An Evolutionary Strategy for Free Form Feature Identification in 3D CAD Models

T.R.Langerak, J.S.M.Vergeest, H.Wang, Y.Song

Delft University of Technology

Landbergstraat 15

2628 CE Delft

The Netherlands

t.r.langerak@tudelft.nl

ABSTRACT

Nowadays, most 3D CAD systems support the use of form features. The main advantage of form features is that they provide parametric, high-level support for shape manipulation. When the parametric information of a shape is not available, it can be retrieved using a feature recognition procedure. In this procedure, a target shape is recognized as an instance of one or more features in a pre-defined feature library. The speed and accuracy of the feature recognition procedure significantly improve when the target feature type is known. This information can be provided by a user, but we propose a new method that identifies the feature type automatically. This method uses an evolutionary algorithm to find the optimal feature type. The algorithm randomly generates a population of instances of every type of feature available in the feature library. From this initial population of feature instances, successive populations are generated using the principle of natural selection. The algorithm was tested for different settings and was found to correctly identify features in between one and three minutes.

Keywords

Feature recognition, evolutionary algorithms, feature library

1. INTRODUCTION

One of the trends in geometric modeling in the past decades has been to bring shape manipulation routines to a higher level, meaning that a modeler is able to perform a certain shape operation with less effort. A means to do this is by using *form features*. A form feature is a parameterized shape (part) that can be manipulated in pre-defined ways by adapting parameter values. Many, if not all, currently used geometric modeling applications use form features.

Information on form features is stored and maintained as a model is built up, to ensure that form features are available at every step of the modeling process and always lead to a valid result. However, there are situations in which the form feature information is not available, such as when a form feature is an unintended result of other modeling operations or when there is a domain change, e.g. when using different modeling applications. Form feature information is also not available when raw geometrical data is imported, such as when data is obtained from physical objects by a 3D measuring device. In these cases, form feature information can be retrieved through the process of *form feature recognition*. As a special version of shape recognition, form feature recognition does not only retrieve shape information, but also a semantic understanding of the (possible) parametric constitution of the shape. In the practical sense, feature recognition means matching a target shape to the information that is available in a *feature library*, the collection of available features.

In the past decades, many algorithms have been developed for feature recognition in the domain of machining features. These algorithms are mostly applied to communication between Computer-Aided Design applications and Computer-Aided Manufacturing applications, but are not useful for free form modeling applications.

Efforts are being made to extend feature recognition to the free form domain, but many of the techniques used for machining features can not, or only in highly adapted forms, be used for free form features. Some methods for free form feature recognition have been developed ([Tho99], [Song05], [Lan05], [Pal05]), but the results are few and preliminary.

More so than for machining features, the recognition of free form features is aided by user input. Because free form features are more complex than machining features, computation times are unavoidably larger. Extra information that is given by the user can help in reducing the computation time. Also, the set of theoretically correct solutions is larger than for machining features but the user is typically interested in only one or a few results. User input is needed to select the desired output from the set of feasible outputs.

One of the pieces of information that lead to more efficient algorithms is the type of the feature to be recognized. When the feature type of the target shape is known, the solution space of the problem is known and the search strategy can be adapted. This information can easily be asked of the user when the feature library is small and distinct, but when the number of features to choose from is large and when the distinction between features is small, then providing the right feature type requires an expert user. In this paper we propose a method for retrieving the feature type of a target shape automatically. The method makes use of the well-known principle of evolutionary computation.

Evolutionary computation is a popular technique that is based on the principle of natural selection. It applies 'survival of the fittest' to populations of possible solutions to a problem, in order to evolve towards an optimal solution. Survival of the fittest is a biological mechanism in which organisms with certain genetic elements have a higher chance to procreate and are therefore more likely to pass on these genetic elements to a next generation. Evolutionary algorithms have been applied to many computational issues in the past. Good results have been reported, but part of the popularity of evolutionary algorithms can only be explained by their elegance. The only application of evolutionary algorithms to feature recognition known to the authors is that of Pal et al. [Pal05]. However, this method makes far-going assumptions on the input data and does not result in a parametric description of the recognized feature.

2. PREVIOUS WORK

In the past two decades, research after form feature recognition has been fueled by its application in Computer-Aided Process Planning (CAPP), linking design to manufacturing. Many different techniques for form feature recognition have been developed, of which good overviews can be found in ([Shah01], [Sub95]). Among the techniques that have been developed are graph-based feature recognition ([Jos88], [Chu90], [Flo89]), hint-based feature recognition ([Van93], [Van94]), volume decomposition ([Woo82], [Kim92], [Sak95], [Sak96]) and neural networks ([Pra92], [Nez97]). Graph-based feature recognition expresses target shapes in the form of connectivity graphs and then uses graph-based heuristics to search these graphs for features. Hint-based feature recognition uses a two-step approach to feature recognition; in the first step, hints of possible features are retrieved and in the second step these hints are processed and combined. Volume decomposition expresses target shapes in the form of a tree in which each node is defined as the difference between its parent and its parent's convex hull. Features can be found in the leafs of the tree.

More recently, techniques have been proposed for free form feature recognition. Thompson et al [Tho99] proposed a method to recognize features from point clouds that are the result of scanning real objects. This was extended by Song et al. [Song05], who implemented a method that recognizes features by fitting a template feature to the point cloud data. Once an optimal configuration is found, the parameter types and values of the recognized feature match that of the template feature. They also show how the template feature can be used to manipulate the recognized feature. Recently, Langerak et al. [Lan05] proposed a method to recognize styling features in polyhedral data. This method uses surface curvature analysis to locate features and offers a sketch-based manipulation tool for the identified feature.

There is literature on evolutionary computation in abundance. Goldberg [Gol89] and Davis [Dav91] are good early references on the subject. A more recent overview of the actual problems in evolutionary computation is given by Ghosh and Tsutsui [Gho02].

Evolutionary algorithms have been recently applied to feature recognition by Pal et al. [Pal05]. Their method extracts features from collections of feature surfaces by generating a population of individuals with a random number of surfaces from the given collection. Consecutive generations inherit feature surfaces, which thus accumulate to form an entire, recognized feature. They report good results, but their method is only applicable to polyhedral inputs and assumes it to be know what surfaces belong to a feature. In addition, their method is not able to handle feature interference.

3. METHOD OUTLINE

Feature identification can be seen as a minimization problem of a shape similarity function in the parametric domain of a feature. The minimization problem is a well-known problem. Several methods for solving it have been proposed in the past, but as the solution space is different for each problem there is no generic method that works for all problems. A short overview and implementations can be found in Press et al. [Pre02]. A popular method for the minimization problem is the conjugate gradient method, which bases its search strategy on the derivative of the fitness function. This method is not applicable to the feature identification problem, because a derivative for the shape similarity function can not be computed. Other minimization methods, such as the direction-set method or the simplex method are slow and inefficient. We therefore choose to apply an evolutionary strategy to feature identification, which does not rely on derivative information and can be, depending on its implementation, relatively fast.

Evolutionary algorithms solve a problem by regarding some element of the problem as an individual, onto which the mechanisms of evolution are applied. Each individual has a genetic mockup that reflects the variables that play a role in the problem that is to be solved. When an individual procreates, its genetic constitution is inherited by an individual in the next generation. The chance that an individual procreates is derived from its fitness, which is a value that indicates how well the individual represents a solution to the problem. How well an evolutionary algorithm performs depends on the shape of the solution space of the problem, of course, but also on the quality of the genetic mechanism.

In the case of feature identification, populations are formed by individual features instances. The shape similarity between a feature instance and the target shape serves as the fitness function in the evolutionary computation. Because the appearance of feature instance is determined by its parametric configuration, it is only natural to consider parameters to be a feature's genes. In this section we will first describe the constitution of a feature instance in more detail, and then define the mechanisms that render successive feature populations.

3.1 A Free Form Feature Definition

Because the evolution of a feature is managed through its parameters, in defining a feature there is a strong emphasis on the parametric constitution of a feature. However, because all parameters need to be treated similarly in an evolutionary computation procedure, we use a two-layered system in which one layer defines the amount of parametric influence, whereas the second layer, which we call the *parameter mapping*, defines the nature of the influence. As an added benefit, this allows for a generic feature definition system, in which the parameter mapping can be made domain-independent.

As a means to represent features, we use nurbs patches. Parametric influence is defined through the nurbs control points, so as to mirror a natural genetic structure, where each gene has a limited global influence. The following definition is used:

A free form feature is a parametric description of a shape, $\{E, P, \mu\}$, containing at least

- *a set of nurbs control points* $E = \{e_1 = (x_1, y_1, z_1), \dots, e_n = (x_n, y_n, z_n)\}$ each with a basic state $e_i^0 = (x_i^0, y_i^0, z_i^0)$, $1 \le i \le n$
- *a parameter set* $P = \{p_1, \dots, p_m\}$, $1 \le j \le m$ and $p_j = \{name_j, value_j\}$
- *a parameter mapping, defined as the set* $\mu = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \{\mu_j^i (e_i \in E, p_j \in P)\}$, *where* μ_j^i *is a function with the type* $(e, p) \to e$ *that defines the influence of parameter* p_j *on element* e_i. *The configuration of each element can be computed by combining all the mappings defined on the element, so that:*
  $$e_i = \mu_1^i(\mu_2^i(\dots(\mu_{m-1}^i(\mu_m^i(e_i^0, p_m), p_{m-1})\dots), p_2), p_1)$$

In our implementation, we express the nurbs control points in homogenous coordinates and define the parameter influence in the form of 4 by 4 transformation matrices. Each function μ_j^i consist of two parts: a transformation matrix and a function mask matrix that defines how parameter values influence the elements of the transformation matrix. For example, when a control point e has a basic state $e^0 = (0,0,0)$, then a parameter that defines a vertical movement (in the z-direction) is defined by a transformation matrix $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ and a function mask matrix:

$$\begin{pmatrix} const & const & const & const \\ const & const & const & const \\ const & const & const & var \\ const & const & const & const \end{pmatrix},$$

where $const(x,p) = x$ and $var(x,p) \; = \; x \cdot p$.

This definition allows for orientation and location of a feature to be regarded as a parameter of the feature as well, by using the cosine and sine functions in the function mask matrices. Although traditionally orientation and location are not considered to be part of the parametric space of a feature, they do play an important role in the identification of a feature. Location and rotation settings do alter the shape representation of a feature, and there is therefore no theoretical objection to regarding them as parameters. This enables orientation and location to be treated identical to other parameters. They do, however, differ from other parameters in that they have identical parameter mappings for all nurbs control points.

Our implementation allows for both rigid body control of the feature as well as deformation, but for the sake of simplicity in this paper it is assumed that only rigid body transformation is used.

3.2 Evolutionary Computation

The goal of the feature identification procedure is to find the type of a feature that best matches a certain target shape. It may be that only part of the target shape represents a feature. The more 'material' there is that does not belong to the feature, the higher the computation time will be and the less accurate the result. In our implementation we therefore provide a mechanism for a user to select a region of interest. This mechanism asks a user to place a bounding box roughly at the location of the feature. Only the shape data that is contained by this bounding box is used in the feature identification procedure. As will be shown in section 4, this mechanism is not needed to guarantee the correctness of the feature identification, it is merely a means to bring back the computation time.

At the start of the evolutionary computation, a population of features is generated. Each feature is a copy of a randomly chosen feature from the feature library, so the initial feature population consists of a mix of all the features that are present in the feature library. The features are initiated with random parameter values in the domain [-1000, 1000]. This domain was used with the assumption that, in practical cases, no parameter values outside the domain are used. If necessary, the scale can be adapted by using larger values in the transformation matrix. The size of the population is variable and influences the success of the identification procedure. In section 4, it is shown how different population sizes affect the success of the procedure.

From this initial population, successive populations are generated using the following steps:

1. The fitness $f()$ of each individual is computed as the mean directed Hausdorff distance between the feature and the target shape. The directed Hausdorff distance is defined as the smallest maximal distance between two shapes. Because the Hausdorff distance is most efficiently computed on point sets, we use point set samples of both target shape and feature shape to compute the fitness.

2. The individuals in the population are ranked by fitness, so that $f(F^0) \geq \ldots \geq f(F^n)$, where the F 's are the features in the population, f is the fitness function and n the population size. If the terminating conditions are met, the algorithm terminates. For brevity, we will denote $f(F^o)$ as f^0 .

3. For each of the *n* individuals in the next generation, two parents (a 'mother' and a 'father') are selected. An individual feature can be parent to more than one individual in the next generation. The chance that an individual is selected to be a parent is determined by a one-sided Gaussian distribution of the fitness: $$P(f^{parent}) = \frac{2}{\sigma\sqrt{2\pi}} e^{\frac{-(f^{parent})}{2\sigma^2}}$$

   The standard deviation σ determines how fast the chance to procreate decreases with the fitness and is also known as the *selection size*. Individuals with a low fitness are less likely to become parents to a next generation.

4. For each individual in the new generation, parameters are randomly copied from one of the two parents. To prevent the degrading of the gene pool, the collection of available genes, in a population is subject to mutation. The chance that mutation occurs is indicated as the *mutation probability*. When mutation does occur, the value of the copied gene is distorted by Gaussian noise with a standard deviation that is also known as the *mutation* rate. The two parents may be of different types and thus have a different number of parameters. In this case, without loss of generality, assume that $\left|P^{mother}\right| \geq \left|P^{father}\right|$.

Then $\left|P^{child}\right|$ receives a random value between $\left|P^{mother}\right|$ and $\left|P^{father}\right|$ and $p_i = p_i^{mother}$ if $i > \left|P^{father}\right|$.

5. When a new population of features is generated, the algorithm proceeds at step one.

While going through an iterative process, the algorithm keeps track of the ancestry of a feature. As in natural evolution, each individual feature can be placed in a family tree and by backtracking it can be determined from what ancestors an individual stems. Instead of backtracking for each individual, this information is passed on and added to when new individuals are created.

The algorithm terminates when one of the following conditions are met:

1. More than 75% of the ancestors of the 10% fittest individuals in a population have the same feature type. This type is returned as the result of the feature identification.
2. The fitness of the fittest individual drops below a certain threshold. In this case, it is very likely that a large majority of the ancestors of this feature have the same type, so the type that occurs the most in the feature's ancestral history is returned as the identified feature type.
3. The fitness of the fittest individual in the population is lower than or equal to that of the previous generation. In this case, the algorithm is likely to be stuck in a local minimum. As a best guess, the feature type that occurs in 50% of the ancestors of the 10% fittest individuals is returned as the identified feature type. If no such feature exists, the algorithm fails.

4. RESULTS AND ANALYSIS

The algorithm was tested on a feature library that consists of 8 features (see figure 1). This library contains features that have been the subject of past work on features (Bump, Ridge) and other features known from the modeling world (Step, Blend) as well as newly constructed features that we expect to have a significantly different search space than the other features (Cross, Wave, Crown). The cross feature is a combination of two ridge features, with two perpendicular regions of leveled geometry. The Wave feature is a special version of the ridge feature, with an added parameter that causes the ridge to lean over. The Crown feature is a special version of the Bump feature that does not only have a parameter for the central bump, but for the middle and the corner of its edges as well. Finally, a Plane feature without parameters was included to test if the algorithm is able to distinguish cases of other features with small parameter values.

Each feature is defined using a 5 by 5 patch of nurbs control points. Each feature has a height parameter that defines a translation of the control points in the z-direction. The translation in the x- and y-direction is combined in the parameter *radius* (for the Bump and Crown feature) or defined separately as the parameter *width* and the parameter *length*

Feature identification starts with asking the user to place a bounding box on a target shape to determine the rough position and orientation of the feature. The bounding box also indicates what parts of the target shape assumingly belong to the feature to be identified.

4.1 Testing on Artificial Target Shapes

Because testing the algorithm on a large number of target shapes is very time-intensive, we simulated target shapes by instantiating a random feature from the library and setting its parameter values to random values.

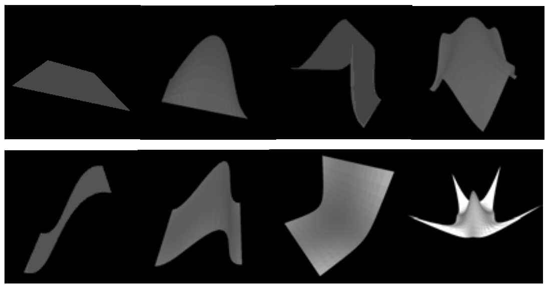


Figure 1: From the top left to bottom right: Plane, Bump, Ridge, Cross, Step, Wave, Blend and Crown feature.

Possible 'excess' material was simulated by creating two extra layers of control points around the control points used for the feature. The entire feature was converted to a triangular mesh with 2500 triangles and the points of the mesh were distorted by Gaussian noise. We conducted 100 tests for each combination of five different population sizes and five different selection sizes (which indicate the standard deviation of the Gaussian curve used for parent selection), amounting to 2500 tests in total. The results are given in Table 1.

The selection size is expressed as a percentage of the total population, and determines how selective the fitness values are. It indicates the percentage of individuals for which the chance of being selected as a parent is within two standard deviations of that of a prefect individual. The best result occurs for a selection size of 10%. The algorithm was implemented in C++ and tested on a computer with a 3GHz Pentium processor and 1GB RAM. The average computation times for the different population sizes were 58, 86, 112, 139 and 165 seconds for the respective population sizes. As the initial orientation and location of the feature are assumed to be given by the user, the 6 parameter values for translation and rotation are set to zero. To correct for the inaccuracy of the initial estimation of the user, Gaussian noise was added to the parameter values. The variables mutation probability and mutation rate did not have a significant influence on the success of the feature identification or the computation time.

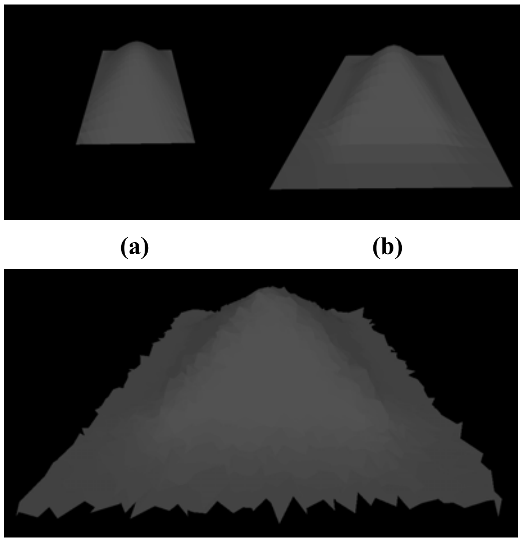


Figure 2: (a) A target feature, randomly chosen from the feature library (b) with added 'material' (c) distorted with Gaussian noise

Feature type ***(# of params)***	Correctly identified	Incorrectly identified	Not identified	Total
Plane ***(6)***	308 (96%)	13 (4%)	1 (0%)	**322**
Bump ***(8)***	291 (94%)	14 (5%)	3 (1%)	**308**
Ridge ***(9)***	274 (88%)	32 (10%)	5 (2%)	**311**
Cross ***(9)***	324 (96%)	8 (2%)	4 (1%)	**336**
Step ***(8)***	296 (95%)	14 (5%)	1(0%)	**311**
Wave ***(10)***	212 (69%)	73 (24%)	23 (7%)	**308**
Blend ***(8)***	301 (96%)	9 (3%)	2 (1%)	**312**
Crown ***(10)***	284 (97%)	4 (1%)	4 (1%)	**292**
Total	**2290 (92%)**	**167 (7%)**	**43 (2%)**	**2500**

Table 2: Results of the feature identification per feature type

A possible explanation for this is that the algorithm converges to a solution very fast (in on average 2.41 generations). The effect of mutation rate and probability is more long term and therefore not apparent in our tests. The effect of the initial inaccuracy of the translational and rotational parameters was significant, but very small. This is probably due to the large size of the domain in which parameter values are instantiated.

Table 2 shows, for each feature type, the number of features that were correctly or incorrectly recognized or for which the algorithm failed. The performance of the algorithm for the wave feature is considerably worse than for the other feature types, because it is often recognized as a ridge feature. This is no surprise, because the wave feature is a special version of the ridge feature, with one added parameter.

Population size/ Selection size	1000	1500	2000	2500	3000	Total
5%	78	91	96	100	100	**463**
10%	83	96	100	100	100	**479**
20%	82	94	95	99	100	**470**
30%	76	88	88	97	100	**449**
40%	62	83	89	93	100	**429**
Total	**381 (76.2%)**	**452 (90.4%)**	**468 (93.6%)**	**489 (98.8%)**	**500 (100%)**	**2290**

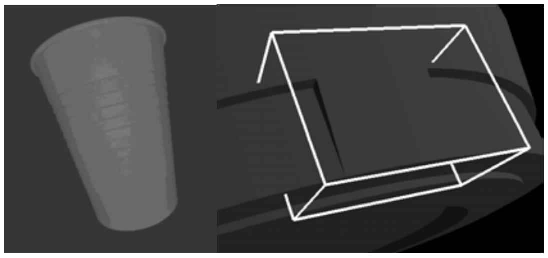
Table 1: Number of successful feature identifications per population and selection size

If this parameter has a small value, the wave feature behaves almost identical to the ridge feature. This also explains the slightly worse performance for the ridge feature.

From the results it can be concluded that the method correctly identifies a large portion of the test shapes, and that the computation time of the method is reasonable. Feature identification cannot be compared to other methods since no similar method exists, known to the authors, in current literature.

4.2 An Application Example

To demonstrate that the algorithm also works in non-artificial situations, we show an application example in which the algorithm is applied to a CAD model of a plastic coffee cup (see figure 3a). One of the features in this model, a wedge on the edge of the bottom of the cup, is shown in figure 3b, indicated by the user with a bounding box. To successfully be able to recognize the wedge-shaped feature, a new feature was added to the library, as is shown in figure 4. With this addition, the library consisted of 9 features.



(a) (b)

Figure 3: (a) CAD model of a coffee cup (b) A region of interest

At the start of the procedure, the user was asked to place a bounding box at the region of interest, as shown in figure 3b. The bounding box contained 113 polygons. From there on, a feature identification procedure was started. The feature identification used a feature population of 3000 and a selection size of 10%. For the identification, a mutation rate and probability of respectively 0.1 and 10% were used. The feature identification procedure was repeated 10 times. In all repetitions of the procedure, the feature was correctly recognized as a wedge feature. In 8 of these repetitions, three generations were needed to come to a correct identification, which cost on average 172 seconds. In two cases, only two generations were needed, in on average 118 seconds. In all cases, the feature identification procedure terminated because 75% of the ancestors of the top feature in the last generation were of the type Wedge.

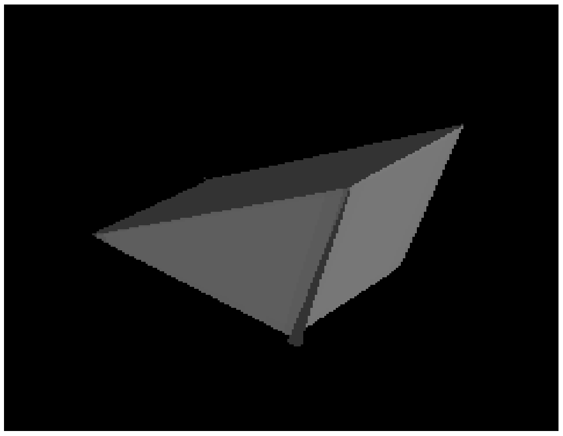


Figure 4: A wedge feature

5. CONCLUSIONS AND FURTHER WORK

We have proposed and implemented a method for feature identification using an evolutionary algorithm. The algorithm is successful in retrieving the feature type of a target shape and its computation time is reasonable. The algorithm was tested on a small feature library, but there is no theoretical limitation for using it on a larger, even interactive feature library. We intend to use the feature identification in a broader free form feature support system, in which users define and compose their own features, which can then be recognized in target shapes of their choice. We are currently completing an extension of the presented techniques to a free form feature recognition algorithm. In this procedure, the feature type is assumed to be known, but the parameter values are not. The free form feature recognition algorithm retrieves the parameter values in an evolutionary process that is similar to that presented in this paper.

An even more advanced application of the presented evolutionary technique that we are working on is in feature type construction. In this procedure, the target shape does not match a feature type that is available in the feature library. The shape is analyzed and a new feature type is automatically constructed that fits the target shape. To obtain a mechanism that is flexible enough to automatically construct a feature type, the genes are no longer formed by the parameters, but by the elements of the parameter mappings, increasing the number of genes from m to mn, where m is the number of parameters and n the number of nurbs control points. Also, the principle of mutation is extended to allow new individuals to have more or less genes than their parents.

The proposed feature identification method can be applied to retrieve the feature type prior to any (free form) feature recognition system. In future work it will be investigated if other user-input can be automated, such as the location of a feature or an estimation of its parameter values. When more parts of the work with form features is automated, an even higher-level approach to feature recognition can be implemented that in turn allows a user to more easily manipulate shapes.

6. REFERENCES

[Chu90] 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.

[Dav91] Davis, L, editor, Handbook of genetic algorithms, New York, Van Nostrand Reuinhold, 1991.

[Flo89] 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.

[Gho02] Ghosh, A. and Tsutsui, S., editors, Advances in Evolutionary Computing, Natural Computing Series, Springer, 2002.

[Gol89] Goldberg, D.E., Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, Massachusetts, 1989.

[Jos88] Joshi, S., Chang, T.C., Graph-based heuristics for recognition of machined features, Computer-Aided Design, Vol. 20, No.2, pp.58-66, 1988.

[Kim92] Kim, Y.S., Recognition of form features using convex decomposition, Computer-Aided Design, Vol. 24, No. 9, pp. 461-476, 1992.

[Lan05] Langerak, T.R., Vergeest, J.S.M., Song, Y., Parameterising styling lines for reverse design using free form shape analysis.", Proceedings of IDETC/CIE, ASME 2005.

[Nez97] Nezis, K., Vosniakos, G., Recognizing 2½D shape features using a neural network and heuristics, Computer-Aided Design, Vol. 29, No. 7, pp. 523-539, 1997.

[Pal05] 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.

[Pra92] Prabhakar, S., Henderson, M.R., Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models, Computer-Aided Design, Vol. 24, No. 7, pp. 381-393, 1992.

[Pre02] 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.

[Sak95] Sakurai, H., Volume decomposition and feature recognition: part 1 – polyhedral objects, Computer-Aided Design, Vol. 27, No. 11, pp. 833-843, 1995.

[Sak96] Sakurai, H., Dave, P., Volume decomposition and feature recognition: part 2 – curved objects, Computer-Aided Design, Vol. 28, No. 6, pp. 517-537, 1996.

[Shah01] 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.

[Song05] Song, Y., Vergeest, J.S.M., Bronsvoort, W.F., Fitting and manipulating freeform shapes using templates, Journal of computing and information science in engineering, Vol 5, No. 2, pp. 86-95, 2005.

[Sub95] 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.

[Tho99] 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.

[Van93] Vandenbrande, J.H., Requicha, A., 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.

[Van94] Vandenbrande, J.H., Requicha, A., 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.

[Woo82] Woo, T., Feature extraction by volume decomposition, Proc. Conf. CAD/CAM Technology in Mechanical Engineering, Cambridge, MA, USA, March 1982.