techniques for computer-aided process planning, Computers in Industry, 26, 1995, 1-21.
[17] 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, 15(1), 1999
[18] Wilson, P. R.; Pratt, M. J.: A taxonomy of features for solid modeling, In M. J. Wozny and J.L. Encarnacao (eds.), Geometric Modeling for CAD Applications, Elsevier Science Publishers B.V., Holland, 1988, 125-136.
[19] Wu, M. C.; Liu, C. R.: Analysis on Machined Feature Recognition Techniques Based on B-Rep, Computer-Aided Design, 28(8), 1996, 603-616.