Automated Learning of Model Classifications

Cheuk Yiu Ip William C. Regli Leonard Sieger Ali Shokoufandeh
Geometric Intelligent and Computing Laboratory
Department of Computer Science
College of Engineering
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104

ABSTRACT

This paper describes a new approach to automate the classification of solid models using machine learning techniques. Existing approaches, based on group technology, fixed matching algorithms or pre-defined feature sets, impose a priori categorization schemes on engineering data or require significant human labeling of design data. This paper describes a shape learning algorithm and a general technique for "teaching" the algorithm to identify new or hidden classifications that are relevant in many engineering applications. In this way, the core shape learning algorithm can be used to find a wide variety of model classifications based on user input and training data. This allows for great flexibility in search and data mining of engineering data.

Categories and Subject Descriptors

H.3.3 [**Information Storage or Retrieval**]: Information Search or Retrieval; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; I.2.6 [**Artificial Intelligence**]: Learning
; J.6 [**Computer Applications**]: Computer-Aided Engineering

General Terms

Experimentation

Keywords

3D Search, Shape Recognition, Shape Matching,
Solid Model Databases, Machine Learning.

1. INTRODUCTION

This paper introduces an adaptation of machine learning and data mining techniques for identification of arbitrary geometric and manufacturing categories in CAD database. Recent work in industry [17] has explored the use of neural networks to identify parts (fasteners) based on multiple 2D views. To our knowledge, no past research exists on how to use machine learning techniques to train 3D shape recognition system with CAD data.

SM'03, June 16–20, 2003, Seattle, Washington, USA.

Our approach is to use learning algorithms (decision tree learning and reinforcement learning) to develop weighted similarity functions based on an underlying shape distribution-based shape model matching algorithm.

In practice, indexing of parts and part families had been done with group technology (GT) coding [18]. Group technology was designed to facilitate process planning and cell-based manufacturing by imposing a classification scheme on individual machined parts. These techniques were developed prior to the advent of inexpensive computer technology, hence they are not rigorously defined and are intended for human, not machine, interpretation. Some of the early work on feature identification from solid models aimed to find patterns in model databases or automate the GT coding process . The common aspect of all of these techniques is that they are all *post priori*; one runs their algorithm on model and it produces the category or label for it. There are many issues with this: If my categorization scheme changes? Do I need an entirely new algorithm? It would be nice if one had an algorithm that one could train to find arbitrary categories.

2. RELATED RESEARCH

2.1 Comparing 3D Models

There are two basic types of approaches for matching and retrieval of 3D CAD data: (1) *feature-based* techniques and (2) *shape-based* techniques. The feature-based techniques [6, 16], going back at least as far as Kyprianou's thesis [10], extract engineering features (machining features, form features, etc.) from a solid model of a mechanical part for use in database storage, automated GT coding, etc. Elinson et al. [4] used feature-based reasoning for retrieval of solid models for use in variant process planning. Cicirello and Regli [3] examined how to develop graph-based data structures and create heuristic similarity measures among artifacts; this work was extended in [2] to manufacturing feature-based similarity measurement. McWherter et al. [13] have integrated these ideas with database techniques to enable indexing and clustering of CAD models based on shape and engineering properties.

The shape-based techniques are more recent, owing to research contributions from computational geometry, computer vision and computer graphics. A shape-based approach works as the representational "lowest common denominator": STL or VRML (or other) polygon mesh. From the polygon mesh, measures of similarity can be computed among 3D models. Thompson et al. [19] examined reverse engineering of designs by generating surface and machining feature information off of range data collected from machined parts. Sipe, Hilaga et al. [7] present a method for matching 3D topological models using multi-resolution Reeb graphs. The

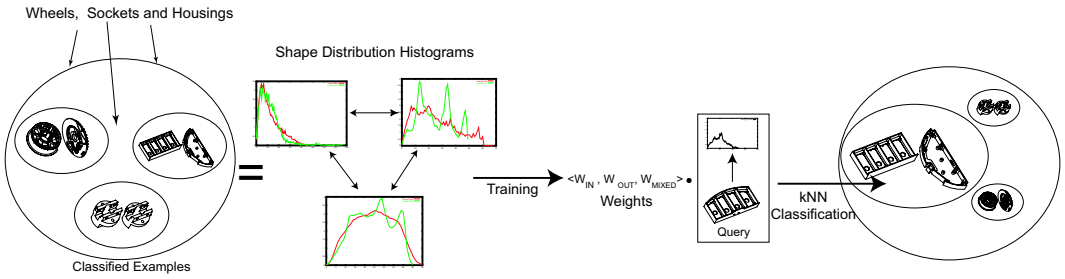

Figure 1: An Overview of our approach: Training with classified examples to classify query models with *k*NN technique.

method of Osada, Funkhouser et al [15] is a basis for the work in this paper. Their method creates an abstraction of the 3D model as a probability distribution of samples from a shape function acting on the model. Their technique is generally robust under model degradation, but it is a rigid technique and is a poor discriminator among model classes because it analyzes gross model shape, rather than the discriminatory features that are common for CAD/CAM data. In addition, these recent published studies [7, 15] have focused exclusively on a very limited set (several hundred) of heterogenous (planes, trees, phones, etc.) and manually-classified 3D graphics, animation and rendering models; a set that does not include any models that are specifically engineering, solid modeling or mechanical CAD oriented. It should be noted that the CAD/CAM dataset addressed in this paper is significantly more complex (i.e., polygon meshes are done at a very high degree of refinement) and homogeneous (i.e., models are often very similar from a gross-shape standpoint, hence discrimination is difficult and not always based on pure shape). All models used in this paper are available through the National Design Repository at `http://www.designrepository.org`.

2.2 Mining and Knowledge Discovery

Knowledge Discovery is the overall process of extracting non-trivial relations from large amounts of data. In [1], Agrawal outlined the three most important issues of many database-mining systems: *classification*, *association*, and *sequencing*. Fayyad et. al. Perhaps most well-studied problem among all these is the clustering problem, the main goal of which is to generate a compact description for subsets of data. A popular method of partition clustering is called *k*-clustering [5]. While the general problem is known to be NP-Hard, many effective heuristics have been developed, including the *k*-means algorithm [11] and the *k*-medoid [9] method.

Nearest Neighbor Machine Learning. The *k* nearest neighbor learning algorithm [14] (*k*NN) learns classifications by storing training examples and classifies query instances according to examples that are closest to the query instances. This algorithm is an instance based, unsupervised machine learning algorithm, typically used for pattern recognition, learning complex functions and data mining. The *k*NN algorithm requires:

- *A set of example instances*, to be used as the model answers for classifying query instances.
- *A distance metric* returning a numerical value describing the dissimilarity between two data instances.
- *k* example instances to be inspected to determine the classification of a query instance.
- A *locally weighting function* for combining the *k* local points into the result.

Data instances are described by *n* attributes, projected into an *n*-dimensional data space as a vector $< a_1, a_2, \cdots, a_n >$ and then given as input to *k*NN. Similar data instances are expected to fall into the same categories and distribute close to one another in the data space, forming clusters of parts that represent different categories. *k*NN learning works off this assumption and classifies query instance according to the classification of the *k* nearest example instance of the query instance. Hence, given a set of example instances and their corresponding classifications, the *k*NN learning algorithm proceeds as follows:

1. Store $\{s_1, s_2 \cdots s_n\}$ and $\{c_1, c_2 \cdots c_m\}$;
2. Accept an unclassified query instance, s_q;
3. Calculate the distances between s_q and $\{s_1, s_2 \cdots s_n\}$;
4. Return the classification of s_q given by the locally weighted function and classifications of the *k* nearest example instances.

3. PROBLEM FORMULATION

Given a set of solid models and corresponding categories, our work attempts to extract the related attributes, features or patterns to automatically construct a model classifier, as illustrated in Figure 1. Our approach integrates learning with 3D model matching so that, using a small subset of example models as training data, *k*NN can tune the matcher to perform classifications based on the input examples. In this way, the matching algorithm can learn arbitrary classification schemes.

Using kNN to Classify Solid Models. Let S be a set of solid models $\{s_1, s_2 \cdots s_l\}$ which is to be classified into m categories $\{c_1, c_2 \cdots c_m\}$. In order to apply *k*NN, we need (1) a subset of S to use for training examples; (2) a number k; (3) and a function to serve as a distance metric among models in S. Distance metrics measure the dissimilarity between models, and a number have been developed in previous research [8, 12, 15].

Tuning Parameters for Comparisons. Next, one needs to adjust the parameters of the model comparison algorithm based on the specified classification, i.e., to return short distances if, for example, s_2 and s_7 are both in the same class, c_5; and larger distances if models are not in the same class. Hence, given solid models $\{s_1, s_2, s_3\}$, a category c_1 and the distance metric $D(s_1, s_2)$, *k*NN requires $s_1, s_2 \in c_1 \wedge s_3 \notin c_1 \Rightarrow D(s_1, s_2) < D(s_1, s_3)$.

We develop a general method for adjusting the distances returned by $D(s_1, s_2)$ to satisfy this condition. We take advantage of the fact most model distance functions are based on measuring the distance between n comparable attributes, usually represented as a vector $< a_1, a_2, \cdots, a_n >$. We assess the discriminatory power of each attribute and assign weights to each attribute in order to maximize the ability of the classifier to discriminate among the classes given in the training set. Hence, the distance in between a pair of models is a weighted distance among n attributes:

$$D(s_1, s_2) = \sum_{i=1}^{n} w_i \cdot D(s_{1_i}, s_{2_i}) \quad (1)$$

$D(s_{1_i}, s_{2_j})$ represents the distance between attribute i of models s_1 and s_2, w_i represents the weight of the attribute. The trick of how to calculate the w_i's is discussed in Section 4.2.

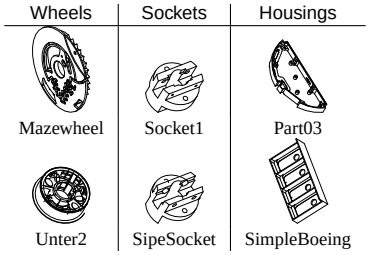

Figure 2: Training examples.

4. APPROCH

To illustrate the approach, we use distance metrics based on shape distribution functions [8, 15] in conjunction with an example dataset consisting of six mechanical parts from three distinct categories: *Sockets*, *Wheels* and *Housings*. All six example parts and their respective categories are shown in Figure 2. We briefly review the underlying technique, based on [8], for comparing solid models then show how it can be a basis for training automated classifiers.

4.1 Enhanced Shape Distribution Matching

A mesh representation of a solid model is stochastically sampled by placing random point-pairs on its surface and generating the the probability distribution of distances between these points. As reported in [8], the techniques of [15] can be significantly enhanced to discriminate among globally homogeneous (both locally diverse) models by classification of these point-pair distances as:

- *IN* distances: The line connecting the two points lies completely *inside* the model.
- *OUT* distances: The line connecting the two points lies completely *outside* the model.
- *MIXED* distances: The line connecting the two points passes both *inside and outside* of the model.

With the statistics of the classification of points and their Euclidean distances, we construct normalized *probability* vs. *distance* histograms for each distinct *IN*, *OUT*, and *MIXED*. The accumulated distributions of classifications are also recorded as percentage ratios of point pairs falling into *IN*, *OUT*, and *MIXED* categories for the sampled model (*IN*% + *OUT*% + *MIXED*% = 100%) These are used to assess the significance of *IN*, *OUT*, and *MIXED* distribution histograms, i.e., big differences in the *IN*% for two models would diminish the significance if a close measurement between the *IN* histograms, etc.

Distribution Example. Point pairs are sampled and classified to construct histograms as shown in Figure 3. Shape distributions histograms are compared to produce dissimilarity measures. *IN*, *OUT* and *MIXED* histograms of the models are mapped to a three attribute vector $< h_{IN}, h_{OUT}, h_{MIXED} >$. The dissimilarity in between models is represented by a per bin L_1 norm Minkowski distance in between their corresponding shape distribution histograms, computed using across each of the j histogram bins as:

$$L_1(h_1, h_2) = \frac{\sum_{i=0}^{n} |h_{1_i} - h_{2_i}|}{j}$$

This is done for each of the *IN*, *OUT*, and *MIXED* histograms. The differences *IN%*, *OUT%*, and *MIXED%* are used to scale L_1 norm

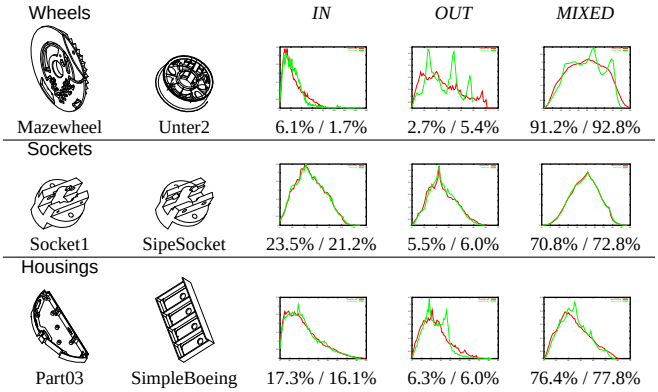

Figure 3: Shape distributions for the example models.

histogram distances to reflect the significance of correlations based on the differences in the sample sizes in each category of *IN*, *OUT*, and *MIXED*.

Matching Example. Averaged and scaled distances across all pairs of example parts in the earlier example are shown relative to the Mazewheel part in Table 1:

Part	*IN*	*OUT*	*MIXED*	Average	Scaled
Unter2	15.7	72.2	31.8	39.9	122.9
Socket1	57.9	58.9	78.1	65.2	232.5
SipeSocket	56.0	68.7	77.0	67.2	232.5
Part03	16.2	32.1	39.1	*29.2*	*97.63*
SimpleBoeing	19.7	49.6	33.9	34.4	110.44

Table 1: Distances in between Mazewheel part and other training examples.

Note that the Mazewheel part could easily be mis-classified in this simple example if one uses just average distance or scaled distances. Since the closest, in average and scaled distances, part to Mazewheel is Part03, instead of Unter2, which is supposed to be the only part shares the *Wheels* category with Mazewheel. This illustrates the shortcoming of untrained shape matching algorithms as well as the deficiencies of simple naive combinations of measures. Neither average or scaled distance measure produced the categorical grouping correctly.

4.2 Learning Categories

Classifying models with *k*NN can easily be done based on the measures between the histograms and the sampling percentages. The problem is that these distance metrics are composed of information from just the two models that are being compared. To perform classification, one needs to learn a set of attribute weights to enhance the discrimination across *all* models in the dataset and across *all* classes. Using the *IN*, *OUT* and *MIXED* histograms, and the *IN*%, *OUT*% and *MIXED*% sampling data, our approach is to derive a set of weights to combine these measures into a distance metric that maximize discrimination among model classes.

4.2.1 Training Process

A small subset of models, including representatives from each category, is used to learn the weights appropriate for the categorization. Shape distribution histograms of all training examples are computed and compared to determine the frequencies of *IN*, *OUT* or *MIXED* distance to serve as the representative distance between the classes. The training of weights on the distances of *IN*, *OUT* or *MIXED* histograms is done as follows:

1. Build *IN*, *OUT* and *MIXED* histograms of training models;

2. Compute distances between all pairs of training models;
3. Select, using the categories, an *IN*, *OUT* or *MIXED* distance for each pair of training models as the representative distance;
4. Normalize the frequencies of *IN*, *OUT* or *MIXED* distances to be the weights.

Three distances, *IN*, *OUT* and *MIXED*, are calculated for each pair of models. From each triplet of distances, only one suitable representative distance is selected for computing weights:

- If two models fall into the same category, select the shortest distance among *IN*, *OUT* and *MIXED*;
- Otherwise, select the longest from *IN*, *OUT* and *MIXED*.

The sampling frequencies of *IN*, *OUT* and *MIXED* reflects the chance of the respective *IN*, *OUT*, and *MIXED* histogram L_1 distance being appropriate for use in an aggregate distance computation. The weight triplet $< w_{IN}, w_{OUT}, w_{MIXED} >$ is derived as:

$$w_i = \frac{\#i}{\#IN + \#OUT + \#MIXED}, i \in \{IN, OUT, MIXED\}$$

Hence, different weight triplets are produced for each category. When computing the distance in between a query model and a training model, the weight triplets that correspond to the training model's category is used to scale the distance between *IN*, *OUT* and *MIXED* histograms.

Category	Part	*#IN*	*#OUT*	*#MIXED*
Wheel	Mazewheel	1	1	3
	Unter2	1	0	4
Sockets	Socket1	0	0	5
	SipeSocket	0	0	5
Housing	Part03	1	0	4
	SimpleBoeing	1	1	3

Table 2: Frequencies of *IN*, *OUT* and *MIXED* being selected as the appropriate distances.

Category	w_{IN}	w_{OUT}	w_{MIXED}
Wheel	0.2	0.1	0.7
Sockets	0	0	1.0
Housing	0.2	0.1	0.7

Table 3: Weights of *IN*, *OUT* and *MIXED* for example dataset.

Example. The frequencies and weights of a set of mechanical parts for "Wheels", "Sockets" and "Housings" categories are shown in Table 2 and Table 3. Recall that both average and scaled distance functions returned parts from *Housing* group as the nearest to the Mazewheel part which, instead, belongs to the *Wheel* group. The weight triplet $< w_{IN}, w_{OUT}, w_{MIXED} >$ revises distance between Mazewheel and Part03 to be 39.1 and distance between Mazewheel and Unter2 reduces to 33.9. Distances of other parts are shown in Table 4. Using $< w_{IN}, w_{OUT}, w_{MIXED} >$ favors the *MIXED* distance and decreases the influence of *OUT* distance which scale up the largest difference *MIXED* between Mazewheel part and Part03 yet suppresses the largest difference, *OUT* for Unter2.

Part	Average	Scaled	Revised
Unter2	39.9	122.9	**33.9**
Socket1	65.2	232.5	78.1
SipeSocket	67.2	232.5	77
Part03	29.2	97.63	39.1
SimpleBoeing	34.4	110.44	37.1

Table 4: Revised distances in between Mazewheel part and other example models.

4.3 *k*NN Classification

Given the set of training models and the set of weight triplets, one can accept and classify new query instances, s_q:

1. Sample and construct shape distribution histograms from s_q;
2. Compute the *IN*, *OUT* and *MIXED* distances in between the query model and all training models;
3. Compute the distance in between example models and query model by scaling *IN*,*OUT* and *MIXED* distances with distribution differences and weights triplet $< w_{IN}, w_{OUT}, w_{MIXED} >$;
4. Select the nearest *k* examples for classification.

Example. The Socket2 part is a minor variation on Socket1, shown in Figure 4. Shape distribution histograms are constructed for Socket2 and compared to histograms of the training models. Representative distances are compute with weights $< w_{IN}, w_{OUT}, w_{MIXED} >$, as shown in Table 5. Query model Socket2 is considered to be closest to example query models Socket1 and SipeSocket in this dataset.

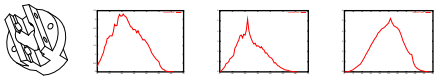

Figure 4: Socket2 with shape distribution histograms.

Category	Part	Distance
Wheel	Mazewheel	90.05
	Unter2	39.1
Sockets	Socket1	**4.58**
	SipeSocket	6.99
Housing	Part03	73.9
	SimpleBoeing	60.48

Table 5: Distances between Socket2 and example models.

4.3.1 *Classifying Query Models*

The classification of the query model s_q is the ultimate goal. The *k* closest example models are used to increase the robustness of the classification and reduce the effect of possible outliers or noise in the example model. Based on the categories of the *k* nearest example models, the query model s_q is classified by a locally weighted function provided to *k*NN learning algorithm, in our work either *Majority* and *Gaussian Kernel Regression*.

The *Majority* method returns the majority of categories of *k* nearest training examples neighbors as the classification. All *k* example neighbors "vote" for their classification and the classification with the highest votes will be returned as the classification of s_q. *Gaussian Kernel Regression* assigns weights to the *k* nearest example neighbors,s_i, according to a Gaussian kernel function. $D(k_i, s_q)$ is the distance between example model s_i and query model s_q, with a standard deviation σ is:

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{D(s_i,s_q)^2}{2\sigma}}$$

The category with the highest accumulated weight is returned as the classification of s_q.

Example. For $k = 3$, the categories of the 3 closest example models to Socket2 were considered to the classification of Socket2. The standard deviation of models, 28.5, was used as the standard deviation in the Gaussian function. From Table 6 both Majority and Kernel Regression classifications returned Sockets to be the category of query model Socket2, which is obviously correct.

5. EXPERIMENTAL RESULTS

This method has been implemented in Java/Perl and executed on Solaris platforms. Experiments were conducted using mechanical part data from the National Design Repository (`http://www.`

Category	Example	Vote	G(x)
Wheel	Unter2	1	0.007
Total		1	0.007
Sockets	Socket1	1	0.123
	SipeSocket	1	0.129
Total		**2**	**0.0252**

Table 6: Classifying Socket2

designrepository.org). Specifically, we illustrate how one can create classifiers using two different sets of mechanical CAD models under two different classification schemes:

- Classify models based on appearance or functionality;
- Classify models based on general manufacturing properties.

Both datasets were initially classified by hand and training examples were randomly selected based on this classification. The objective was to see if the system could learn weights from the training examples and then classify the non-training examples in a manner consistent with the human classification. All test models are available at http://www.designrepository.org/SM03.

Shape and Functionality Classification. 85 CAD models were first manually classified into 12 categories according to the general properties of shape or function for the models. The *k*NN classifier was trained using a subset of the models and then used to classify the remaining ones. The number of training examples per class was chosen proportional to the size of the class. The classifier performed extremely well at classifying Linkage_arms from the variable radius Spectrometer assembly from the National Institute of Standards and Technology (NIST). As shown in Table 7(a), more than 70% of *k*NN classifications were correct in our experiments: i.e., given a model, over 70% of the time the classifier gave it the correct class label as the top choice. The *k*NN classification of CAD models along with their categories is shown in Figure 5.

Process Classification. 56 CAD models were hand classified into 4 categories according to general properties of the manufacturing processes that would be used to create them: rotational parts, injection molded parts, cast-then-machined parts, and rotational-machined parts. The *k*NN classifier performed similarly in all categories in our selected example. Even though the average correctness of the *k*NN classifier on this example, as shown in Table 7(b), was not as strong as the *Shape and Functionality* example, both examples' best performances were nearly equal. Indicating that if particular sets of example models were provided, our approach could perform equally well on both example classifications. The *k*NN classification of our test models is shown in Figure 6

Shape	Correctness
Highest	80.70%
Lowest	64.91%
Average	72.30%
Std-dev	4 %

(a) Shape

Process	Correctness
Highest	79.70%
Lowest	59.11%
Average	66.71%
Std-dev	8 %

(b) Process

Table 7: *k*NN classifications statistics.

6. CONCLUSIONS

This paper described a new approach to classification of solid models using machine learning techniques, enabling the automatic categorization of models, archived components or brand-new parts, using a classification schema based on training data. It showed how weight learning with training examples and *k*NN can be used for pattern recognition and data mining in the challenging new domain of CAD databases. As part of our work, these techniques have been applied to a large, heterogeneous CAD model database—The National Design Repository—to learn and adapt it to various categorization and browsing schemes.

There are several issues for discussion and future research. Clearly, the performance of *k*NN classifier depends on the quality of training examples. This paper simply uses random selection—with more experience, we hope to identify how to select training data to maximize discrimination abilities. Additionally, while our initial success rates of 70%-80% and above seem low, they are in line with expected results for use of *k*NN in this context and certainly considerably higher in information content than just random assignment. Our subsequent experiments suggest that success rates can be improved above 85%-90% using this technique. With refinement, we hope that eventually users will be to *train* a retrieval system to recognize arbitrary categories.

Acknowledgments. This work was supported in part by NSF CAREER Award CISE/IIS-9733545, ONR Grant N00014-01-1-0618, Honeywell FM&T and Lockheed Martin. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or the other supporting organizations.

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 1–10. ACM, May 1993.

[2] V. Cicirello and W. Regli. Machining feature-based comparisons of mechanical parts. In *International Conference on Shape Modeling and Applications*, pages 176–187. ACM SIGGRAPH, the Computer Graphics Society and EUROGRAPHICS, IEEE Computer Society Press, Genova, Italy, May 7-11 2001.

[3] V. Cicirello and W. C. Regli. Resolving non-uniqueness in design feature histories. In D. Anderson and W. Bronsvoort, editors, *Fifth Symposium on Solid Modeling and Applications*, New York, NY, USA, June 8-11 1999. ACM, ACM Press. Ann Arbor, MI.

[4] A. Elinson, D. S. Nau, and W. C. Regli. Feature-based similarity assessment of solid models. In C. Hoffman and W. Bronsvoort, editors, *Fourth Symposium on Solid Modeling and Applications*, pages 297–310, New York, NY, USA, May 14-16 1997. ACM, ACM Press. Atlanta, GA.

[5] D. Fasulo. An analysis of recent work on clustering algorithms. Technical Report 01-03-02, Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195, April 1999.

[6] J.-H. Han, W. C. Regli, and M. J. Pratt. Algorithms for feature recognition from solid models: A status report. *IEEE Transactions on Robotics and Automation*, 16(6):782–796, December 2000.

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH*, pages 203 – 212, New York, NY, USA, August 2001. ACM, ACM Press.

[8] C. Y. Ip, D. Lapadat, L. Sieger, and W. C. Regli. Using shape distributions to compare solid models. In *Seventh ACM Symposium on Solid Modeling and Applications*. ACM SIGGRAPH, ACM Press, Jun 17-23 2002.

[9] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons,

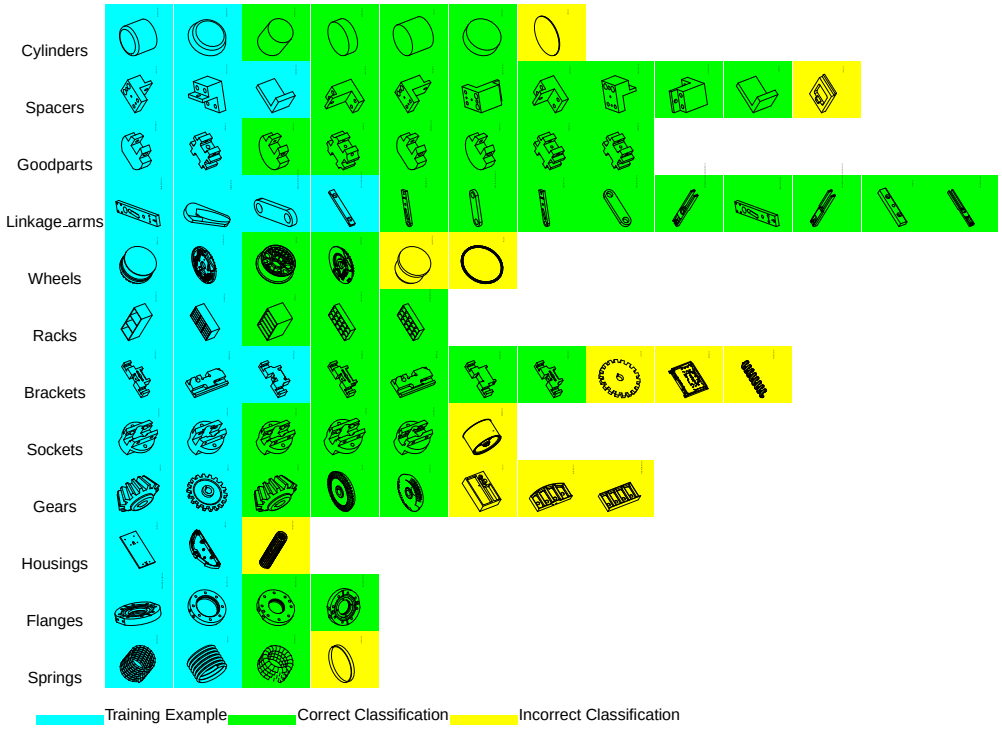

Figure 5: *k*NN shape or functionality classifications.

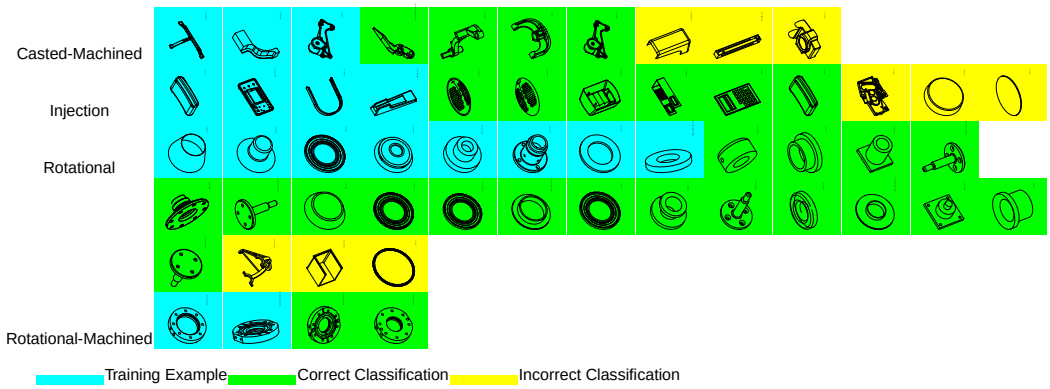

Figure 6: *k*NN manufacturing process classifications.

Inc., 1990.

[10] L. K. Kyprianou. *Shape Classification in Computer Aided Design*. PhD thesis, Christ College, University of Cambridge, Cambridge, United Kingdom, July 1980.

[11] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium*, pages 281–297, 1967.

[12] D. McWherter, M. Peabody, A. Shokoufandeh, and W. Regli. Database techniques for indexing and clustering of solid models. In D. Dutta and H.-P. Seidel, editors, *Sixth ACM/SIGGRAPH Symposium on Solid Modeling and Applications*, pages 78–87. ACM, ACM Press, June 4-8. Ann Arbor, MI 2001.

[13] D. McWherter, M. Peabody, A. Shokoufandeh, and W. Regli. Solid model databases: Techniques and empirical results. *ASME/ACM Transactions, The Journal of Computer and Information Science in Engineering*, 1(4):300–310, December 2001.

[14] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[15] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, Octobor 2002.

[16] J. Shah, D. Anderson, Y. S. Kim, , and S. Joshi. A discourse on geometric feature recognition from cad models. *ASME Transactions, the Journal of Computer and Information Science in Engineering*, 1(1):41–51, March 2001.

[17] S. D. G. Smith, R. Escobedo, M. Anderson, and T. P. Caudell. A deployed engineering design retrieval system using neural networks. *IEEE Transactions on Neural Networks*, 8(4):847–851, July 1997.

[18] C. S. Snead. *Group Technology: Foundations for Competitive Manufacturing*. Van Nostrand Reinhold, New York, 1989.

[19] W.B.Thompson, J. Owen, H. de St. Germain, S. Stark Jr., and T. Henderson. Feature-based reverse engineering of mechanical parts. *IEEE Transactions on Robotics and Automation*, 12(1):57–66, Feburary 1999.