Automatic Classification of CAD Models

A Thesis
Submitted to the Faculty
of
Drexel University
by
Cheuk Yiu Ip
in partial fulfillment of the
requirements for the degree
of
Master of Science in Computer Science
February 2005
# Table of Contents

List of Tables ........................................................................................................... v
List of Figures ........................................................................................................ vi
Abstract .................................................................................................................. viii

1. Introduction ......................................................................................................... 1
   1.1 Motivation .................................................................................................... 1
   1.2 Research Goals ......................................................................................... 3
   1.3 Approach .................................................................................................... 4
   1.4 Overview of Thesis ................................................................................... 5

2. Related Work ....................................................................................................... 6
   2.1 3D Models Representation ....................................................................... 7
       2.1.1 Solid Model Representations ......................................................... 7
       2.1.2 Shape Model Representations ....................................................... 7
   2.2 Extracting and Comparing Features from 3D CAD Models ....................... 8
       2.2.1 Extracting and Comparing Features of Solid Models ..................... 8
       2.2.2 Extracting and Comparing Features of Shape Models ................... 9
   2.3 Machine Learning and Classification ....................................................... 9
       2.3.1 Nearest Neighbor Classification .................................................... 10
       2.3.2 Support Vector Machines ............................................................. 11

3. Framework for Classifying CAD Models ......................................................... 14
   3.1 Pattern Classification System .................................................................. 14
       3.1.1 Feature Extraction .......................................................................... 16
       3.1.2 Classification .................................................................................. 18
       3.1.3 Feature Relevance .......................................................................... 21
   3.2 Example CAD Classifications .................................................................. 23
       3.2.1 Shape and Appearance ................................................................. 23
3.2.2 Manufacturing Processes ................................................. 23
3.2.3 Functions ........................................................................ 25
3.3 CAD Classification Example Approaches ......................... 28
3.4 Summary ............................................................................ 28
4. kNN Shape Classification with Multiple Shape Distributions ......... 30
  4.1 CAD Classification Using Multiple Shape Descriptors .......... 30
  4.2 Extracting Shape Features Using Shape Distributions .......... 32
    4.2.1 Primer on Shape Distributions ..................................... 32
  4.3 Learning a Weighted Classifier from Examples .................. 35
    4.3.1 Training Process ..................................................... 36
  4.4 Classification Using Nearest Neighbor Classifier ............... 39
    4.4.1 k Nearest Neighbor Classification ............................. 40
  4.5 Experimental Results ....................................................... 41
    4.5.1 Mechanical CAD Models Data Set ............................... 42
    4.5.2 Performance .......................................................... 42
    4.5.3 Shape and Functionality Classification ......................... 43
    4.5.4 Precision and Recall Evaluation ................................. 44
  4.6 Summary ............................................................................ 46
5. Classifying Prismatic Machined and Cast-then-Machined Artifacts .... 47
  5.1 Prismatic Machined and Cast-then-machined ...................... 47
  5.2 Process Classification with Shape Matching .................... 49
  5.3 Summary ............................................................................ 50
6. SVMs Manufacturing Classification Using Surface Curvature Statistics .... 51
  6.1 Extracting Surface Curvature Features .............................. 51
    6.1.1 Surface Curvature on Mesh Surfaces .......................... 52
    6.1.2 Estimating Curvature from Mesh Models ..................... 54
List of Tables

3.1 Example classification approaches .................................................. 28
4.1 Distances in between Mazewheel part and other training examples. ............ 35
4.2 Frequencies of \textit{IN}, \textit{OUT}, and \textit{MIXED} being selected as the appropriate distances. 38
4.3 Weights of \textit{IN}, \textit{OUT} and \textit{MIXED} for example dataset................................. 38
4.4 Revised distances in between Mazewheel part and other example models. ...... 38
4.5 Distances between query model Socket2 and example models. ................. 40
4.6 Classifying Socket2 with majority and kernel regression methods. ............ 41
4.7 \textit{kNN} shape classification statistics. .................................................. 44
5.1 Classification accuracies of shape descriptors. ....................................... 49
6.1 SVMs and curvature statistics classification accuracy. ............................ 63
6.2 Nearest neighbor and curvature statistics classification accuracy. ............. 64
# List of Figures

1.1 Example online catalog for mechanical parts. ........................................... 2  
2.1 SVMs classification margin. ................................................................. 12  
3.1 Pattern classification system. ............................................................... 15  
3.2 Feature extraction and classification in Shape distribution. ....................... 16  
3.3 3D zernike descriptor. ........................................................................ 18  
3.4 Reeb graph representation of CAD models. ........................................... 18  
3.5 Scalespace decomposition of a CAD model. ............................................ 19  
3.6 Clustering .......................................................................................... 20  
3.7 $k$ nearest neighbor classifier. (Q is an query instance)............................. 21  
3.8 Examples of models from a shape classification dataset. .......................... 24  
3.9 Examples of models from a manufacturing classification dataset ............... 25  
3.10 Functional representation of an assembly. ............................................. 26  
3.11 Functional descriptions of parts and assembly in OWL. ........................... 27  
4.1 Overview of weight learning with shape distribution. ............................... 30  
4.2 Training examples. ............................................................................... 32  
4.3 Shape distribution of example models ................................................... 34  
4.4 Query model Socket2 with shape distribution histograms. ....................... 39  
4.5 $k$NN shape or functionality classifications. .......................................... 43  
4.6 Precision recall plot for weighted shape distributions. ............................ 45  
5.1 Manufacturing classification dataset ....................................................... 48  
5.2 Manufacturing equipment ...................................................................... 48  
5.3 Similar cylindrical parts made by different processes. ............................. 50  
6.1 Machine stocks and resulting artifacts from different processes .................. 52  
6.2 Artifacts manufactured by different processes share similar features .......... 52  
6.3 CAD models colored by curvature values. ............................................. 55
6.4 Curvature shaded models and descriptors. .............................................. 57
Abstract
Automatic Classification of CAD Models
Cheuk Yiu Ip
William C. Regli, Ph. D.

This thesis research introduce a framework for classifying CAD models in a pattern classification model. This model separated feature extraction and classification from 3D CAD and shape classification procedures. This framework enable the incorporation of different invariant shape features and classifiers, according to example CAD/CAM oriented classifications. Two example approaches are used to demonstrate how CAD models can be classified according to shape and manufacturing for interest in the CAD/CAM domain using supervised machine learning classifiers. The first approach describes how to combine multiple shape descriptors using a linear weighting formulation. A training procedure is introduced to estimate appropriate weights. Then models are classified using the nearest neighbor classifier. The second approach aimed at classifying prismatic machined and cast-then-machined artifact. Normal surface curvature is first shown to be a relevant feature that discriminates the manufacturing classifications, then statistics of surface curvature are collected to represent models. Non-linear support vector machines were applied to successfully learn the classification of different manufacturing processes.
1. Introduction

This thesis research applies a pattern recognition/classification model with supervised machine learning techniques for identification of shape and manufacturing classifications in CAD database. The goal is to improve databases in engineering using low level approximated geometry of CAD models. It facilitates the retrieval and archival of artifacts for engineers, encourages reuse of parts and design to improve product life cycle management, and accelerates product design and manufacturing processes.

Automatic CAD classification can be modeled under a pattern recognition/classification framework, such that, different features of models and classifiers can be plugged in to produce different classification schemes. Shape matching research has presented various descriptors that capture shapes of models. This thesis research introduces a training technique to combine multiple descriptors for fitting some given shape classification using nearest neighbor classifier. This thesis also shows why and how to use surface curvature and support vector machines classifiers distinguish prismatic machined and cast-then-machined artifacts under the new domain of manufacturing classification.

1.1 Motivation

Retrieval and archival of engineering artifacts facilitate part reuse in engineering. It is estimated that 75% of design activities comprises of case base design—designs that are variations and improvements upon previous designs. CAD knowledge-bases are vital to engineers who search through vast amounts of corporate legacy data and navigate online catalogs to retrieve the appropriate components for assembly into new products.

Current engineering parts catalog, Figure 1.1, describe artifacts by different attributes, such as, manufacturers, product series, and model numbers. These meta attributes require
significant annotations. To reduce the need of human inputs, joint efforts from computer vision, graphics and mechanical engineering research aim to automate part classification.

Figure 1.1: Example online catalog for mechanical parts.

In practice, indexing of mechanical parts and part families had been done with group technology (GT) coding [49, 1, 23, 6, 8, 9, 21, 24, 27, 46, 50]. 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 [36] or automate
the GT coding process [24, 3, 25, 2]. Recent work in industry [48] has explored the use of neural networks to identify parts (fasteners) based on multiple 2D views. Research on how to use machine learning techniques to train 3D shape classification system with CAD data is rather brief.

Research on computer vision has been successfully recognize, compare, and indexed 2D images of different faces and household objects. Computer graphics developed different shape descriptor and comparison algorithm to distinguish general 3D models. This research combines these research effort to produce engineering relevant classifications using geometry information provided by 3D CAD models.

1.2 Research Goals

A framework for CAD models classification is introduced by this thesis research. Under this framework, one can extract features that describe shape/solid models and apply a generic machine learning classifier to group models according to a specific categorization or discover potential classifications. The following specific problems will be investigated for modeling and application of the framework:

- Formulate the CAD model classification problem using a pattern classification model. (Feature extraction then machine learning classification).
- CAD model classifications in the interest of engineering.
- Combine multiple shape descriptors using a weighting formulation and machine learning procedure.
- Observe and extract relevant features to classify models according to a new categorization schema.
1.3 Approach

This research attempts to classify the mesh representation of artifacts. Previous research, such as, feature recognition and process planning, focused on extracting manufacturing information from solid models. Mesh models have become a useful CAD representation thanks to the development of rapid prototyping and laser scanning acquisition technologies. Mesh models allow us to compare models generated by any CAD system.

CAD classification procedure follows a generic pattern classification model. Models are abstracted into sets of features for the application of machine learning classifier. The challenge of CAD classification is that the categories can be both subjective and objective. Artifacts can be subjectively classified by their shape and appearance or objectively classified according to the manufacturing processes and functionality.

To classify model by shape, this thesis research introduce the use of a training procedure to estimate weighted similarity functions for combining multiple shape model descriptors. Then an instance based $k$ nearest neighbor ($k$NN) machine learning technique is applied to classify query models base on shapes. Multiple shape distribution descriptors of CAD models will be used to illustrate how the proposed procedure adaptively combines different descriptors to fit an example model and classification.

Manufacturing classification of CAD models is accomplished by applying curvature descriptors and support vector machines in this thesis. This thesis research investigates in prismatic machined and cast-then-machined artifact classifications. The surface curvature of artifacts is found as an discriminating feature among the manufacturing processes. Support vector machines are used to learn the classification in between the targeted processes to classify other models. The experimental results shows the combination of surface curvature and support vector machines out perform existing shape descriptors and classifiers on the targeted prismatic machined and cast-then-machined classification.
1.4 Overview of Thesis

The remainder of this thesis will be organized as follows: A review on related work in CAD model representation and comparison is provided in Chapter 2. Chapter 3 discusses performing CAD classification using a generic pattern classification model. \(k\)NN shape classification with weighted shape descriptors is presented in Chapter 4. Chapter 5 describes the prismatic machined and cast-then-machined manufacturing classification, show, and explain why current shape descriptors fail to classify model according these manufacturing properties. In order to classify these two processes, surface curvature and support vector machines are introduced in Chapter 6. Chapter 7 concludes thesis research and discusses possible future work.
2. Related Work

The target of this thesis research is to model CAD classification as a pattern classification system. The proposed approach learn its classifier from a given a set of classified examples and be able to assign query models into appropriate categories. The goals are similar to systems performing person identification in computer vision, speech and text classification systems in natural language processing and information retrieval. They are collectively known as pattern recognition/classification systems.

Pattern recognition is the research area that studies the operation and design of systems that recognize patterns in data. It encloses sub-disciplines such as statistical pattern recognition pattern recognition. Statistical pattern recognition focuses on discriminant analysis, feature extraction, error estimation, cluster analysis.

In particular, the approach described in this thesis apply pattern a recognition model to classify CAD models, related research comes from different areas to accomplish two goals to complete a classification system.

1. Feature extraction

2. Classification

Feature extraction abstracts a CAD model into a set of representative attributes. The classification component find the separation of different categories through machine learning techniques. The rest of this section provides the details of the related work.

Due the extensive use of the terminology “feature” in both pattern recognition and mechanical computer aided design(MCAD) domains, in this thesis, “features” refer to representative attributes in the sense of machine learning or pattern recognition, where as descriptions will always prepends to MCAD features, such as “machining features” or “surface features”. To understand what features can be extracted from CAD models for
pattern classification, different 3D models representations will be briefly reviewed.

2.1 3D Models Representation

Most CAD models are defined parametrically as solid models. However, approximated shape models represented by a polygonal mesh are becoming another useful representation thanks to the development of rapid prototyping from approximated models and the acquisition of shape models through 3D scanning.

2.1.1 Solid Model Representations

CAD models are traditionally exact representations of 3D solids, which are suitable for creating physical models. In commercial CAD systems like Pro/Engineer and I-DEAS, models are dominantly represented by boundary representations (B-Rep). Objects are represented by a data structure that gives information about each of the object’s faces, edges, vertices, and how they are joined together. Under B-Rep, two types of information are recorded: (1) a topology record of the connectivity of faces and edges; (2) and a set of parametric equations that describes the geometry and the location of vertices, faces, and edges (e.g. NURBS). Solid models give a complete and compact representation for design, simulation, and manufacturing purposes. Yet these models are usually stored in proprietary data formats across different CAD/CAM systems. Thus, for example, comparing models generated on I-DEAS against Pro/Engineer involves some lossy data exchange process, through conversion from STEP to IGES, or approximated shape models.

2.1.2 Shape Model Representations

3D shape models are approximated models characterized by a mesh of polygons for presentation or rendering purposes in computer graphics. Rather than exact parametric equations, polygons are used to approximately curved surfaces. Only the geometry of
triangles are stored without any topological information. In contrast to proprietary solid model formats, open mesh file formats such as VRML and STL, are widely available. Although shape models are not suitable for modeling physical properties or simulations in CAD/CAM systems, polygonal meshes can serve as the lowest common denominator in comparing CAD models, by faceting solid models generated by different modeling systems. Shape models of objects can also be acquired easily by using laser scanners or CT to enable comparison of digital and physical artifacts.

2.2 Extracting and Comparing Features from 3D CAD Models

3D Model comparison research extract and compare representative features from models according to different interest. Two basic types of approaches for comparing and retrieval of 3D data: (1) *Engineering feature* based techniques and (2) *Shape* based techniques.

2.2.1 Extracting and Comparing Features of Solid Models

Engineering feature based techniques [22, 47], dating from late 1970s [36], extract engineering features (e.g. machining features, form features) from a solid model of a mechanical part for use in database storage, automated GT coding. Elinson et al. [17] used engineering feature based reasoning for retrieval of solid models for use in variant process planning. Cicirello and Regli [13] examined how to develop graph-based data structures and create heuristic similarity measures among artifacts; this work was extended in [14] to a manufacturing feature based similarity measurement. McWherter et al. [38] have integrated these ideas with database techniques to enable indexing and clustering of CAD models based on shape and engineering properties. Cardone et al. [10] compared machining features of solid models for manufacturing cost estimation.
2.2.2 Extracting and Comparing Features of Shape Models

The shape based techniques are more recent, owing to research contributions from computational geometry, computer vision, and computer graphics. From the polygon mesh, different transformation invariant attributes can be extracted as the means of similarity among 3D models. Thompson et al. [55] examined the reverse engineering of designs by generating surface and machining feature information off of range data collected from machined parts. Hilaga et al. [26] present a method for matching 3D topological models using multi-resolution reeb graphs. The method of Osada et al. [42] creates an abstraction of the 3D model as a probability distribution of samples from a shape function acting on the model. Novotni and Klein [40] demonstrated the use of 3D Zernike descriptors. Kazhdan et al. [34] compares 3D models with spherical harmonics. Tangelder and Velkamp [52] constructed weighted point sets by sampling Gaussian curvature or normal statistics to compare triangular mesh. For further information, Velkamp and Tangelder published a recent survey on shape retrieval methods [54]. While these techniques target to compare general 3D models, Ip et al. [29, 30] and Bespalov et al. [4] are focused on comparing shape models of CAD with shape distributions and scale-space representations. Iyer et al. [32, 33] presented a CAD oriented search system. Surveys of shape comparison for engineering parts were published by Iyer et al. [31] and Cardone et al. [11].

2.3 Machine Learning and Classification

Machine learning techniques construct classifiers through recognizing patterns on example data. Many learning algorithm has been proposed in the rich research history. Two successful classifiers were applied in this thesis:

1. Nearest Neighbor Classification

2. Support Vector Machines
2.3.1 Nearest Neighbor Classification

Nearest neighbor classification is an example of instance based machine learning technique. It classifies query instances by assigning them the classification of their closest examples. \( k \) nearest neighbor (kNN) classification assigned the classification according to \( k \) closet examples of the query instance. This classifier assumes that closely distributed data are similar and forming clusters of instances that represent different categories.

**kNN classification 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 instances into the result.

\( k \)NN classification use a distance function to evaluate the distance in between instances. Example common distance functions are Minkowski \( L_N \) norm distance for comparing points, earth mover distances for comparing point sets, and Kullback-Leibler measures relative entropy on different instances.

Given a set of example instances, \( s_1, s_2 \cdots s_n \), and their corresponding classifications, \( c_1, c_2 \cdots c_m \), kNN classification 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.

The major computation cost for $k$NN classification occurs at classification time for finding the closest neighbors. No explicit training process is required. $k$NN classification imposes no requirement on the format of shape descriptors, at a minimum, a distance value has to be computed between two different shape descriptors. The search for nearest neighbors can be greatly accelerated if the descriptors live in some vector space, which allows the examples to be indexed by spatial data structures, such as r-tree [19] or kd-tree.

2.3.2 Support Vector Machines

Support vector machines is an example of linear discriminant classifier proposed by Vapnik [15, 7]. It finds the best hyperplane, with the largest margin, that separates instances from different categories. It relies on the fact that given an appropriate non-linear mapping to a sufficiently high dimension, data from two categories can always be separated, therefore if the original data points are not linearly separable (e.g. XOR function), they can be projected to a higher dimension for separation. Figure 2.1(a) and 2.1(b) shows examples of possible classifiers for a binary classification. Only the classifier with the maximum margin is optimal (Figure 2.1(b)), the objective of SVMs is to find this particular classifier.

Consider linearly separable example data, $x_i \in \mathbb{R}^n, i = 1 \cdots l$, from a binary classification, SVMs attempt to find a classifier that satisfies $w \cdot x + b = 0$. The closest training examples, also called support vectors, defines the classification margin with a width of $\frac{2}{||w||^2}$, such that it returns a label $y_i$ for $x_i$ satisfying inequality $y_i(w \cdot x_i + b) - 1 \geq 0$.

The maximum margin, hence minimizing $||w||^2$, can be found by minimizing the Lagrangian formulation of the classifier. Positive Lagrangian multipliers $\alpha_i \cdots \alpha_l$ are introduced for each example to form a convex quadratic programming problem.
Figure 2.1: SVMs classification margin.

\[ L_P = \frac{1}{2} - \sum_{i=1}^{l} \|w\|^2 - \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i \]

subject to:

\[ \sum_{i=0}^{l} \alpha_i y_i = 0 \]

which is equivalent to maximizing the dual of \( L_P \)

\[ L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \]

The general SVMs formulation is ideal for data exhibits a proper margin, yet real world classification data are often noisy and not completely separable. SVMs can be extended to learn the separation by allowing errors during training. Slack variables \( \xi_i \) are introduced to account for the errors. \( y_i (w \cdot x_i + b) - 1 + \xi_i \geq 0 \). This formulation minimizes \( \frac{\|w\|^2}{2} + C(\sum_{i=1}^{l} \xi_i) \). \( C \) is a user defined penalty parameter, the error penalty increases when \( C \) increases. \( L_D \) does not change and still maximizing:

\[ L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \]
but subjects to:

\[ 0 \leq \alpha_i \leq C, \quad \sum_{i=0}^{l} \alpha_i y_i = 0 \]

To generalize SVMs to classify non-linear data, one can project the data into a higher dimensional space via a mapping function \( \Phi \). Note that \( L_D \) only depends on \( x_i \cdot x_j \), therefore only values of \( \Phi(x_i) \cdot \Phi(x_j) \) needs be computed without directly working in the projected high dimensional space. Functions that computes \( \Phi(x_i) \cdot \Phi(x_j) \) are called kernel functions, \( K(x_i, x_j) \), common examples are high degree polynomials, radial basis functions or sigmoid function. By substituting \( x_i \cdot x_j \) by \( K(x_i, x_j) \), the SVMs separates the training data in high dimension space which may results in non-linear classifiers in the input space.

\[
L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]

As long as the kernel function can be computed in constant time, the complexity of SVMs do not increase for non-linear cases.

In contrast to \( k \) nearest neighbor classification, the major computation cost for SVMs classification occurs at training time for solve the quadratic programming problem. The computation cost at classification time depends the of number of support vectors.
3. Framework for Classifying CAD Models

To outline the important attributes for computers to classify CAD models, this research follows a model of pattern classification or recognition systems and investigates the relationship between information provided by the model representations and various themes in CAD classifications.

Pattern classification for CAD models consists of two steps. (1) Abstract CAD models by extracting relevant attributes that distinguish different categories. (2) Label models with appropriate categories. The first step is largely related to the work of solid and shape models matching as described in Section 2. Shape matching research built transformation invariant model descriptors. The second step benefits from a rich history of research in machine learning. A wealth of techniques is available to separate the model descriptors into different categories.

For the purpose of CAD classification, three criteria illustrate common classification targets. A typical classification of CAD models is by their appearance, this is similar to the goal of computer vision and computer graphics, retrieving models that look similar from human standpoint. Moreover, one important distinction in between CAD models and general 3D models is that CAD models are designed to be physically constructed. The manufacturing processes and the use of artifacts are important properties of CAD. CAD models can also be objectively classified by their manufacturing processes and functions, in addition to their subjective appearance.

3.1 Pattern Classification System

The proposed CAD classification system is analogous to a pattern recognition system. This thesis follows a model of pattern recognition systems [16], Figure 3.1. A sample sys-
tem for pattern classification consists of five steps or components: (1) sensing (2) segmentation (3) feature extraction (4) classification (5) post processing. Sensing and segmentation components acquire and separate the object of interest from irrelevant noise, however, they are not necessary in CAD classification, since the CAD models are cleanly created in the form of 3D models and ready for feature extraction. This research focuses on the extraction of features from CAD models and classifying the resulting descriptors.

![Pattern classification system](image)

Any combination of shape feature and classifier can be plugged into the framework to build a model classification system. Current research effort from computer graphics and engineering often focused on feature extraction and distance metrics, however, overlook the possibility on selecting classifiers.

**Example.** Most of the recent shape matching and classification techniques, in fact, largely follow this pattern classification model by introducing a shape feature then apply nearest neighbor classification. Figure 3.2 shows how shape distribution matching can be separated into tasks of feature extraction and classification.

1. **Feature Extraction:** Shape distributions attempts to find the dissimilarities of 3D models using sampled distributions of simple shape functions (e.g., the distance between two random points on a surface). Statistics of the shape function is used as features.
2. **Classification**: Nearest neighbors classification is applied to return the closest (according to Minkowski $L_1$, Manhattan, distance) model/classification.

![Feature Extraction and Classification Diagram](image)

**Figure 3.2**: Feature extraction and classification in Shape distribution.

### 3.1.1 Feature Extraction

Feature extraction selects representative attributes from the classification targets for comparisons. Transformation invariant characteristics relating to the classification targets are picked from models to form shape descriptors, in particular, invariance to rigid body rotation is necessary for matching 3D models.

Shape descriptors are commonly represented as numerical vectors and graphs. In the vector representation, invariant attributes are represented as $n$ independent numerical attributes. Each model can be considered as a point in a $n$ dimensional vector space. This
generic form of descriptors enable the application of a variety of machine learning classifiers to separate data into different categories. Further query and indexing of models can be facilitated by using spatial indices (R-Trees) and computational geometry techniques (Range queries).

Another popular representation for matching 3D models employ graphs to maintain dependency between different attributes. Such descriptors are used to store structural, connectivity and topological information of models. This is a considerable richer representation than independent numerical attributes, but comparing models involves complex graph matching. To alleviate the computational overhead, many research applied heuristic or impose structural constraints in graphs (e.g. Trees instead than general graphs) to facilitate the comparison process. These techniques often come with custom matching functions that limit the application of generic classifiers or indexing of descriptors.

**Examples.** In the context of 3D model classification, many shape matching techniques have developed shape descriptors that represents a model as $n$ numerical attributes, like 3D Zernike descriptor. While techniques such as reeb graph descriptors and scalespace representation used graphs and trees to maintain the structural information.

- Figure 3.3 shows an example of 3D Zernike shape descriptor on CAD models. Zernike moments decompose a model into some orthonormal spherical basis functions. The descriptor is computed through spherical sampling on the voxelization of the model.

- Multi-resolution reeb graph descriptors [26] used graphs to maintain structural information of models. A reeb graph is a topological and skeletal structure for an object of arbitrary dimensions. It is defined by a Morse function (e.g. a height function) on the object’s surface manifold and critical points. Reeb graph feature extraction of two similar parts are shown in Figure 3.4.
• Scalespace representation considers CAD models as metric data and hierarchically decompose them using their spectral properties [4]. Figure 3.5 shows an example scalespace decomposition of a Fork model. Nodes in the tree of Fork represent different portions of the model. Topological matching is performed by matching the scalespace hierarchy of the models.

3.1.2 Classification

A classifier assigns categorical labels to CAD models, a classifier is considered as a function, \( f(s) = c \), where \( s \) and \( c \) are the CAD model (descriptor) and the returning category. In an ideal scenario classification function \( f(s) \) is known such that correct classification can be produced, perfect classifiers can be constructed by sufficient domain knowledge.
Figure 3.5: Scalespace decomposition of a CAD model.

Nevertheless, in CAD/CAM domain, the relationships between shape feature and classification are unclear. Which bin(s) and values of shape distributions signifies a screw or which set of surfaces shows that an artifact can be produced by a prismatic machining process? Although there is no concrete answers to the above questions, a considerable amount of screw models and prismatic machined parts are available. This abundance of data allows the application of pattern recognition and machine learning techniques to approximate appropriate classifiers. The aim is to solve for $f(s)$ through example data to separate models from different categories.
**Examples.** Machine learning procedures for classifier construction can be supervised or unsupervised. Supervised machine learning requires labeled example data to construct classifiers, whereas unsupervised machine learning constructs classifiers with example data without labels. This research specifically focuses on the application of supervised machine learning classifiers.

- **Unsupervised Machine Learning.** Clustering, Figure 3.6, is an example of unsupervised machine learning technique, $k$-means and hierarchical clustering algorithms find the intrinsic grouping of data, each group represents a different category. This approach produces the best possible groupings of models with respect to a feature set and a distance metric.

![Figure 3.6: Clustering](image)

- **Supervised Machine Learning.** The application of $k$NN classifier, Figure 3.7, and SVMs, Figure 2.1(b), in this research are all supervised machine learning classifiers. They explicitly attempt to adjust the returning classification according to the given labeled artifacts.

  - $k$NN classifier uses labeled example instances to infer categories of query model. Figure 3.7 shows an example of $k$NN classification. Q is an query object being
classified as positive under kNN classification. With \( k=5 \), the majority (four) of its closest five neighbors are positive instances.

![Figure 3.7: k nearest neighbor classifier. (Q is an query instance)](image)

- SVMs selects labeled example support vectors to bound the optimal classification margin through an explicit training process by solving a quadratic programming problem. (Figure 2.1, Section 2.3.2)

Under general setup of shape matching or retrieval experiments, models categorized example models are provided. Supervised learning classifier other than kNN classifier, such as, decision trees, neural network, and expectation maximization can also be applied. This research will give an example of classification accuracy being drastically affected by the selection of classifiers.

### 3.1.3 Feature Relevance

Feature relevance is the connection between the low level features and the resulting human oriented classification. The question being asked here is how does the value of a certain feature affect the output classification of a model. For example which value(s) in the physical moments transformation shows a CAD model is piece of screw?

A feature, \( x_i \), is relevant [5] to a targeted classification if there exists a pair of example,
A and B from different categories; A and B differ only in their assignment to feature $x_i$. This means the classification label can be reflected by the feature, $x_i$, in other words, the value of $x_i$ separates the models in the feature space.

In the Section 3.1.2, machine learning attempts to find the classification function that separates examples in to different categories, when more features are relevant to the targeted classification, the more accurate can the machine learning classifiers can learn that function. For this reason, feature extraction and classifiers complements each other. A set of relevant features separates models from different categories in the feature space, therefore, requiring a simpler classifier.

However, if the features are irrelevant to the classification, e.g. data from different categories are evenly distributed in the feature space, then data from different categories become inseparable, no classifier will be able to find a proper separation and perform classification correctly. Although there is an enormous effort in developing advance classifier like SVMs and neural networks, it should be stressed that feature relevance still plays an important role in the classification process.

This feature relevance problem is often tackled by domain knowledge or statistical techniques. The target is to select a subset of features that correlates with target classification. Domain knowledge can be applied to create a better set of adhoc features and it is still by far one of the most efficient approach in selecting relevant features for classification. One example of domain knowledge application in CAD classification is described in [33], skeleton graphs are first classified into topological types curved and loop to assist indexing and classification.

Automatic feature selection attempts to use statistics from training data to show some subset of features exhibits a higher correlation with the labeled classification. A concise discussion on different methods of automatic feature selection and evaluation can be found in [20].
3.2 Example CAD Classifications

In order to understand what features are relevant, three common classification targets in the interest in CAD/CAM domain are discussed.

3.2.1 Shape and Appearance

Classifying parts by their visual appearance is a natural target. Numerous research efforts in computer vision attempted to classify engineering artifacts from their 2D photo images. With respect to 3D models, the computer graphics community used approximated shape models to perform shape matching. Related research focused on comparing meshes of polygons to one another. Attributes related to geometry (locations of vertices and edges of triangles) of the mesh are sampled and have become the focus of representing the model’s appearance.

Example. Figure 3.8 shows mechanical artifacts classified by their shapes. The names of shape categories also reflects some primitive functional characteristics of the models. For instance, brackets are overhanging members that project from a structure and are usually designed to support a vertical load or to strengthen an angle. Linkage arms are motion transferring components from the spectrometer assembly. Nuts, Screws, and Blots are commonly used fasteners. This dataset will be classified by weighted shape distributions in Chapter 4.

3.2.2 Manufacturing Processes

One interest in computer aided engineering is computer aided manufacturing, automating the generation of manufacturing process plans from CAD models. Classification of CAD models according to different classes of manufacturing processes has become an important interest. Recently, Yao et al. investigated in milling cutting processes from 3D scan
Apart from the recent effort, automated process planning is traditionally derived from recognized manufacturing features on CAD models. Feature recognition interprets a part in terms of manufacturing features, such as slots, holes, and pockets. Popular approaches to this problem include graph-based, volumetric decomposition, and hint-based feature recognition. All of them utilize exact topology and geometry provided by solids to generate manufacturing features. Graph based approaches extract the topology of the model as a graph (faces as vertices and edges as edges), then recognize subgraphs that compose features. Hint based approach search for features through the set of hints on the models, then employ a completion procedure according to the nearby geometry and topology to generate manufacturing features. Topology and geometry of solid models directly influence the manufacturing process based classifications. This thesis research will show surface curvature can be relevant for classifying artifacts that are manufactured by different classes of equipments.
Example. Figure 3.9 shows a sample of mechanical artifacts made by two different manufacturing processes. This dataset was classified into (1) prismatic machined parts and (2) parts that are first cast and then have their finishing features machined. This dataset will be classified by surface curvature and support vector machined in Chapter 6.

<table>
<thead>
<tr>
<th>PRISMATIC MACHINED</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image of prismatic machined parts" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CAST-THEN-MACHINED</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Image of cast-then-machined parts" /></td>
</tr>
</tbody>
</table>

Figure 3.9: Examples of models from a manufacturing classification dataset.

3.2.3 Functions

Functional classification describes how engineering artifacts are used. This classification ties a part directly to its application. High level semantics of parts becomes the key attributes for comparison and classification. One important aspect of this problem is the functional relationship in between different artifacts could infer further characteristics of the resulting assemblies. It is unclear if there is a mapping between the low level attributes (e.g. topology, geometry and tolerance) of a CAD model and its function. Current func-
Functional classification of CAD models are highly dependent on human labeling. To make a connection in between the semantics and CAD model attributes, it is essential to construct a set of labeled CAD models that permits one to data mine the associations between low level model attributes and functionalities.

Example. Kopena et al. [35], Szykman et al. [51] and Wood et al. [56] attempted to model the semantics annotation of parts and assemblies formally with knowledge representation, Figure 3.10, and semantic web technologies in OWL(Web Ontology Language) format, Figure 3.11. This representation has little explicit relationship with the models’ geometrical and topological attributes. Efficient evaluation, storage and comparisons are made using logic programming (JESS, PROLOG) and XML databases.
Figure 3.11: Functional descriptions of parts and assembly in OWL.
3.3 CAD Classification Example Approaches

To demonstrate how the propose classification framework can be applied to practical CAD classification, two example approaches with different classification targets, features, and classifier will be presented in this thesis, Table 3.1.

<table>
<thead>
<tr>
<th>Classification Target</th>
<th>Feature Set</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>Weighted Multi Shape Distributions</td>
<td>$k$NN</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Surface Curvature</td>
<td>SVMs</td>
</tr>
</tbody>
</table>

The first approach attempts to combine multiple shape distributions descriptors and use $k$NN classifier to improve shape classification of CAD models. A training procedure is introduced to find a linearly weighted combination of the shape descriptors according to the given example models.

Manufacturing process classification is the focus of the second example approach. Surface curvature and SVMs are applied to classify artifacts that are being manufactured by prismatic-machined and cast-then-machined. Curvature and SVMs have been popular research area in computer graphics and machine learning, this research combines them to perform CAD classification. Chapters 5 and 6 introduce the detail about the prismatic-machined and cast-then-machined manufacturing classification, as well as, justifying why curvature and SVMs are selected to perform this specific classification task.

3.4 Summary

The CAD classification problem is modeled in a pattern classification framework in this thesis research. Existing shape match algorithm provides a wealth of feature extraction
techniques. Generic machine learning classifiers can be applied to approximate the separation of different categories. The appearance, manufacturing and functional properties are example relevant classification for CAD models.

The rest of this thesis presents two different approaches to use different features and supervised learning classifiers to categorize CAD models according their appearance and manufacturing processes.
4. \textit{kNN Shape Classification with Multiple Shape Distributions}

This chapter describes how to apply the model of pattern recognition systems to classify CAD model in respect to their shapes. A generic method is presented to combine and automatically weight multiple shape descriptors by training and nearest neighbors classification.

4.1 CAD Classification Using Multiple Shape Descriptors

Given a set of solid models and corresponding categories, this work attempts to adjust the weights of related shape descriptor for pattern discovery and model classifier construction, as illustrated in Figure 4.1. This approach integrates machine learning with 3D model matching so that, using a small subset of example models as training data, multiple descriptors can be combined and weighted before performing nearest neighbor classifications based on the input examples. In this way, the matching algorithm can adapt to different shape classification schemes.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4.1.png}
\caption{Overview of weight learning with shape distribution.}
\end{figure}
Nearest Neighbor Classification for CAD. 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 nearest neighbors classification, 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 [29, 37, 42].

Tuning Weights for Comparisons. A training procedure is used to adjust the weights of different shape descriptors in the model comparison process based on the specified classification, i.e. to return short distances if, for example, $s_1$ and $s_2$ are both in the same class, $c_1$; 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 $D(s_1, s_2)$, nearest neighbor classification requires:

$$\forall s_1, s_2, s_3, c_1 : s_1, s_2 \in c_1 \land s_3 \notin c_1 \Rightarrow D(s_1, s_2) < D(s_1, s_3) \quad (4.1)$$

A general method is developed for adjusting the distances returned by $D(s_1, s_2)$ to satisfy this condition. Representing $n$ comparable shape descriptors as a vector $< a_1, a_2, \cdots, a_n >$, the discriminatory power of each descriptor can be assessed and appropriate weights are assigned to each descriptor to maximize discrimination among the classes given in the training set. Hence, the distance in between a pair of models is a weighted distance among $n$ descriptors:

$$D(s_1, s_2) = \sum_{i=1}^{n} w_i \cdot D(s_{1i}, s_{2i}) \quad (4.2)$$

$D(s_{1i}, s_{2j})$ represents the distance between descriptor $i$ of models $s_1$ and $s_2$, $w_i$ represents the weight of the attribute. The trick of estimating $w_i$’s is discussed in Section 4.3.
4.2 Extracting Shape Features Using Shape Distributions

Shape distribution comparison functions [29, 41, 42] are used as shape features to illustrate the process of learning weights through a set of example models. Then query models can be classified according to their shapes using $k$NN classification.

A small example dataset is used to exemplify the proposed procedure. This dataset consists of six example parts originally in ACIS SAT format from three distinct categories: Sockets, Wheels and Housings. These parts will be used in the weight learning process and $k$NN classification to infer the categories of query mechanical parts. All six example parts and their respective categories are shown in Figure 4.2.

<table>
<thead>
<tr>
<th>Wheels</th>
<th>Sockets</th>
<th>Housings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazewheel</td>
<td>Socket1</td>
<td>Part03</td>
</tr>
<tr>
<td>Unter2</td>
<td>SipeSocket</td>
<td>SimpleBoeing</td>
</tr>
</tbody>
</table>

Figure 4.2: Training examples.

Shape distributions [29, 42] are briefly reviewed, then the weight parameters learning and classifications will be discussed.

4.2.1 Primer on Shape Distributions

A shape distribution can be viewed as a digital signature for a 3D model. Our approach is to use distribution-based techniques [42], with enhancement for matching CAD mod-
els [29], to perform statistical sampling of the shape properties of CAD models and use these samples to generate meaningful comparison among the models.

Let $S$ be a CAD model, let $T = \{t_1, t_2, \ldots, t_k\}$ be a set of triangular facets that approximate the topology of $S$. The facets of $T$ could be produced by existing mesh generation algorithms, Stereolithography exporters or VRML exporters. The facets in $T$ could also be from an active data acquisition system working off of actual models, as in [55].

Matching two models with shape distributions requires:

1. **Selecting a shape function.** Many different functions were tested by [42] and the $D_2$ shape function (measuring the distance between two random points on the surface of a model) was shown to be the most consistent.

2. **Sampling of random points.** Generate a sufficiently large number of random sample points pairs on surface of model $S$.

3. **Classification of point-pair distances.** [29] In computing the $D_2$, distances are classified based on their interaction with the local geometry and topology. Specifically, there are three kinds of interactions:

   - **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.

4. **Calculate shape distribution histograms.** Construct histograms associated with IN, OUT, and MIXED $D_2$ shape distribution function.
5. **Compare shape distribution histograms.** Using well-known curve matching techniques (Minkowski $L_N$).

**Shape Distributions Example.** To obtain a mesh representation of the example models, all example parts are translated to VRML format. Point pairs are sampled and classified to construct histograms as shown in Figure 4.3.

![Shape Distributions Example](image)

**Comparing Shape Distributions**

Shape distributions histograms are compared to produce dissimilarity measures. 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:

$$L_1(h_1, h_2) = \frac{\sum_{i=0}^{n} |h_{1i} - h_{2i}|}{j}$$
Table 4.1: Distances in between Mazewheel part and other training examples.

<table>
<thead>
<tr>
<th>Part</th>
<th>IN</th>
<th>OUT</th>
<th>MIXED</th>
<th>Average</th>
<th>Scaled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unter2</td>
<td>15.7</td>
<td>72.2</td>
<td>31.8</td>
<td>39.9</td>
<td>122.9</td>
</tr>
<tr>
<td>Socket1</td>
<td>57.9</td>
<td>58.9</td>
<td>78.1</td>
<td>65.2</td>
<td>232.5</td>
</tr>
<tr>
<td>SipeSocket</td>
<td>56.0</td>
<td>68.7</td>
<td>77.0</td>
<td>67.2</td>
<td>232.5</td>
</tr>
<tr>
<td>Part03</td>
<td>16.2</td>
<td>32.1</td>
<td>39.1</td>
<td>29.2</td>
<td>97.63</td>
</tr>
<tr>
<td>SimpleBoeing</td>
<td>19.7</td>
<td>49.6</td>
<td>33.9</td>
<td>34.4</td>
<td>110.44</td>
</tr>
</tbody>
</table>

This is done for each of the \(IN\), \(OUT\), and \(MIXED\) histograms.

**Distance Example.** Averaged and scaled (according to the scaling formula presented in [29]) distances of \(IN\), \(OUT\), and \(MIXED\) shape distributions across all pairs of example parts are computed. Distances between the Mazewheel part and the other parts (Table 4.1) as an demonstration:

Note that the Mazewheel part could easily be misclassified in this simple example if one uses just average or scaled the histogram 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 simple naive combinations of shape distances.

### 4.3 Learning a Weighted Classifier from Examples

In this section classification information is collected from the example models classifications and to derive the significance of different histograms.

Recall from Equation (4.1) and Equation (4.2) that the distance function should produce short distances for parts sharing the same category and long distances for parts falling into different categories, such that the weighted distance of histograms, \(w_{IN} \times IN + w_{OUT} \times OUT + w_{MIXED} \times MIXED\) minimizes the aggregate distance within a category and maximizes the aggregate distance across different categories.
Ideally, if the categories of the models being compared is known in advance, the aggregate distance in a category can be minimized by assigning a weight of one to the shortest distance among IN, OUT, or MIXED histograms and assign weights of zero to the other two histograms to represent the distance in between two models. On the other hand, assign a weight of one to the longest distance among IN, OUT, or MIXED histograms and assign weights of zero to the other two histograms to represent the distance in between two models from different categories.

Nevertheless, the categories of query models are not known at query time for selecting the representative distance, alternatively, the probability of the distance among IN, OUT, or MIXED being assigned a weight of 1 to be the corresponding weight of that distribution can be considered. With a set of classified example models, weight probabilities can be estimated by the frequency of assigning a weight of 1 to IN, OUT, and MIXED distances, during comparisons of all pairs example models from the same category, to derive weights for the corresponding category.

4.3.1 Training Process

Shape distribution histograms of all training examples are being compared to find out the frequencies of IN, OUT, or MIXED distance being selected to be the representative distance.

Provided a set of training examples models, the training of weights probability for IN, OUT, or MIXED histograms can be carried out as follows:

1. Profile the example models and construct IN, OUT, and MIXED histograms as signatures of each model.

2. Compute the $L_1$ Minkowski distances in between all pairs of model histograms, IN to IN, OUT to OUT and MIXED to MIXED.
3. Pick the appropriate IN, OUT, or MIXED distance for each pair of models, according to the corresponding categories, as the representative distance.

4. Normalize the frequencies of IN, OUT, or MIXED distances being selected as representative 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 weights computation.

- Select the shortest distance among IN, OUT, and MIXED when two models fall into the same category
- Select the longest distance among IN, OUT, and MIXED when two models fall into the different categories

The frequencies of IN, OUT, and MIXED distances being selected as representative distances reflects the chance of IN, OUT, and MIXED distance being the appropriate distance for aggregate distance computation. The weight triplet \(<w_{IN}, w_{OUT}, w_{MIXED}>\) is derived using:

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

Different weight triplets are produced to fit each category specifically. When computing the distance in between a query model and an example model, the weight triplets that correspond to the example model’s category will be used to scale the distance between IN, OUT, and MIXED histograms.

**Training Example.** The frequencies and weights of our example set of mechanical parts for Wheels, Sockets, and Housings categories are shown in Table 4.2 and Table 4.3.
Table 4.2: Frequencies of \textit{IN}, \textit{OUT}, and \textit{MIXED} being selected as the appropriate distances.

<table>
<thead>
<tr>
<th>Category</th>
<th>Part</th>
<th>#IN</th>
<th>#OUT</th>
<th>#MIXED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel</td>
<td>Mazewheel</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Unter2</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Sockets</td>
<td>Socket1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>SipeSocket</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Housing</td>
<td>Part03</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>SimpleBoeing</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.3: Weights of \textit{IN}, \textit{OUT} and \textit{MIXED} for example dataset.

<table>
<thead>
<tr>
<th>Category</th>
<th>(w_{IN})</th>
<th>(w_{OUT})</th>
<th>(w_{MIXED})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel</td>
<td>0.2</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>Sockets</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>Housing</td>
<td>0.2</td>
<td>0.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

In Section 4.2.1, average and scaled distance function returned parts from \textit{Housing} group as the nearest of Mazewheel part which, instead, belongs to the \textit{Wheels} group.

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 4.4.

Revising distances with \(<w_{IN},w_{OUT},w_{MIXED}>\) favors the \textit{MIXED} distance and decreases the influence of \textit{OUT} distance which scale up the largest difference \textit{MIXED} be-

Table 4.4: Revised distances in between Mazewheel part and other example models.

<table>
<thead>
<tr>
<th>Part</th>
<th>Average</th>
<th>Scaled</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unter2</td>
<td>39.9</td>
<td>122.9</td>
<td>33.9</td>
</tr>
<tr>
<td>Socket1</td>
<td>65.2</td>
<td>232.5</td>
<td>78.1</td>
</tr>
<tr>
<td>SipeSocket</td>
<td>67.2</td>
<td>232.5</td>
<td>77</td>
</tr>
<tr>
<td>Part03</td>
<td>29.2</td>
<td>97.63</td>
<td>39.1</td>
</tr>
<tr>
<td>SimpleBoeing</td>
<td>34.4</td>
<td>110.44</td>
<td>37.1</td>
</tr>
</tbody>
</table>
tween Mazewheel part and Part03 yet suppresses the largest difference, \( OUT \) for Unter2.

### 4.4 Classification Using Nearest Neighbor Classifier

With a set of example models and a set of weight triplets stored in a database, the system is ready to accept and classify new query instances. Given a new query model \( s_q \),

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 example models.

3. Compute the distance in between example models and query model by scaling \( IN,OUT \), and \( MIXED \) distances with distribution differences and weight triplet

\[
< w_{IN}, w_{OUT}, w_{MIXED} >
\]

4. Select the nearest \( k \) examples for classification considerations.

**1NN Classification Example.** Socket2, a variation of Socket1 is presented as a query model (Figure 4.4). Shape distribution histograms are constructed from Socket2 and compared to histograms of example models. Representative distances are compute with weights

\[
< w_{IN}, w_{OUT}, w_{MIXED} >
\]

Table 4.5 shows the distances.

![Shape distribution histograms](image)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21.7%</td>
<td>5.6%</td>
<td>72.6%</td>
</tr>
</tbody>
</table>

Figure 4.4: Query model Socket2 with shape distribution histograms.
Table 4.5: Distances between query model Socket2 and example models.

<table>
<thead>
<tr>
<th>Category</th>
<th>Part</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel</td>
<td>Mazewheel</td>
<td>90.05</td>
</tr>
<tr>
<td></td>
<td>Unter2</td>
<td>39.1</td>
</tr>
<tr>
<td>Sockets</td>
<td>Socket1</td>
<td>4.58</td>
</tr>
<tr>
<td></td>
<td>SipeSocket</td>
<td>6.99</td>
</tr>
<tr>
<td>Housing</td>
<td>Part03</td>
<td>73.9</td>
</tr>
<tr>
<td></td>
<td>SimpleBoeing</td>
<td>60.48</td>
</tr>
</tbody>
</table>

Query model Socket2 is considered to be closest to example query models Socket1 and SipeSocket in this dataset. Its classification will be determined by the nearest example models in the next section.

4.4.1 *k* Nearest Neighbor Classification

The classification of the query model *s*<sub>q</sub> is computed and returned after this process. Considering the categories of the *k* nearest example models, the classification of the query model *s*<sub>q</sub> is determined by a locally weighted function provided to the nearest neighbor classification. Here are two simple variations of the function.

- The *Majority* method simply returns the majority of categories of *k* nearest training examples neighbors as the classification. All *k* example neighbors “vote” for their classification. The classification with the highest votes will be returned as the classification of *s*<sub>q</sub>.

- *Gaussian Kernel Regression* assigns weights to the *k* nearest example neighbors, *s*<sub>i</sub>, according to a Gaussian kernel function. *D*(*k*<sub>i</sub>, *s*<sub>q</sub>) as the distance in between, example model *s*<sub>i</sub> and query model *s*<sub>q</sub>, with a standard deviation *σ*:

\[
\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{D(s_i,s_q)^2}{2\sigma^2}}
\]
The category with the highest accumulated weight is returned as the classification of \( s_q \).

**kNN Classification Example.** For instance, pick \( k = 3 \), the categories of the 3 closest example models to Socket2 were considered to the classification of Socket2.

Table 4.6: Classifying Socket2 with majority and kernel regression methods.

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>Vote</th>
<th>G(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel</td>
<td>Unter2</td>
<td>1</td>
<td>0.007</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1</td>
<td>0.007</td>
</tr>
<tr>
<td>Sockets</td>
<td>Socket1</td>
<td>1</td>
<td>0.123</td>
</tr>
<tr>
<td></td>
<td>SipeSocket</td>
<td>1</td>
<td>0.129</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td><strong>2</strong></td>
<td><strong>0.0252</strong></td>
</tr>
</tbody>
</table>

The standard deviation of models distances, 28.5, was used as the standard deviation in the Gaussian function. From Table 4.6 both Majority and Kernel Regression classifications returned Sockets to be the category of query model Socket2, which is obviously correct.

### 4.5 Experimental Results

Experimental results are presented to demonstrate the accuracy and performance of classifying solid models with our machine learning technique. Performance of the shape distributions, weight learning and \( k \)NN classification processes were tested to ensure they are practical to provide real-time classification of query models. In this evaluation CAD models classified according to shape or appearance.
4.5.1 Mechanical CAD Models Data Set

Test data consists of mechanical CAD models available from the National Design Repository (http://www.designrepository.org/). The dataset were entirely classified by hand, training examples for weight learning were randomly chosen and $k$NN classification was performed with the majority method, $k = 3$. The system classified all models other than the training examples. The hand classification of the datasets validated the results of the returned $k$NN classifications. Each experiment was repeated to verify the robustness of our approach.

4.5.2 Performance

The performance of our approach was tested through the sequence of, constructing shape distributions, learning weights $< w_{IN}, w_{OUT}, w_{MIXED} >$ and $k$NN classifying of models. On average, construction of shape distribution and weight learning for each example model took one minute. Classification of query models can be performed in less than one minute per model.

Sampling and histogram constructions of shape distribution took about one to one minutes per model, comparison of two models took less than one second. Experiments in Section 4.5.1 took 20 minutes to sample and construct histograms for 28 training examples models. Training of $< w_{IN}, w_{OUT}, w_{MIX} >$ for all categories took about one minute. In practice, classification of each query model can be performed in about one minute, including histogram construction time.

Shape distributions sampling and histogram constructions were implemented in Java. Histograms comparisons, weight learning and classification were done in Perl along with PostgreSQL to store the example model information, all experiments were run on a Sun Solaris UltraSPARC III workstation.
4.5.3 Shape and Functionality Classification

85 CAD models were first hand classified into 12 categories according to the shape or functions of the CAD models. A typical classifications of CAD models using weight learning and multiple shape distribution along with their categories are shown in Figure 4.5.

![Figure 4.5: kNN shape or functionality classifications.](image)

In this Shape Classification example, the weighted kNN classifier performed extremely well at classifying Linkage_arms from the variable radius Spectrometer assembly from the National Institute of Standards and Technology (NIST). In general, more than 70% of kNN classifications were correct in our experiments. Statistics are presented in Table 4.7.
As with any automated classification system, performance depends on the quality of training examples and the make up of the overall data set. In our experiments, we used data from the National Design Repository. While the CAD models selected are all real engineering artifacts, they exhibit considerable heterogeneity and variability within classes. This technique could perform better in a more realistic setting where the datasets would be much larger and more structurally homogeneous within classes. Better training sets can be identified. For example, instead of providing random examples as in our experiments, hand picking examples by experts can possibly improve the accuracy of classification.

### 4.5.4 Precision and Recall Evaluation

The performance of weighted shape distributions were evaluated by recall and precision measures for information retrieval systems. The recall and precision values at different thresholds are computed as follows:

\[
\text{recall} = \frac{\text{Retrieved and Relevant instances}}{\text{Relevant instances}}
\]

\[
\text{precision} = \frac{\text{Retrieved and Relevant instances}}{\text{Retrieved instances}}
\]

Under this experimental setting, the factors of recall and precision computation become:

- **Relevant models**: The number of models that fall in to same category as the query model.
- **Retrieved models**: The number of models returned by a query.

- **Retrieved and Relevant models**: The number of models returned and that fell into the same category as the query model.

Recall and precision values were first computed per model at different $k$ values. For each $k$, the arithmetic mean of the recall and precision across all models in a dataset was used as a representative value. To illustrate the results, precision is plotted against recall on different datasets and comparison techniques.

Ideally, a retrieval system should retrieve as many relevant models as possible, both high precision as well as high recall are desirable. Precision-recall graphs, plotting precision against recall, of weighted shape distribution and original shape distributions in Figure 4.6 (solid line represents original shape distributions, dotted line represents weighted shape distributions). It shows the trade-off between precision and recall. Trying to increase recall, typically, introduces more irrelevant models into the retrieved set, thereby reducing precision. Rightward and upward precision-recall curves indicates a better performance.

Figure 4.6: Precision recall plot for weighted shape distributions.
The precision of weighted shape distributions was higher than the original shape distributions [42] at a low recall rate, but it gradually dropped below original shape distributions as recall increases. This shows the accuracy was increased by the training process when a small amount of models were retrieved. This improvement was demonstrated for recall rates up to 0.4 (40% of the dataset). Engineers are mostly interested in the top ten to twenty hits of a model query, an improvement up to recall at 0.4, 34 models in our dataset shows a practical advantage of the weighted shape distributions over the original shape distributions.

4.6 Summary

In this chapter, a weighting technique is presented to combined multiple shape descriptors though training against classified example models. kNN classification is used to separated models into different categories. Experiments confirm this weighting technique can combined IN, OUT, and MIXED shape distributions for shape classification. On average 72% of the resulting classification was correct, the accuracy can reach 80%, depending on the given example configurations.

Weighting and shape distribution classification meets the appearance criteria of CAD classification in section 3.2.1. In chapter 6, this thesis research will continue to explore the possibilities of automated model classification according to manufacturing processes.
5. Classifying Prismatic Machined and Cast-then-Machined Artifacts

This chapter describes a CAD relevant manufacturing classification that cannot be separated by current shape matching techniques. Prismatic machined and cast-then-machined artifacts are the classification target. Various shape descriptors and nearest neighbor classifier are shown to classify a process classified model dataset with low accuracy. Characteristics of the manufacturing processes shows why current shape matching features fail to separate artifacts produced by the processes.

5.1 Prismatic Machined and Cast-then-machined

The classification of prismatic machined and cast-then-machined parts assist manufacturing cost estimation. Prismatic machining manufactures artifacts by material removal operations on regular shaped stock, such as rectangular blocks or cylinders. Cast-then-machined manufacturing first cast out stocks with the gross shape of the resulting parts, then let machining operations to construct the details. Figure 5.1 shows a small sample of the 100 process classified models in our dataset.

The engineering rationale in this classification is that parts that are exclusively machined are usually high-precision parts, or parts made in small batches (i.e., for custom jobs). Cast-then-machined parts are typically from larger production runs and generally have much looser tolerance considerations for the non-machined surfaces of the object. In this case the investment of physical plant is larger, as is the manufacturing production plan (i.e., one needs to machine a mold with which to do casting). Figure 5.2 shows example equipments to do casting and machining.

Machining is a high precision and time consuming process. It requires a process plan to route tools for material removal operations. The application of casting procedure loosely
constructs the shape of parts, reduces the need for machining, hence decreases the complexity of process plans and manufacturing time and costs.
5.2 Process Classification with Shape Matching

Current research on shape matching techniques focuses on extracting features to match the gross shape of mesh models. For example, shape distributions sample distances in between surfaces, zernike descriptors record the transformation of models from a sphere. Reeb graph technique segments and match shape by their topology. Scalespace decomposition separate models by shape features. However, they do not adequately distinguish artifacts manufactured by different processes. Table 5.1 shows nearest neighbor classification accuracy for some mesh based shape matching techniques. (The classification of closest example model is assigned to the query model.)

Table 5.1: Classification accuracies of shape descriptors.

<table>
<thead>
<tr>
<th>Shape Matching Techniques</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape Distributions [42]</td>
<td>57%</td>
</tr>
<tr>
<td>Scalespace [4]</td>
<td>54%</td>
</tr>
<tr>
<td>Reeb Graph [26]</td>
<td>55%</td>
</tr>
<tr>
<td>Zernike Moments [40]</td>
<td>52%</td>
</tr>
</tbody>
</table>

Similar setup of manufacturing equipments are able to produce a variety of artifacts in different shapes. Further, artifacts’ shapes can be drastically different when the same set of volume removal operations is applied on different shapes of stock. The resulting shapes of artifacts no longer strongly ties with their respective manufacturing processes. This explains why shape matching techniques performed mediocrely when the objective of retrieval targeted at similar manufacturing processes.

Figure 5.3 shows two parts of similar shapes but made by different manufacturing processes. Both models are cylindrical, the model shown in Figure 5.3(a) a is made by removing material from a cylinder stock, where as, the model shown in Figure 5.3(b) is manufac-
tured by casting out a stock with its shape (This stock will be shown in Figure 6.1(b)), then finish constructing the part by machining operations.

(a) Prismatic machined.  (b) Cast-then-machined.

Figure 5.3: Similar cylindrical parts made by different processes.

5.3 Summary

Prismatic machined and cast-then-machined manufacturing processes classification is shown to be indistinguishable by current shape descriptors and nearest neighbor classification. To fulfill the interest of manufacturing classification in CAD/CAM domain, there is a need to research on new manufacturing relevant features and classifiers for process classification.
6. SVMs Manufacturing Classification Using Surface Curvature Statistics

This chapter introduces the use of surface curvature as distinguishing features for manufacturing classification of CAD models using support vector machines. In last chapter, parts cannot be classified by recent shape descriptors with respect to prismatic-machined and cast-then-machined processes. This chapter demonstrates the relevant use of surface curvature and its application with support vector machines to discriminate the classification targets.

6.1 Extracting Surface Curvature Features

This research proposes the use of surface curvature and support vector machines to classify prismatic-machined and cast-then-machined models. Surface curvature is introduced as a relevant feature for distinguishing the two processes.

Considering the limited accessibility of cutting tools in 2.5D machining processes, material removal machining operations can only construct a finite set of surfaces. In contrast, the casting process allows a larger variety of surfaces. Figure 6.1 shows parts manufactured by prismatic-machined and cast-then-machined processes and their respective stocks before the part getting machined.

Without the casting process, prismatic-machined parts are manufactured from uniform stocks like rectangular and cylindrical blocks. Cast-then-machined processes cast out the stock before being machined. The stocks for cast-then-machined parts are more customized. A larger variety of surfaces are generated before machining. The challenges for classifying these two processes is that both processes share a set of surfaces generated by machining processes. Figure 6.2 shows parts manufactured by prismatic machine and cast-then-machined processes with their a machining features (holes) highlighted.
6.1.1 Surface Curvature on Mesh Surfaces

The curvature of a point on a planar curve is defined as the reciprocal of the radius of the osculating circle at that point. Extending this to regular surfaces, normal curvature, $\kappa_n$, is the curvature of the intersection on the smooth surface with a plane in direction $(s,t)$. $\kappa_n$ satisfies the following equation.

$$\kappa_n = \left(\begin{array}{c}s \\ t\end{array}\right) \left(\begin{array}{c}e & f \\ f & g\end{array}\right) \left(\begin{array}{c}s \\ t\end{array}\right) = \left(\begin{array}{c}s \\ t\end{array}\right) \mathbf{II} \left(\begin{array}{c}s \\ t\end{array}\right)$$
Principal curvatures is the largest and smallest curvature at a point in all directions, the principal directions are the directions, which the principal curvatures occur. If \((s,t)\) are expressed in the principle directions, the formulation for \(\kappa_n\) becomes:

\[
\kappa_n = \begin{pmatrix} s' \\ t' \end{pmatrix} \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix} \begin{pmatrix} s' \\ t' \end{pmatrix} = \kappa_1 s'^2 + \kappa_2 t'^2
\]

This formulation shows \(\kappa_1\) and \(\kappa_2\) are the eigenvalues of second fundamental tensor \(\mathbf{II}\), where \((s',t')\) are eigenvectors. Meaning that principal curvatures and directions can be found by eigenvalues decomposition on \(\mathbf{II}\).

Different representative curvature values can be computed from principal curvatures.

1. Maximum Curvature \(\kappa_1\)
2. Minimum Curvature \(\kappa_2\)
3. Mean Curvature \(H = \frac{\kappa_1 + \kappa_2}{2}\)
4. Gaussian Curvature \(K = \kappa_1 \kappa_2\)

Estimating curvature from mesh has been a great interest for computer graphics and visions. Curvature information has been used in a variety of applications, mesh smoothing, repairing surfaces, crest detection, re-meshing and non-photorealistic rendering. Exact curvature information can be computed from parametric surfaces, while mesh representation are piecewise approximation of surfaces, curvatures are also approximated. Some of the recent work on curvature estimation for mesh includes: Taubin [53] and Page et al. [43] estimated the curvature tensor with a weighted average of normal curvatures of neighboring vertices. Meyer et al. introduced discrete differential geometry operators [39]. Goldfeather and Interrante presented a cubic method for estimating principal directions [18]. This research employs Rusinkiewcz’s method [45] for fast, single pass estimation of curvature from smooth mesh.
6.1.2 Estimating Curvature from Mesh Models

Curvatures are estimated from mesh representation of CAD models. Rusinkiewicz’s algorithm works particularly well on smooth mesh models, such as those produced by faceting solid models. It estimates curvatures per vertex only by its immediate neighbors. To enable curvature estimation on coarse and noisy mesh models, such as those obtained by 3D laser scanning, curvature estimation algorithms need to consider a larger geodesic neighborhood, one example is presented in [43].

Following [45], per vertex curvature is computed by weighting the curvature of triangle faces adjacent to the vertex. Assuming each triangle on mesh surface is a smooth curved surface, curvature can be computed by solving $\mathbf{II}$ with constraints setup by the normals.

1. Compute per triangle face curvature by solving for $\mathbf{II}$ according to the constraints of difference between three normals.

2. Transform the coordinates with respect to the local coordinate frame of the vertex.

3. Weight the contribution of each adjacent face according the their voronoi area of the triangle.

4. Find the eigenvalues and eigenvectors of the tensor to determine the principal directions and curvatures.

To visualize the surface differences in between prismatic-machined and cast-then-machined parts, curvature values are computed for all vertices on the samples models, Figure 6.3 shows sample models that are colored according to different curvature values, minimum, maximum, mean, and Gaussian curvatures. Regions with zero curvature are colored in white, non-zero curvature regions are colored according to curvature values.

The following observations can be made:
<table>
<thead>
<tr>
<th></th>
<th>Prismatic-Machined Parts</th>
<th>Cast-then-Machined Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_1$</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>$H$</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>$K$</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 6.3: CAD models colored by curvature values.

- Cast-then-machined process produces artifacts with a higher portion of curved (shaded $\kappa_1$) surfaces.
- The minimum and Gaussian curvature, $\kappa_2$, $K$, of machining features, holes, slots, and pockets, surfaces are zero (white). Resulting in almost zero curvature for all prismatic machined parts.
- Cast-then-machined parts possess a higher variation of curvature values (more colors).
- The difference of minimum and maximum curvatures (colors) for prismatic-machined
parts is smaller than cast-then-machined parts.

6.1.3 Curvature Feature Relevance

Different curvature values signify different type of surfaces on the CAD models. While both manufacturing processes produce similar surfaces through machining operations, cast-then-machined process is expected to construct a larger variety of surfaces. Therefore, some distinctive curvature values should only be found on cast-then-machined parts but not prismatic-machined parts. These unique features separate the two manufacturing processes. Support vector machines are applied to recognize the optimal separation by supervised training.

6.1.4 Curvature based Shape Descriptor

Statistics of per vertex curvature are used to construct shape descriptors for classification. Curvature values naturally varies from $-\infty$ to $\infty$. To avoid extreme values and numerical problems, curvature values, $c$, is mapped from to $[-1, 1]$, similar to [52]

$$c' = \frac{c}{||c||} \left( 1 - \frac{1}{1+||c||} \right)$$

Equal width bins divide range $[-1, 1]$ to record the frequencies of different curvature $c'$ values. Frequencies of curvature values are normalized. Curvature bins are aligned to produce meaningful comparisons, e.g. Frequencies of planar surfaces ($c = 0$) always align to the same bin. Figure 6.4 shows a sample of CAD models and the corresponding descriptors. Cast-then-machined parts shows a larger variation in terms of any curvature statistics. Note that one might worry the center bar, $c' = 0$ often dominates the statistics in the histogram, this zero bar shows the proportion of (intrinsically, for $\kappa_2$, $K$) planar surfaces, the significance of this feature will be automatically determined by the machine learning classifier through the training process.
6.1.5 Invariance of Curvature Statistics

Curvature statistics is a rotational invariant feature set, because the curvature values are local measures on the surface of models, rotating a surface does not affect the curvature measure. Frequencies of the curvature measure are normalized to enable comparison of models of different number of vertices, and different mesh resolutions. However, the curvature statistics varies with the scale of model. Since type of surfaces are aligned in our formulation, the same model in different scales affects the curvature statistics. Although scale invariance is generally desirable in the context of shape matching, consider manufac-
turing two simple rings of the with different sizes. The smaller one consist of a hole of diameter 1 cm whereas the larger hole diameter is 50 cm. It is more likely that the smaller one is constructed by drilling, while the large one is done by a casting process. Nevertheless, scale invariance can be achieved by simple rescaling the model to a fixed volume, for example an unit cube.

6.2 Classifying Curvature Descriptors

Classification of prismatic-machined and cast-then-machined processes can be learned by different classifiers. SVMs with a non-linear kernel function is the choice of classifier in this thesis research. In addition to the use of SVMs, commonly used nearest neighbor classifier is also provided as a comparison evaluation.

This research choose to apply supervised machine learning techniques to find the classification of different manufacturing processes with curvature descriptors. Given that all parts share some subset of curvature features statistics due to machining operations but casting procedure generates surfaces with curvature values that signifies the cast-then-machined process. It is intuitive that applying a classifier that would weight attributes differently should produce better performance. SVMs possess this desirable weighting property, weights for the projected features are learned through an explicit training process. Distance based classifiers such as kNN and unsupervised machine learning by clustering the models often consider equal weighting of attributes during distance computation. SVMs learning is more appropriate to adapt to the characteristic of this classification problem.

The overall machine learning procedure of classifying prismatic machined and cast-then-machined CAD models can be summarized as follow.

1. Compute per vertex curvature and construct curvature shape descriptors for training models.

2. Training:
(a) SVMs: Train a classifier using SVMs with the classifications and corresponding curvature shape descriptors of example.

(b) Nearest Neighbors: No training is required.

3. Compute per vertex curvature and construct shape descriptors for a query model.

4. Classify the query model shape descriptors:

   (a) SVMs: Feed the query model shape descriptors to the trained SVMs classifier to get a classification.

   (b) Nearest Neighbors: Return the classification of the nearest example model(s).

### 6.2.1 Support Vector Machines Classification

The separation in between prismatic-machined and cast-then-machined processes can be learned by feeding example curvature shape descriptors and their corresponding categories to SVMs. Although, it is unclear about the linear separability of the curvature statistics descriptors from different manufacturing processes. The SVMs framework allows descriptors to be non-linearly projected into a high dimensional feature space for the algorithm to find a linear separating margin. This may result in a non-linear separating margin in the low dimensional input space.

In Section 2.3.2, the learning algorithm takes advantage of $L_D$ only depending on dot products of $x_i \cdot x_j$, rather than direct protection of examples to a high dimensional feature space, $x \rightarrow \Phi(x_i)$, it replaces the dot product computation with a kernel function, $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$, that computes the dot products of the projected examples in constant time. This approach allows the SVMs to find non-linear classifications without increasing its complexity.
Kernel functions need to satisfy the Mercer condition:

\[ K(x_i, x_j) > 0 \]

This ensures the matrix is positive semi-definite in the quadratic programming formulation, therefore a solution must exist.

This research experimented with different common kernel functions for SVMs learning.

- **High Degree Polynomial** — \((\gamma (x_i \cdot x_j) + r)^d, \gamma > 0\)
- **Radial Basis (Gaussian) Function** — \(e^{-\gamma ||x_i - x_j||}, \gamma > 0\)
- **Sigmoid Function** — \(\tanh(\gamma (x_i \cdot x_j) + r), \gamma > 0\)

\(\gamma, r\) and \(d\) are free parameters of the kernel functions.

For the radial basis function kernel, \(N_s\) (number of support vectors) Gaussian functions are centered at the selected support vectors. The weights and threshold for each Gaussian is determined during SVMs training. Sigmoid function models a specific kind of two layers neural network. the first layer consist of \(N_s\) sets of weights, each set consisting of \(D\) (data dimension) weights. The second layer consist of \(N_s\) weights. Evaluation takes a weighted sum of sigmoid functions.

Free parameters of kernel functions and the penalty parameter \(C\), mentioned in Section 2.3.2, needs to be determined by a model selection process. This research follows the advice of Lin [28] on performing an exhaustive grid search and cross validation for optimal parameters. Parameters are estimated by trying a growing sequence. Cross validation divides training data into \(n\) folds, a classifier is trained on \(n - 1\) folds for classifying the remaining fold. Accuracies are averaged across different classifications to predict testing performance. Parameters with the highest cross validation performance are selected for training the final classifier.
6.2.2 Nearest Neighbor Classifier

Apart from SVMs, common nearest neighbor approach is presented as a comparison for classifying curvature statistics descriptors. Nearest neighbor classifier returns the classification of the query model’s closest example model(s). Minkowski $L_2$, Euclidean, distance is used to determine the distance in between curvature shape descriptors. $L_2$ distance between models $t$ and $q$ is computed as:

$$L_2(t, q) = \left( \sum ||t_i - q_i||^2 \right)^{\frac{1}{2}}$$

The closest examples model(s) of the query model determines the returned classification. Nearest neighbor classifier is included in the evaluation as an alternative to SVMs and provides a performance comparison in between different classifiers along the use of curvature statistics descriptors.

6.3 Experimental Results

Curvature statistics descriptor and SVMs is applied to learn and classify a dataset of 100 hand classified prismatic-machined or cast-then-machined CAD models. Experiments were conducted using a set of the mechanical part data sampled from the National Design Repository. All models can be retrieved at http://www.designrepository.org/datasets/. Implementation of the curvature estimation and SVMs were provided by Trimesh2 [44] and libsvm [12].

6.3.1 Experimental Setup

Experiment was repeated perform on randomly selected training models to confirm the robustness classification. In addition to SVMs, $k$NN is also performed to provide a comparison in between SVMs and shape matching popular $k$NN classifier.
Half of the manufacturing classified model dataset were split in to two random halves for training and testing. Similar to the weighted shape descriptors experiment, this experiment was repeatedly performed to confirm the robustness. High fidelity mesh representation of CAD models were prepared by faceting the ACIS SAT solid models using Geomagic Studio. On average 150,000 triangles were used to approximate the surfaces per solid models to ensure curvature computation is accurate.

Curvature for every vertex on training example were estimated to construct the curvature shape descriptor. Using C++ implementation of trimesh2. Curvature histograms used in this evaluation consists of 100 bins.

A C++ support vector machines implementation was provided by libsvm. The two classes of models were feed into the machine learner as some 100 dimension curvature histogram vectors. All experiments were performed on Linux platform using a single 1.5GHz AMD Opteron processor with 1 GB of memory.

6.3.2 Performance

The proposed curvature estimation and SVMs learning procedure took 5 minutes to build a classifier from half of the labeled dataset. Constructing curvature shape descriptor and classify all the query models using SVMs classifier took five seconds. Estimating per vertex principle curvatures and construct curvature histograms took approximately two secs per model with 100,000 triangles. The SVMs training process took 0.2–0.5 minute depending on the example model set. Most of the training time was spent on model selection for optimal parameters using grid search and cross validation as described in Section 6.2.1.

6.3.3 Curvature and Support Vector Machines Classification

Support vector machines classification of curvature shape descriptors along with different non-linear kernel functions (radial basis function, polynomial, and sigmoid functions)
was experimented. $\kappa_1$, $\kappa_2$, mean curvature, $H$, and Gaussian curvature, $K$, histograms were computed for this evaluation. Each experiment was repeatedly performed for 40 times to confirm the robustness of the approach. The objective was to verify that the system could stably learn the classification by randomly selecting sets of training examples then accurately classify the non-training parts. Average and maximum accuracies are provided to illustrate the performance. Higher maximum and average accuracies shows better classification rate, lower standard deviation shows the classifier being more stable. Table 6.1 shows a summary of the results.

Table 6.1: SVMs and curvature statistics classification accuracy.

(a) Polynomial Kernel Function.

<table>
<thead>
<tr>
<th></th>
<th>$\kappa_1$</th>
<th>$\kappa_2$</th>
<th>$H$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum%</td>
<td>81%</td>
<td><strong>87%</strong></td>
<td>79%</td>
<td>74%</td>
</tr>
<tr>
<td>Average%</td>
<td>69.7%</td>
<td>75.5%</td>
<td>70.95%</td>
<td>65.2%</td>
</tr>
<tr>
<td>Stdev %</td>
<td>6.57%</td>
<td>4.81%</td>
<td>5.39%</td>
<td>6.12%</td>
</tr>
</tbody>
</table>

(b) Radial Basis Kernel Function.

<table>
<thead>
<tr>
<th></th>
<th>$\kappa_1$</th>
<th>$\kappa_2$</th>
<th>$H$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum%</td>
<td>85%</td>
<td><strong>87%</strong></td>
<td><strong>87%</strong></td>
<td>76%</td>
</tr>
<tr>
<td>Average%</td>
<td>67.85%</td>
<td><strong>77.1%</strong></td>
<td>71.15%</td>
<td>59.2%</td>
</tr>
<tr>
<td>Stdev %</td>
<td>8.13%</td>
<td><strong>4.75%</strong></td>
<td>7.9%</td>
<td>6.7%</td>
</tr>
</tbody>
</table>

(c) Sigmoid Kernel Function.

<table>
<thead>
<tr>
<th></th>
<th>$\kappa_1$</th>
<th>$\kappa_2$</th>
<th>$H$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum%</td>
<td><strong>87%</strong></td>
<td>85%</td>
<td>81%</td>
<td>72%</td>
</tr>
<tr>
<td>Average%</td>
<td>69.7%</td>
<td>75%</td>
<td>67.5%</td>
<td>59.35%</td>
</tr>
<tr>
<td>Stdev %</td>
<td>7.41%</td>
<td>5.87%</td>
<td>7.24%</td>
<td>7.28%</td>
</tr>
</tbody>
</table>

Using minimum curvature, $\kappa_2$, features along with radial basis function SVMs produced the highest classification rate of 87% with an average of 77.1% and lowest standard deviation of 4.75%. These statistics show this combination produced a more accurate classification with a higher stability. This is a 20–30% increase over existing 3D shape matching
algorithms (Table 5.1, accuracy 53–57%). \( \kappa_2 \) performed the best in this classification because most curved surfaces were generated by volume removal operations, such as holes, slots, and pockets, these machining features have zero minimum curvature. Therefore, \( \kappa_2 \) facilitated classification by minimizing the variety of curvature statistics for prismatic-machined parts.

The results of experiments shows radial basis function performed slightly better than polynomial or sigmoid kernel functions. The classification accuracies (>75%) of minimum curvature were always higher better than other curvature statistics despite which different kernel function was used. Maximum and mean curvature performed similarly with average 70% accuracy. Gaussian curvature performed the worst (average 60%) in the experiments, this shows Gaussian curvature statistics did not properly separate the CAD models according to the prismatic-machined and cast-then-machined manufacturing classifications.

### 6.3.4 Curvature and Nearest Neighbor Classification

Nearest neighbor classifier was applied in place of non-linear SVMs to provide a comparison of the performance of SVMs and popular 3D shape matching nearest neighbor classification using the proposed curvature shape descriptor. The classification of the query model is determined by its nearest example neighbor. The experimental setup was the same as the SVMs experiments. Table 6.2 shows a summary of the results.

<table>
<thead>
<tr>
<th></th>
<th>( \kappa_1 )</th>
<th>( \kappa_2 )</th>
<th>( H )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum%</td>
<td>69%</td>
<td>55%</td>
<td>59%</td>
<td>58%</td>
</tr>
<tr>
<td>Average%</td>
<td>54.59%</td>
<td>48.62%</td>
<td>51.63%</td>
<td>52.1%</td>
</tr>
<tr>
<td>Stdev%</td>
<td>6.66%</td>
<td>3.89%</td>
<td>5.19%</td>
<td>4.62%</td>
</tr>
</tbody>
</table>
Nearest neighbor classification did not perform satisfactorily with curvature shape descriptors. Only 50–60% of models were classified correctly for each run of the experiment. In contrast, radial basis function SVMs and minimum curvature performed 20–30% better. This result demonstrated the SVMs classifiers found significantly better separation among the curvature shape descriptors with respect to the targeted classifications. This experiment also shows the choice of classifiers could considerably affect the performance in CAD classification, although, by no means that SVMs are always superior to \textit{k}NN classification. It is important to search for a combination of features and classifier that produces the best accuracy.

6.4 Summary

This chapter presented an approach to classify CAD models according to prismatic-machined and cast-then-machined manufacturing processes. Surface curvature and support vector machines were used to learn and perform classification. To summarize the experimental results from Tables 5.1, 6.1, and 6.2, the combination of curvature descriptor and non-linear support vector machines demonstrated a considerable improvement over recent shape descriptors and near neighbor classification on prismatic-machined and cast-then-machined manufacturing CAD models classifications.
7. Conclusions and Future Work

To conclude, this thesis research presented a framework for classifying CAD models using a pattern classification model. 3D CAD and shape matching techniques are modeled in a two-step procedure, (1) feature extraction and (2) classification. Two example approaches are presented to classify polyhedral representation of CAD models according to different engineering interests (appearance and manufacturing). First approach described how to combine appearance oriented shape descriptors using a weighting formulation and estimate the appropriate weights using a training procedure and classify CAD models using nearest neighbor classifier. Second approach was aimed at classifying prismatic-machined and cast-then-machined parts, statistics of surface curvature and non-linear support vector machines were used to classify the models.

In the future, it is important to study the correlations in between features and classification targets in the interest of CAD/CAM. It is necessary to analyze which classifier’s model fits better to the features of CAD/CAM classification, for establishing an error bounding model on classifying engineering artifacts. Further, to help direct comparisons of different matching and classification techniques, standard procedures and dataset should be proposed to evaluate the performance of combinations of features, classifier, and specific targeted classification.

7.1 Contributions

The contributions of this research include: (1) Modeling 3D shape matching in a pattern classification framework (2) Weight learning for shape descriptors and nearest neighbor shape classification of CAD models. (3) Classification of prismatic-machined and cast-then-machined manufacturing process with surface curvature and support vector machines.
The matching 3D models can be separated into two sub-procedures (1) feature extraction, and (2) classification. Appearance and shape matching were improved by combining recent shape feature using a weighted learning approach. New features and classifiers, surface curvature and SVMs, are introduced to perform manufacturing process classification.

Weight learning across shape descriptors attempted to take advantage of using multiple feature sets to fit a particular classification target. This approach combined existing shape descriptors from different recent shape matching research. The contribution of this approach proposed a simple technique to join multiple shape descriptors for generating a better shape classification system at a minimal cost. The weight parameters are learn from examples to bring different shape descriptors closer to a targeted classification using a linear combination.

Surface curvature and SVMs are shown to be relevant for manufacturing process classification. By observing the models and classification criteria, different types of surfaces are generated by different manufacturing techniques, this makes surface curvature relevant to distinguish the processes according to difference of the surfaces. Non-linear support vector machines were then shown to discover the separation of curvature descriptor with respect to process classification. Experimental results showed that radial basis function support vector machines performed the best and demonstrated a 20–30% improvement over classification using recent shape descriptors.

7.2 Future Work

Research in this thesis on CAD classification can be improved in three areas: (1) Better combination of shape descriptors, (2) Extraction of new features and classifiers, and (3) Standard classification procedure and datasets.

Shape descriptors are combined with a weighted linear combination in this thesis research. Each descriptor is considered as one unit of attributes. Future work may consider
non-linear combination of shape descriptors or weighting each feature from the shape descriptors using automatic feature selection or variable ranking techniques.

The process classification experiments showed popular shape descriptors and nearest neighbor classifier did not performed satisfactorily on some non-shape related engineering target classification. This implies the importance of target oriented features and classifiers, there is a need to better the understanding of relationships between CAD classifications and geometric attributes. Engineering domain knowledge can play a more important in choosing features to ensure descriptors are relevant to the target CAD/CAM classification.

To facilitate the evaluation and comparison of research in information retrieval or classification in CAD/CAM domain. Open standards of classification procedure and datasets should be constructed to allow direct comparison and evaluation across different approaches.
Bibliography


