Geometric Feature Deletion Through Freeform Feature Recognition

Thomas R. Langerak
Delft University of Technology
Landbergstraat 15
2628 CE Delft
Netherlands
T.R.Langerak@tudelft.nl

ABSTRACT

The deletion of geometric features is a much used operation in the process of shape modeling; by deleting geometric features, the smoothness of a surface can be increased. In this paper, a new method is presented for the deletion of geometric features, which is based on a morphological understanding of the feature. The method first parameterizes the feature and then deletes it using a parameter-based shape manipulation. The advantage of this new method is that a geometric feature can be deleted while maintaining the semantics of both the feature and its embedding shape. To be able to parameterize the feature, an improved version of an existing evolutionary feature recognition procedure is developed, which constructs a feature template that matches a feature on the target model. Application examples are given and the robustness of the method is discussed.

Keywords

Freeform features. Feature recognition, Feature deletion, Evolutionary computation

1. INTRODUCTION

When thinking of a surface with distinct geometric patterns, it is often natural to think of this surface as a base surface with features attached to it. For example, if one would view a landscape with rock formations, then it is only natural to see the rock formations as being *placed* on the landscape (see Figure 1). When a piece of skin swells up after an insect bite, then the bump that originates is seen as 'additional' shape and the original skin surface is still perceived, regardless of the fact that, of course, no additional material has been created.

The geometric patterns of the rock formations and the bump in these examples are *features*. Features can be defined as connected regions of the surface that can be easily separated from the rest of the surface [Rib01]. Geometric features are often used as a modeling tool by which distinct characteristics can be added to an otherwise smooth surface. Features can even be further formalized by adding parameters to them that have a functional relation to the feature shape.

Figure 1: Rock formations in Monument Valley, Arizona

In other words, parameter values can be used to modify the shape in a pre-defined and predictable way. This way features can be used to significantly improve the efficiency of shape manipulations. In addition, the recognition of the features can be used or is sometimes even necessary in a process of reverse engineering. However, geometric features can also occur as noise or byproducts of other geometric modeling operations, and in this case can be classified as unpredicted, unnecessary and even unwanted regions of a surface. If this is the case, then a logical wish is to remove them from the surface and to (re)create a smooth surface. Smooth surfaces can be more efficiently represented and as a result the

computation time for many operations on the model is significantly decreased.

Current methods for dealing with geometric features are based on an geometric approach to the problem of feature deletion, where features are considered to be a geometric anomaly. However, in this case, an 'understanding' of the feature is lacking, and consequently the result of the feature removal procedure is imprecise and requires post-processing of the geometric data. In this paper, a new approach to feature deletion is proposed that first recognizes a target feature. The parameterization of the feature that can be derived from this recognition can then be used to delete the feature using a parameter-based shape manipulation.

2. EXISTING LITERATURE

To find approaches to remove features, one does not have to look far. Different approaches exist, and within each approach different techniques can be used. Ribelles et al. [Rib01] propose a method that combines several of these techniques for the removal of features from polyhedral data. In their work, they first split a polyhedral model into several subsections using an intersection plane. They then classify and cluster triangles in the polyhedral model corresponding to their position relative to the intersection plane. In the clustered data, they identify features and separate these from the rest of the polyhedral model. Finally, they use a hole filling algorithm to repair the holes that are left behind by the removal of the feature. Although the method neatly combines several techniques, it is only demonstrated for use with regular polyhedral models, i.e. models with only surfaces that are either perpendicular or parallel to the other surfaces in the model. This implies that the technique cannot be used in freeform models.

Another approach that targets regular features is that of Venkataraman et al. [Ven02a]. The authors detect features by finding a closed boundary loop, i.e. a closed loop of edges along the boundary of the feature. They then use extension and shrinking of the faces that border on this boundary loop to remove the features. In addition to the fact that their method only targets regular features, it is also unclear to what degree the approach can be automated. In [Ven02b] the authors present a similar method that specifically targets blend features.

Lee et al. [Lee05] propose a feature suppression method that uses a cellular modeling approach to identify features and then collapses the geometric elements of a feature such that it can be removed safely from the shape model. They store every collapse operation, in order to later be able to unsuppress the feature. Again, this method mainly targets regular features and suffers from all the disadvantages of history-based modeling.

3. PROBLEM DEFINITION

In this paper we assume that freeform surfaces can be described by a discrete set of elements, which we call *shape configuration elements*. A B-spline surface can be described by a set of control points; polyhedral meshes can be described as a collection of points, vertices and faces. This means that a surface that can be described with n elements, be it a polyhedral mesh or a B-spline surface, can be formalized as $S = \{s_1, \ldots, s_n\}$. Note that the shape configuration elements are not part of the shape themselves. For each representation type, it is assumed that an *evaluation function* exists, which translates the shape configuration element to actual geometry, e.g. for a B-spline representation the shape evaluation function is the well-known de Casteljau's algorithm.

In our approach, features are embedded in a freeform target surface. If the discrete set of elements that represents the feature shape, denoted E, is a subset of that of a surface S, then S is called the *base surface* of E. The deletion of the feature can now be defined as the transformation $S^{original} \rightarrow S^{deleted}$. Note that for each element s_i it holds that if $s_i^{original} \neq s_i^{deleted}$, then it holds that $s_i \in E$. Hence, to delete a feature from a surface, an operation on the feature is needed that transforms the feature shape (and with it the shape of its base surface) from its original state to the 'deleted' state. A large advantage of this approach compared to that of Lee et al. [Lee05] is that the feature information is not removed from the model. The only difference between the 'deleted' state of the feature and the original state of the feature is that its parameter values have been set to a state that corresponds to $S^{deleted}$. Therefore, to reinstate the feature, all is needed is another parameter change.

Features in this paper are taken to be parametrically controlled shapes. This means that, besides the shape information, a feature also contains parameter information and, more importantly, information on how the parameters influence the shape. We define a feature as follows:

*A **feature** $F\left(E^0, P, \mu\right) = E$ is a function that, given a basic shape configuration E^0, a set of parameter values P and a parameter mapping μ, results in a shape configuration E, where:*

 ISBN 978-86943-15-2

- $E^0 = \{e_1^0, \ldots, e_n^0\}$ *is the set of shape configuration elements of the basic shape configuration*
- $P = (p^1, \ldots p^m)$ *is a vector of parameter values*
- $\mu(E,P) = \bigcup_{i=1}^{n} \bigcup_{l=1}^{m} \{\mu_i^l (e_i \in E, p^l \in P)\}$ *is a parameter mapping*
- $E = \{e_1, \ldots, e_n\}$ *is the resulting shape configuration*

The basic shape configuration E^0 can be any valid shape configuration of the feature. Without loss of generality we therefore assume that the basic shape configuration occurs when the vector of parameter values $P^0 = (0, \ldots, 0)$. This definition implies that a feature can have multiple states, each of which can be derived from an original state (the basic shape configuration of the feature), a vector of parameter values and the parameter mapping.

Information on how the parameters influence the shape is contained in the parameter mapping, which is a set of nm functions μ_i^l that describe the influence of parameter p^l on shape configuration element e_i. Although we have implemented more sophisticated parameter mapping functions, in this paper it suffices to assume that a parameter mapping function μ_i^l has the form of a translation vector T_i^l, such that $\mu_i^l(e_i, p^l) = p^l T_i^l e_i$.

Combining this with what has been said earlier in this section, we can pose the following problem statement:

Feature deletion is the problem of, given an original surface $S^{original}$ *that contains one or more features* $E \in S^{original}$, *to find an alternative state* $S^{deleted}$ *of the surface, for which it holds that* $E^0 \in S^{deleted}$.

Note that this problem only requires us to find the feature $F(E^0, P, \mu) = E$, for once E, P and μ are known, E^0 can be backwards computed as $E^0 = F(E, -P, \mu)$. Hence, the problem of feature deletion can be partly solved by using feature recognition.

4. FEATURE RECOGNITION

Feature recognition has been a popular research topic in the last decades of the previous century. However, although many approaches to the recognition of regular features have been proposed, very little work has been done on the recognition of freeform features.

Song et al. [Son05] and Vergeest et al. [Ver01] have proposed a *template matching* approach, in which an instance of a pre-defined feature type, a feature *template*, is iteratively compared with a target surface. They construct a function that takes the parameter values of the template feature as input variables and results in a measurement of the geometric distance between the shape of the template feature and that of the target surface. By then applying a function minimization routine, parameter values can be computed for which the geometric distance between template and target surface is minimal (see Figure 2). The template that is used in the procedure is an instance of a user-selected feature type from a collection of pre-defined features, the *feature library.*

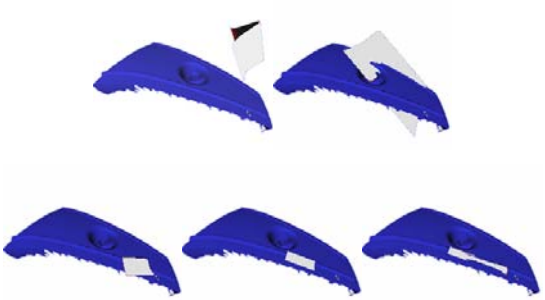


Figure 2: Consecutive steps in a template matching procedure, in which a template is iteratively matched to a shape model

Although the template matching method is successful in finding relatively simple features, the usefulness is limited by the size of the feature library. Before the template matching method can be used to recognize a feature on a target surface, a feature type that resembles this feature must first be defined and added to the library. Part of the recognition process therefore lies with whoever designs the feature types in the feature library.

An improvement on the template matching method has been proposed by the author [Lan07]. In this method, an evolutionary approach was used to find a minimal configuration for the template feature. In this paper, a more elaborate version of this method will be given and it will be discussed how this method can be used for the problem of feature deletion.

Evolutionary computation is a technique that has often been used in the past and that has its roots in the field of biology. Evolutionary methods mimic the

biological concept of natural selection, which is based on the principle of 'survival of the fittest'. In nature, some organisms have higher chances of survival and procreation because they are better adapted to their environment. The fact that they are better adapted to their environment is due to the fact that their genetic structure, their 'DNA', contains specific genes. Because their chance of procreation is bigger than that for other individuals, they are likely to pass on these genes to a next generation. As a result, organisms in this offspring generation are even better adapted to their environment. 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 solution to the problem. In an evolutionary computation method, natural mechanisms that play a role in evolution must be implemented. The most important of these mechanisms are: genetic structure, crossover, mutation, fitness and selection.

Genetic structure

In the feature recognition problem, the 'problem' under consideration is that of shape matching. The organisms in an evolutionary freeform feature recognition problem are, logically, feature instances, and the genetic structure therefore contains all the variables that are needed to compute the shape of a feature: the parameter values of the feature, but also its parameter mapping functions. We define the gene sequence $\gamma = \left\{\gamma_1, \ldots, \gamma_{(n+1)m}\right\}$, such that $\gamma_i = P_i$ for $i \le m$ and $\gamma_i = T^{\lceil (i-m)/n \rceil}_{\mathrm{mod}(i-m,n)}$ for $i > m$.

Crossover and mutation

Crossover and mutation determine how the inheritance of genes from one generation to the other takes place. The gene sequences of two individuals, the parents, are combined such that the gene sequence of each of their offspring is a (different) combination of parts of the gene sequences of their parents. Sometimes, during the process of inheritance, the information in the gene sequence is distorted. This process is called mutation and ensures that the 'gene pool', the collection of all genes that are available in a population, is constantly in motion. Mutation introduces new properties into the gene pool, both good and bad. However, only the good properties survive the natural selection process. Mutation plays a particularly important role in evolutionary computation methods. Because the available computation time and memory is limited, populations are often simulated with less-than-natural sizes. To counter the effect of inbreeding, mutation rates are typically higher than in nature.

Fitness and selection

The extent to which an organism is adapted to its environment is expressed in the *fitness*. In the case of feature recognition, the fitness is implemented as the geometric distance between feature shape and target shape. As a measurement of this distance, the Hausdorff distance is used, which is defined as $H(S_1, S_2) = \max_{s_1 \in S_1}\left(\min_{s_2 \in S_2}\left(dist\left(s_1, s_2\right)\right)\right)$, where *dist* is the Euclidean distance between two points. The higher the fitness of an organism, the higher the probability that it will generate offspring.

Several variables play a large role in any evolutionary computation:

- The *population size* Π indicates the number of organisms in a population. 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 *selection size* σ indicates the extent to which the fitness has an influence on the chance of selection. If σ is set to a high 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.
- The mutation probability χ indicates how likely it is that mutation occurs during the creation of an offspring population. If χ is high, then a higher number of new genes is introduced in each generation. This can either be an advantage or a disadvantage: the higher the number of new genes, the less likely the feature recognition is to get 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.
- 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.

Considering what has been said in this section, an evolutionary feature recognition algorithm can be given as follows:

1) An initial feature population $gen_1 = \{F^1, \ldots, F^\Pi\}$ is generated, where each F is an instance of a pre-defined type from the feature library. A random value is assigned to the parameter values.
2) The fitness of each feature is computed and features are ranked according to their increasing fitness value $f(\ )$, such that $f(F^1) \le \ldots \le f(F^\Pi)$
3) 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.
   - 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. The chance of mutation is computed using a uniform distribution, and if mutation occurs, then the value of a gene is multiplied by a factor that is computed using a Gaussian distribution with a mean of 1 and a standard deviation χ.

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 an acceptable solution to the feature recognition problem, i.e. $f(F^1)$ in gen_i is smaller than a user-given threshold ε_2.

Using the evolutionary algorithm, features can be recognized providing the variables of the algorithm are set to a correct value. However, if both parameter mapping functions and parameter values are included in the genetic structure of a feature, then the amount of genes is large. As a result, the population size must be set to a very large value, which in turn leads to a large computation time. To counter this problem, we divide an evolutionary feature recognition procedure into two steps:

Step 1

During the inheritance of genes, only the genes that represent the parameter values are subjected to mutation. As a result, the genes that represent the parameter mapping functions remain stationary and the population size can be kept small. The result of this step is a configuration of a feature template that roughly matches the target shape, similar to that of the template matching method. Although the match is not perfect, the end result of this step is a good starting point for step 2.

Step 2

Starting from the configuration that was found in step 1, a new evolutionary procedure is started. Because this procedure starts from a reasonable approximation of the optimal configuration of a template feature, the population size can again be kept small. This time, the genes that represent parameter values are fixed and the genes for the parameter mapping are included in the inheritance. During this step, a much more accurate match is found. Because the parameter mapping functions are now subjected to mutation, the resulting feature is no longer an instance of a feature that is available in the library. The shape of the feature has been adapted to that of the target surface, while maintaining the functional relation between parameters and shape.

Examples of the result of the two steps are shown in Figure 3. The main advantage of dividing the feature recognition process into two steps is that the population size can be kept small and hence the computation time is reduced. In step 1, only a small amount of genes is used; in step 2, the initial configuration of a feature is already a reasonable match of the target surface.

(a) (b) (c)

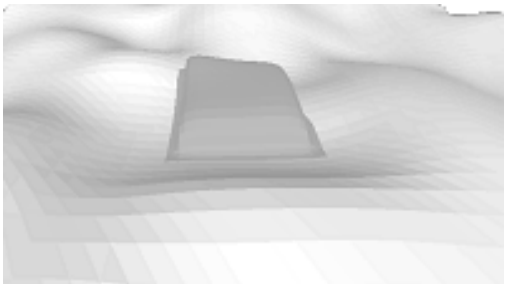
Figure 3: Results of the two steps of the evolutionary feature recognition algorithm, (a) the target shape model, (b) a rough match of a feature template to the shape and (c) an improved match, in which the feature template is adapted to the target shape model.

Evolutionary feature recognition is an improvement on existing feature recognition techniques in that it does not require a rigid similarity between a feature on the target surface and a pre-defined feature in the feature library. Although at the starting point of the recognition procedure, instances from pre-defined feature types are used, during the procedure these become adapted to the target surface and the feature definition is slowly changed. However, although the recognition procedure results in an accurate match between a feature template and a target surface, the result is still a template, not a portion of the target surface. In other words, the shape configuration of the feature template E is not a subsection of S. This means that we are still not able to use the feature template that results from the recognition process to delete the feature on the target surface. In the next section we therefore describe a technique to find the base surface of the feature, using the recognized feature as a starting point.

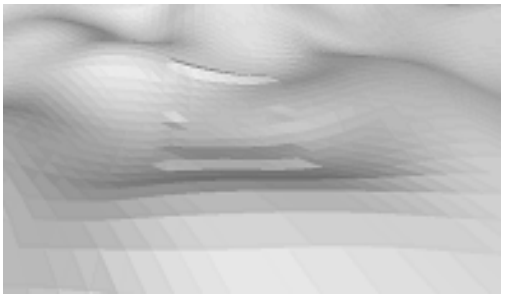
5. BASE SURFACE DETECTION AND FEATURE DELETION

To be able to use the feature template that was found in the feature recognition procedure, the observation must be made that although the shape configurations of the feature template $F'\left(E'^0, P', \mu'\right)$ and the target feature $F''\left(E''^0, P'', \mu''\right) = E^{target} \subseteq S$ are not identical, this does hold to a high degree for the parameter mapping functions, such that $F'\left(E^{target}, -P', \mu'\right) \approx E''^0$. Remember that E''^0 is the surface that remains after the feature F'' has been deleted. This observation allows us to find an estimate for the base surface of the target feature.

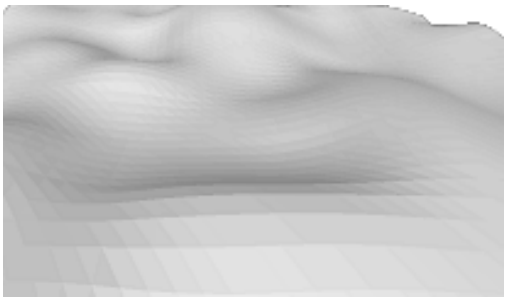
Song et al. [Son05] show how a matched template can be used to deform the feature that it has been matched to. They use a variant of lattice-based deformation that was proposed by Sederberg and Parry [Sed86].



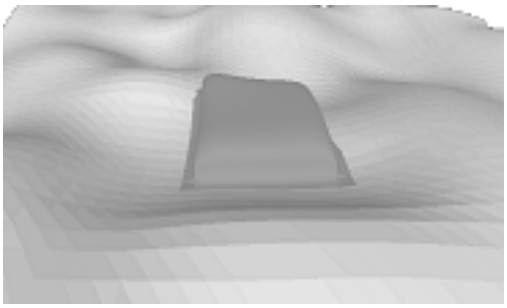
(a)



(b)



(c)



(d)

Figure 4: Steps in the base surface detection: (a) the original surface, (b) after deletion of the feature, with some residual effects, (c) after smoothing the surface (d) after reconstruction of the feature.

By using this approach, the recognized feature can be deleted by setting the parameter values of the template to 0, but because the match between the template and the target surface is not perfect, there is some residual effect (see figure 4b).We have implemented an approach to improve on this result, so that a feature can be removed without residual effect:

1. An evolutionary feature recognition is used to find the configuration of a feature template such that it has a minimal distance to the target surface. This step equals the two steps mentioned earlier.
2. The parameter values for the template feature are set to 0 and the correspondence between template and target surface is used

to deform the target surface using a lattice-based approach.

3. The resulting surface is smoothed
4. A new feature recognition procedure is started, in which the basic shape configuration of the feature is a subset of the surface that results from step 3.

For the smoothing in step 3, any known smoothing algorithm can be used. There is not a 'correct' solution to the base surface detection problem, and therefore the exact result of the smoothing procedure is irrelevant, as long as it leads a smooth surface. We used a simple Laplacian smoothing algorithm, which in practical cases provided a sufficiently smooth base surface.

Note that the feature recognition in step 4 can be done much faster and with much more efficiency than the recognition procedure in step 1, because:

- The configuration of the feature is constrained by its position on the target surface.
- The exact feature type of the target feature is known, as it was constructed during step 1.
- Approximate parameter values for the feature are known.

As a result, the population size for the second feature recognition procedure can be kept very small, and the computation time is small as well.

In the end result of step 3 , the targeted geometric feature has been deleted, leaving a smooth surface. However, earlier it was stated that the preservation of the semantics of the deleted feature is important, for example if one wants to undo the deletion. For this reason, the surface that results from step 3 is assumed to be the base surface of the original feature; in step 4 the feature is reconstructed using this base surface. Although the reconstructed feature that is the result of step 4 is not an exact match of the original target feature, we found that the distance between the two was very small and could not be visually discerned (see figure 4a vs. figure 4d).

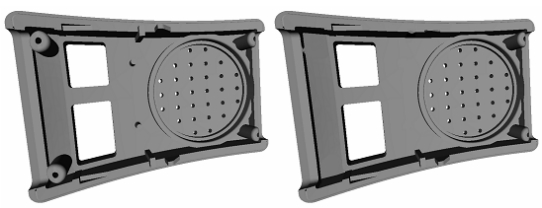


Figure 5: Test models (left) and the result of the feature deletion procedure (right).

To test the proposed method, it was subjected to several test models, of which three are shown in figure 5. The models were in the form of polyhedral meshes containing in between 45000 and 76000 polygons. To be able to recognize and delete the features on these models, first feature type definitions were created; to guard the objectivity of this definition, i.e., to guarantee that the definitions were not an a priori match to the features on the test models, the definitions were created by a colleague on the basis of a verbal description of the feature in question. Then, to guarantee that the correct feature was deleted, first regions of interest were selected around the targeted features on the test models. Finally, these regions of interest were subjected to the proposed method, the result of which can be seen in figure 5.

From the results of the tests, it was concluded that the proposed method can successfully be used to delete geometric features. However, some situations were found in which a feature could not be deleted, due to the fact that (a) the feature interfered with other features or parts of the shape, (b) the feature was attached to multiple base surfaces, and (c) the feature was nested, i.e. its base surface could be considered to be another feature (see Figure 6).

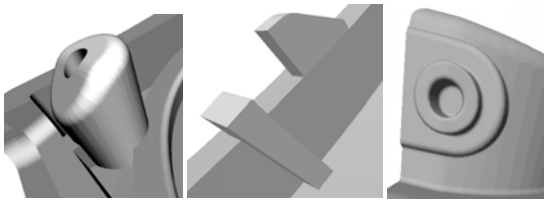


Figure 6: Three cases in which the proposed method failed: (a) interference with other shape parts, (b) multiple base surfaces and (c) nested features

6. RESULTS AND DISCUSSION

A method was proposed that deletes a feature from a target surface. The method first recognizes the feature by matching a feature template to the target shape, and then uses the correspondence with the template feature to reduce the feature to its basic shape configuration. The resulting surface is smoothed and in a final step the original feature is reconstructed. The method was implemented for use on polyhedral meshes, but can be generalized to B-spline surfaces because in the theory no specific assumptions were made on the shape representation type. To implement the method for B-spline surfaces, only the smoothing of a surface must be addressed.

It was found that the feature deletion method is successful in deleting features from polyhedral meshes. In the testing experiments, several problems occurred, some of which were shown in figure 6. First, it was found that the feature deletion method does not fully work in the case where two or more features interfere. Dealing with feature interference is a known problem of feature recognition and it can be expected that if improved feature recognition methods are developed, then the deletion of these features also improves. However, the difficulty of dealing with features that have multiple base surfaces is a minor but systematic shortcoming of the proposed method, which should be addressed in future research. Nested features cannot be unambiguously deleted without additional user input on what feature exactly has to be deleted.

Another problem with the feature deletion method presented in this paper is that it is unable to deal with eliminative features, i.e. holes. This is a consequence of a shortcoming of the proposed feature recognition method, not with the methodology that was proposed for feature deletion. A different approach to feature recognition is necessary for this type of features, which is also left to future research.

7. ACKNOWLEDGMENTS

The research project 06240 is supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs, The Netherlands.

8. REFERENCES

[Lan07] Langerak, T.R., Vergeest, J.S.M, An Evolutionary Strategy for Free Form Feature Identification in 3D CAD Models, Proceedings of the WSCG conference, Plzen, 2007

[Lee05] Lee, K.Y., Armstrong, C.G., Price, M.A., Lamont, J.H., A small feature suppression/ unsuppression system for preparing B-rep models for analysis, Proceedings of the 2005 ACM symposium on Solid and physical modeling, June 13-15, Cambridge, USA, 2005

[Rib01] Ribelles, J, Heckbert, P.S., Garland, M., Stahovich, T., Srivastava, S., Finding and removing features from polyhedra, Proceedings of the ASME DETC conference, September 9-12, Pittsburgh, USA, 2001.

[Sed86] Sederberg, T. W., Parry, S. R., Freeform Deformations of Solid Geometric Models, Computer Graphics, Vol. 20, No. 4, pp. 151–160, 1986

[Son05] 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.

[Ven02a] Venkataraman, S., Sohoni, M., Reconstruction of feature volumes and feature suppression, Proceedings of the 7th ACM symposium on Solid modeling and applications, June 17-21, Saarbrucken, Germany, 2002

[Ven02b] Venkataraman, S., Sohoni, M., Rajadhyaksha, R., Removal of blends from boundary representation models, Proceedings of the 7th ACM symposium on Solid modeling and applications, June 17-21, Saarbrucken, Germany, 2002

[Ver01] 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