Feature Extraction and Analysis for 3D Point Cloud-Based Object Recognition

by

Seyed Khatamian Oskooei

(Under the Direction of Hamid R. Arabnia)

Abstract

Object recognition is one of the most problematic challenges in computer vision, robotics, autonomous agents and others. Image Processing and Machine Learning collaborate to solve this problem from various perspectives. Most systems operate on 2D projections to recognize 3D objects. The author proposes a novel methodology that performs on 3D point clouds to extract signatures and to recognize possible existing objects. 3D scanning devices can produce 3D point cloud of any object to collect a dataset; PDA devices such as Google Tango and scanners associated with 3D printers provide the scanning ability. Our objective is to build a system that recognizes objects utilizing properties of 3D point clouds, to prove such a system exists and to address some of the shortcomings in the commonly-used approaches. Moreover, some methods measure the features learnability and the impacts of the properties to analyze the proposed attributes—geometrical or topological—and to assess the recognition procedure and to emphasize the proof of concept.

Index words: Computer Vision, 3D Point Cloud Processing, Object Recognition, Machine Learning, Data Analysis

Feature Extraction and Analysis for 3D Point Cloud-Based Object Recognition

by

Seyed Khatamian Oskooei

B.Sc., Iran University of Science and Technology, Iran, 2011

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2016

Feature Extraction and Analysis for 3D Point Cloud-Based Object Recognition

by

Seyed Khatamian Oskooei

Approved:

Major Professor: Hamid R. Arabnia

Committee: Walter D. Potter
Lakshmish Ramaswamy
Rabia Jafri Ali

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
Aug 2016

ACKNOWLEDGMENTS

I would like to thank Prof. Hamid R. Arabnia for his valuable advice, support, passion and encouragement. I am also thankful for my wise and supportive members of committee; Prof. Walter D. Potter for his inspiration, advice and help. Prof. Lakshmish Ramaswamy who has helped me and guided me to recognize my path and step into that and Dr. Rabia Jafri for being always helpful, motivational and unconditionally supportive. I would like to thank my family and friends for their support and kindness, as well.

I would also thank and dedicate this work to a special person in my life, anonymously (D), who helped me, supported me, and stayed beside me along this long and difficult journey.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Object recognition is one of the most problematic challenges in computer vision, robotics, autonomous agents and others. Image Processing and Machine Learning collaborate to solve this problem from various perspectives. This chapter presents an overview of existing strategies that address the object detection/recognition problem, our motivations to tackle this issue from a different perspective, and the big picture of the system architecture.

1.1 Background and Motivation

More than often, scientists use the terms "object recognition" and "object detection" interchangeably, although they pursue different objectives. Object detection means to answer a binary (YES/NO) question; thus, a detective system tends to project if at least an object exists in a given input—most frequently a captured image from a scene—without considering the actual meaning of the object. Obstacle avoidance and path planning in robotics in addition to collision prevention in the new technologies of the regular and self–driving cars—to increase the passengers' safety—and surveillance systems exemplify the object detection concept [80, 24, 109, 42, 121, 124]. As opposed to object detection, object recognition leads to distinguish the objects by their names, categories or any other particular definitions; in other words, object recognition enhances object detection by adding symantics to the objects

(e.g. assistant robotics, visual perceptual applications, security checking and authorization techniques)[106]. Object recognition systems work based on single or multi object classification. In the former, one and only one individual object exists in the input data, whilst in the latter several objects may appear in the scene and the system recognizes the objects within a reasonable range of accuracy.

Almost all the current object recognition strategies work based on (RGB) images[106]. Some studies perform object recognition on a single captured image, which is the 2D projection of a camera's 3D view on its focal plane. Image Processing methods process the 2D projection (image) to segment the image into the concepts (objects) and to describe the objects with appearance characteristics or inherent features. Either a learning model trained by similar concepts (Machine Learning) recognizes the objects or a database of real objects is queried to retrieve the analogous concepts [105, 59, 56, 107, 94]—in both single and multi RGB–based object recognition systems. This strategy works only where the 2D projection and the 3D view of an object from all different angles match together. Therefore, if the 2D projection of an object from the front view differs from its corresponding side view, this approach fails because it classifies the object based on a single 2D view from the capturing angle.

Other techniques utilize a sequence of consecutive captured images from different angles to extract and to project the depth data on the front view (source image) for the purpose of recognition. These systems use the so called two and a half dimension processing approach known as RGB–D [24]—RGB as the front view image and D as the projected depth data—and the same procedure of RGB–based methods. The two dimensions belong to the front view image and the half dimension corresponds to the extracted and projected depth data. The projected depth data add half of a dimension, because the depth information derive from processing the sequence of images (2D data) and projection of the resultants into a single image (front view), which is a 2–dimensional entity. This strategy works but utilizing missing dimensions may improve the performance of the recognition system. In addition, RGB–D

approaches involve parameters like capturing positions and orientations in order to measure the depth data.

3D point cloud of an object materializes in the form of a set of points in 3D space, lying on the surface of the object. Even though the 3D point cloud excludes any information about the texture and color of the object surface, it is a full 3–dimensional piece of data. Looking into the object recognition problem through the 3D point cloud processing window establishes the foundation of this research. This view port will no longer consider any RGB data or 2D projection of depth data.

The proposed methodology—3D point cloud–based feature extraction and object recognition—eliminats the existing ambiguity in the recognition systems based on 2D projections, utilizes the missing dimensions in the recognition systems with regard to RGB–D approaches, and facilitates the depth data measurement by removing the parameters like capturing positions and orientations of cameras. This research focuses on 3D point cloud representation, adequate descriptors assignment to 3D point clouds, and measurement of the learnability of the provided descriptors. Moreover, the ultimate goal of this study is to find out how 3D model or 3D point cloud processing may affect the object recognition procedure.

Apart from introducing a novel methodology for object recognition, our contribution includes collecting a dataset of 3D point cloud of objects, proposing features and feature vector generation methods from 3D point clouds, and proving the learnability and usability of the features by finding the best object classifier and measuring the impact of each feature in the proposed vectors with intuitive strategies.

1.2 System Overview

Most object recognition systems work based on a three–phase architecture, shown in Figure 1.1. This architecture appears in the form of a pipeline structure with three phases: (1) Data Collection, (2) Feature Extraction and (3) Learning.

First and foremost, in the data collection phase, a set of objects in a specific format forms

the dataset. Secondly, the system processes the dataset to generate object descriptors, in the feature extraction phase. Lastly, the learning phase trains a model based on the second phase extracted features, for recognition purposes.

Figure 1.1: Three–phase object recognition procedure

In this study, we focus on a single object recognition system; thus, each entitiy in the dataset contains only an individual object. Therefore, a collection of 3D point clouds of singular objects forms the dataset. The system performs some feature extraction methods on each object in the dataset (described in Chapter 4) to produce two separate descriptors—to measure and compare the learnability and descriptive power of same features described in different ways. The system trains multiple learning models—to compare different models and to conduct to the best classifier—based on the extracted descriptors. Finally, according to the best selected classifier, we provide three methods to measure the influence of the proposed attributes, to compare each individual feature with others, and to conclude the most effective feature in each descriptor.

1.3 Literature Review (Chapter 2)

Chapter 2 presents a comprehensive literature review about Image Processing, Cognitive Systems, Machine Learning, 3D models and 3D point cloud surface reconstruction methods that are involved in this research. To brevity, Chapter 2 is about how Machine Learning uses Image processing to build a recognition system and how research trends utilize 3D models and 3D point cloud processing methods for cognitive purposes. Chapter 2 ends up with a short discussion which implicates what this dissertation aims to prove and how this goal structures the whole document.

1.4 Data Collection Procedure and Resources (Chapter 3)

Different resources provide 3D point clouds of objects. Chapter 3 presents a brief introduction about these resources, the procedure of generating point clouds using a 3D scanner (Google Tango), and an encountering issue in this process. Moreover, Chapter 3 provides details about the collected dataset using available resources. A short discussion—about a computationally expensive problem in data collection from 3D scanner devices—and future potential works in this area draw Chapter 3 to its close.

1.5 Feature Extraction and Selection Criteria (Chapter 4)

Chapter 4 studies feature extraction—as the most fundamental phase in the three–phase architecture—discusses feature selection criteria, and defines the proposed geometrical and topological features. Since the dataset contains of a collection of point sets, feature selection and extraction phase of the recognition procedure concentrate on the geometric properties of the point sets. Chapter 4 winds up with a brief discussion and potential future works.

1.6 Learnability Examination and Feature Impacts (Chapter 5)

A recognition system is useful, only if the extracted features are learnable by a model. Chapter 5 revolves around the third phase of the recognition process. It discusses which learning model classifies unseen objects more accurately, how much a feature in a descriptor impacts the learning process, and which descriptor owns the highest discriminative power. A short discussion and explanation of possible future studies in the learning phase of the recognition procedure bring Chapter 5 to its end.

1.7 Contributions, Algorithms, and Findings

In this research, we contribute to: (1) collect a dataset of point clouds of various objects using multiple resources, (2) utilize mathematical and approximation approaches to extract inherent properties of objects from their point clouds and to provide discriminative feature vectors for objects, (3) prove the learnability of the proposed feature vectors employing Machine Learning algorithms, and (4) provide intuitive methods for measuring the impact of each attribute in the feature vectors.

In order to extract the inherent properties of the objects and generate representative feature vectors, we utilize mathematical approaches and approximation strategies such as Gaussian, Fourier, Sinusoid, and Polynomial. Machine Learning algorithms, from basics including decision trees, random forests, and neural networks; to advanced based on ensemble learning models such as bagging, boosting, and stacking, prove the learnability of these feature vectors. Last but not least, Intuitive analytical approaches measure the impact and effectiveness of each attribute in the feature vectors.

Our important findings include: (1) the restrictions in data collection phase of this study such as, complexity of point cloud generation using 3D laser scanners, constraints in dataset size, difficulties in modeling uniform 3D point cloud from various resources, and limits in making a complete dataset which includes any type of object; (2) the benefits of geometric and topological properties and the process of normalization like, making the features scale-, rotation-, and translation-invariant, efficient computation, and representing objects based on the points on their surfaces; (3) the advantages of using ensemble learning models for precise recognition results; and (4) the influence of attribute representation and the interesting effectiveness of each attributes in the recognition procedure.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Object recognition has become a well–studied field of science which combines Image Processing and Machine Learning. 3D Object recognition replaces Image Processing with 3D model analysis. In this study, the lowest level of informative 3D models (3D point set of objects/models) is used for the purpose of recognition. For the sake of better realization of the concept, different subjects and field of studies including Image Processing, Machine Learning and 3D point cloud surface reconstruction were reviewed. The following sections discuss Image Processing and recognition, Machine Learning and recognition, in addition to 3D point cloud surface reconstruction due to their involvement in 3D object recognition field of study.

2.2 Image Processing and Recognition

Image Processing and Computer Vision have been used and are still in use in the area of intelligent decision making and vision–based understanding/learning. Autonomous agents—in the context of understanding (a) concept(s) with the aid of image/vision—need to employ Image Processing and Computer Vision with respect to Machine Learning in order to

get a sense of the image or the video that is used to represent the subject. Pattern recognition, object recognition, content–based image classification, localization and mapping, image segmentation and many more others are some aspects of Image Processing and Computer Vision that come into the consideration when we are looking for solutions to make machines with better understanding of visual entities.

Machine Learning plays a remarkable role in any of these aspects. Image Processing is used to extract features and Machine Learning is employed to understand and learn the concept from the features. Concept can be an object, an action or anything else. Deep Learning as a special part in Machine Learning is considered of a solution to learn deeply from entities and their characteristics, with hierarchical relationships, to provide better understandings of things for machines.

2.2.1 Applications

Search engines are autonomous agents that receive requested queries and retrieve corresponding results. Search engines for image retrieval are divided into two categories: (1) text–based querying engines and (2) image–based querying engines[53]. In both categories, Image Processing is an inevitable part of the engine. A text–based querying engine utilizes the text query to understand the concept(s), to find the concept(s) in image databases, and to retrieve images that are highly related to the concept(s)[53]. An image–based querying engine uses an image as a query to export the concept(s) from the query image, to match the concept(s) with the images in image databases, and to retrieve the images that are contextually related to the query image[53]. Content–based image classification, a broad field of study, focuses on how to extract the content of an image, how to know an image is a representation of what, how objects in an image are related to each other, and what concepts or objects are present in an image[53].

Localization and mapping may utilize Image Processing in the field of robotics and intelligent agents in order to find relative location of a robot or an agent in its operating

environment, to create a map of the environment, and even to position the surrounding objects[113]. Micro aerial vehicles, unmanned autonomous vehicles and autonomous robots are some of the applications of localization and mapping field of study where Image Processing may be used to enhance the capabilities of the system, and to get more accurate results for precise decision making[113].

Identity verification (Biometrics) is another aspect in Image Processing which tries to identify a person through the appearance of the person[102]. Iris analysis, face recognition, and fingerprint matching are some of the applications in the area of identity verification that may use Image Processing to create an intelligent agent that receives the required information about the subject, matches the description with the registration, and validates the authority of a person[102].

In addition, Image Processing has an aspect in neuroscience and medical brain image analysis[113]. To understand brain imaging and neurological reactions in human brain, Image Processing is used to analyze brain images, to recognize neurological patterns, and to understand how to distinguish different reactions or brain issues[113]. Medical diagnosis is being done through intelligent pattern recognition on brain imaging and neurological modeling systems[113].

Object recognition focuses on how to make a machine to learn from visual entities and to distinguish general objects—as visional sense of human brain works. Face recognition, human body recognition, and activity recognition are some other aspects of Image Processing that are utilized in autonomous agents in order to enhance the agents' understanding of visual entities[102, 109, 42, 121, 124].

2.2.2 Image Understanding

What understanding means? Literally, understanding is perceiving the meaning, significance, explanation or cause of something. So, image understanding means finding the definition of an image. "It has been assumed that image understanding is a human spe-

cific ability to explain and interpret content of image on the basis of image perception, object recognition, knowledge, experience, reasoning, culture, association, context analysis, etc."[113] Image understanding in the context of machine ability needs domain knowledge, context consideration, linguistic representation of contents, reasoning and learning[82, 55].

An image can be understood from its appearance information or context information[82, 55]. Properties like size, color, texture, shape or edge are appearance features of an image. Mathematical equations formulate and quantize these features to describe an image. Appearance features identify object classes in an image to some certain extent, however they are unable to describe meaning of an image and nor the objects or the contexts present in an image. Context information, interaction of objects in an image or global statistic knowledge about an image, may remove ambiguity in object recognition due to reasoning by only appearance information[82, 55].

Contextual information of an image is the semantic of objects and the interactions between the objects; i.e., contextual features are produced from the properties other than the appearance of an image. Context features—with respect to the information that they provide—are classified into three different types: (1) Semantic, (2) Spatial and (3) Scale[55].

Semantic Context: Human vision system and the recognition task in our brain work based on the expectation probability of an object in a scene[55]. Semantic context is the probability that an object can be seen in an image or in a higher level the likelihood of appearance of an object in existence of another object in a particular scene[55]. Semantic context information may have three different sources: pre–defined rules, implicit pixel–level relations and co–occurrence probability of two objects in a scene[55].

Spatial Context: The probability of observing an object in a specific position of another object is considered as spatial feature—similar to semantic context with some additional information[55]. Spatial features may decrease the amount of ambiguity in a recognition task if an object in a scene has a unique and general meaning[55]. Moreover, proper spatial relations between objects in a scene may reduce the error rate in a recognition task[55]. Spatial

features can be stored in a graph–like structure in which objects are the nodes and the relations are the edges of the graph[55]. As an example, sky in a scene can be a reference object and the positioning of the other objects can be recognized with respect to this reference object (reducing the amount of ambiguity by considering an object in a scene with a specific meaning)[55]. Furthermore, sky is always at the top of a scene and observing trees right under the sky is highly probable (decreasing error with proper spatial relations)[55].

Scale Context: The probability of appearing an object in certain size with respect to the other objects is the scale feature of that object[55]. Prior knowledge about the size of an object in a scene improves the recognition task and reduces required multi–scale evaluation[55]. Because the scale context is a relational feature, scale features demand the presence of at least two objects[55]. For instance, "a sofa is bigger than a lamp" statement indicates a scale context for sofa with respect to lamp[55]. Presumably, scale context is the hardest context to be obtained from a scene as it needs higher level of relational information between objects of the scene[55]. Semantic context is used implicitly in spatial and scale contexts, as they need to know how objects are related to each other in a scene[55].

Human perceptual system works in a hierarchical mechanism. First it sees an image as a whole then it focuses on a particular section of the scene to obtain further information[55]. This hierarchical mechanism defines contextual levels of information: Global context explains an image as a whole and Local context sees an object and its neighboring in that scene[55].

Context information can be obtained from the interactions of the entities in a scene[55]. When it comes to local context information, these interactions are pixel–to–pixel, region–to–region, or object–to–object interactions[55]. On the other hand, object–to–scene interactions are global contextual interactions[55]. Pixel level interactions are based on the assumption that having a pixel with a same label of its neighborhood's is more probable except in the presence of discontinuities—boundary detection algorithms work with respect to this pixel–to–pixel interaction[55]. Regional interactions describe the interactions among mean-

ingful and separable neighboring regions of a scene[55]. Object interaction—as the most important one—defines interactions among actual objects of a scene[55]. Object–to–scene interaction—as the only global interaction—is the most extensive one due to considering the relationship between actual objects and the entire scene[55].

2.3 Recognition and Machine Learning

Machine Learning is an inevitable part of any image understanding method. Machines need to have learning ability to obtain the required information from the subject, to perceive, to reason and to act accordingly[113]. Classification and clustering are the two ways of learning; the former is considered as supervised learning and the latter as unsupervised[91]. Traditional classifiers in machine learning such as Decision Trees, Neural Networks, Bayesian Networks, k–Nearest Neighbors, Linear Regressions and Support Vector Machines can be used to understand an image from its features by proper training datasets[91]. Clustering methods such as k–Means, as one of the well–known ones, also can be employed to realize image content by its belonging cluster's properties[91]. Deep Learning, mostly considered as the extension to Neural Networks, becomes one of the newest machine learning mechanisms for image understanding, which utilizes hierarchical and structured learning[108].

If a system tends to learn all the image features—including appearance and context that we discussed earlier in this section (Image Understanding)—to understand different scenes, as human beings do, the number of features system must learn will be enormous[113]. That makes reducing the dimension of the features in the most Machine Learning methods an important preprocessing step. Large number of features or long feature vector in Machine Learning is considered as curse of dimensionality which may cause slower training time in the most traditional Machine Learning algorithms[91]. Deep Machine Learning algorithms become general approaches which allow us to build a system that learns from a high dimension of features by utilizing multi–layer and hierarchical processing information mechanism[108].

Deep Learning has defined with analogous but different statements. Deep Learning is "a

class of Machine Learning techniques that exploit many layers of non–linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification."[34] In another phrase, "Deep Learning is a set of algorithms in Machine Learning that attempt to learn in multiple levels, corresponding to different levels of abstraction. It typically uses artificial neural networks. The levels of concepts, where higher–level concepts are defined from lower–level ones, and the same lower–level concepts can help to define many higher–level concepts."[1]

To breviry, Deep Learning is a set of Machine Learning techniques that employ multiple layers of processing units in order to interpret hierarchically related input data. These techniques depending upon their intended usage can be classified into three major groups[108].

2.3.1 Unsupervised/Generative Deep Learning Networks

This group of techniques tends to extract high level correlation of the input data for pattern analysis or sampling purposes when there is no label information about the data[108]. Restricted Boltzmann Machines or RBMs, Deep Belief Networks or DBNs and Deep Boltzmann Machines or DBMs are some of the generative deep networks that can be used for sampling purposes[108]. Sum–Product Networks or SPNs and Recurrent Neural Networks or RNNs are some of the deep networks for unsupervised learning[108].

A DBM is a special type of general Boltzmann Machine (BM)—a complex multi–layer Neural Network with symmetrical connections between processing units of hidden layers and the visible ones[108]. Unlike BMs, DBMs discard the connections between the units in a same layer[108]. In many cases of Neural Networks, there exist two visible layers: input and output; and at least one intermediate layer as hidden layer of the network, however in unsupervised/generative networks the output layer is also a hidden layer[108]. General BMs are very complex and though very slow in training phase[108]. In a DBM, each hidden layer extracts high level correlation between the lower layer processing units[108]. The hierarchical correlation extraction from the input layer through the multiple hidden layers ends up at the

last hidden layer to get the desired information from the input data[108]. This hierarchical structure of DBMs makes them potentially capable of exploiting internal representations of complex data, such as what is aimed in object recognition and speech recognition[108].

If the number of hidden layers in a DBM is reduced to one, the network becomes a Restricted Boltzmann Machine or an RBM[108]. Similar to DBMs, no connection exists between the units in the only hidden layer and the units in the only visible layer of RBMs[108]. By combining multiple RBMs, where each hidden layer of the lower RBM is considered as the visible layer of the higher RBM, Deep Belief Networks or DBNs are obtained[108].

2.3.2 Supervised Deep Learning Networks

Supervised deep networks intend to directly classify the input data with the available posterior knowledge about target classes for discriminative purposes[91]. Deep networks such as Deep Neural Networks or DNNs, Convolutional Neural Networks or CNNs and Deep Stacking Networks or DSNs are some of the networks in this category[108].

A DNN in actuality is an Artificial Neural Network (ANN) with multiple hidden layers between the input and output layers[108]. Inreasing the number of the hidden layers improves the discriminative power of the network[108]. Learning methods adjust the weight values of the connections between the units of the layers; for which the well–known method is feed–forward/back–propagation algorithm[108, 91]. A DSN is constructed by stacking shallow networks such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), one on top of the other[108]. A CNN is also an ANN consisting of modules in which each module has a convolutional and a pooling layer[108]. The modules are stacked on top of each other to build a CNN[108]. In each module, the convolutional layer generates weights and the pooling layer subsamples the output weights of the convolutional layer to reduce the amount of data pouring into the upper layer[108]. Combination of convolutional layer and pooling layer gives a CNN some invariance properties such as translation invariance which is very useful in computer vision and object recognition[108].

2.3.3 Hybrid Deep Learning Networks

This group of networks aim to perform discrimination, more often assisted by the outcomes of unsupervised or generative networks—the output of a generative model can be used as an initial parameter set for a supervised deep network[108]. Some of the hybrid models published in the literature are DBN–DNN and RBM–DNN[108]. Training the supportive networks in most cases is called "pre–training" step[108]. Pre–training step initializes the supervised network which improves the discriminative power of the network in comparison with randomly initialized network[**?**]. The supportive network can be replaced with RBM or any appropriate unsupervised network[108]. Pre–trained CNN using DBN also also helps to increase the discriminative power of the randomly initiated network[108].

In some literature, Deep Learning techniques are divided into two classes: (1) discriminative models and (2) generative or unsupervised models[108]. The two way classification of Deep Learning techniques hides the strength of the third category but it may illustrate the advantages and disadvantages of the two classes: the discriminative models such as Deep Neural Networks or DNNs are more efficient in terms of training and testing, less–constrained in construction, and more appropriate to learn complex systems; on the other hand, generative or unsupervised models are easier to understand, easier to include domain knowledge, easier to combine, and easier to handle uncertainty, but difficult to learn complex systems.

2.4 3D Models and Descriptors

In late 1990s and early 2000s, the number of 3D models; designed and created by 3D designers with the use of 3D modeler applications, were increasing. Therefore, some researches emerged in the field of 3D model retrieval to search through 3D model databases to retrieve 3D models similar to a query model. Previous works in the area of 3D model recognition have focused on matching and retrieval of similar 3D model to a query model which is used

in search engines[126, 53, 115, 54, 120, 98, 64]. The main idea behind the proposed methods was representing two 3D models; the query model and the one stored in a database, with defined descriptors and compare the two descriptors to measure the similarity or dissimilarity of the two models[53, 54, 98, 64].

The most important problem in 3D model matching is how to represent a 3D model that the representation be computationally efficient and representatively expressive. 3D models appear as mesh–based, polygonal or voxel–oriented representation of real objects. The possible degeneracies in the models—due to scanning step of building the 3D models—may cause the calculation of some fundamental geometric features of a 3D model such as volume impossible[53]. As an example, the first 3D model, the Utah teapot, is bottomless[53]. What would be the volume of an object without defined boundaries? One way to solve the degeneracy problem in a model is an additional reconstruction step before descriptor calculation—computationally expensive and human–intervention–required task[53].

In general, 3D shape descriptors are classified into three broad non–disjoint categories: (1) feature–based descriptors, (2) graph–based descriptors and (3) geometry–based descriptors[115]. Geometry and topology properties of a 3D model form the foundation of feature–based descriptors[115]. Graph–based descriptors represent topological characteristics of a 3D model[115]. Geometry–based descriptors formulate the attributes coming from geometric operations on the 3D model, or properties of geometric components of the model[115].

2.4.1 Feature–Based Descriptors

In feature–based descriptors, features themselves or the distribution of features are used to describe a 3D model[115]. Feature–based descriptors are classified into four categories according to the features' essence: (1) Global Features, (2) Global Feature Distribution, (3) Spatial Map and (4) Local Features[115]. After describing a 3D model with the first three feature–based descriptors, the signature of a 3D model becomes a d–dimensional feature vector[115]. To compare and measure the similarity of two d–dimensional feature vectors,

they are considered as points in space and the distance between the two points is accounted as the dissimilarity measurement of the two descriptors[115]. The problem of finding best matches of a query model described with a d–dimensional feature vector is solving a k–nearest neighbors problem[115]. In this case, the distance between the query descriptor and each descriptor in the database is measured and the k–lowest distant descriptors are returned as the k–best matches of the query descriptor[115].

Global features are properties of the overall shape of a 3D model[115]. Probability distribution of the global features is the main component of the second category in 3D shape descriptors[115]. Spatial maps are characteristics of spatial location and orientation of an object; spatial maps are rotation–variant[115]. Finally, a local feature–based descriptor is a bag of descriptors each describing a 3D model, locally; i.e., each descriptor describes a 3D model shape around a number of surface points[115].

2.4.2 Graph–Based Descriptors

Feature–based descriptors, discussed in previous section, more or less, describe 3D models according to the geometry features of 3D models[115]. Graph–based methods try to illustrate a 3D model according to the topology of the model with respect to its geometry[115]. The most well–known topological description of things is graph[115]. Graph–based descriptors use graph to show the way that the components of a 3D model are linked together[115]. Graph–based methods are subdivided into three different methods: (1) Model Graphs, (2) Skeleton Graphs and (3) Reeb Graphs[115]. Graph–based descriptors are compared to each other through graph matching and graph isomorphism[115].

Model graphs are more representative for solid models made by CAD systems rather than natural shapes like humans and animals[115]. Skeleton graph represents geometric and topology of a 3D model[115]. Reeb graph is another way of describing topological characteristics of a 3D model[115].

2.4.3 Geometry–Based Descriptors

Geometry–based descriptor is another approach of perceiving 3D models; using geometric primitives such as points, lines, etc. to describe a 3D model or applying geometric operations on a 3D model and to get the required information[115]. Geometry–based methods are categorized into four classes: (1) View based methods, (2) Volumetric Error methods, (3) Weighted Point Set and (4) Deformation methods[115]. Two arbitrary 3D models are similar if they are geometrically close to each other[115].

In view based methods, the descriptor of a 3D model is characteristics of the 3D model from different viewing angles[115]. Volumetric error is defined as a similarity measurement approach in which two models are compared with the volumetric attributes[115]. Weighted point also describes a 3D model with weighted points[115]. Finally, deformation method is a comparison approach of two 3D model in which the required amount of deformation of one model in order to register into the other model measures the dissimilarity of the two objects[115].

2.5 3D Point Cloud Surface Reconstruction

3D surface reconstruction is an important concept and one of the fundamentals in 3D modeling and 3D model reconstruction[87, 81, 76, 40, 57, 4, 68, 2, 5, 6, 36, 72, 58, 37, 25, 3, 77, 99, 8, 49]. Popularity of 3D scanner devices added more value to 3D surface reconstruction methods dealt with 3D scatterd points—the raw products of 3D scanners. More specifically, 3D surface reconstruction from point cloud is used to build more realistic 3D models of objects, scanned by the 3D scanners[4, 25, 3, 49]. Before 3D scanner devices, 3D surface reconstruction was studied to build a 3D model from a set of planar or spatial points. Inception of 3D scanner devices initiated a series of studies which were aiming to utilize the raw data generated by these devices, in the form of point cloud, in order to produce a 3D visualization of the scanned objects.

The importance of 3D surface reconstruction from a set of points comes from the fact that 3D scanner devices and many other resources of 3D point data are available but the raw data derived from these resources is not representative nor useful for ordinary people[25]. 3D surface reconstruction may be used to visualize the 3D point cloud generated by scanners or obtained from other resources. For instance, 3D printers use 3D surface reconstruction to build a real 3D object from a provided point set.

3D scanners use different technologies to generate a point cloud—a set of disjoint points in space appearing where there exists an object in a scene. One of the most commonly used and more accurate techniques is laser–based scanning technology[49]. Invisible laser beams are emitted from a 3D scanner and recollected after reflection from the objects in the scene. The traveling time of a beam since it is emitted from the device until gets back to the device is a measurement of depth. The distance, the direction of the beam, and the position of the laser pointer on the device form the components of measuring a single depth point in space which lies on the object's surface. 3D scanners release thousands of laser beams in multiple directions in a second. Therefore, the result of the scanning process is called point cloud.

2.5.1 Surface Reconstruction

Surface reconstruction is a process of retrieving a 3D model of a real object from the input data which is provided by 3D scanner devices[5, 66, 37, 49]. Input data is usually in the form of scattered points in 3D space. Points can be either structured or unstructured[5, 4, 77]. An unstructured point set is raw which means the coordinates of the points are the only information that are provided[4]. On the other hand, a structured point set includes additional information either geometrical, such as partial connectivity or normal vectors of faces defined by at least three point coordinates, or topological like overall shape that the points may fit into including cylindrical, spherical and so on[4, 66]. According to the input data type, an appropriate method should be used to reconstruct the surface.

Input data is derived from 3D laser scanners that produce a set of points corresponding to

the positions where the laser beams hit the objects in a scene. A pre–processing step may be required before surface reconstruction to remove noises, to filter out the unwanted points, to simplify the input data and to segment the original data set into clusters representing actual objects. Different input data type can be treated with various reconstruction methods and the results may change drastically upon the selected method and the data type. Problems that may rise upon the data type are discussed later in this section.

A surface or the method that may result to a surface, is described or qualified in different ways. Surfaces, and correspondingly the methods that are producing such surfaces, in literatures, are classified into implicit vs. explicit, approximated vs. interpolated and isotropic vs. anisotropic.

Explicit vs. Implicit Surfaces

[40] describes an explicit surface as a prescription of the precise location of the surface. It means that an explicit representation of a reconstructed surface from a point cloud lies on the points that are scanned from a real object and are present in the cloud. The surface is usually in the triangulated format and these points are the vertices of the triangles or place on the triangles of the surface.

There are two different kinds of explicit surfaces: (1) Parametric and (2) Triangulated[41]. A parametric surface is the deformation of a primitive model that covers an arbitrary portion of the points[41]. Some of the primitives being used are B–Spline, NURBS, plane, sphere or ellipsoid[43, 33, 63, 95, 117, 118]. Parametric surfaces are topologically limited by the initial model which means complicated surfaces are not representable easily and they may need a mixture of primitives[41].

Triangulated surfaces are the most intuitive version for surface representation in which the surface is described by connected triangles made from the input points using k–nearest neighbors of a point to build the connectivity[51, 128, 27]. A triangulated representation of a surface is called simplicial surface [66].

An implicit surface is defined by a function which one of its isocontours is a close estimation to represent the input data. Implicit surfaces are defined as particular isocontours of scalar functions in [40]. More specifically, implicit surface reconstruction is essentially finding a function that best fits the input data. Implicit representation of a surface needs to be post–processed in order to be visualized; marching cube is the most well–known method to generate a triangulated surface from the implicit representation of the surface[41].

Implicit surfaces are typically a summation of radially symmetric basis functions. This symmetricity is an issue when a surface has corners or edges; i.e., a typical implicit surface suffers from incapability of presenting asymmetric characteristics of the surface like edges and corners[41]. Therefore, these surfaces are not able to represent complex surfaces, well enough[41].

Variational implicit surfaces, despite of typical ones, use different types of basis function to address the inability of describing drastic changes on the surface; corners and edges [41]. Volumetric regularization or energy minimization is how a variational implicit surface represents an object; i.e., the surface is the isocontour of the basis function where the function is minimized [41]. In fact, the implicit surface is the zero set of the basis function. This kind of representation is more powerful than the typical one in the case of complex shapes with twisted topologies [41].

Different types of functions are being used for implicit surfaces; distance functions and convolution of symmetric functions. The former is more about energy minimization and relative to variational surfaces [41]. While the latter is about combination of symmetric functions with different parameters to find the best fit for the given points [41].

Least Square (LS)[104], Partial Differential Equation (PDE)[43], Hausdroff Distance (HD)[128] and Radial Basis Function (RBF)[25] are some of the variational implicit surface representation functions which are used in surface reconstruction methods. The zero set of these functions on input data are the approximation of the surfaces[128]. Mixture of Poisson distribution with different parameters[72, 23, 78], Gaussian kernel combination with

various μ and σ[41], in addition to blending heterogeneous polynomial functions are some of the typical implicit surface representation approaches.

Interpolated vs. Approximated Surfaces

While term "interpolation" is used for the surfaces that may suffer from non–uniformity of the input data or undesired holes due to opaque; to resolve the abnormality of the given data, an interpolated reconstructed surface is the one that at least a subset of the input points lie on the surface[7, 5, 49]. On the other hand, an approximated surface is the representation in which none of the points may place on the surface[26, 128, 41, 49]. That surface is just an estimation of topological circumcircle of the input set.

Almost all the explicit surfaces are considered as interpolated surfaces, specifically, triangulated surfaces in which the input points rest on the surface[128, 41]. Implicit surfaces are often approximated as the spirit of the reconstruction methods of these types of surfaces are about energy minimization or combinatorial function fitting[41, 128].

Anisotropic vs. Isotropic Surfaces

Isotropy or anisotropy attribute of a surface is a characteristic coherent to the topology of the surface[41]. A model is called isotropic when it has very smooth surface without any sharp edges or corners[41]. In fact, isotropy refers to a set of features that do not change in different directions; i.e., a surface feature is isotropic if that feature carries the same value where it measures in various directions[41]. Therefore, a surface is isotropic where all the measurable features of the surface are isotropic[41]. Isotropy and homogeneity or symmetricity are all equivalent.

An anisotropic model is a model with at least one anisotropic feature[41]. An anisotropic property differs when it is measured in different directions[41]. The distance between points in presence of sharp edges and corners ia an anisotropic feature of a surface because the distance of the points along an edge varies with the distance across the edge[41]. Anisotropy

and heterogeneity and asymmetricity can get used interchangeably.

2.5.2 Issues and Constraints

Issues and limitations in surface reconstruction techniques are co–dependent to the input data type. Some methods have addressed the issues by making some assumptions about the input data and have accepted the imperfection, inaccuracy, and incorrectness of the result. On the other hand, some other approaches have overcome the constraints by sacrificing the computation speed and space complexity.

Unstructured Data

More than often, the input data is unstructured; it means that it includes just the coordinate of the points and nothing more than that[4, 68, 5]. Dealing with unstructured data has become one of the main factors in qualification of surface reconstruction methods. Problems in surface reconstruction from unstructured point cloud; caused by the non–uniformity characteristic of the input set, includes implementation difficulties, inaccuracy of the results and high computational demands due to less information about the overall structure of the scanned objects[4, 68, 5].

Some of the reconstruction techniques address the aforementioned issues of dealing with unstructured data by feeding a dense point cloud to the method in order to infer the connectivity information and to ease the surface computation. Although dense sampled input set works in many applications, it may cause some other issues like increasing the input size, computation cost in terms of time and memory space plus having redundancy in the input set which may result in undistinguishable noises that produce outliers in the reconstructed surface. On the other hand, dense sampled data is not always available. Surface reconstruction methods that use condensed data to resolve the unstructured–ness of the input suffer from incompleteness and inaccuracy when the data is not dense enough.

Dataset Size

The other major constraint of surface reconstruction methods is the dataset size[4]. Some methods are not able to deal with a large data set due to the implementation of the method and the data structure that is used to store the input data and the reconstructed surface; i.e., an inadequate data structure and implementation limit the approach to available hardware resources. These surface reconstruction methods with upper limits in the data size suffer from low speed because of the computation cost and memory usage of the algorithm. Even a method handles an enormous data set it may require high memory space to reconstruct the surface. Therefore, there is a tradeoff between accuracy and performance.

Incrementing the amount of data may result higher accuracy and perfect surfaces, whilst it may lower the performance of the method either with respect to computation time or acquired memory space. Even using condensed data set is used in some techniques as a resolution for surface reconstruction from unstructured points, these methods may suffer from low speed computation and high memory consumption or in the worst case they may not be able to handle the large input sets.

Dataset size problem is addressed by filtering and simplification which both try to reduce the input size. Filtering is removing some parts of the input data which are undesired or unnecessary in the reconstructed surface. Simplification is also reducing the amount of the data by uniformly sampling the input. Filtering and simplification are the two steps of preprocessing in some reconstruction methods which is also considered as an overhead on the whole process.

Performance

As it may be known, performance is always a bottleneck in all computer–based applications. Computer software solutions are qualified and ranked with respect to their performance. Performance has three sides: (1) time efficiency, (2) memory efficiency and (3)

accuracy and completeness[57, 77].

All of these three aspects, together, may not get satisfied in any surface reconstruction technique. But there are some methods which keep them all in an arbitrary level of satisfaction (tradeoff between input set size and performance). In brevity, the larger the dataset gets, the lower the speed, the higher the memory space and the better the accuracy will be.

Some surface reconstruction techniques resolve the time and memory efficiency issues by preprocessing steps such as filtering and simplification on the input set, while they suffer from the third aspect of performance: accuracy and completeness. On the other hand, accurate and complete methods may sacrifice the two other aspects: time and memory efficiency.

Redundancy vs. Sparseness

Redundancy happens when the input set is dense; therefore points in the set may overlap[5]. Although it may ease the surface reconstruction implementation due to resolving the unstructured–ness of the points, it may affect the performance. Moreover, redundancy leads to concentrated dataset, which also impacts the performance of the system: degradation in time and memory efficiency vs. promotion in accuracy.

Additionally, redundant data may influence preprocessing steps for noise reduction. When there is a redundancy in the data, the noisy area is also condensed. That means the noisy area and the noise–free area of the data are not readily distinguishable. That generates outliers in the output surface. Solutions to resolve the issue in most cases is simplification methods in the preprocessing step.

Dispersion in the data input is called sparseness[5]. The reason that some point sets are sparse is sometimes weaknesses in scanning devices including lower sampling rate and sampling quality. Scattered points are the opposite of redundant data and highly dense one. Sparseness affects the accuracy, increases the chance of noise inclusion in surface reconstruction and may produce coarse surfaces[5].

As of the other issues, preprocessing is the first idea to overcome the sparseness issue.

Interpolation the points to increase the density and as a result producing smoother surface with incompleteness and outlier–inclusion payoffs, is one way of resolving the issue in the preprocessing step.

Noisy Data

Scanning devices are not perfect; therefore, the input data may include erroneous and noisy points. Noises affect the result of the surface reconstruction procedure[87]. It may create outliers in the reconstructed surface or even worse; the result may not sound the scanned object at all[87]. In fact, noise and error in the input are equivalent to inaccuracy. Assuming that the input is noise–free; which sacrifices the accuracy and produces outliers, is one resolution for noisy data in some reconstruction techniques. On the other hand, filtering in preprocessing steps is another idea that may reduce erroneous data[87, 76, 26, 38].

Incompleteness

Incompleteness in the reconstructed surfaces occurs when the input data is missing some parts due to opaque, hollows or inappropriate scanning devices[87, 76]. A surface is incomplete when it is not dual of the real object. Missing a hole on the surface due to highly dense points and having extra holes on the surface because of opaque are the two major types of incompleteness.

To resolve incompleteness, preprocessing steps such as reproducing or interpolating the missing data to remove the unwanted holes, or simplification to extract the actual holes must be done. Though, here again there is a tradeoff between accuracy and performance.

2.6 Discussion

The comprehensive literature review of this chapter elucidates Image Processing's and Machine Learning's collaboration in object recognition systems; and 3D point cloud process-

ing methodology requirements of building a cognitive model based on the 3D point cloud of objects. Therefore, we aim to find out if there exists a procedure to utilize Image Processing and Machine Learning methods, and the concept of 3D point clouds, to provide a 3D point cloud–based object recognition model. In other words, is this possible to propose point cloud processing methods—as the preceding units for learning models—to prepare an object recognition system? So, this dissertation points out to solve the object recognition problem by processing the unstructured 3D point cloud of objects—not the 2D projection—to extract features, to feed the learning models, and to build the recognition system. The following chapters signify the data collection procedure, the feature extraction methods, and the learning phase of the proposed recognition system.

CHAPTER 3

DATA COLLECTION PROCEDURE AND RESOURCES

3.1 Introduction

Data collection, as the first phase in the recognition architecture shown in Figure 1.1, aims to prepare and specify a set of sample entities required either for validation or verification of the system. In the third phase (Learning), the collected dataset trains, validates, and evaluates the learning models studied in Chapter 5. Since the proposed recognition system works based on the unstructured 3D point clouds, this chapter revolves around what are the different resources to collect 3D point cloud of objects, what objects are present in the collected dataset, and how are the distribution of instances in the dataset across the object classes (labels) and two other attributes described in Chapter 4—size class and symmetricity degree. Moreover, this chapter describes the general procedure of producing a 3D point cloud of an object and the details about one of the resources (Google Tango) used as a benchmarking platform.

The rest of this chapter follows the succeeding organization. Section 3.2 presents various available resources to collect 3D point clouds. Section 3.3 describes the procedure of building a 3D point cloud of an object utilizing laser–based 3D scanner devices. Section 3.4

introduces Google Tango—one of the accessible resources and an example of laser–based 3D scanners—and explains how Google Tango produces scattered points and how these data are transformed into a 3D point cloud. Section 3.5 provides the collected dataset's specifications: the size; the number of object classes (labels); and some analytical descriptions of the collected dataset. Section 3.6 draws this chapter to its end by a short discussion about an encountering issue in 3D point cloud data collection phase using laser–based 3D scanners.

3.2 3D Point Cloud Resources

Several resources contribute in collecting a dataset of 3D point clouds: 3D laser–based scanner devices; 3D models scanned by other scanner devices; and 3D models created by 3D modelers using 3D software packages.

3D laser–based scanner devices exist in the desktop version for 3D printers and modeling purposes. The popularity of laser–based scanning technology, provoke its usability even in mobile devices such as Google Tango, Microsoft Kinect and Occipital Structure. Therefore, handheld and desktop 3D laser–based scanners constitute two major 3D point cloud data collection resources.

Other types of 3D scanner devices utilizing touching probes, white light and Computerized Tomography (CT) techniques may generate 3D models of real objects. These methods represent the 3D models in mesh–like structure. Preprocessing steps including information removal, simplification and registration—converting into and storing in a consistent format—are performed in order to transform these mesh–like objects into their corresponding 3D point cloud structures.

Last but not least, 3D models generated by 3D modelers using 3D software packages based on the real objects compose the third available resource for 3D point cloud data collection. These models appear in the mesh–like format. Preprocessing steps—likewise the second approach—transform these mesh–like models into the 3D point cloud data structures.

3.3 3D Point Cloud Scanning Process

Data collection using 3D laser–based scanners requires the following scanning procedure to produce complete, accurate and smooth 3D point clouds.

- 360 degree horizontal scans from different angles
- Pose registration[1] of horizontal scans to build a uniform side point cloud
- 360 degree vertical scans from different angles
- Pose registration of vertical scans to build a uniform top–bottom point cloud
- Pose registration of side and top–bottom point clouds to generate the complete point scan

3.4 Google Tango

Low energy laser sensor equipments on Google Tango—a handheld device in the form of regular tablets—transform the device into a mobile 3D laser scanner. Therefore, Google Tango, as one of the resources for data collection in this study and a benchmark for mobile 3D scanners, produces some of the 3D point cloud of objects in the collected dataset. This section describes the procedure of point collection using a Google Tango device.

Google Tango works with two coordinate systems, one for data collection and the other for localization and orientation examination—local and global coordinate systems. The former is attached to the device and therefore when the device moves the coordinate system moves along. In this system, the z–axis, the axis of depth measurement, defines the line of gaze. On the other hand, location and orientation of the device are formulated according to the global coordinate system (a fixed point).

Google Tango uses Quaternion coordinate representation for localization and orientation calculation. Quaternion coordinate representation consists of three components: (1) A fixed point, (2) A rotation vector which runs by the fixed point and (3) A scalar rotation angle.

[1]Pose registration is the process of matching and stitching consecutive scans to make a uniform scan

The following P and Q annotate these components.

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$Q = \begin{bmatrix} q_x \\ q_y \\ q_z \\ \omega \end{bmatrix}$$

Since Google Tango collects all the 3D points according to a momentary local coordinate system, these locally collected point scans ought to be presented in the global coordinate system in order to produce a complete model. This procedure includes a translation and a rotation operation performed on each individual locally measured depth point. The following formulates these matrix multiplication–based operations.

$$R_{4\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2(q_xq_y+\omega q_z) & 2(q_xq_z-\omega q_y) & 0 \\ 2(q_xq_y-\omega q_z) & 1-2q_x^2-2q_z^2 & 2(q_yq_z+\omega q_x) & 0 \\ 2(q_xq_z+\omega q_y) & 2(q_yq_z-\omega q_x) & 1-2q_x^2-2q_y^2 & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

$$P_g = R_{4\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + \begin{bmatrix} p_x \\ p_z \\ -p_y \\ 1 \end{bmatrix}$$

In which $\begin{bmatrix} X & Y & Z \end{bmatrix}$ vector represents each individual point, p_x, p_y and p_z are x, y and z elements of momentary location of the device in the Quaternion representation. ω is the scalar rotation angle and q_x, q_y and q_z are the components of the rotation vector in Quaternion representation and P_g annotates the given point in the global coordinate system.

P and Q are measured by Google Tango in every 0.3 second with respect to origin of the global coordinate system.

3.5 Collected Dataset

Utilizing all the resources mentioned in Section 3.2 and performing preprocessing operations; including filtering process to remove unwanted data and making the so called unstructured 3D point cloud—a set of point coordinates in 3D space, a 3D point cloud dataset consisting of 300 objects in 50 different classes is collected. The dataset includes a variable number of objects in the same class with different orientations, translations and scales. That makes the dataset general and proper for evaluation of the recognition system discussed in Chapter 4 and 5.

In the preprocessing step, additional information of those objects obtained from resources providing mesh–like models are filtered out to generate models represented only with 3D point coordinates or the so called point cloud. Moreover, all objects—essentially objects made by something other than 3D scanner devices—are scaled into their real–world size. The unit of measurement of 3D scanners is meter; therefore all objects are transformed into the meter scale according to their realistic size. Finally, the preprocessed objects form a dataset of 3D point cloud entities. These objects vary from very small ones like paper clips to very large ones such as cars and trucks.

Table 3.1 describes the range of object classes existing in the collected dataset and the number of instances from each class. Figure 3.1 shows the inherent normal distribution of instances in the collected dataset across the object classes. Figure 3.2 and 3.3 illustrate the distribution of objects across the size class and symmetricity degree attributes described in Chapter 4.

Table 3.1: Object classes and number of instances of each class in the collected dataset

#	Object Class	# Instances	#	Object Class	# Instances
1	armchair	4	26	light bulb	4
2	barrel	4	27	lounge	4
3	bench	4	28	mixer truck	4
4	beverage can	8	29	mug	4
5	book	8	30	office chair	4
6	boot	4	31	oil tank	4
7	broom	4	32	ottoman	8
8	brush	4	33	pencil	4
9	candle holder	12	34	plastic bottle	4
10	car	24	35	ring	4
11	ceiling fan	4	36	ruler	4
12	cello	4	37	shoe	4
13	cellphone	16	38	slipper	4
14	chair	8	39	soap	4
15	chandelier	4	40	soccer ball	8
16	circle table	4	41	sofa	4
17	coffee cup	4	42	sport car	4
18	clip	4	43	spray can	8
19	door handle	4	44	sugar bowl	4
20	fruit case	4	45	table	4
21	garbage can	4	46	traffic light	4
22	glass bottle	4	47	truck	4
23	gun	16	48	wedge shoe	4
24	knife	12	49	wine container	4
25	ladder	4	50	wine glass	20

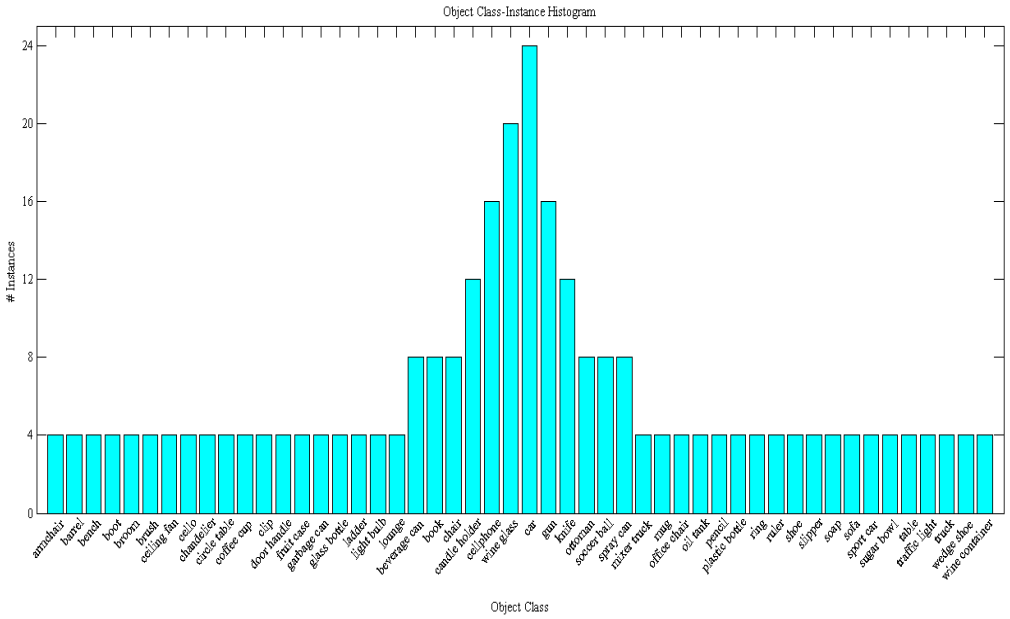

Figure 3.1: Normal distribution (histogram) of object instances across the object classes in the collected dataset

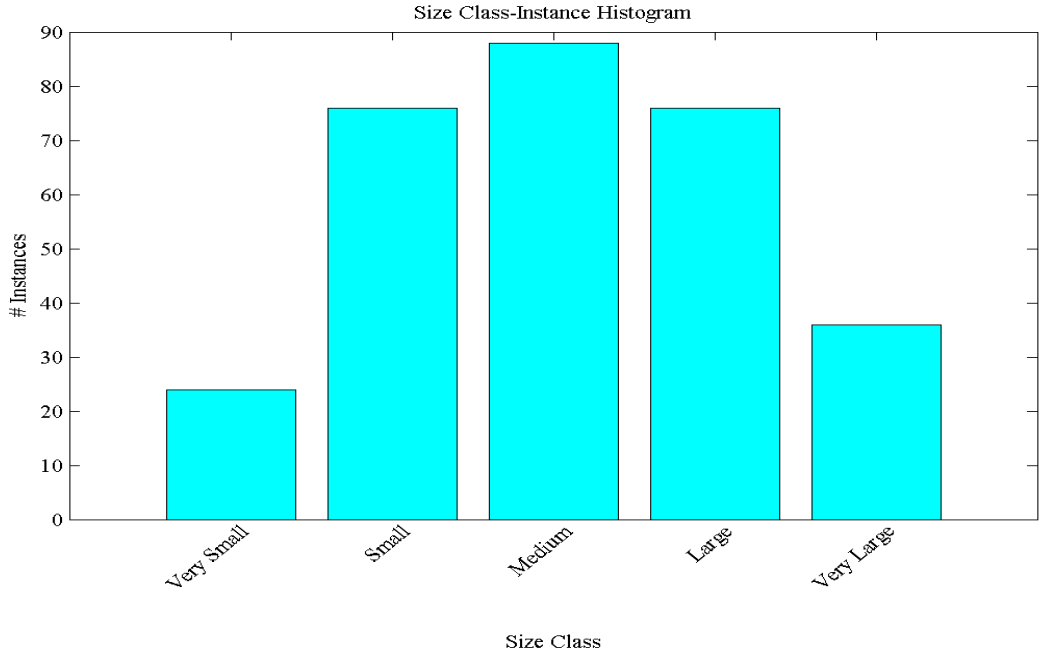

Figure 3.2: Normal distribution (histogram) of object instances across size class attribute

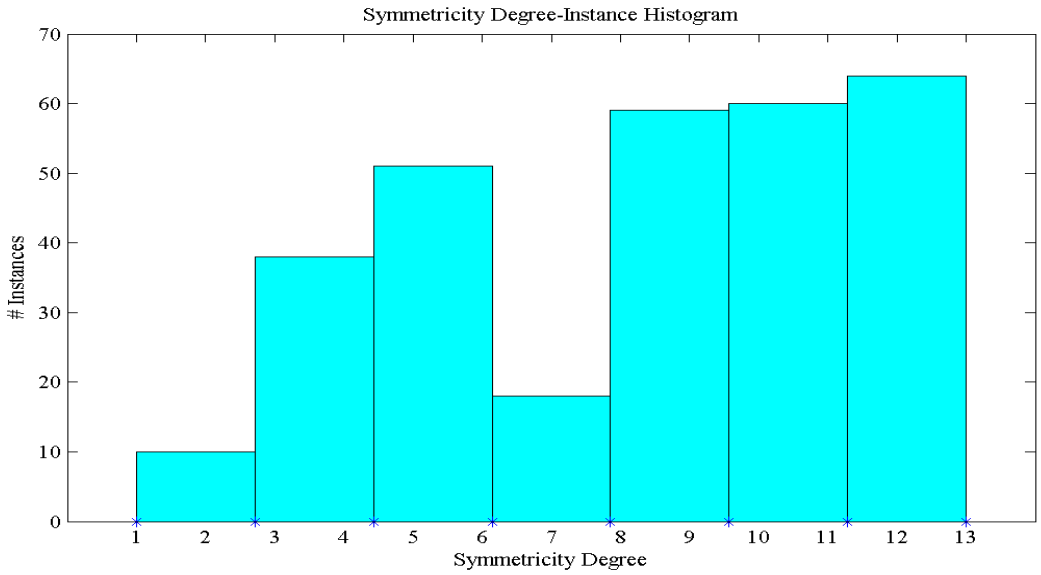

Figure 3.3: Distribution (histogram) of object instances across symmetricity attribute

3.6 Discussion

Due to the difficulties associated with pose registration, the most challenging part in data collection was the use of handheld or mobile 3D scanner devices. Since some of the mobile 3D scanners like Google Tango work based on the estimated localization data, between two consecutive scans of an object from different angles, a non–deterministic drag was observed. Computing the exact amount of rotational and translational difference between two scans—to match, to bring them together and to build a complete and smooth point cloud of an object—becomes a problematic challenge in data collection from mobile 3D laser scanners. Moreover, preprocessing steps like filtering out the surface or texture information, scaling the models to make them consistent to each other, and background removal were other issues in the data collection phase.

The main purpose of this chapter was providing a brief description of the collected dataset

and the variety of objects in the dataset, the available resources, and an introduction of a particular 3D scanner device (Google Tango). The collected dataset are examined in the following chapters for feature extraction, building the recognition system, and evaluation of the system and the extracted features.

CHAPTER 4

FEATURE EXTRACTION AND SELECTION CRITERIA

4.1 Introduction

The second phase in the object recognition architecture, shown in Figure 1.1, revolves around the definition of feature selection criteria and the feature extraction methods. Feature selection criteria define the features' required conditions; whereas they aim to empower the capabilities of the system. For instance, scale–invariant, rotation–invariant, and translation–invariant features may become the points of interests in some applications. Therefore, the system designer—in the earlier steps of the system design—must determine the attribute selection criteria and explore the properties of the dataset that meet the defined conditions.

This chapter presents the attribute selection criteria and describes the proposed features of this study. Due to the geometric nature of the collected dataset, we focus on exploring the geometric properties of the point sets in the dataset, during the feature extraction phase.

Primitive geometry properties such as distance, angle, torsion, area and volume play a significant role to produce the point clouds' descriptors, in this study. Each of these geometric attributes measures a general property of the input point set of an object. As an example, the experimental results imply that distance, area and volume estimate the shape

of an object, while angle measures the density of the points in a cloud, and torsion analyzes the shape and the density of the points on the surface of an object. Applying several estimation functions on each feature component generates two separate numerical sets of values: Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors. Analysis of these vectors measures the classification capability of each vector along the power of each estimation function; Chapter 5 discusses the analysis methods and procedures.

In addition, this chapter introduces a naive but novel approach to calculate the symmetricity degree of a point cloud. This topological attribute provides a quantitative measurement to describe how symmetric the object is. This approach is called spherical ray–based degree of symmetricity calculation. Degree of symmetricity contributes in providing an overall estimation of an object's shape and distinguishing the objects with higher degree of symmetricity from the lower's.

Moreover, an implicit reconstructed surface of an input point set—as an approximation of the object's surface shape—adds more descriptive/discriminative information to the feature vectors.

The rest of this chapter follows the succeeding orders. Section 4.2 describes the acquired specifications of the desired features for a recognition system and defines the proposed attributes' standards. Section 4.3 presents the feature extraction process flow and introduces the activities in this pipeline. A brief discussion, in Section 4.4, and future works in the feature extraction phase, in Section 4.5, conclude this chapter.

4.2 Feature Selection Criteria

In general, shape descriptors have to be invariant to translation, rotation, reflection and scale; robust to noise, perturbation and possible degeneracies; efficient in terms of computation and general—independent to representation, topology or application domain while descriptive enough to discriminate different objects [126, 53, 115, 54, 120, 98, 73, 64, 82, 102, 55]. This section introduces some of the useful descriptors/signatures and their categories, avail-

able in the literature.

Geometric, topological, mathematical and statistical characteristics can describe 3D models; from the basic geometric properties such as volume, center of mass, circularity, eccentricity, and algebraic moments to the complex functions like spherical harmonics, extended Gaussian images, spin images, wavelet and geometric hashing [98, 73, 64, 82, 102, 55].

Volume, area, statistical moments of the boundary, the volume of a model, volume–to–ratio, the Fourier transform of the volume, and the boundary of the model are some of the global features of 3D models [82]. Descriptors like bounding boxes, cords–based and wavelet–based are also global features of 3D models [82]. A bag of three global feature–based descriptors (defined in Section 2.4); aspect ratio, binary 3D shape mask, and set of edge paths, were used to describe 3D models in search engines [82]. Spherical functions like extended Gaussian image and complex extended Gaussian image are the two other global feature–based descriptors [82, 73]. Global features as their name imply describe a 3D model globally; whereas the descriptors lack the details [82].

Attributes according to distance, angle, area and volume between random points on the surface of a 3D model signify as global feature distribution descriptors [98, 82]. Angle between three random points on the surface of a 3D model, distance between a fixed point and a random point on the surface, distance between two random points on the surface, square root of the area of the triangle between three random points on the surface, and cube root of the volume of tetrahedron between four random points on the surface are some of the global feature distribution descriptors[98, 82]. Shape histograms like moment of inertia about an axis, average distance from a surface to an axis, and variance of the distance from a surface to an axis are also global feature distribution descriptors [82]. Thickness histogram and surface partitioning spectrum distribution are also known as global feature distribution descriptors [82].

Distance Map and Surface Penetration Map are two spatial feature maps to describe 3D models [126, 82]. Distance Map is a traveled distance of rays projected from a point on the

surface of a 3D model toward the surrounded sphere's surface [126]. The number of surfaces penetrated by rays traveling from the centroid of the 3D model (center of the sphere) toward the sphere's surface is recorded in the Surface Penetrated Map [126]. The former is considered as geometric feature map and the latter as topological [82]. Spherical harmonics and Fourier transform coefficients of Spherical harmonics of a 3D model are considered as spatial feature maps [73, 82]. For voxel–based 3D models, three spatial feature maps were introduced by partitioning a voxel grid of a model into disjoint cells: (1) volume features, (2) solid–angle features and (3) eigenvalue features [82]. Volume feature for each cell is the occupied volume of the grid by a voxel, solid–angle for each cell is the convexity of the voxel boundary and eigenvalue for each cell is obtained by applying Principal Component Analysis (PCA) method to the voxels of the model [82].

In this study, we focused on features which are scale–, rotation– and translation invariant. Robustness to outliers comes as the next level of consideration in feature selection phase. The rest of this chapter explains the feature extraction flow and the proposed feature vectors—mostly geometric and a few topology–oriented.

4.3 Feature Extraction

Feature extraction builds the foundation of the learning process. According to the system architecture, the recognition system requires to explore the features satisfying the selection criteria. This process contains four sequential phases. The following subsection defines the feature extraction process flow.

Feature extraction process aims to generate and assign a distinctive signature to each object class fed into the system. These signatures describe the objects and need to distinguish various object classes from others. Chapter 5 presents further studies and analysis about the proposed feature vectors, how impactful they are in the learning process, and how descriptive they can be to discriminate an object class from others.

4.3.1 Process Flow

A pipeline of certain activities forms the feature extraction phase of the recognition process: (1) Bounding Box Calculation, (2) Size Classification, (3) Normalization and (4) Feature Vector Generation. Figure 4.1 shows the flow diagram of this process.

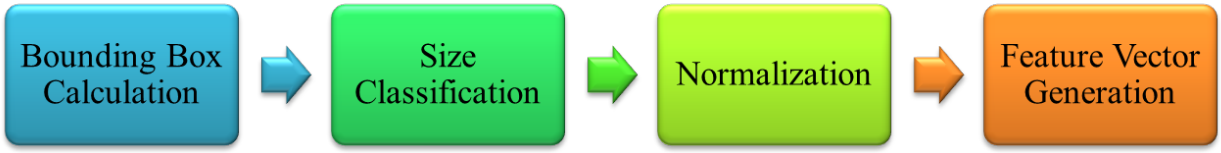

Figure 4.1: Feature Extraction Process Flow

Bounding box calculation and size classification are executed before normalization step —explained in detail later in this chapter—as it changes the scale of the input object. Normalization combines generalization steps to make the rest of the procedure and the feature vectors invariant to scale, translation, and and some other properties. These characteristics satisfy the major factors mentioned in the attribute selection criteria.

In the feature vector generation phase, numerical vectors of attributes including features obtained from geometrical and topological properties of an object are extracted. These numerical features plus size classification of the object acquired in the second step of the flow (shown in Figure 4.1) produce a complete feature vector for a given object. According to the approximation functions used to estimate the probability distribution–based features and implicit surface reconstruction methods; the length of the two generated feature vectors varies from 203 to 263. Gaussian, Fourier, Sinusoid and Polynomial estimation functions[1] built two different feature vectors from same set of proposed descriptors for further evaluations—like which estimation function is more descriptive than the others. The one with 203 numerical–nominal attributes is called Fourier–Polynomial feature vector; as all geometric descriptors are approximated by a particular Fourier transformation and the implicit

[1]Gaussian, Fourier, Sinusoid and Polynomial estimation functions are described in details in Appendix A.

reconstructed surface is modeled by a special 3D Polynomial function. While the other is named Gaussian–Sinusoid–Polynomial feature vector because of using Gaussian function to approximate the distance, area and volume descriptors, Sinusoid function to estimate angle and torsion attributes and a particular 3D Polynomial function to represent the implicit reconstructed surface of an object.

4.3.2 Bounding Box Calculation

First activity in the feature extraction pipeline of the recognition is bounding box calculation (Figure 4.1). In order to calculate the elongation of the object in the form of 3D point cloud, the rectangular cubic boundary of the cloud is measured. The surrounding boundary box of 3D point set is a rectangular cube which its length equals the maximum elongation of the object along the x–axis, its width is the maximum extension of the object along the y–axis and its height is equivalent to the longest distance along the z–axis. In other words, for each axis, the two farthest points are selected and the distance along the axis is measured by subtraction of the two values along that axis. The resultant rectangular cube is called the bounding box of an object. The dimension of the bounding box is considered as the size of the point set of an object. Moreover, the rectangular bounding box measures the space occupation of objects.

As an example, for the five different 3D point sets of objects in the collected dataset[2] including boot, chair, clip, glass bottle and truck, Table 4.1 shows the calculated bounding box for each object. Figure 4.2 to 4.6 display the 3D scatter point view of these objects. These figures illustrate the distance between the two farthest points in the XY view projected on the x–axis as the length of the bounding box, the difference between the minimum and maximum values in the XY view along the y–axis as the width and the maximum extension along the z–axis in XZ view as the height of the surrounding rectangular cube.

[2]The examination of the features was performed on all the objects in the collected dataset; however the rest of this chapter refers to these five objects (boot, chair, clip, glass bottle and truck) for illustration purposes.

Table 4.1: Bounding box calculation of armchair, boot, car, clip and glass bottle from the dataset

Object	Length (mm)	Width (mm)	Height (mm)
Boot	142.5	337.1	300.6
Chair	488.3	865.5	611.7
Clip	9.2	0.9	34.1
Glass Bottle	44.4	171.1	44.4
Truck	1855.8	1730.2	4623.1

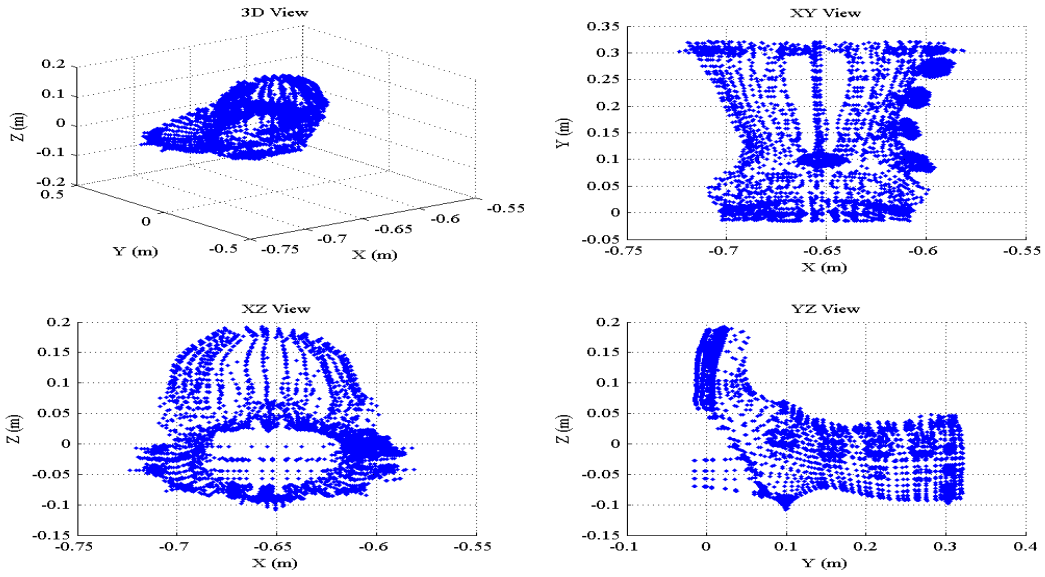

Figure 4.2: 3D scatter, XY, XZ and YZ view of boot

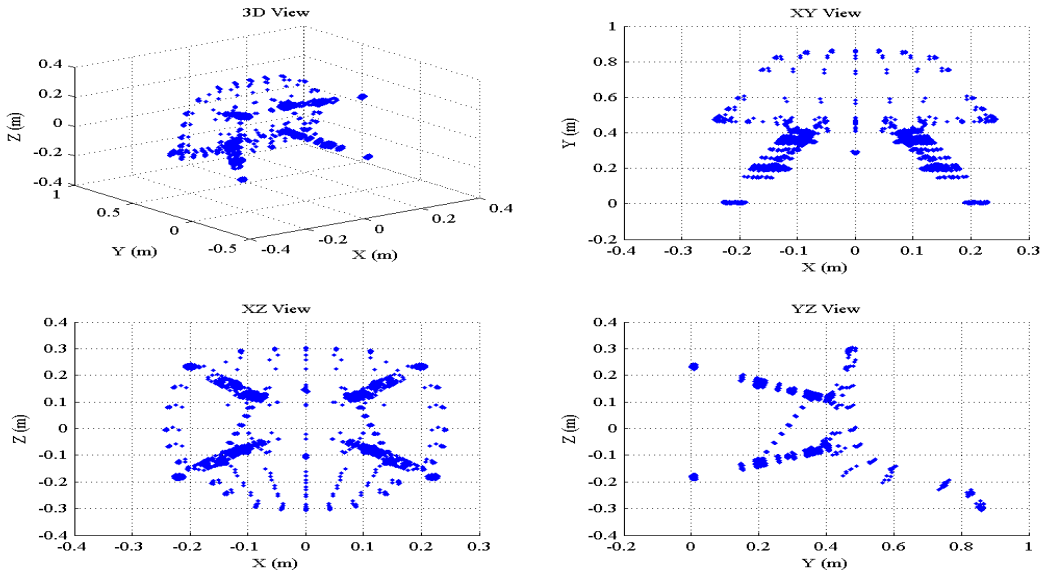

Figure 4.3: 3D scatter, XY, XZ and YZ view of chair

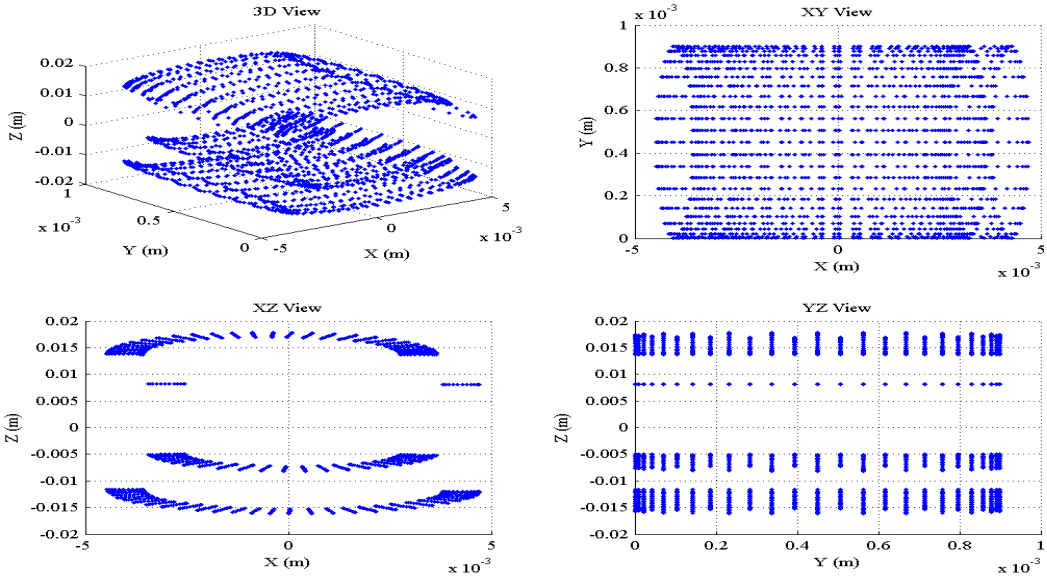

Figure 4.4: 3D scatter, XY, XZ and YZ view of clip

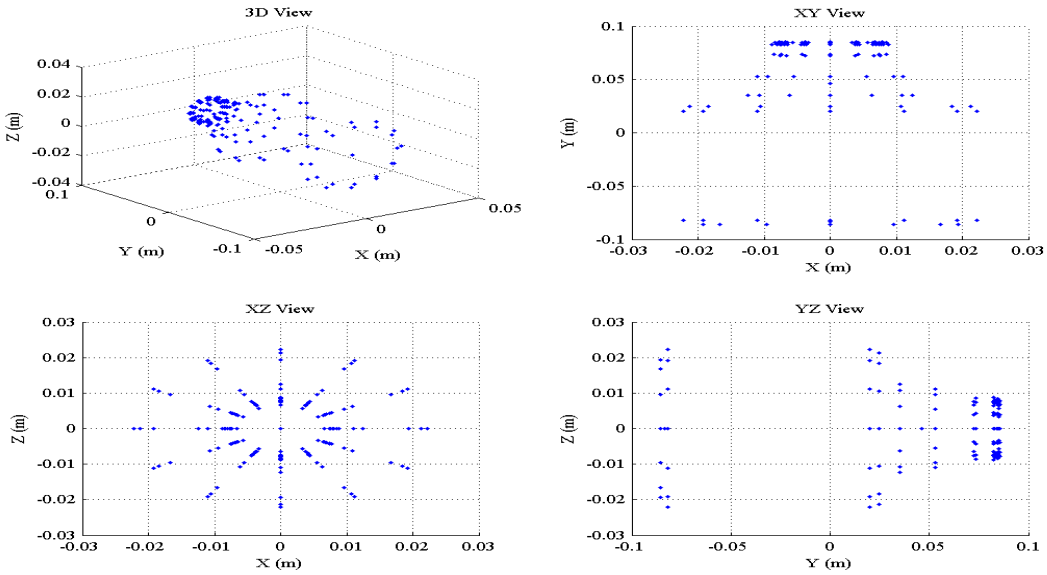

Figure 4.5: 3D scatter, XY, XZ and YZ view of glass bottle

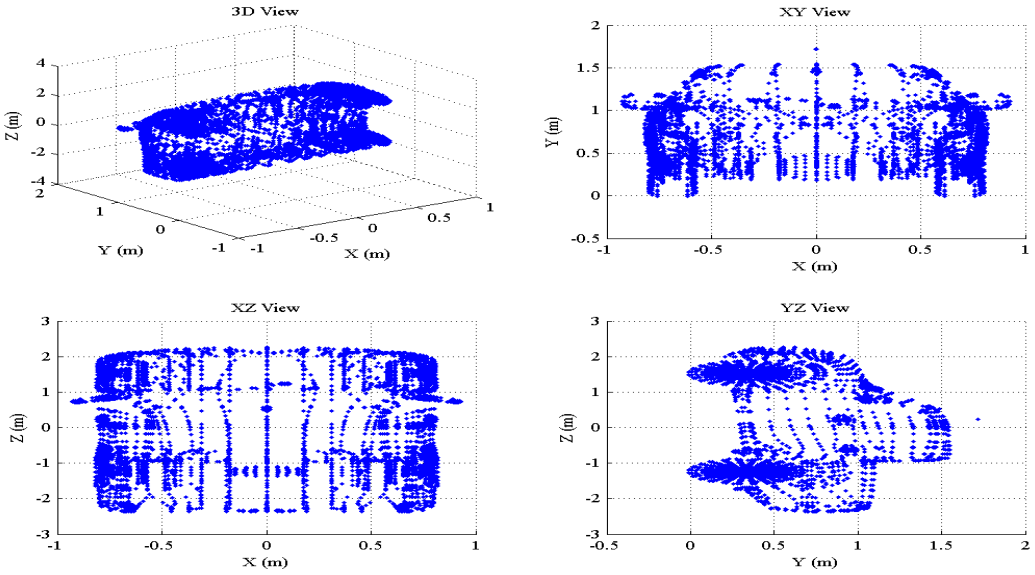

Figure 4.6: 3D scatter, XY, XZ and YZ view of truck

4.3.3 Size Classification

In the pipeline flow of feature extraction, size classification comes as the second process (Figure 4.1). Size classification step—according to space occupation of an object (point cloud of the object)—classifies each object into one of the five standard categories: (1) Very small, (2) Small, (3) Medium, (4) Large and (5) Very Large.

The space occupation of an object defines the volume of the rectangular cubic box surrounding around the object—the dimension of this box comes from the previous activity in the pipeline. In other words, space occupation measures the required room to place an object in space without any collision to other objects. The threshold of the size classes were obtained by trial and error. Table 4.2 illustrates the minimum and maximum space occupation values for each category and the category assigned to the five selected objects using the size classification method and the space occupation thresholds.

Table 4.2: Size classification thresholds and the class of selected objects

Class	Minimum Cubic Volume (cm^3)	Maximum Cubic Volume (cm^3)	Object
Very Small	0	200	Clip
Small	200	$1,000$	Glass Bottle
Medium	$1,000$	$100,000$	Boot
Large	$100,000$	$10,000,000$	Chair
Very Large	$10,000,000$	$50,000,000$	Truck

Size classification replaces the volumetric value of space occupation with a nominal attribute in the feature vector. That means all scalar properties of an object with respect to its size are reduced to a verbal attribute. Although this step may categorize same objects in various sizes into different classes, the size class along with other attributes sounds a significant property for classification purposes—Chapter 5 evaluates the impact of this feature.

4.3.4 Normalization

Normalization is the third activity in the feature extraction process flow (Figure 4.1). Normalization itself contains three steps executed in a sequential manner. Figure 4.7 shows a pictorial representation of the normalization activity in the feature extraction flow. Normalization aims to make the point cloud of an object scale and translation invariant.

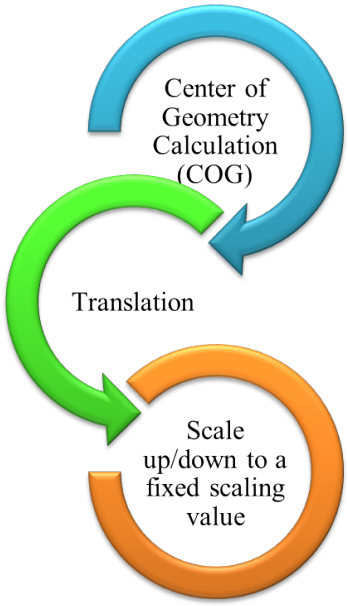

Figure 4.7: Normalization flow diagram

In the first step of normalization the center of geometry (CoG) of the object is calculated. CoG is defined as the arithmetic mean of all the points in the point set of an object. Second step of normalization computes a translation vector in order to move the CoG of the object to the origin of the global coordinate system. All the points in the point set of the objects are translated along the calculated translation vector. Therefore, the new CoG for the translated point set stays on the origin of the global coordinate system.

Third and last step in the normalization activity of the feature extraction pipeline executes scaling operations. In this step, the translated point set is scaled up/down into a fixed–size cube to make the proposed features in the following step (feature vector genera-

tion), scale invariant. Therefore, small objects are scaled up to fit into the fixed–size cube and the large objects are scaled down into that same cube. Why is it important to scale the point set of an object to a fixed–size cube? The scalling step makes all the geometric features indifferent to the actual size of the object—already determined in the previous activity in the pipeline (size classification).

4.3.5 Feature Vector Generation

Generating feature vectors from the inherent properties of the input data (3D point cloud of objects) marks the last activity in the feature extraction pipeline (Figure 4.1). The generated feature vectors consist of each object's signature; the main component used to distinguish objects of different categories in the learning phase pf the object recognition system. Our feature vectors compound of an only non–numerical attribute obtained in the size classification step and multiple numerical elements based on the geometric and topological characteristics of the point set of an object. These properties of the feature vectors inherit from distance, angle, torsion, area, volume and degree of symmetricity along the description for reconstructed surface of the point cloud of an object (the following subsections explain each feature in detail).

In this chapter, we use several estimation functions to quantize the proposed descriptors and to generate two separate feature vectors. Chapter 5 describes the learning models fed by these feature vectors, analyzes each model's performance, and provides the experimental observations.

Gaussian and Fourier estimation functions (described in Appendix A) estimate distance, area and volume; Sinusoid and Fourier transformation quantize angle and torsion; and 3D Polynomial function represents the implicit reconstructed surface of an object. Integrating Gaussian, Sinusoid and Polynomial functions for the geometric descriptors produces Gaussian–Sinusoid–Polynomial feature vector with 262 numerical values, while Fourier and Polynomial functions, together, create Fourier–Polynomial signature for the objects with 202

numerical attributes.

Distance Descriptors

Distance describes one of the primitive properties in geometry. Distance value of the points in the point cloud of an object contains of discriminative information to distinguish an object from others. To utilize the distance operator in this study, we proposed two types of distance measurement:

1. The probability distribution of the Euclidean distance between two points, uniformly at random selected from the point set (D_1).

2. The probability distribution of the Euclidean distance between a single point, uniformly at random selected from the point set, and the center of geometry of the point set (D_2).

In order to compute the probability distribution of the distance between two points from the point set, uniformly at random selected, or in abbreviation D_1, first a population of pairs of points is selected. The size of the population is defined by a sampling rate variable which in this study is equal to 50 percent of the total number of points in the given point set. In other words, 50 percent of all points in the point set are examined to compute the required probability distribution. The selection technique is based on a random number generator utilizing a uniform distribution. The Euclidean distance between each pair in the population is computed and the calculated distance is rounded to the closest fraction. The frequency of the fractions is defined by a constant value which makes the probability distribution calculation fine or coarse. The probability distribution of the Euclidean distance for each distance fraction is measured by counting the number of appearance of each distance fraction in the range of $[0, l]$—l is the length of the surrounding fixed–size cubic box in the third step of normalization—and dividing the counted values by the population size. In order to calculate D_2, the same technique is applied with a single difference. Instead of selecting pairs of points, the sampling population cosists of individual points. The Euclidean distance between these

single points and the center of geometry of the point set, the origin of the global coordinate system after normalization, is considered as D_2.

These probability distribution descriptors are estimated by two different estimation methods to fill the two separate feature vectors—Gaussian–Sinusoid–Polynomial and Fourier–Polynomial analyzed in Chapter 5. Gaussian and Fourier estimation functions are the two approximation methods to get numerical attributes from the computed distance probability distributions. The higher order of these estimation functions are utilized, since the calculated probability distributions are non–deterministic. The coefficients of these estimation functions (the numerical values) form a part of the proposed feature vectors. Gaussian estimation produces 24 numerical values and Fourier transformation generates 18. The distance descriptors of our study provide 48 numerical attributes in Gaussian–Sinusoid–Polynomial feature vector and 36 elements in Fourier–Polynomial feature vector.

Figure 4.8 to 4.12 show D_1 descriptor in blue dots, the Gaussian estimation in green solid curve and the Fourier approximation in red solid line for the five illustrative elected objects. Figure 4.13 to 4.17 display the descriptor D_2 and its estimated curves for these objects. Comparing the distance descriptors along various objects implies that distance descriptors are appropriate shape estimators. Moreover, from the experimental results, it is inferred that D_2 is more descriptive than D_1, as it describes the probability distribution of distance from the centroid.

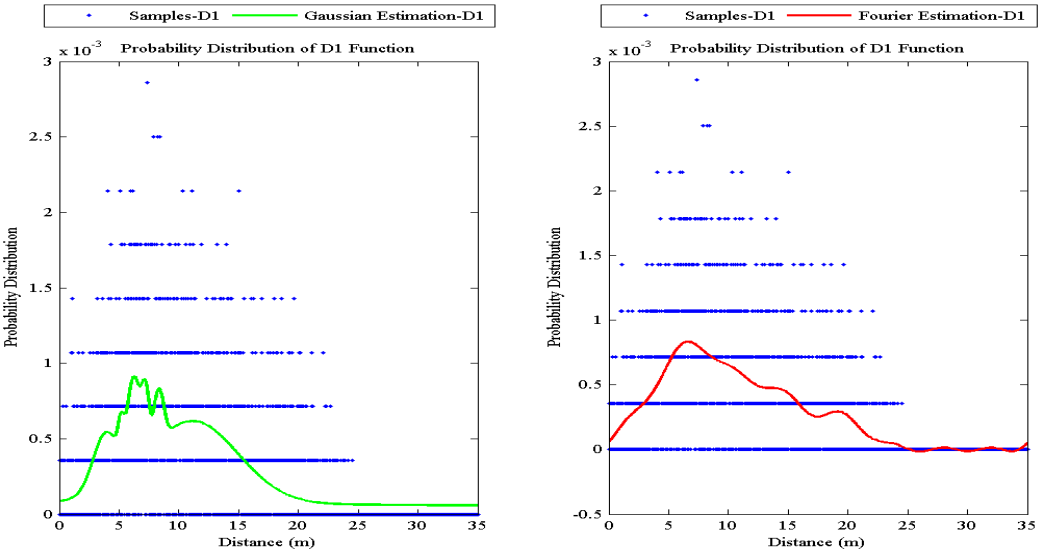

Figure 4.8: D_1 descriptor and the corresponding Gaussian and Fourier estimations for boot

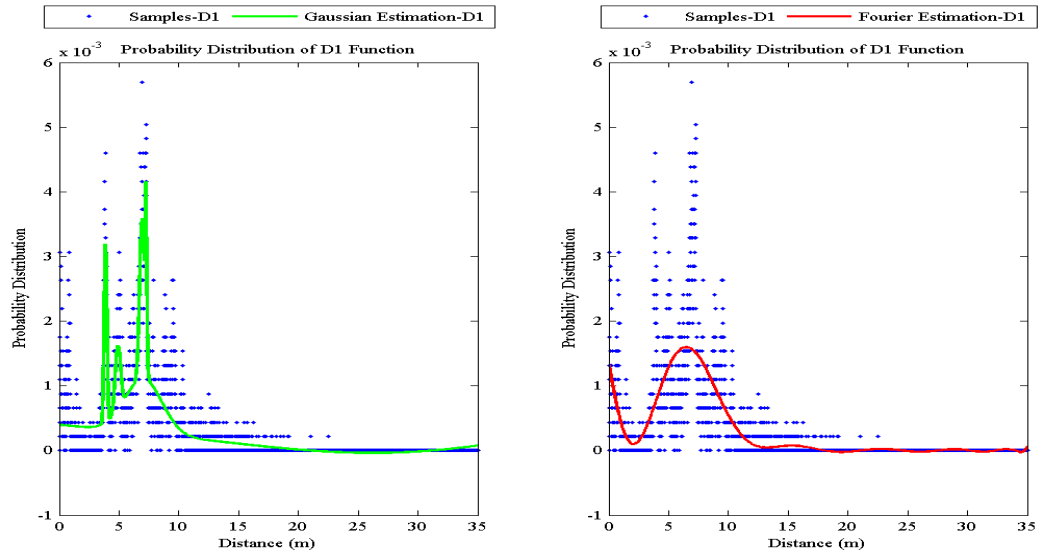

Figure 4.9: D_1 descriptor and the corresponding Gaussian and Fourier estimations for chair

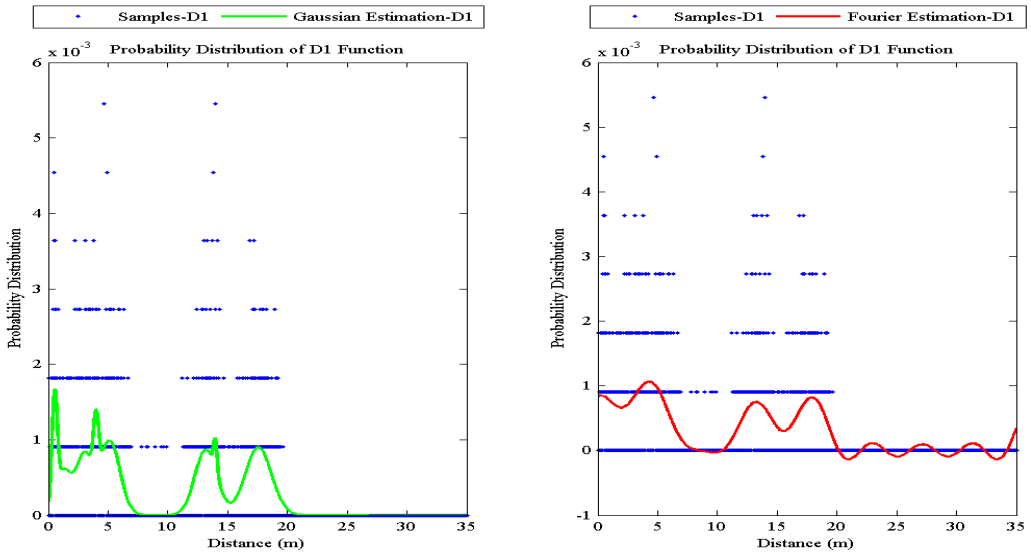

Figure 4.10: D_1 descriptor and the corresponding Gaussian and Fourier estimations for clip

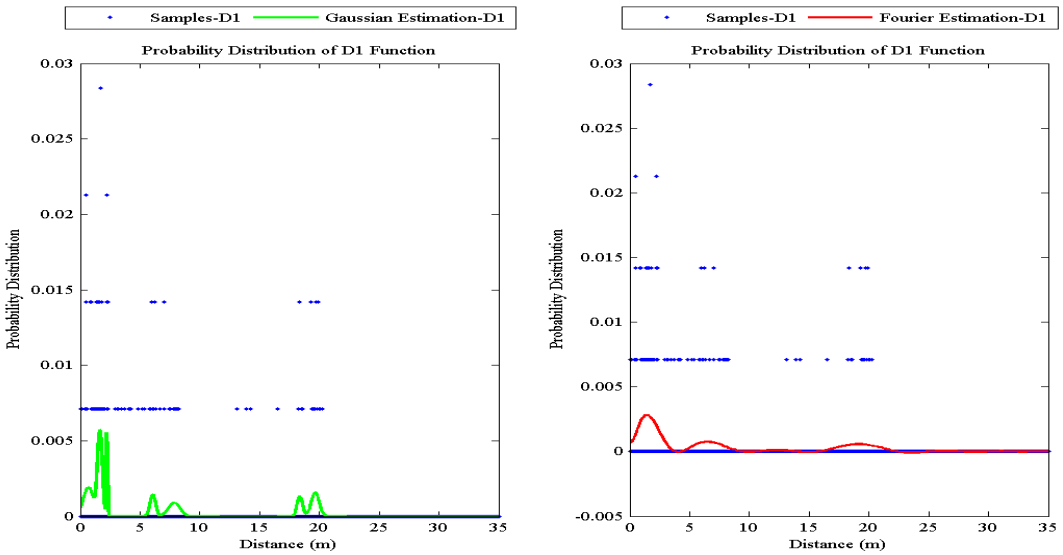

Figure 4.11: D_1 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

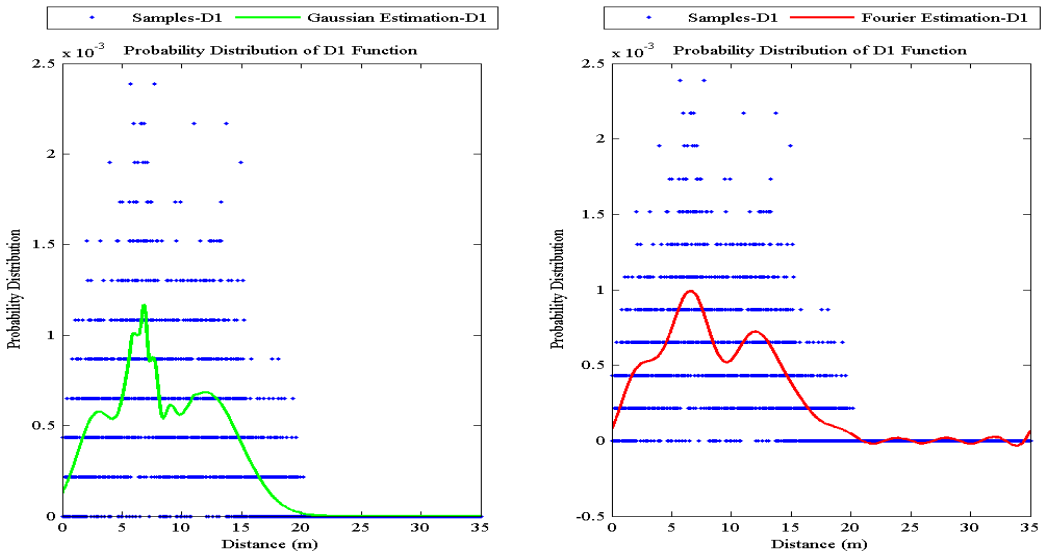

Figure 4.12: D_1 descriptor and the corresponding Gaussian and Fourier estimations for truck

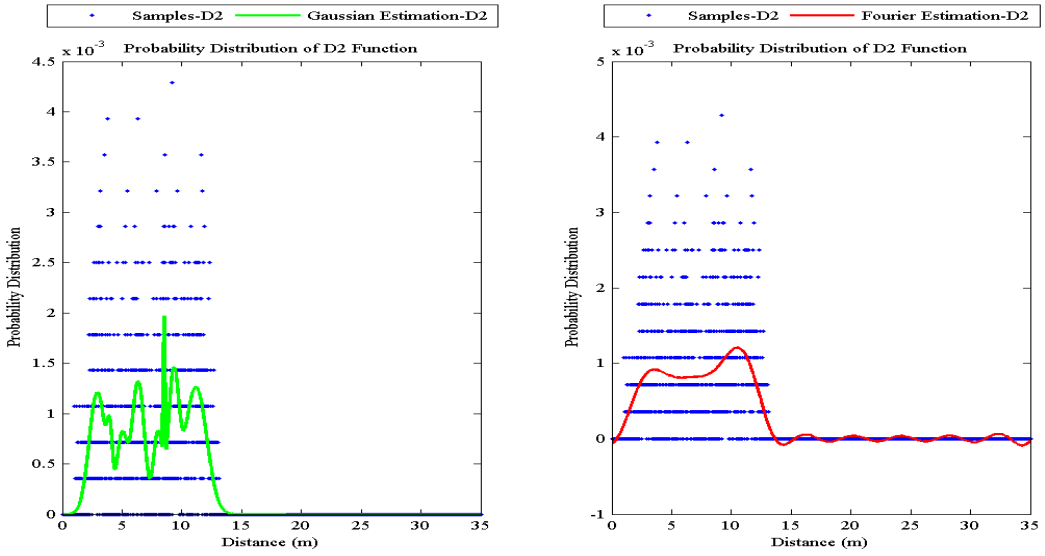

Figure 4.13: D_2 descriptor and the corresponding Gaussian and Fourier estimations for boot

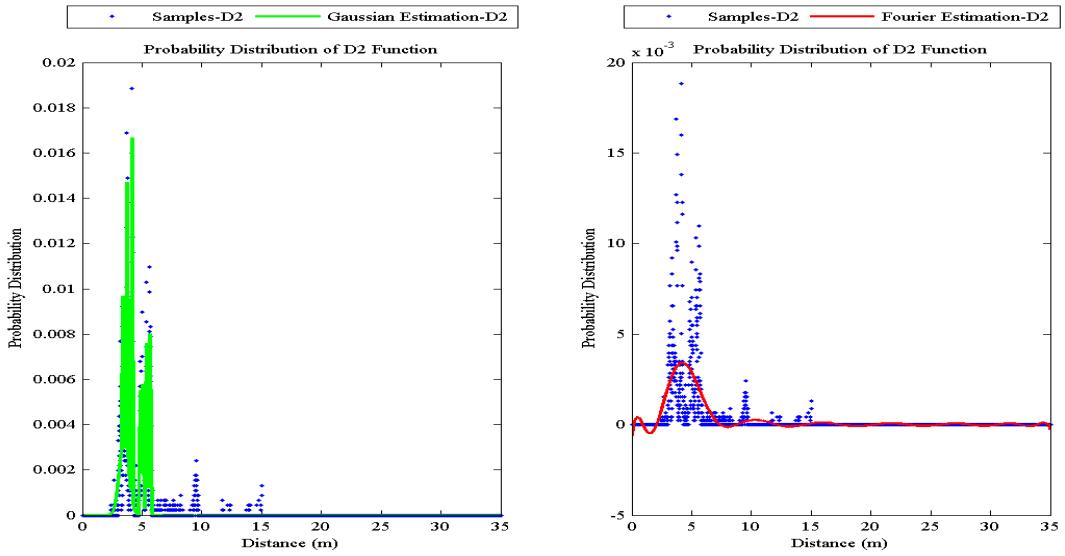

Figure 4.14: D_2 descriptor and the corresponding Gaussian and Fourier estimations for chair

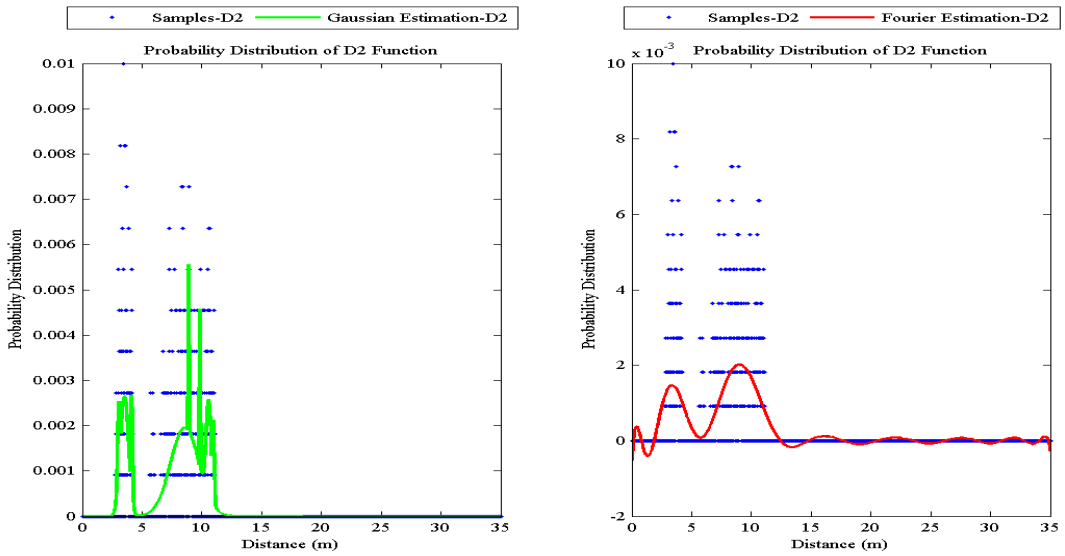

Figure 4.15: D_2 descriptor and the corresponding Gaussian and Fourier estimations for clip

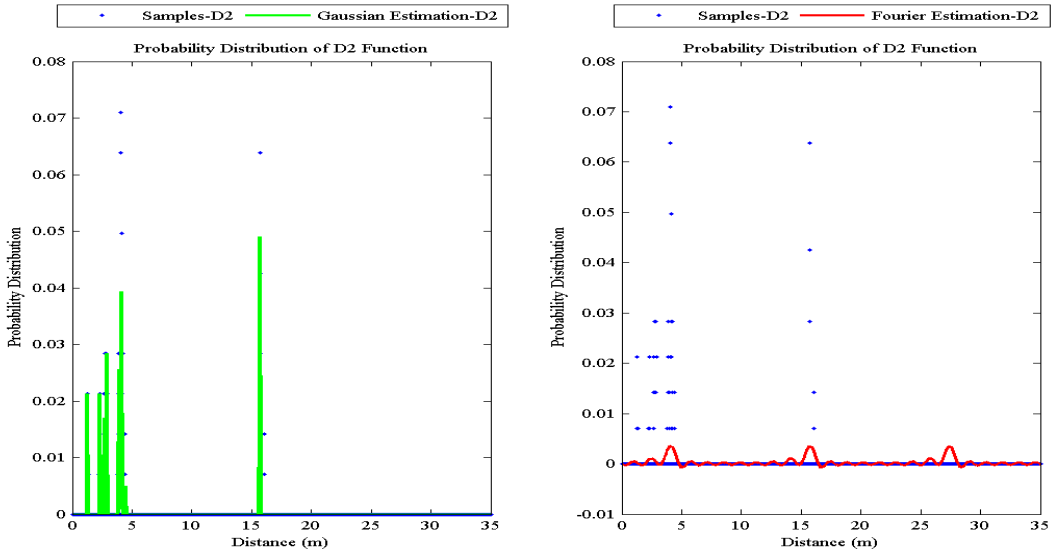

Figure 4.16: D_2 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

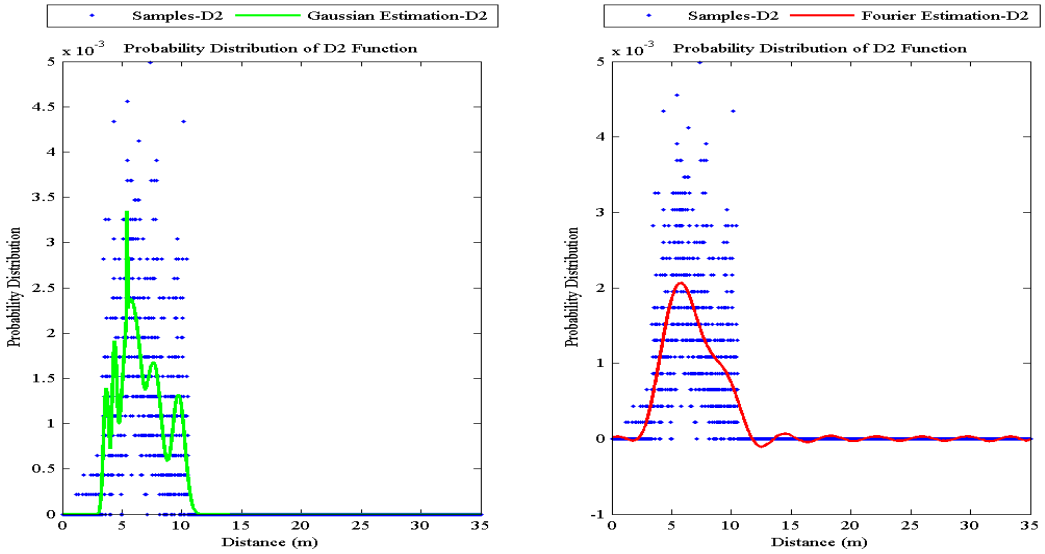

Figure 4.17: D_2 descriptor and the corresponding Gaussian and Fourier estimations for truck

Angle Descriptors

Angle—another primitive geometric attribute—contains of distinguishing values amid different objects. The angle descriptors examine triplets of points from the point set of an object, since an angular value appears between three non–linear points. Angle descriptors that are recruited in this study are the two following probability distribution definitions:

1. The probability distribution of angular measurements between three points, uniformly at random selected from the point set (A_1).

2. The probability distribution of angular measurements between two points, uniformly at random selected from the point set, and the center of geometry of the point set (A_2).

A_1 and A_2 are computed by sampling a population of triples and pairs of points from the point set with the same strategy as in D_1 and D_2. The sampling rate and the selection mechanism are exactly the same as D_1 and D_2 calculation—uniformly distributed random selection. The measured angular values are rounded to the closest angular fraction in the range of $[0, 2\pi]$. A Sinusoid (defined in Appendix A) and a Fourier (defined in Appendix A) estimation functions quantize the angular probability distribution for the Gaussian–Sinusoid–Polynomial feature vector and the Fourier–Polynomial feature vector, correspondingly. The Sinusoid estimator provides 24 approximation coefficients for each angular descriptor and the Fourier transformation function generates 18 numerical values. In total, angular descriptors add 48 numerical values to Gaussian–Sinusoid–Polynomial feature vector and 36 elements to Fourier–Polynomial feature vector.

Figure 4.18 to 4.22 display A_1 descriptor for the five selected objects and the corresponding Sinusoid and Fourier estimation. Figure 4.23 to 4.27 show A_2 descriptor and its approximated curves for the selected objects. The blue dots are the probability distributions, the solid green curve is the Sinusoid estimation and the solid red line represents the Fourier approximation. Angular descriptors estimate the density or the sparsness of the points in clouds.

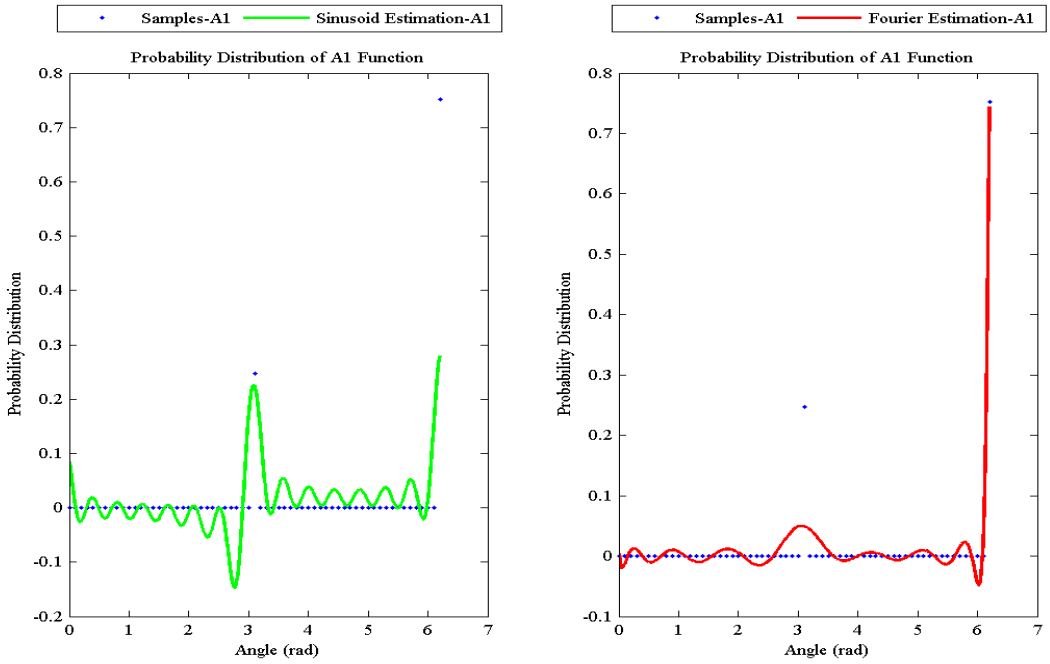

Figure 4.18: A_1 descriptor and the corresponding Sinusoid and Fourier estimations for boot

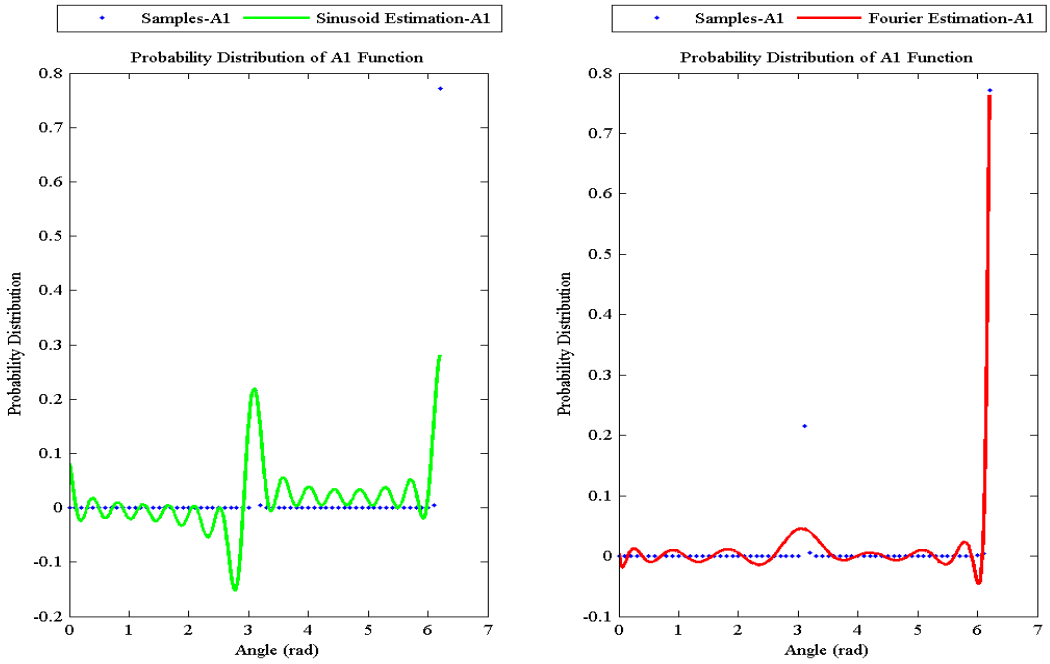

Figure 4.19: A_1 descriptor and the corresponding Sinusoid and Fourier estimations for chair

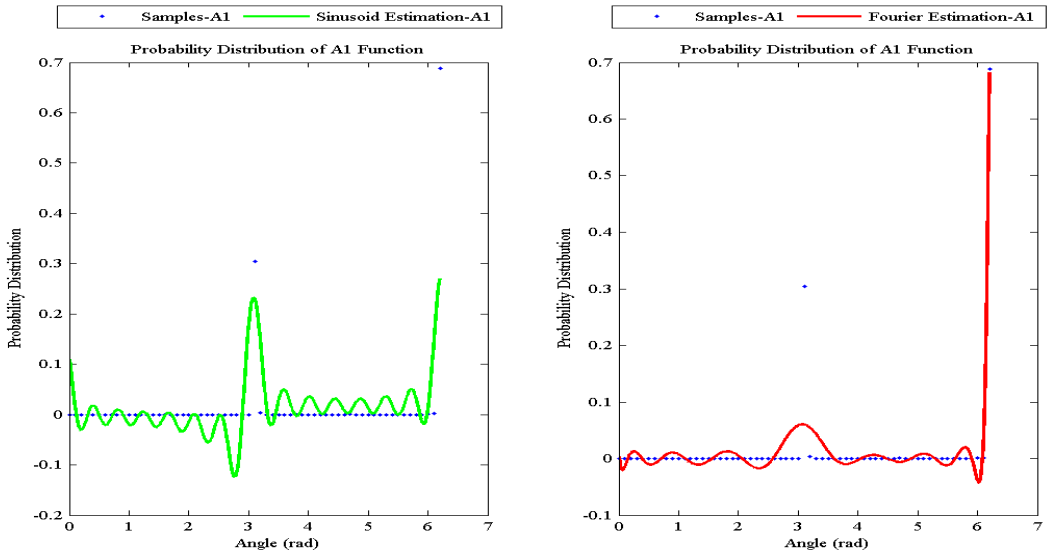

Figure 4.20: A_1 descriptor and the corresponding Sinusoid and Fourier estimations for clip

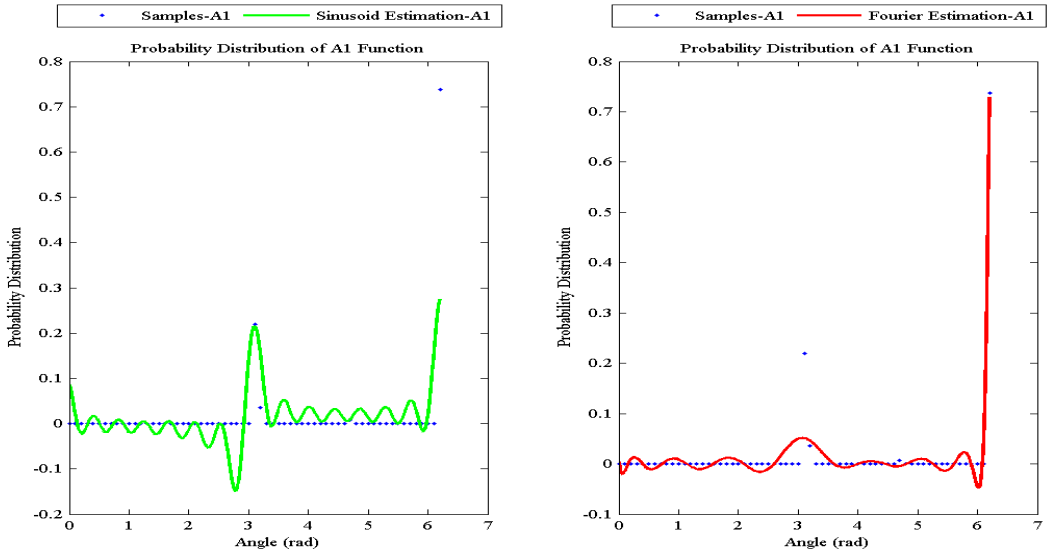

Figure 4.21: A_1 descriptor and the corresponding Sinusoid and Fourier estimations for glass bottle

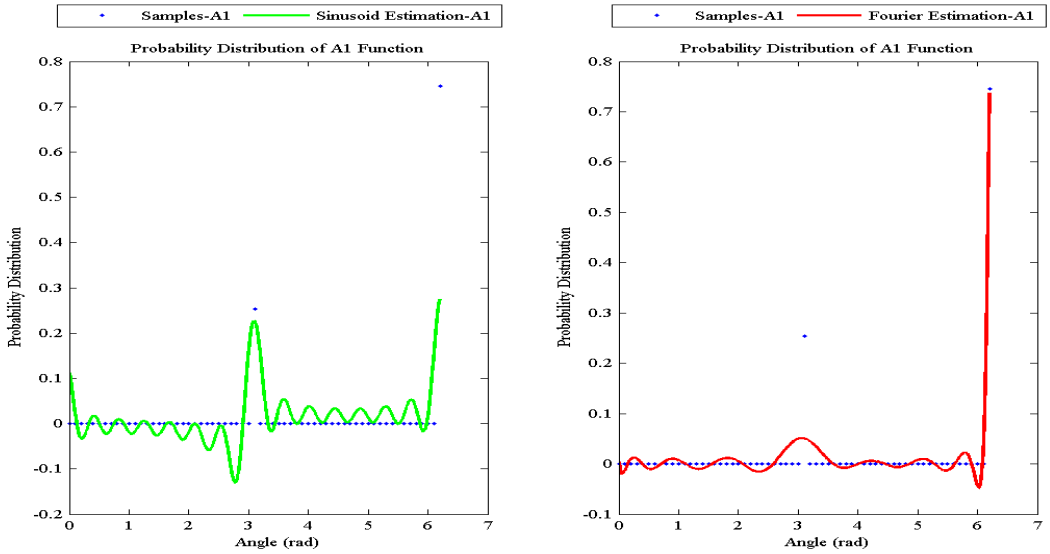

Figure 4.22: A_1 descriptor and the corresponding Sinusoid and Fourier estimations for truck

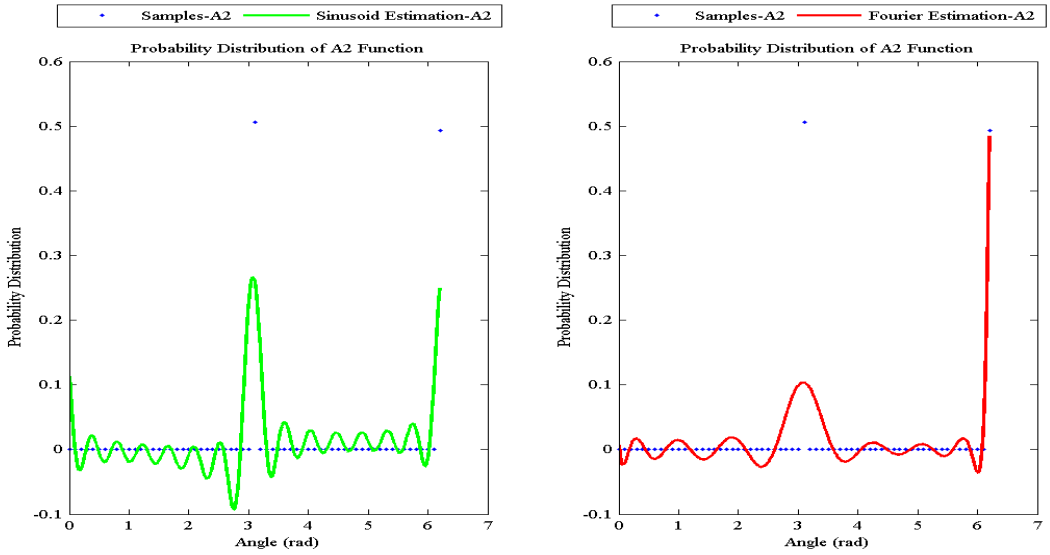

Figure 4.23: A_2 descriptor and the corresponding Gaussian and Fourier estimations for boot

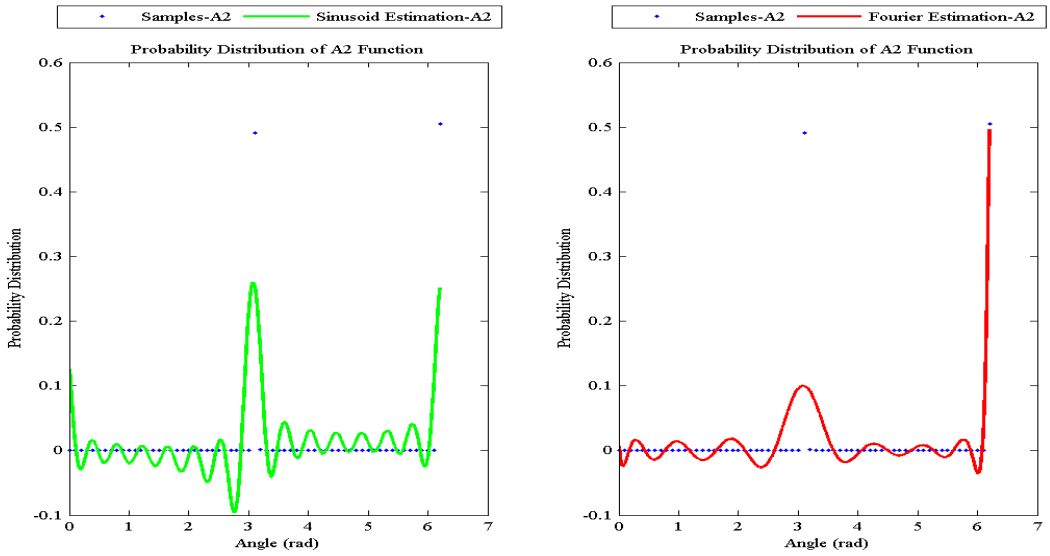

Figure 4.24: A_2 descriptor and the corresponding Gaussian and Fourier estimations for chair

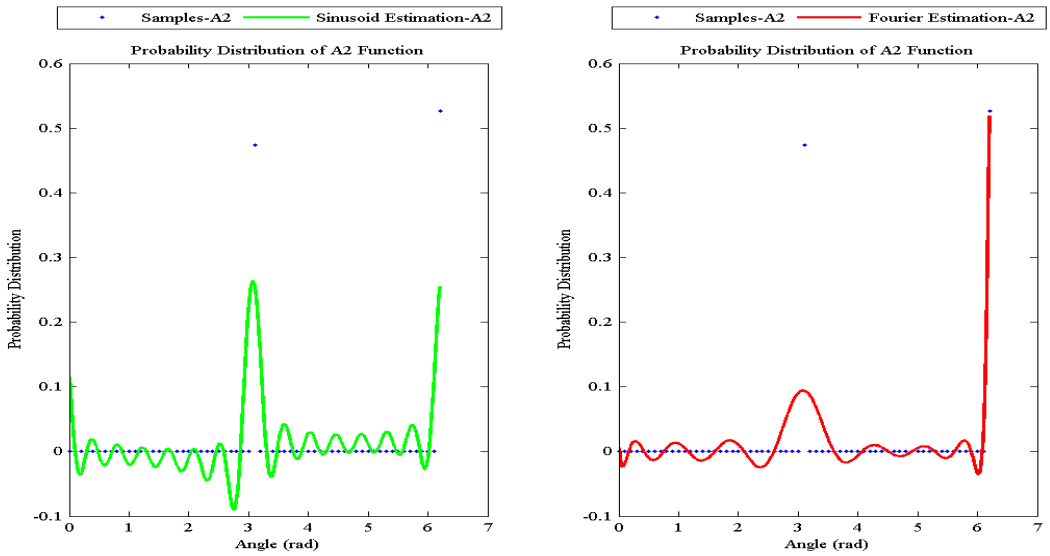

Figure 4.25: A_2 descriptor and the corresponding Gaussian and Fourier estimations for clip

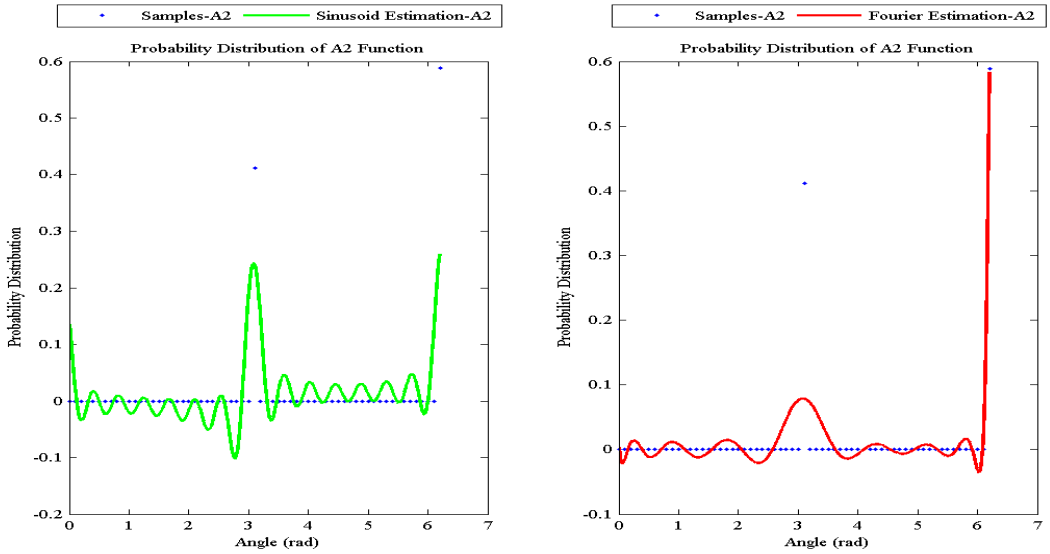

Figure 4.26: A_2 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

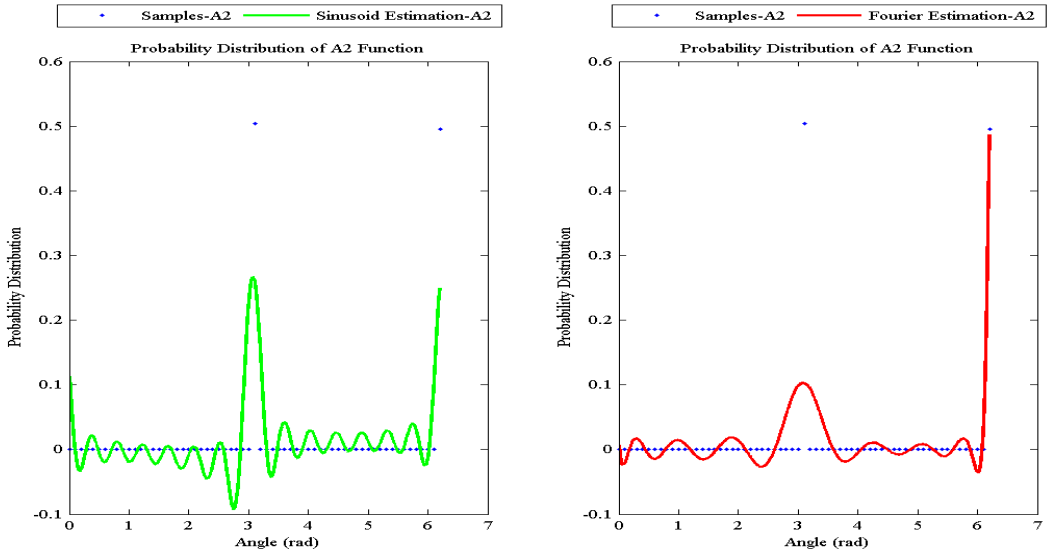

Figure 4.27: A_2 descriptor and the corresponding Gaussian and Fourier estimations for truck

Torsion Descriptors

Torsion or torsion angle represents the angular value between four points in space. The best definition for torsion would be the rotational angle between two triangles with a shared edge in which the vertices of the two triangles are the points in space—the rotational angle between two intersecting planes. Figure 4.28 presents an illustrative explanation of torsion angle. We proposed two torsion descriptors in this study as follows:

1. The probability distribution of the torsion angle measurement between four points, uniformly at random selected from the point set (T_1).

2. The probability distribution of the torsion angle measurement between three points, uniformly at random selected from the point cloud, and the center of geometry of the point set[3] (T_2).

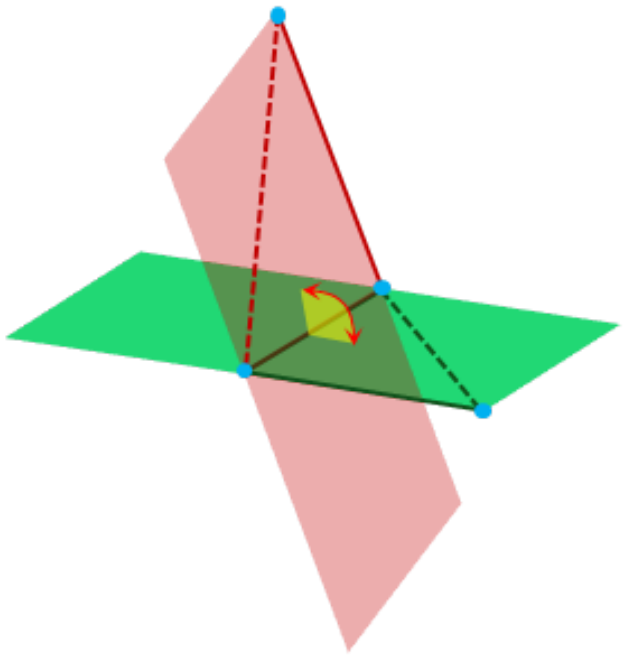

Figure 4.28: Torsion angle definition

An analogous approach to distance and angle descriptors is applied for T_1 and T_2 calculation. The angles are rounded up to the closest angular fraction in the range of $[0, 2\pi]$. Si-

[3]The fixed point is one of the two vertices on the shared edge

nusoid for Gaussian–Sinusoid–Polynomial feature vector and Fourier for Fourier–Polynomial feature vector estimate the torsion descriptors. The Sinusoid estimator generates 24 numerical values for each descriptor and the Fourier one provides 18. The length of Gaussian–Sinusoid–Polynomial feature vector by adding the torsion descriptors becomes 144 and the size of Fourier–Polynomial vector reaches to 108.

Figure 4.29 to 4.33 and 4.34 to 4.38 show the T_1 and T_2 descriptors, and their Sinusoid and Fourier estimation for the selected objects. Torsion descriptors differentiate the shape of the objects in contrast to the angle descriptors.

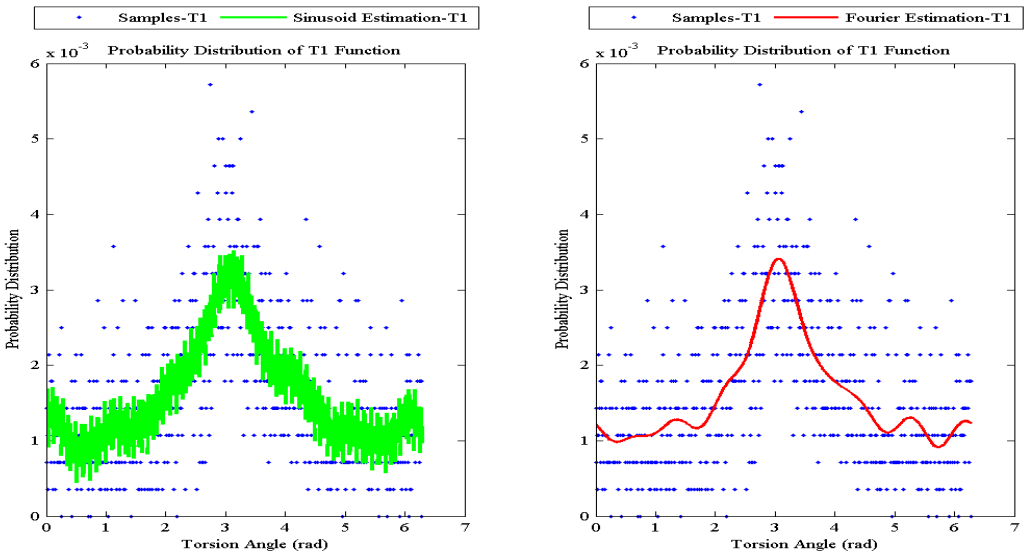

Figure 4.29: T_1 descriptor and the corresponding Gaussian and Fourier estimations for boot

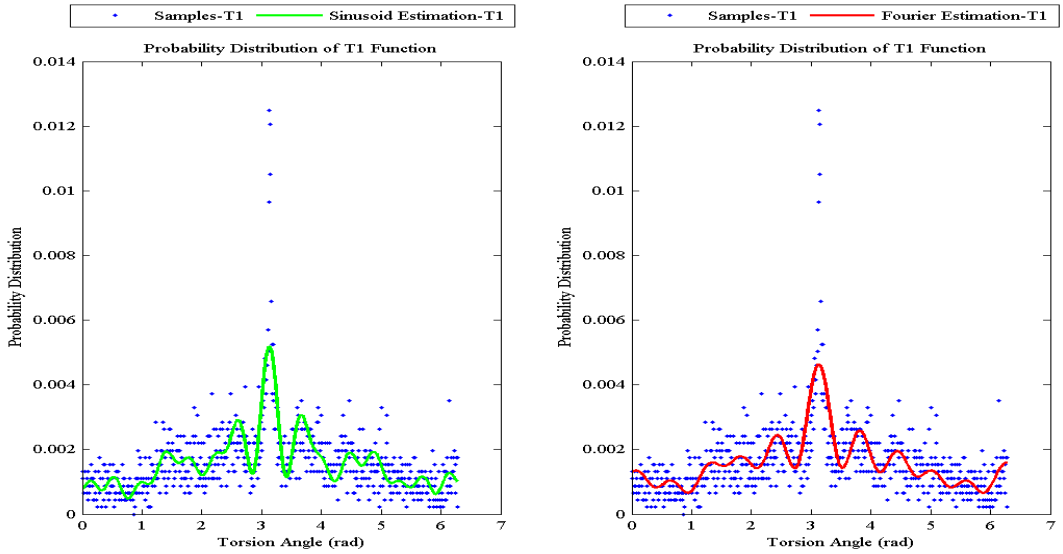

Figure 4.30: T_1 descriptor and the corresponding Gaussian and Fourier estimations for chair

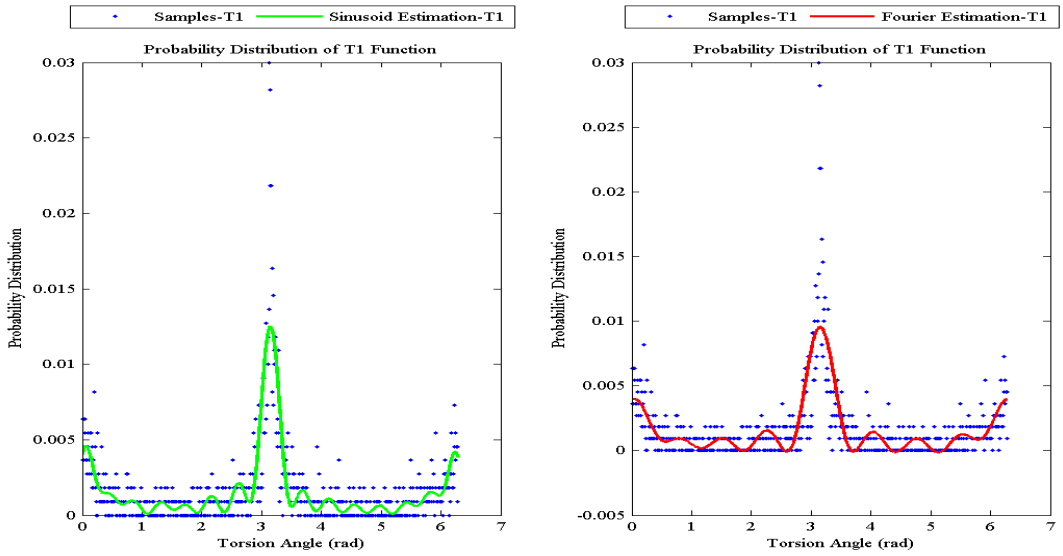

Figure 4.31: T_1 descriptor and the corresponding Gaussian and Fourier estimations for clip

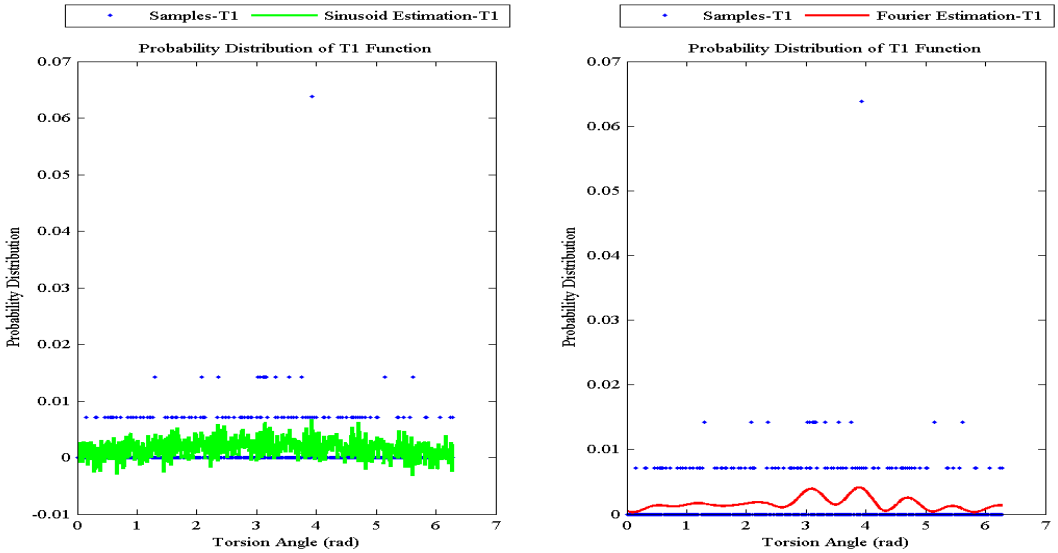

Figure 4.32: T_1 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

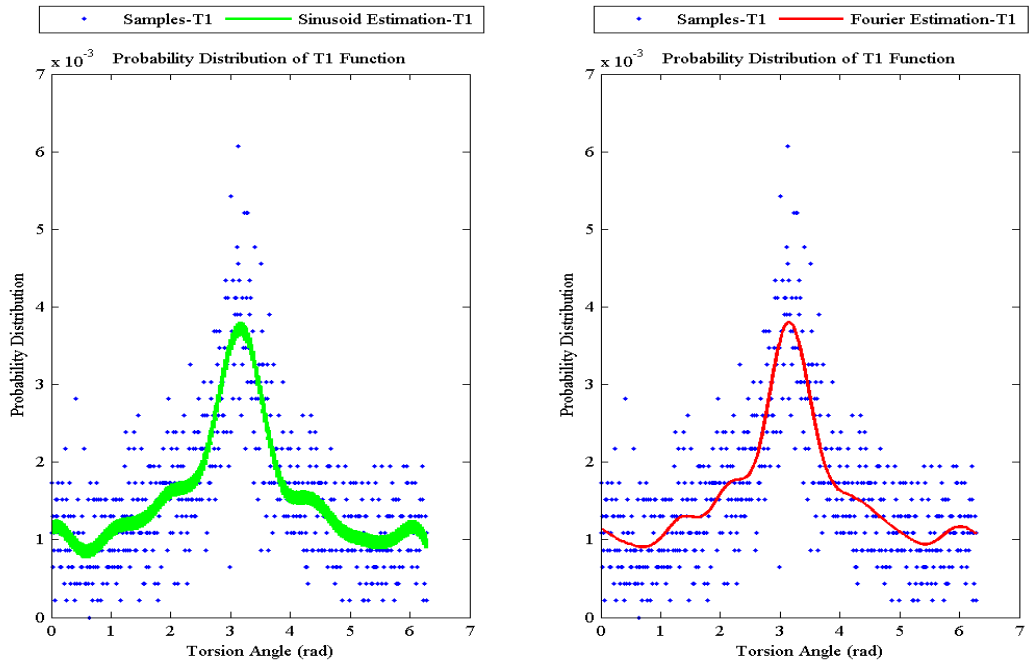

Figure 4.33: T_1 descriptor and the corresponding Gaussian and Fourier estimations for truck

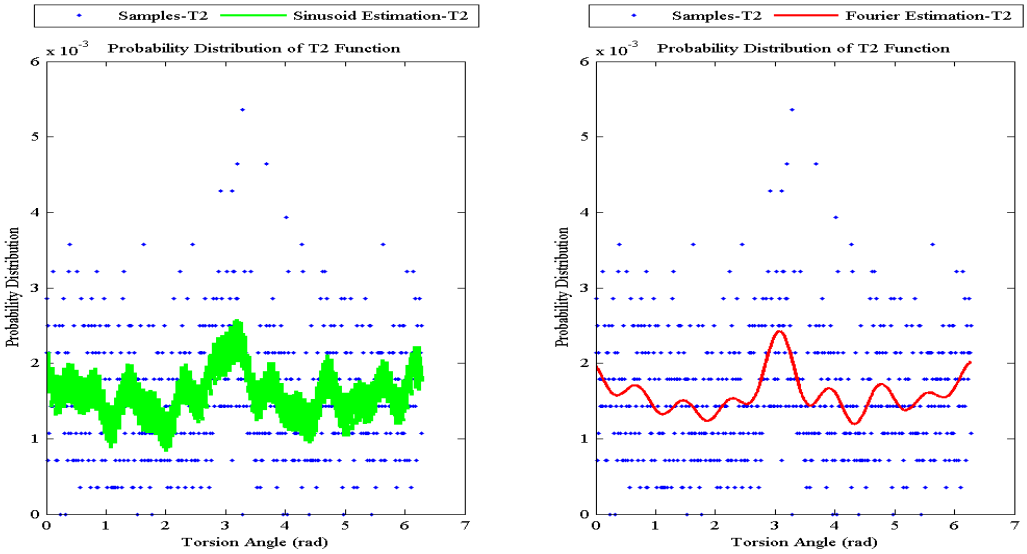

Figure 4.34: T_2 descriptor and the corresponding Gaussian and Fourier estimations for boot

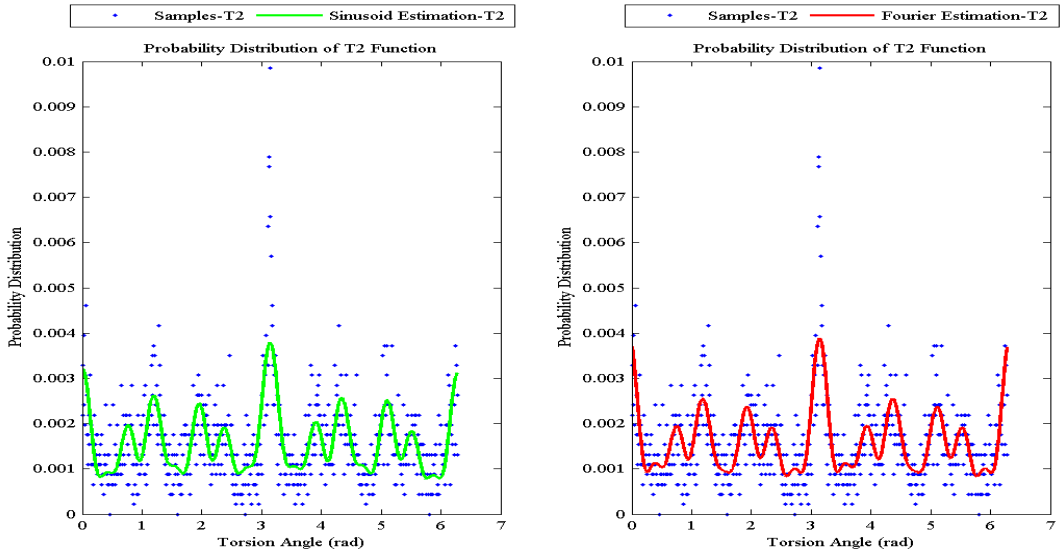

Figure 4.35: T_2 descriptor and the corresponding Gaussian and Fourier estimations for chair

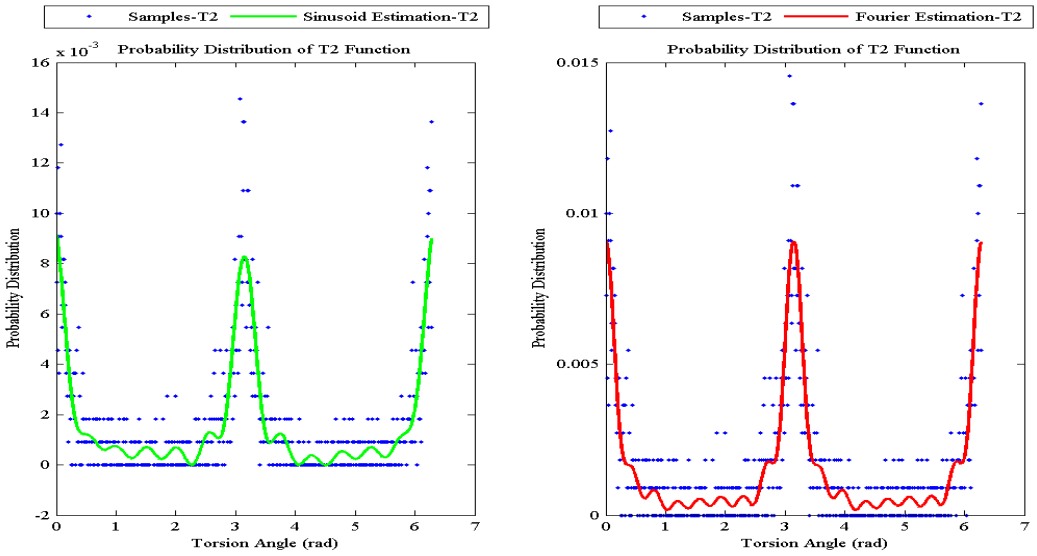

Figure 4.36: T_2 descriptor and the corresponding Gaussian and Fourier estimations for clip

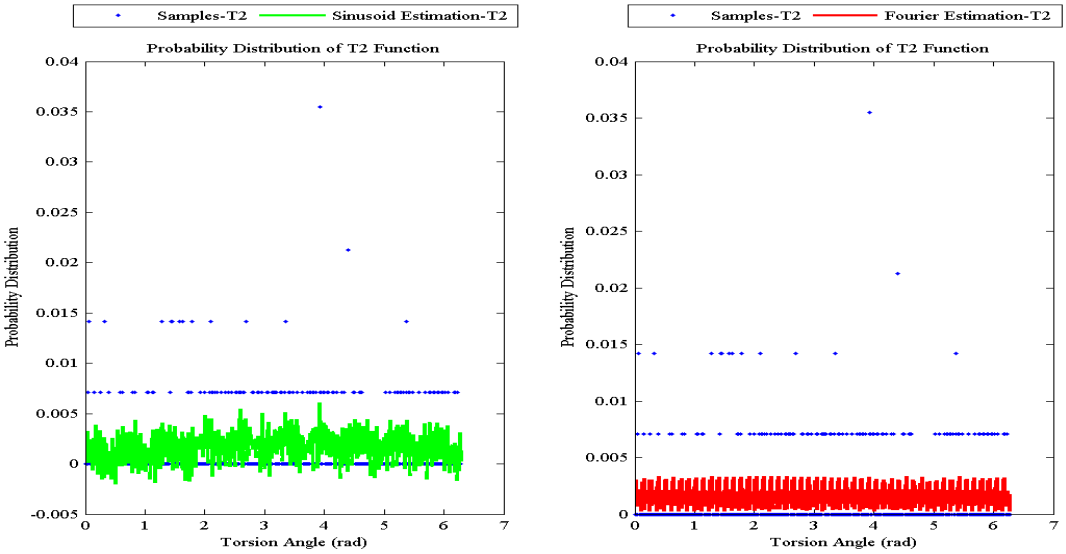

Figure 4.37: T_2 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

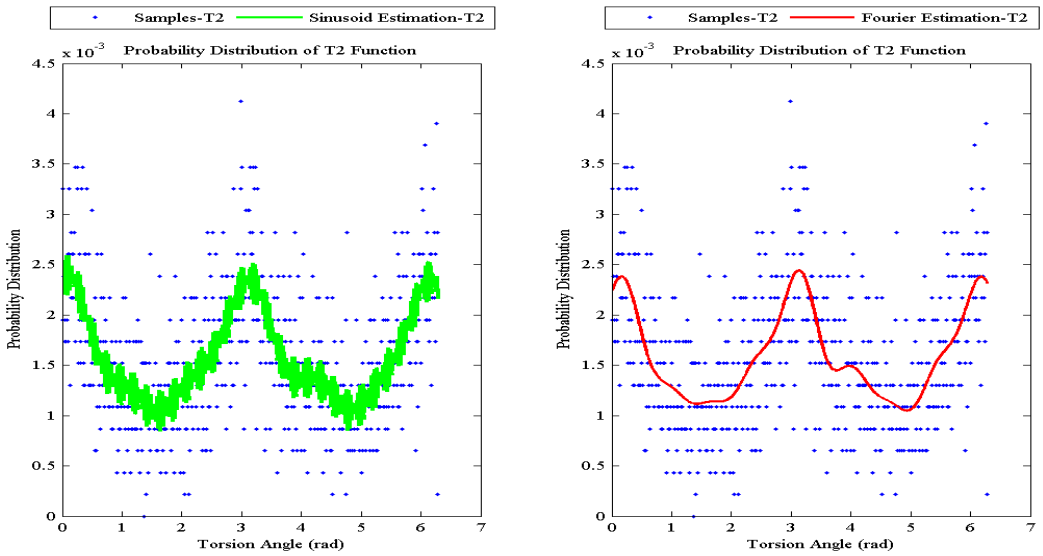

Figure 4.38: T_2 descriptor and the corresponding Gaussian and Fourier estimations for trcuk

Area Descriptors

Any region besieged by three non–linear points on a plane creates a closed surface. The area of this triangular surface makes another point of interest in the feature extraction phase of the recognition system. The area descriptors of our study are defined below:

1. The probability distribution of square root of triangular area measurement of the surface surrounded by three points, uniformly at random selected from the point set (Ar_1).

2. The probability distribution of square root of triangular area measurement of the surface surrounded by two points, uniformly at random selected from the point set, and the center of geometry of the point set (Ar_2).

The computation of Ar_1 and Ar_2 is accomplished likewise the aforementioned descriptors. The calculated square rooted triangular area, enclosed by three points, is rounded to the closest area fraction which varies in the range of $[0, \sqrt{\frac{3}{2}}l]$. In which l is the length of the surrounding cubic box created in the third step of normalization. The area descriptors are estimated by Gaussian and Fourier functions to complement the Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors. Area descriptors append 48 numerical coefficients to the former and 36 values to the latter one.

Figure 4.39 to 4.43 display the Ar_1 descriptor as blue dots, the Gaussian estimation of that in solid green and its Fourier transformation as solid red curve for the selected objects. The corresponding illustration for Ar_2 descriptor presents in Figure 4.44 to 4.48. Comparison of the resultant area descriptors for the objects in the dataset implies that these descriptors provide an estimation of each object's shape.

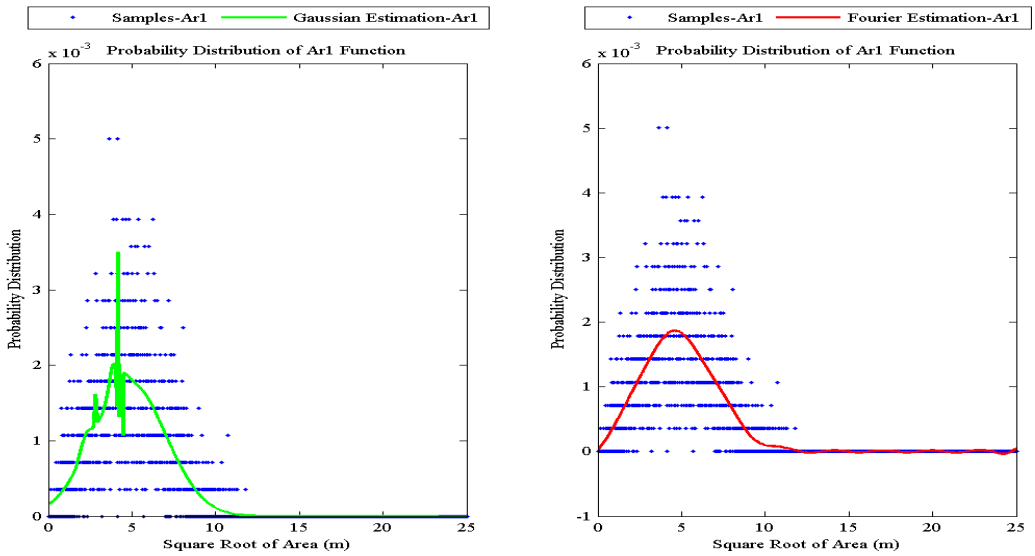

Figure 4.39: Ar_1 descriptor and the corresponding Gaussian and Fourier estimations for boot

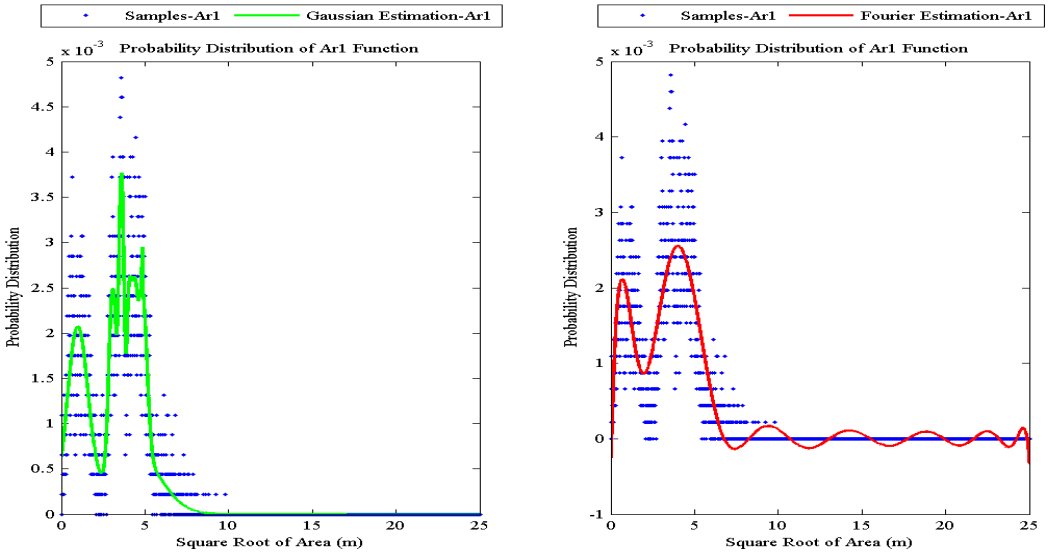

Figure 4.40: Ar_1 descriptor and the corresponding Gaussian and Fourier estimations for chair

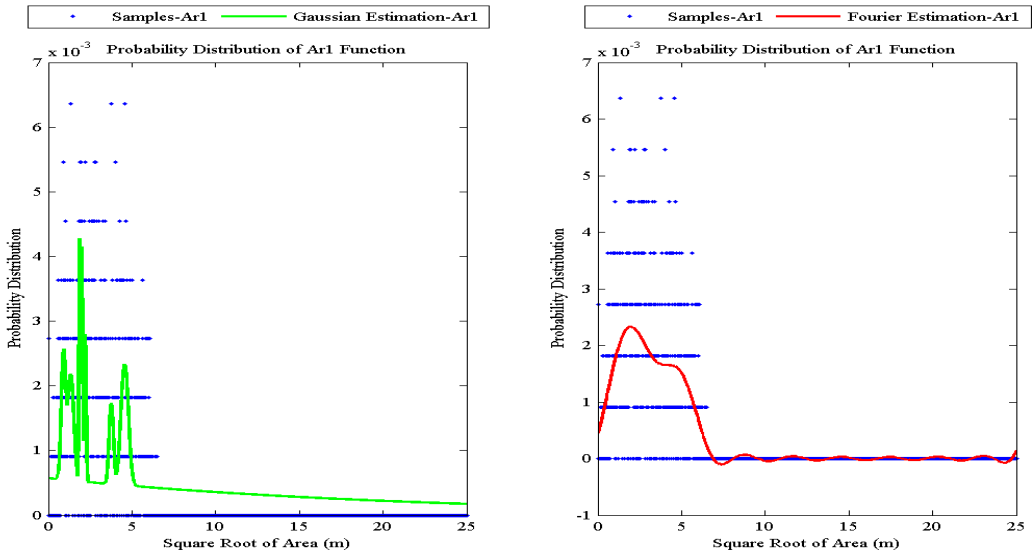

Figure 4.41: Ar_1 descriptor and the corresponding Gaussian and Fourier estimations for clip

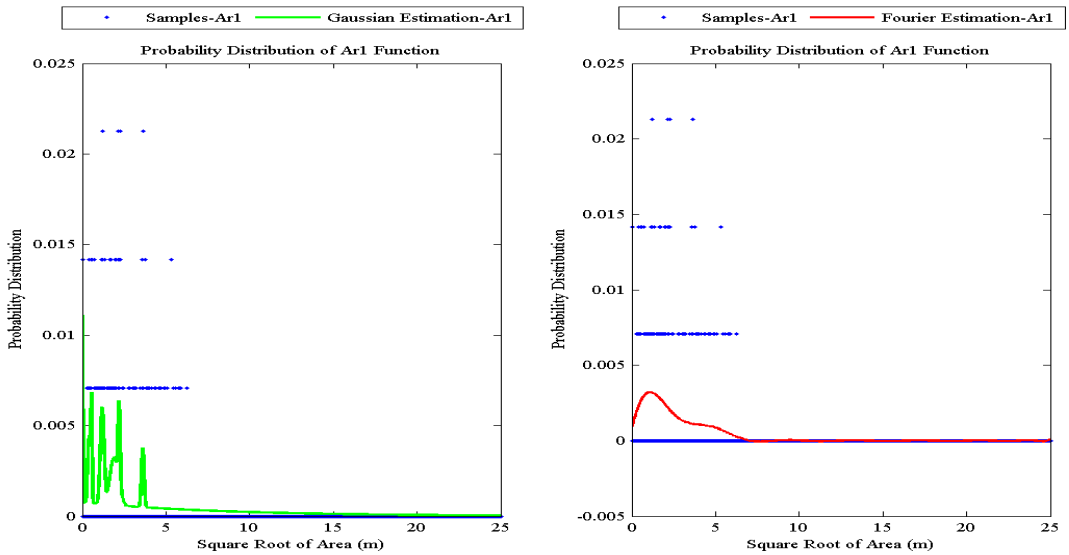

Figure 4.42: Ar_1 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

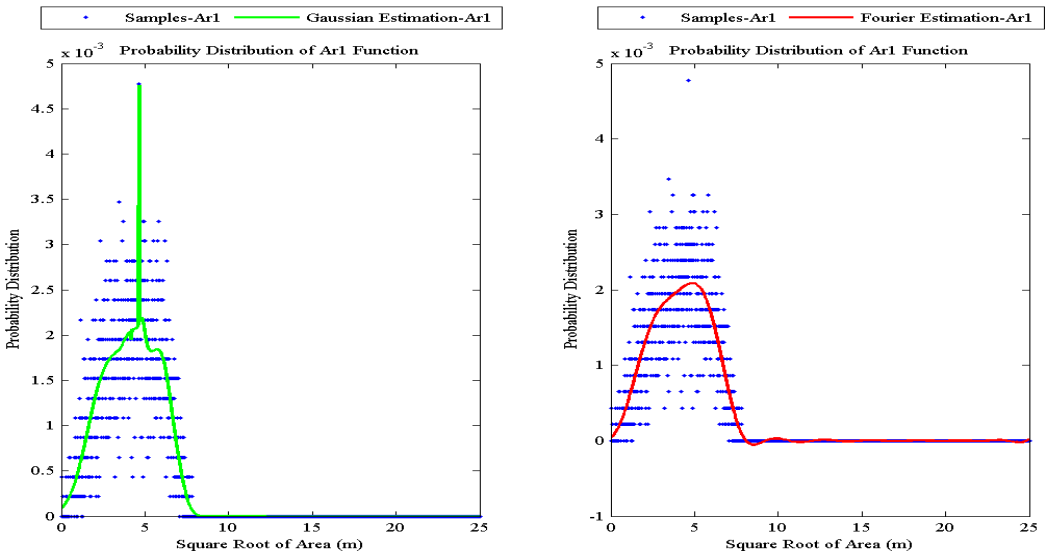

Figure 4.43: Ar_1 descriptor and the corresponding Gaussian and Fourier estimations for truck

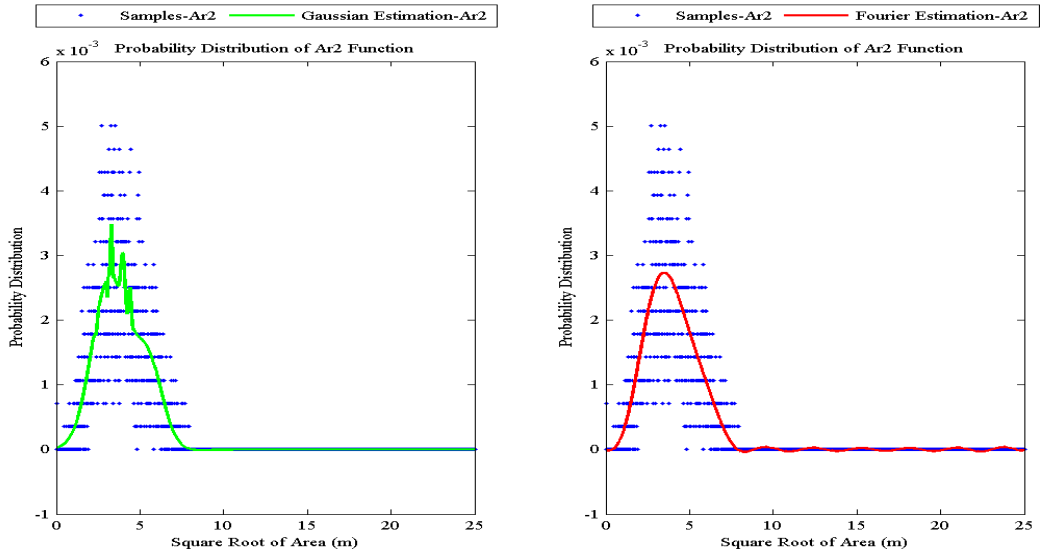

Figure 4.44: Ar_2 descriptor and the corresponding Gaussian and Fourier estimations for boot

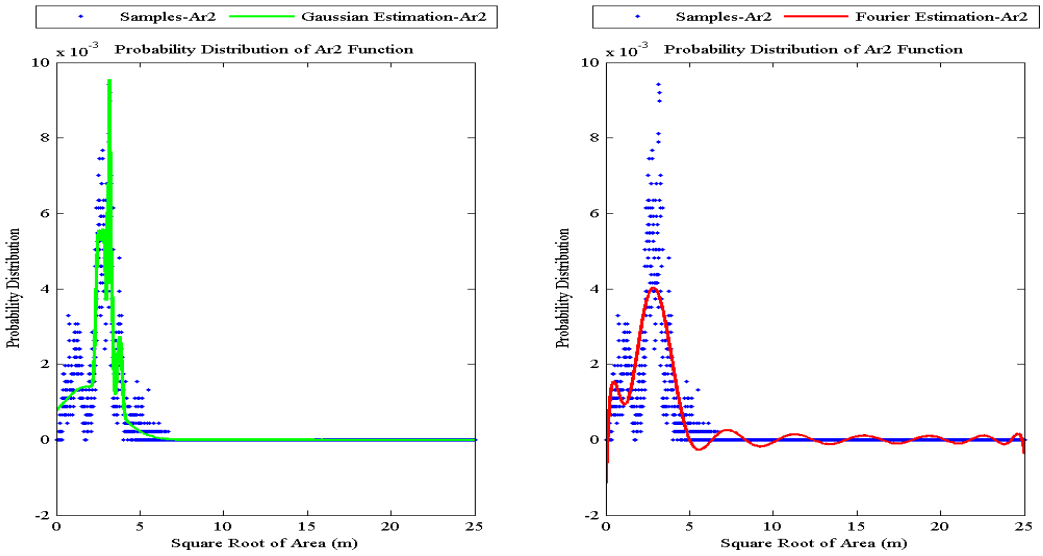

Figure 4.45: Ar_2 descriptor and the corresponding Gaussian and Fourier estimations for chair

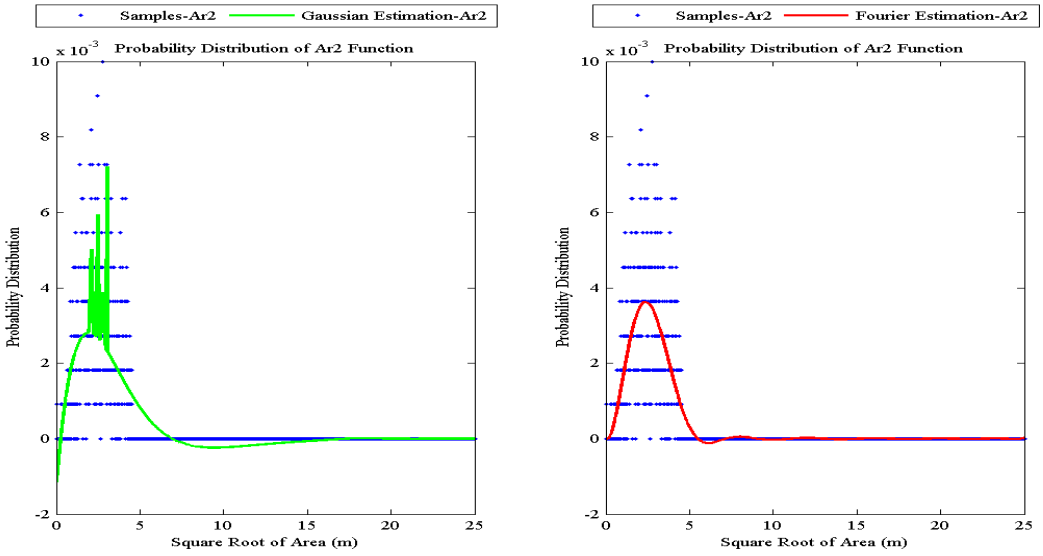

Figure 4.46: Ar_2 descriptor and the corresponding Gaussian and Fourier estimations for clip

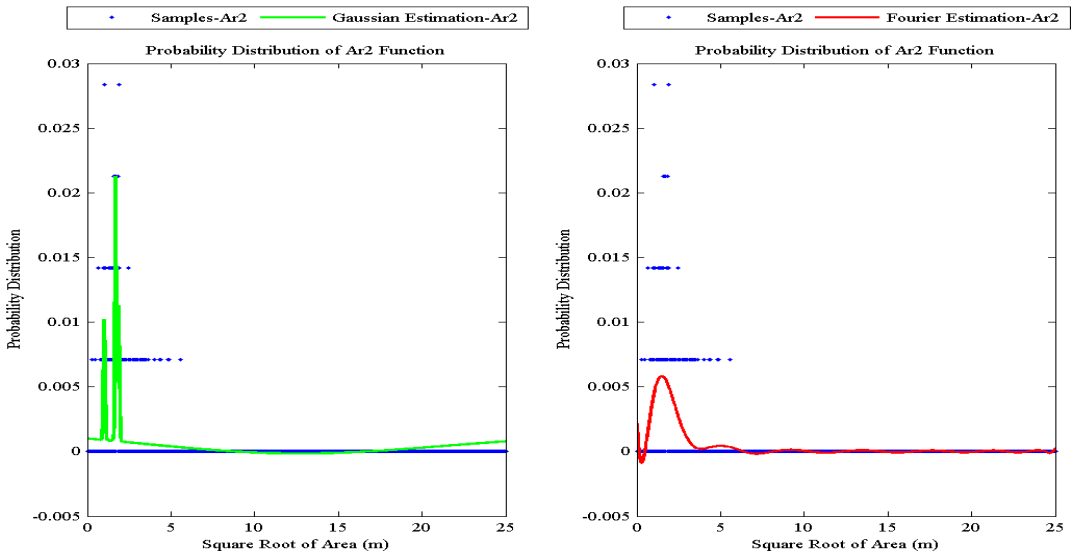

Figure 4.47: Ar_2 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

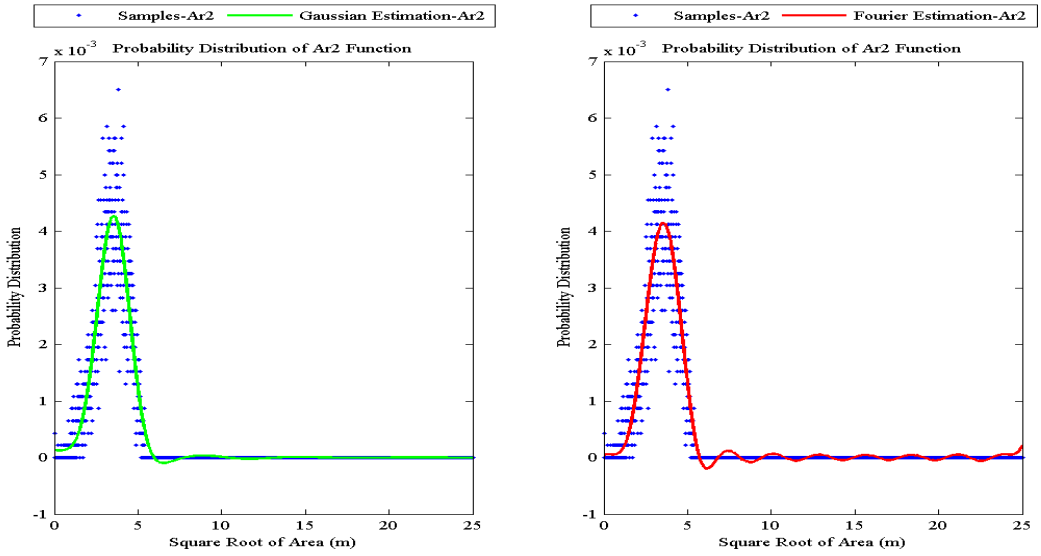

Figure 4.48: Ar_2 descriptor and the corresponding Gaussian and Fourier estimations for truck

Volume Descriptors

Another primitive geometric property known as volume is described by four non–planar points in space. The tetrahedral volumetric space occupation between four randomly selected points includes proper information to distinguish various objects from each other. The followings represent the volume descriptors utilized in the feature vectors of our proposed recognition system:

1. The probability distribution of cube root of tetrahedral volume measurement of the occupied space between four points, uniformly at random selected from the point set (V_1).

2. The probability distribution of cube root of tetrahedral volume measurement of the occupied space between three points, uniformly at random selected from the point set, and a fixed point which is the center of geometry of the point set[4] (V_2).

V_1 and V_2 are calculated with respect to the sampling population and the volume formula for tetrahedron—multiplication of the triangular area of the base and the height of the tetrahedron. The calculated cube rooted tetrahedral volume enclosed by four points is rounded to the closest volume fraction obtained by the fraction frequency varied in the range of $[0, \sqrt[3]{\frac{3}{2}}l]$. In which l is the length of the surrounding cubic box. The descriptors are estimated by Gaussian and Fourier approximation functions to add numerical elements into the Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors.

Figure 4.49 to 4.53 and 4.54 to 4.58 provide pictorial illustrations of V_1 and V_2 descriptors and the corresponding Gaussian and Fourier estimations.

[4]The fixed point is one of the vertices of the base triangle

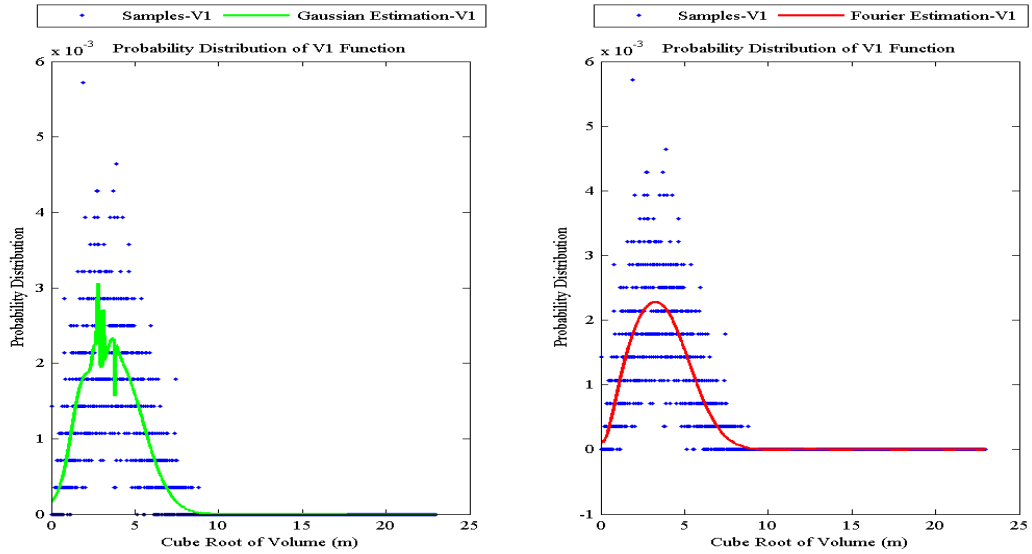

Figure 4.49: V_1 descriptor and the corresponding Gaussian and Fourier estimations for boot

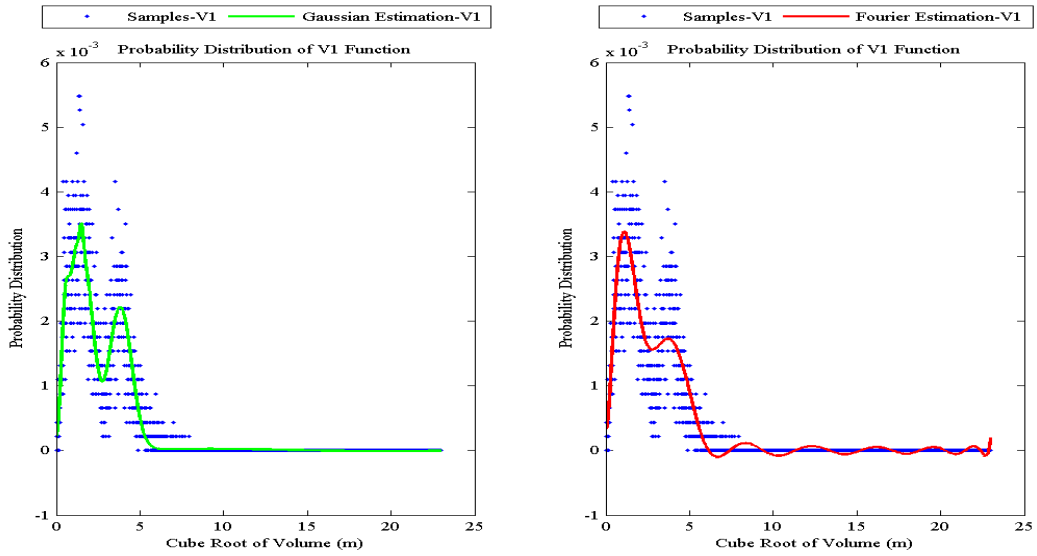

Figure 4.50: V_1 descriptor and the corresponding Gaussian and Fourier estimations for chair

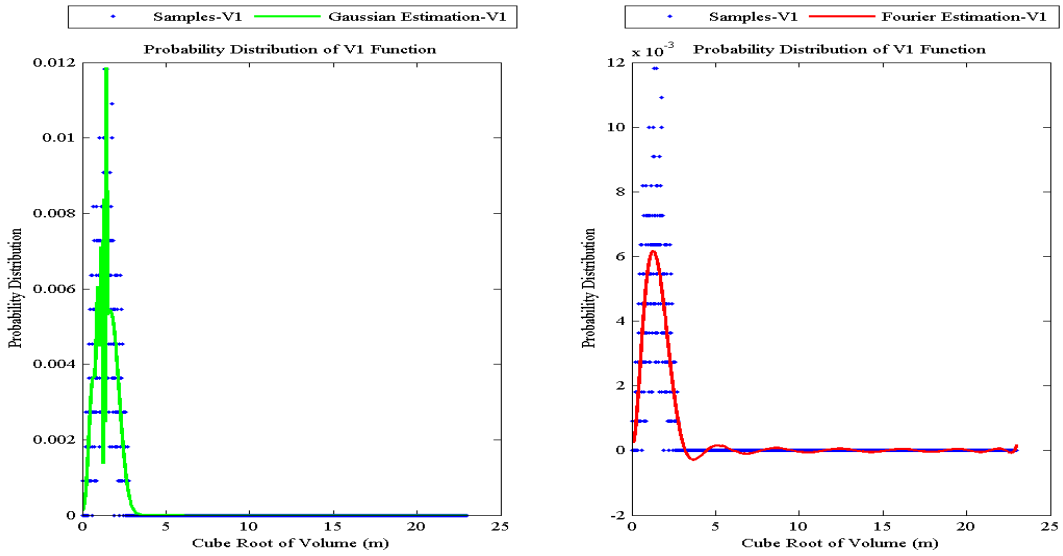

Figure 4.51: V_1 descriptor and the corresponding Gaussian and Fourier estimations for clip

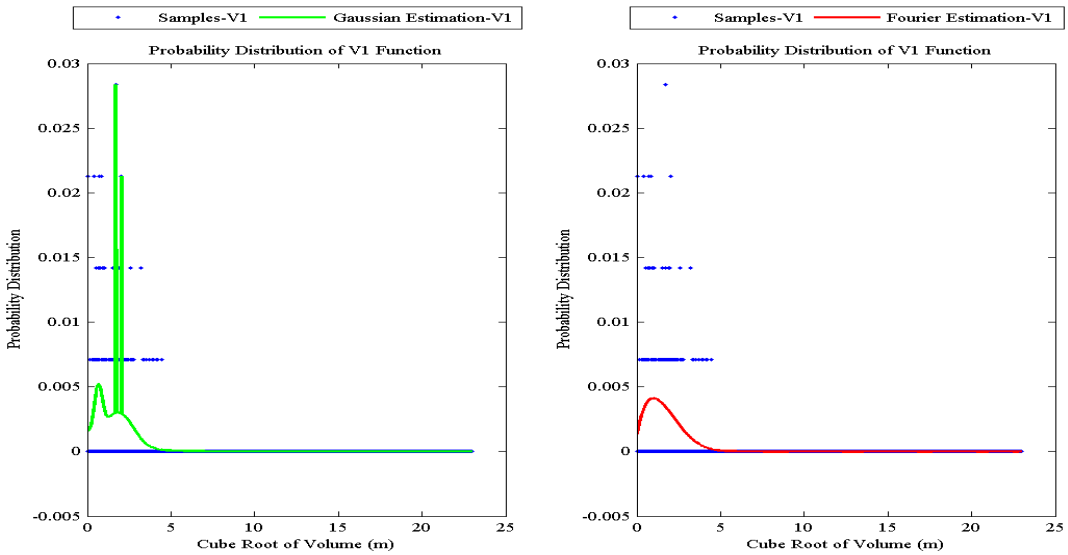

Figure 4.52: V_1 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

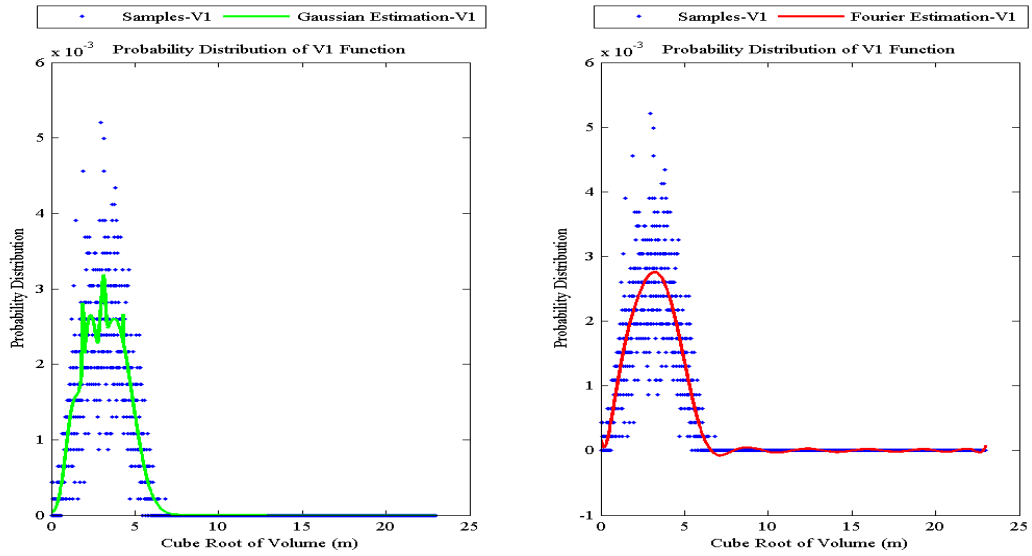

Figure 4.53: V_1 descriptor and the corresponding Gaussian and Fourier estimations for truck

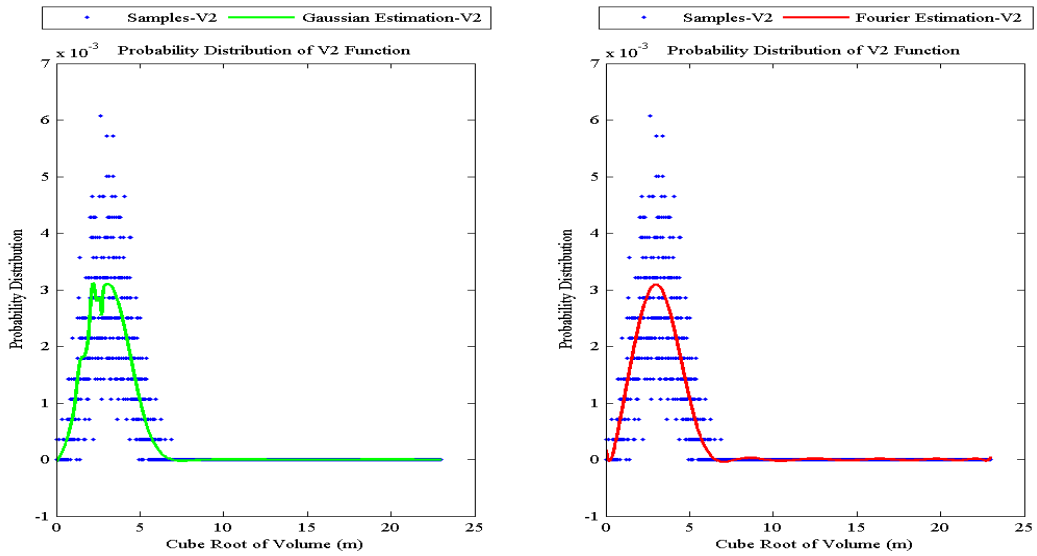

Figure 4.54: V_2 descriptor and the corresponding Gaussian and Fourier estimations for boot

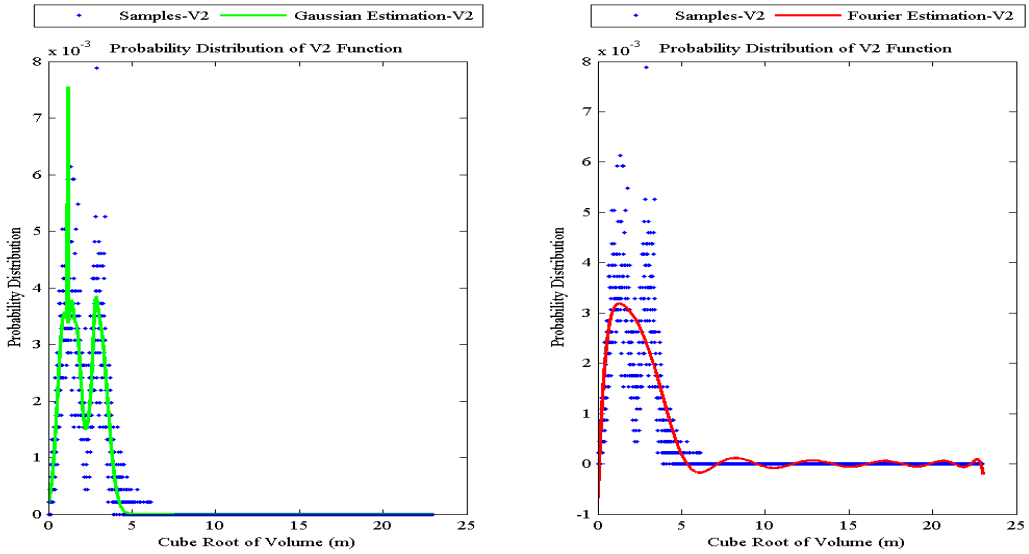

Figure 4.55: V_2 descriptor and the corresponding Gaussian and Fourier estimations for chair

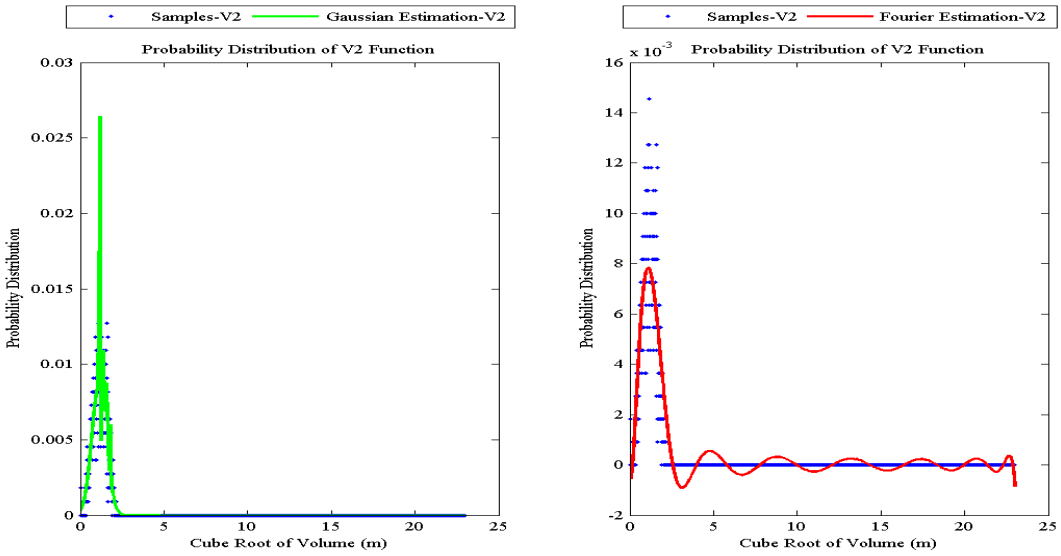

Figure 4.56: V_2 descriptor and the corresponding Gaussian and Fourier estimations for clip

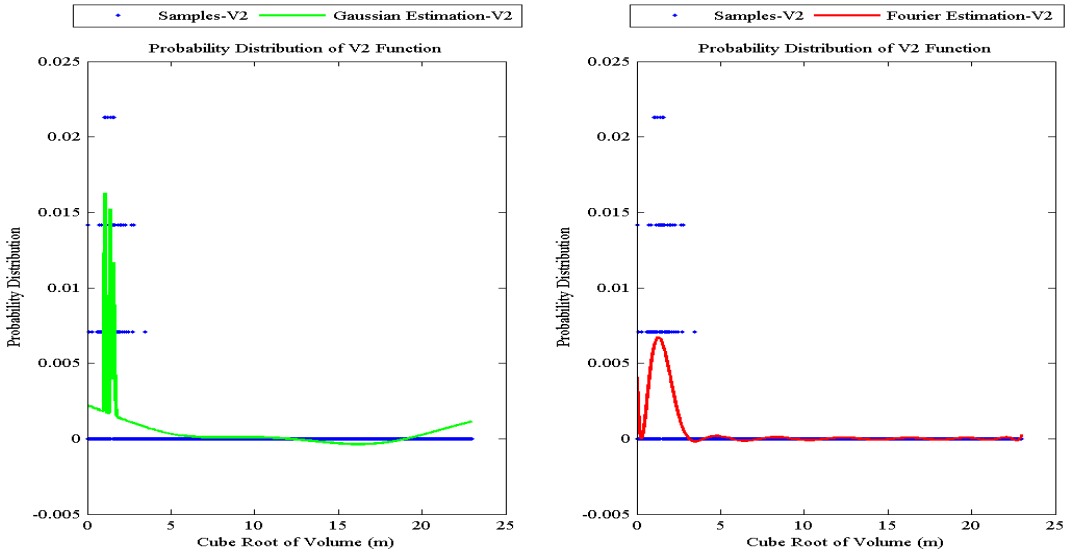

Figure 4.57: V_2 descriptor and the corresponding Gaussian and Fourier estimations for glass bottle

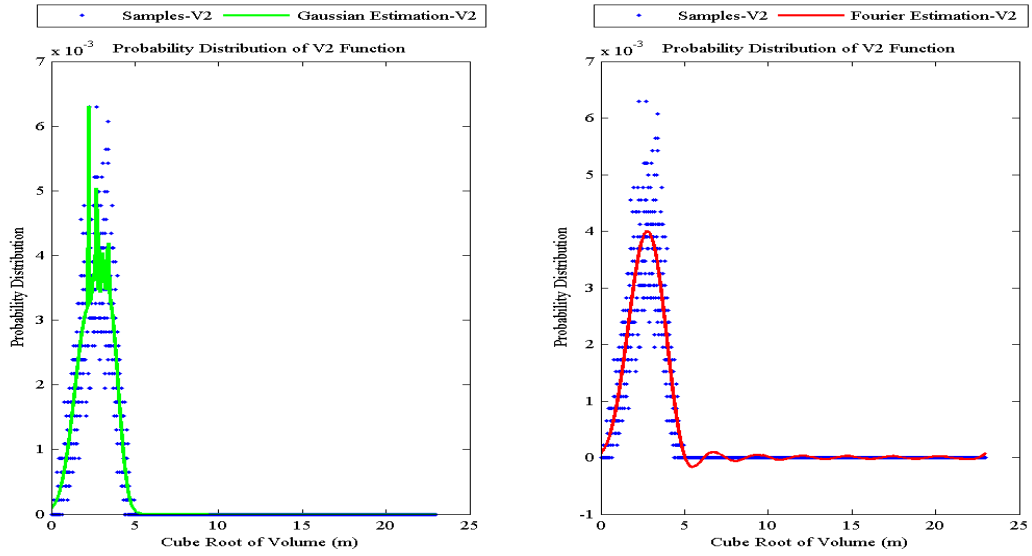

Figure 4.58: V_2 descriptor and the corresponding Gaussian and Fourier estimations for truck

Reconstructed Surface Descriptor

3D point cloud surface reconstruction methods, discussed in Chapter 2, are classified into two broad categories: (1) Explicit and (2) Implicit. Explicit methods triangulate the point cloud's surface of an object; whilst implicit techniques estimate the cloud's surface with functional representations. However these approaches build a closed surface which wraps around the point cloud, another form of representation, similar to implicit surfaces, exists to simulate the scatter points' surface. This representation estimates the surface of a point cloud with an open contour. The zero set of the representative function or convolution of functions defines the open contour—set of coefficients which values almost all the input coordinates to zero. The resultant contour describes the best approximated surface that the points in the cloud could lie on—regression of the points in 3D space.

Gaussian function, convolution of Gaussian functions with different μ and σ, Radial Basis Functions, Polynomial function and sum of Polynomial functions exemplify some of the estimators or open surface reconstructors. In order to approximate the point cloud data in the collected dataset, we employed a particular summation of Polynomial functions[5]. This estimation function produces 21 numerical coefficients. These values are added to both Gaussian–Sinusoid–Polynomial feature vector and Fourier–Polynomial feature vector.

Figure 4.59 to 4.63 illustrate the estimated surface for the selected objects of the collected dataset.

Degree of Symmetricity

One characteristic that human's cognitive system may utilize to recognize objects could include how symmetric an object is. This hypothesis origintes from the thought that a primitive volumetric shape such as cube, cylinder, sphere and etc., can formulate an object according to its symmetricity degree; i.e., an overall shape of an object is recognized with

[5]Appendix A explains the simple and the particular Polynomial functions for 3D surface reconstruction, in detail.

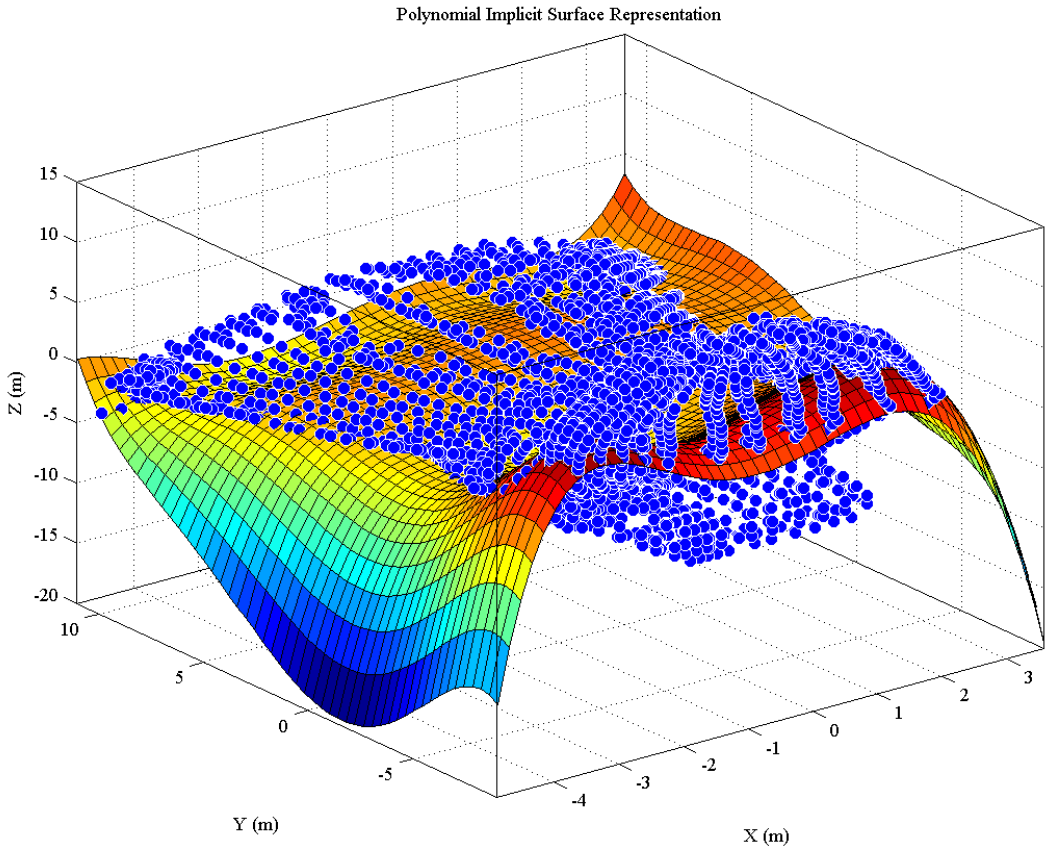

Figure 4.59: Scatter point representation and the corresponding reconstructed Polynomial surface for boot

respect to the symmetricty aspect of the object. This subject proposes a simple and novel methodology to estimate the symmetricity of an object through the point cloud processing. The proposed method performs based on the most important and useful characteristic of sphere. According to the proposed technique, this method is called spherical ray–based degree of symmetricity analysis.

Spherical ray–based degree of symmetricity considers a sphere as an enclosing space around the point cloud of an object. From the center of the sphere (the center of geometry of the cloud), radial rays are emitted along 13 critical directions and their mirrors. In total, 26 directions are inspected to estimate the degree of symmetricity of a cloud. Figure 4.64

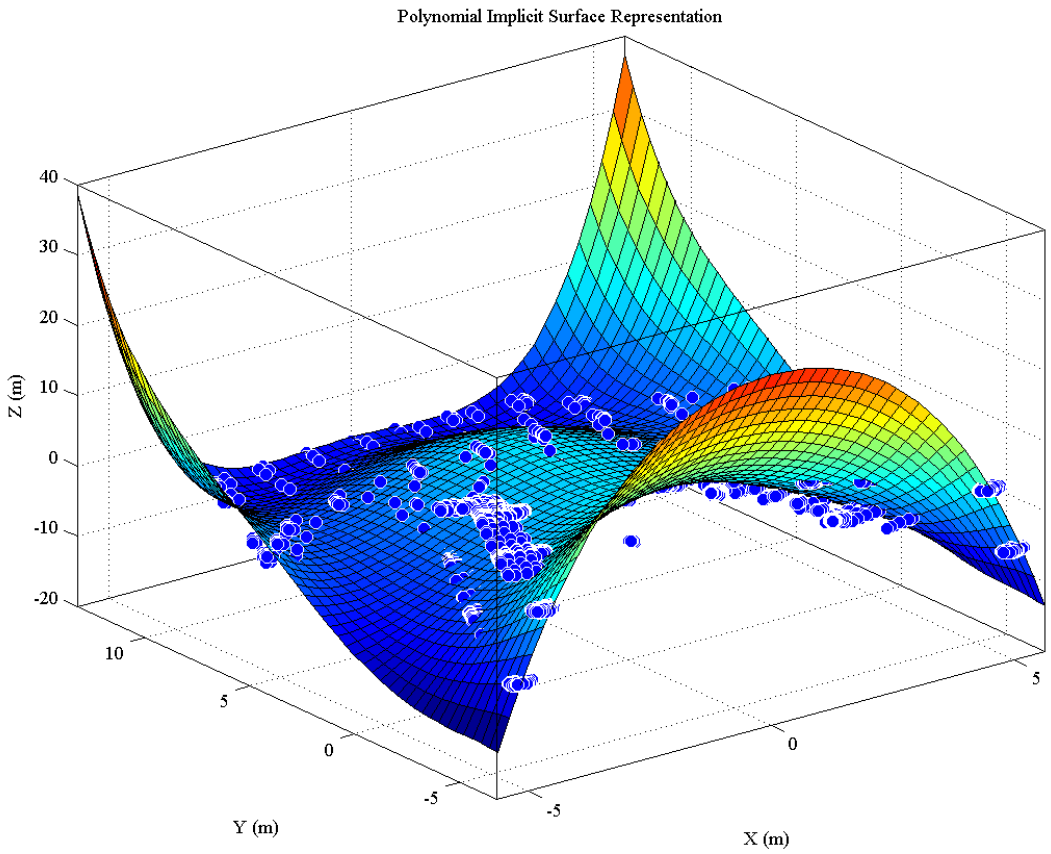

Figure 4.60: Scatter point representation and the corresponding reconstructed Polynomial surface for chair

displays these critical directions in solid arrows and their mirrors in dashed arrows.

The examination process of spherical ray–based technique is as follows.

1. Considering a relatively small sphere on the intersection of the point cloud surface and the emitted ray along a critical direction from the center of the surrounding sphere which happens to be the center of geometry of the cloud (size of the small sphere is a fraction of the surrounding sphere's radius),
2. Inspecting all the points of the cloud inside the small intersection sphere to calculate

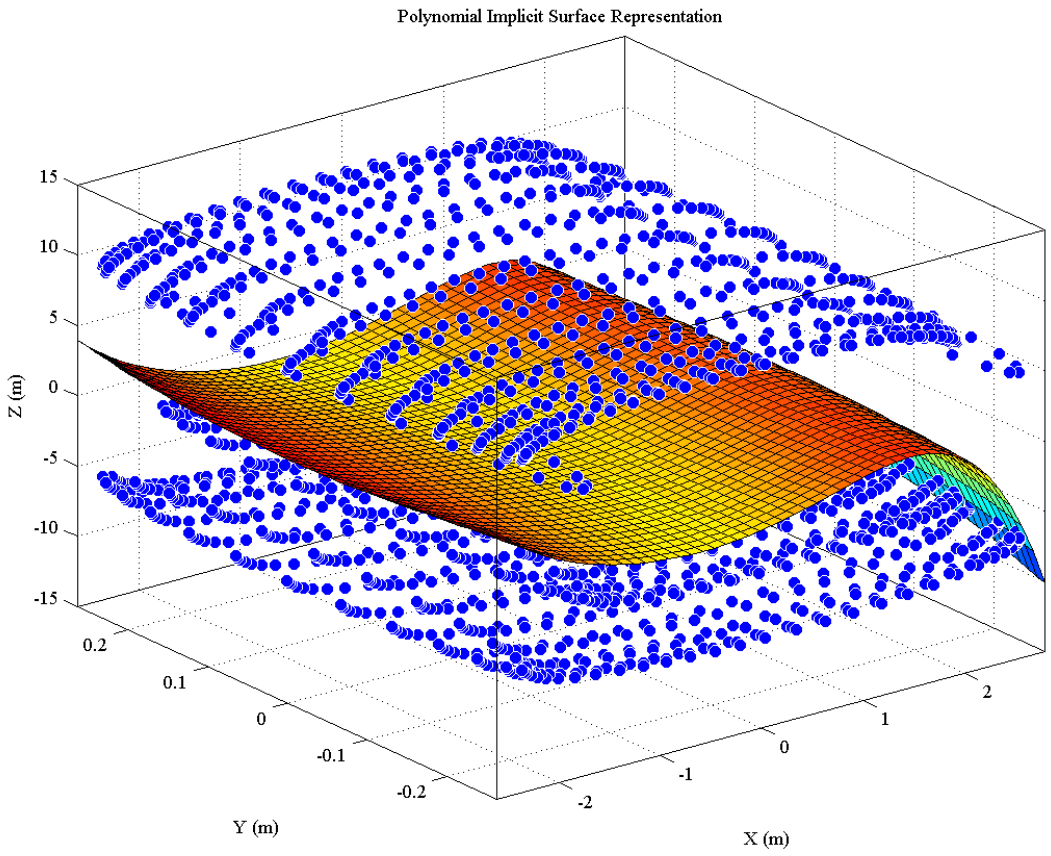

Figure 4.61: Scatter point representation and the corresponding reconstructed Polynomial surface for clip

the distance of the points to the center of the sphere,

3. Choosing the point that owns the maximum distance along the inspected direction, as the elongation or the extension of the object along that direction,
4. Computing the same attribute for the mirror direction of the inspected critical direction,
5. Subtracting the elongation value along the critical direction from the value along the mirror direction,
6. Counting the inspected direction as one degree of symmetricity if the subtraction lies inside a small sphere (same size of the intersecting sphere) around the center of the

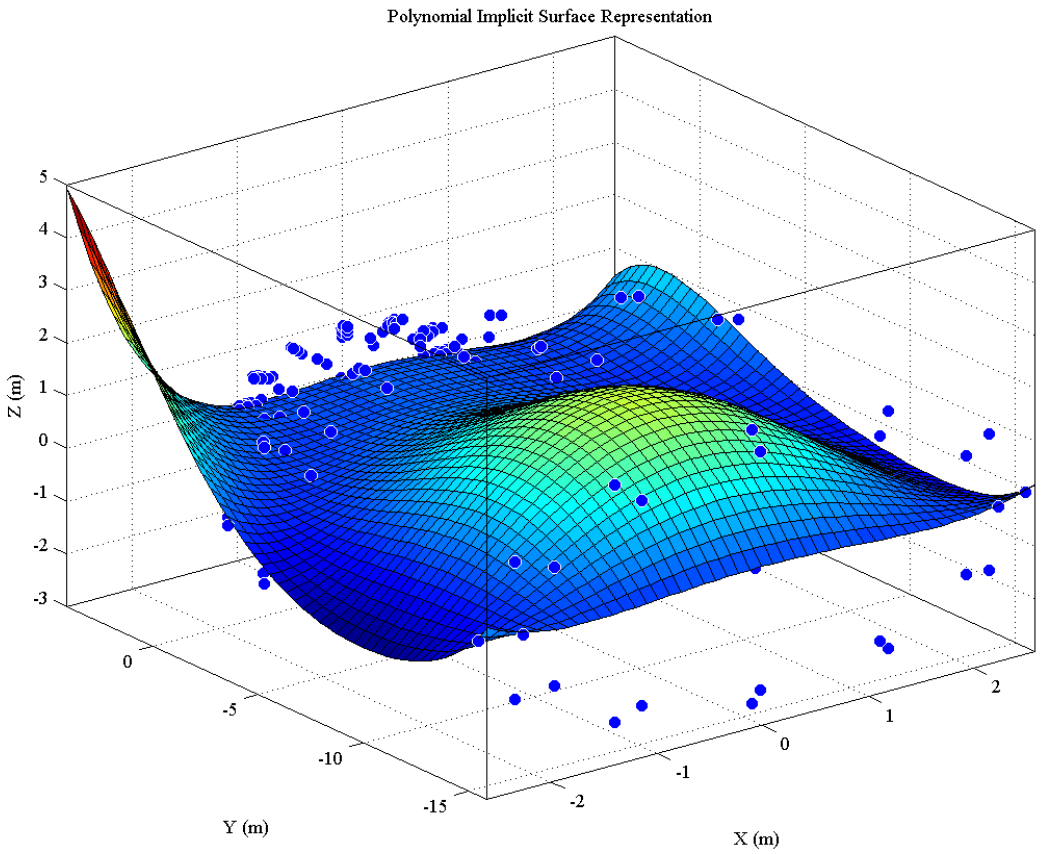

Figure 4.62: Scatter point representation and the corresponding reconstructed Polynomial surface for glass bottle

surrounding sphere,

7. Applying the method for all the critical directions and considering the number of directions that are qualified as symmetric directions as the degree of symmetricity of the point cloud,
8. Rotating the point cloud around the center of geometry by θ degree,
9. Performing the above procedure multiple times and averaging the calculated symmetricity degree of iterations.

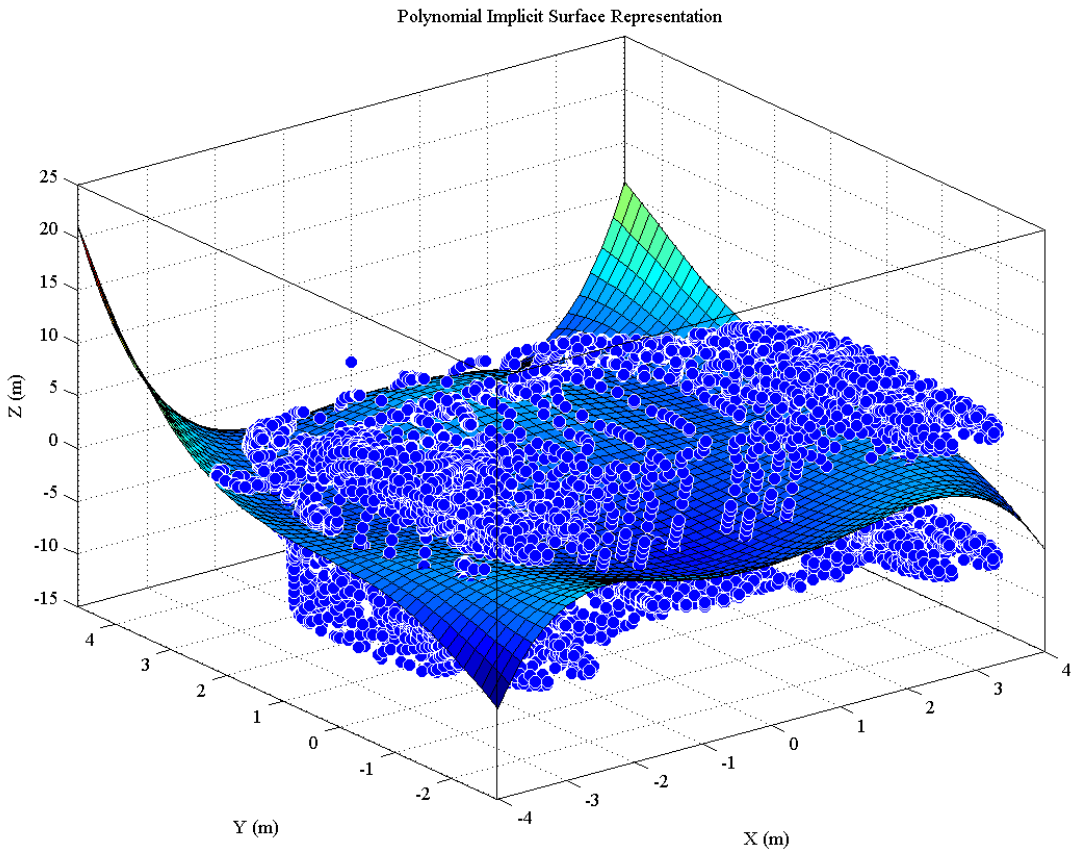

Figure 4.63: Scatter point representation and the corresponding reconstructed Polynomial surface for truck

Degree of symmetricity is quite different than compactness or roundness factors used in literatures; there could be a completely symmetric complex shape with the given definition. For instance, sphere in any rotational angle will have 13 degree of symmetricity by the definition. Figure 4.65 shows the process for a single direction in two–dimensional view which happens to count as one degree of symmetricity. Figure 4.66 displays the case that the direction is a non–symmetric direction. The blue dots are the points inside the small sphere intersecting with the surface of the point cloud. The red dots are the extension values along the critical direction and its mirror. The subtraction of the critical direction and its mirror appears in green dot if it is inside the small sphere around the center of the

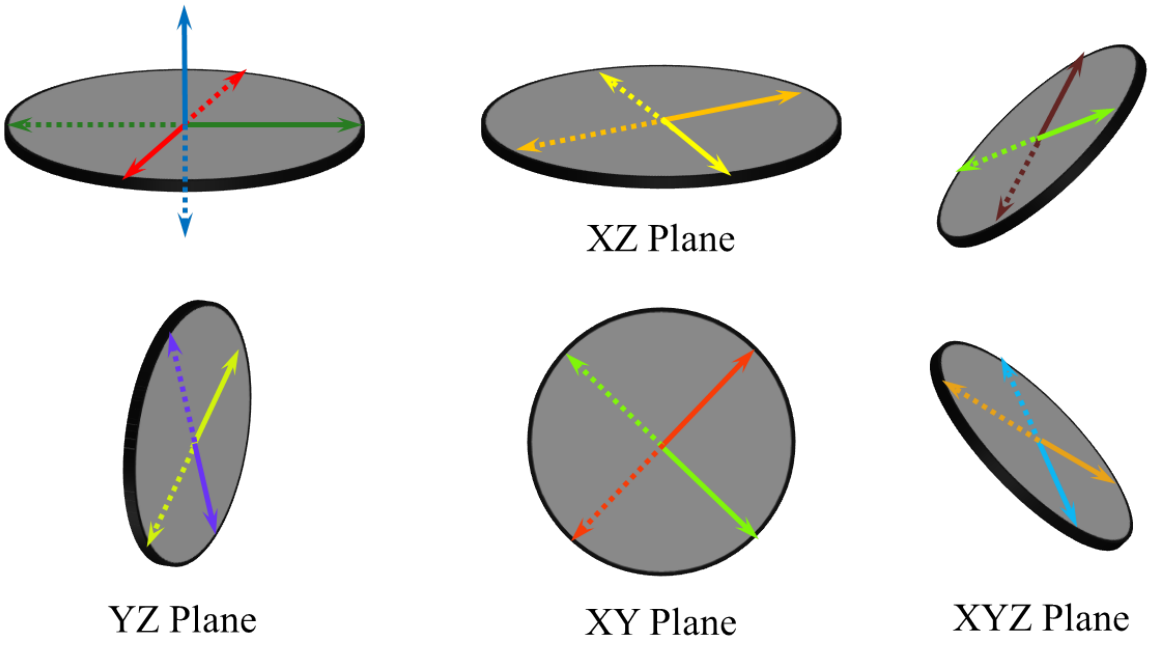

Figure 4.64: Spherical ray–based degree of symmetricity calculation's critical directions and their mirror

surrounding sphere and in dark if it is outside.

At the end of the feature extraction phase, two separate feature vectors were created: (1) Gaussian–Sinusoid–Polynomial and (2) Fourier–Polynomial. The former has a length of 263 attributes including size class as the only non–numerical element, one scalar value as degree of symmetricity and a set of numerical coefficients obtained from the other descriptors and estimation functions that are already being discussed. The latter contains 203 attributes including the size class, degree of symmetricity and 201 numerical coefficients of Fourier and Polynomial approximation functions. Table 4.3 shows the specification of these feature vectors in a tabular format.

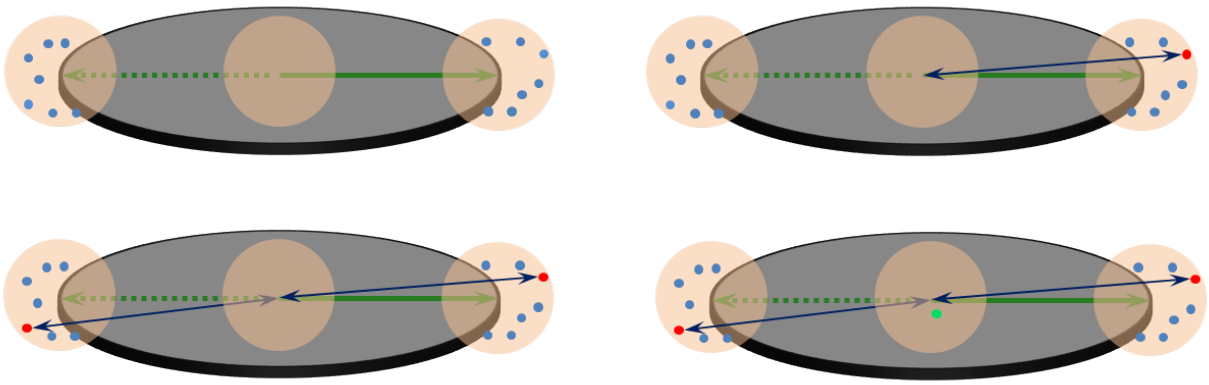

Figure 4.65: Spherical ray–based degree of symmetricity calculation along one symmetric direction

Table 4.3: Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors specifications

Feature Vector	Nominal	Numerical	Length
	Size Class	SymDegree, Descriptors, PolySurface	
Gaussian–Sinusoid–Polynomial	1	$1, 24 \times 10, 21$	263
Fourier–Polynomial	1	$1, 18 \times 10, 21$	203

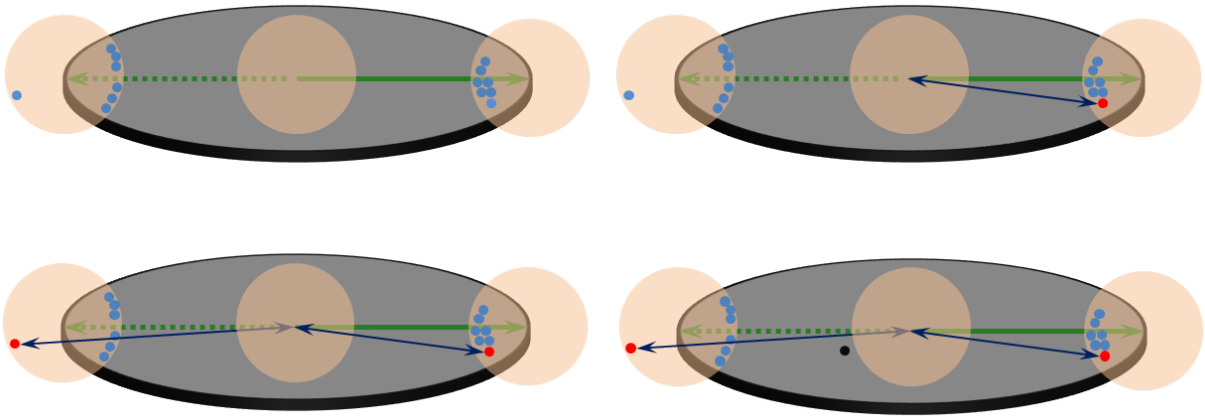

Figure 4.66: Spherical ray–based degree of symmetricity calculation along a none symmetric direction

4.4 Discussion

Feature selection and extraction build the fundamental steps of a recognition system, since the learning model relies on the descriptive attributes in the feature vectors of the objects. Therefore, ultimate goals of the system, such as independency to scale, rotation and translation, determine the feature selection criteria and form the feature extraction methods in a way to produce a set of attributes satisfying the defined constraints.

Due to the geometric nature of the objects in the collected dataset, our proposed features focused on the geometric properties of the dataset. We approached this problem with respect to basic geometric characteristics like distance, angle, torsion, area, volume and some other intuitive features. We believe more informative and distinguishing features exist—like what we extracted from the objects of the collected dataset.

4.5 Future Works

Improving the discriminative power of the recognition system—using other geometric features or topological attributes—establishes our research road map in the feature selection and extraction area. The least number of points required to compute the center of geometry of the cloud as of geometric attributes, and the number of components, the number of deformations needed to build an object from a primitive volume (e.g. cube, sphere, cylinder, etc.), and the object descriptions using primitive models as of topological features pave our future works' roadway.

Other pathways may involve approximation functions for curve estimation of surface reconstruction. For instance, the more accurate and descriptive implicit surface reconstruction methods may worth to explore to improve the discriminative power of the feature vectors and the precision of the recognition system.

In this chapter, first the criteria of selecting features were defined and then some geo-

metric features with respect to these conditions were described. In the following chapter, the two generated feature vectors—Gaussian–Sinusoid–Polynomial and Fourier–Polynomial—are examined and each individual attribute is evaluated to prove the learnability of the proposed feature vectors, to measure the learning impact of the attributes, and to find a learning model with acceptable accuracy.

CHAPTER 5

LEARNABILITY EXAMINATION AND FEATURE IMPACTS

5.1 Introduction

The last phase in the three–phase object recognition architecture includes learning and feature analysis. This phase aims to build a model to classify objects by learning the patterns of the seen entities. So, a learning model explores the inherent patterns of the labeled data using the proposed feature vectors to assure the learnability of the attributes and the existance of a model classifying the unseen objects. This procedure includes labeling the feature vectors obtained from the feature extraction phase (Chapter 4) considering the collected objects in the first phase (Chapter 3); feeding the labeled feature vectors into multiple supervised classifiers with different classification abilities and specifications; and analyzing the proposed attributes with intuitive approaches. This chapter aims to:

- Compare different supervised learning models and select the best classifier,
- Compare each feature vector's classification power,
- Measure the learning impact of each attribute in the feature vectors, and
- Analyze the errors to find out the misclassification reasons.

The rest of this chapter includes the following topics: building and comparing the learning models with respect to the accuracy measurement in the validation phase of the learning process to select the best classifier for each feature vector, measuring the impact of each attribute in the vectors, and analyzing the misclassified objects. Section 5.2 describes the procedure of building different classifiers with the two feature vectors obtained in the second phase (feature extraction), compares the models to measure the power of the features, and selects the best classifier. In addition, the misclassified objects for each learning model are examined to find out the correlation between misconception and the similarity of the objects. Section 5.3 analyzes each attribute of the feature vectors to measure the classification power using three approaches. A brief discussion and possible future works in Sections 5.4 and 5.5 bring this chapter to its end.

5.2 Learnability Examination

A learning model with a reasonable accuracy proves the learnability of the proposed feature vectors by implying that the attributes, assigned to each object in the dataset, apply sufficient discriminative information for classification purposes. In order to achieve this goal, we built various classifiers with different properties and abilities. These models are defined in two categories: (1) Primitive Models and (2) Advanced Models. On one hand, the primitive models include the basic classifiers such as decision trees, neural networks and random forests. However random forest stands in ensemble learning methods, we consider it as primitive learning model in this study. On the other hand, ensemble classifiers including bagging, boosting and stacking form our advanced models examining in this chapter.

Due to the size of the dataset (relatively small), 10–fold cross validation is chosen in order to measure the classification accuracy of the aforementioned models. To brevity, 10–fold cross validation splits the input training set into 10 equally–sized sets and trains a model with 9 folds and validates it with the 10th fold. This process happens 10 times, each time choosing a different set as the validation set. Averaging the 10 models obtained from the

process generates the final model.

5.2.1 Primitive Learning Models

According to Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors from the extraction phase, two separate models are built using decision trees, neural networks and random forests. The following sections describe the specifications of these models, their accuracy measurement, impact analysis approaches, and the experimental observations.

Decision Trees

Feeding the labeled Gaussian–Sinusoid–Polynomial feature vector of the objects in the learning set into an unpruned decision tree trains the model with the inherent patterns and results a model evaluated with the validation set—10–fold cross validation measures the accuracy and evaluates the performance of the model. Our observation implies size class, degree of symmetricity, distance, angle and torsion descriptors as the most significant attributes in the Gaussian–Sinusoid–Polynomial feature vector. Size class is chosen as the root of the tree and the other four attributes come in the second level of the tree.

Same classifier was made feeding the Fourier–Polynomial feature vector evaluated with 10–fold cross validation. The result is quite close to the other feature vector. In this classifier, size class, degree of symmetricity, angle, area and volume descriptors are the most discriminative features. Size class comes as the root of the tree and the rests reside in the next level of the trained model. Table 5.1 shows a tabular representation of the specifications of these two models.

Table 5.1: Specification of the decision tree models for Fourier–Polynomial and Gaussian–Sinusoid–Polynomial feature vectors

Descriptor	Fourier–Polynomial	Gaussian–Sinusoid–Polynomial
Accuracy	75.00%	76.00%
Building Time (s)	0.34	0.62
#Leaves	54	52
#Nodes	104	100
Average ROC Area	0.886	0.890
#Correctly Classified	225	228
#Incorrectly Classified	75	72

Neural Networks

We built a three–layer neural network with the following specification in order to evaluate the proposed feature vectors and to measure the recognition accuracy of the neural network model. The three–layer neural network consists of one input layer, one hidden layer, and one output layer. The number of nodes in the input layer is defined by the length of the feature vectors. In the hidden layer, the number of nodes equals to the average of the number of nodes in the output layer and the number of nodes in the input layer—average of number of attributes in each feature vector and the number of object classes. The training process lasts for 500 epochs and 10–fold cross validation measures the accuracy if the model.

The Gaussian–Sinusoid–Polynomial model, according to the length of the feature vector, forms 157 sigmoid nodes in the hidden layer. On the other hand, the Fourier–Polynomial model works with 127 sigmoid nodes in its hidden layer.

Comparing Fourier–Polynomial and Gaussian–Sinusoid–Polynomial neural network classifiers, the former outperforms the latter in terms of accuracy and memory–efficiency.

Random Forest

Random forest—known as an ensemble learning method—establishes our third basic learning model. Random forest, in fact, is defined as a bagging model utilizing decision tree as the base classifier. Our random forest model includes a bag of decision trees—composed of 10 different trees—trained with randomly selected proportions of the whole dataset. Equally–weighted model of all the trees builds the finalized trained model.

On the one hand, according to the Gaussian–Sinusoid–Polynomial feature vector, the random forest model owns the lowest classification accuracy among the primitive learning models. On the other hand, the Fourier–Polynomial random forest model resides among the top accurate models.

Depending upon the experimental results, Fourier–Polynomial random forest model is highly accurate in comparison to its Gaussian–Sinusoid–Polynomial sibling.

Evaluation

Receiver Operating Characteristic (ROC) curve measures the ration of true positive rate to false positive rate in binary classification problems and evaluates the performance of the system. Since the recognition problem is correlated to non–binary classifiers, the average of the area under the ROC curve across the object classes are used to estimate how accurate the model can classify the objects and also how the misclassified objects are diversified among the object classes. In other words, ROC projects the accuracy of the recognition in each object class and in general. Figure 5.1 and 5.2 illustrate the pictorial representation of classification accuracy and the average area under the ROC curve of the primitive models.

Our observation from average roc area (Figure 5.2) and the accuracy (Figure 5.1) implies that even though neural network and random forest in Gaussian–Sinusoid–Polynomial are less accurate than decision tree, their average roc area bars are taller; which means they classify more object classes truely but perform poorly in a few classes. In other words, while

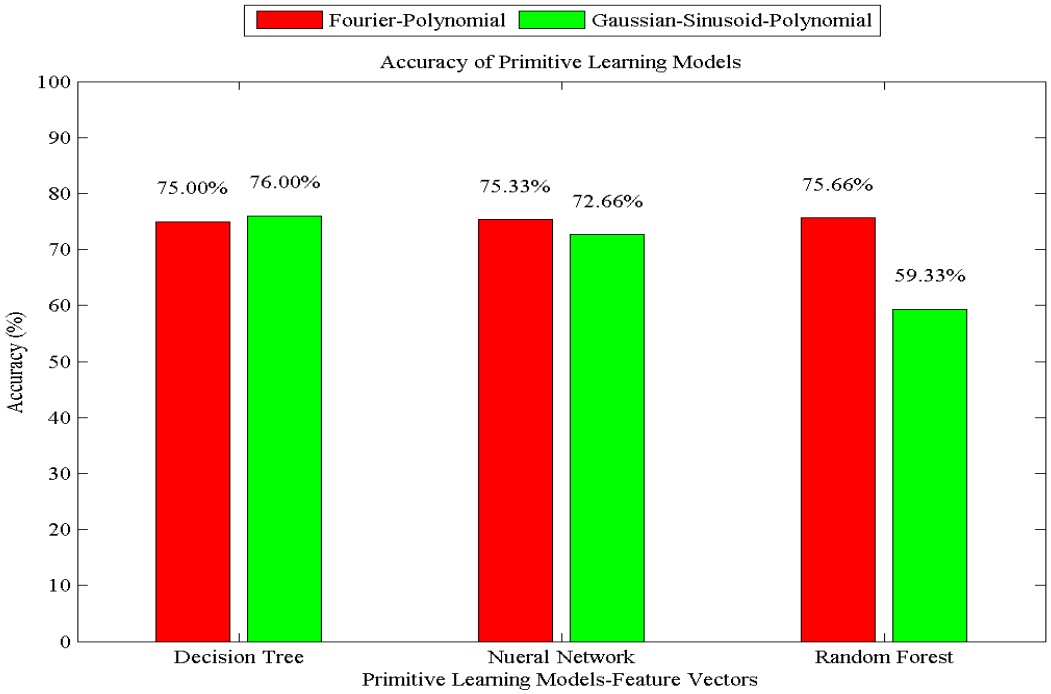

Figure 5.1: Accuracy of the premitive models with respect to the correctly classification rate

the misclassifications in decision tree is diversified uniformly among object classes, neural network and random forest misclassify a few object classes drastically.

According to the experimental results for primitive learning models of each feature vector (Figure 5.1), the best primitive classifier is decision tree with 76% accuracy with regard to Gaussian–Sinusoid–Polynomial feature vector. In addition, the most reliable and learnable feature vector is Fourier–Polynomial with consistent behavior among the different primitive classifiers.

5.2.2 Advanced Learning Models

In this study, customized ensemble learning methods are taken into account as the advanced learning models. The three learning methods including bagging, boosting, and stack-

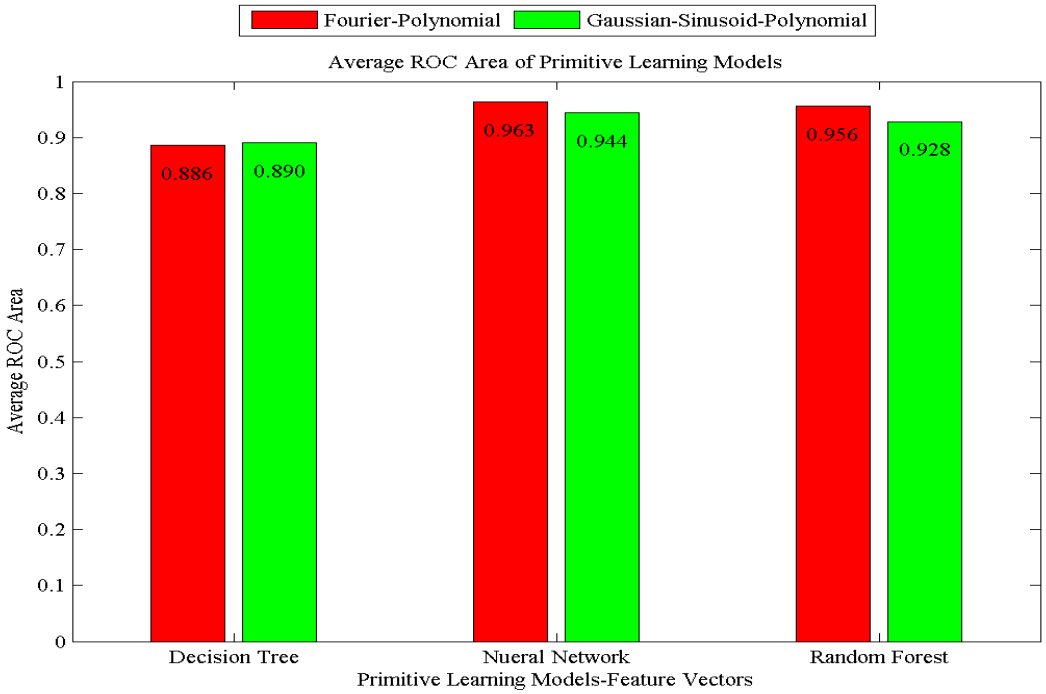

Figure 5.2: Average area under the ROC curve of primitive learning models

ing are employed to build different models for the purpose of measuring the most accurate classifier and the most reliable feature vector of the two (Gaussian–Sinusoid–Polynomial and Fourier–Polynomial).

Bagging divides the dataset into equally–sized randomly–selected training sets. Each division (training set) is used to build a model. The classifier is chosen from the primitive learning methods and it is similar among the models running on each training sets. The final model is built by averaging of all the classifiers trained by the small training sets.

Boosting is quite similar to bagging; however, it carries different methodology for training each classifier and generating the final model. In boosting, multiple same–type classifiers are trained by the whole training set and the final model is built by weighted–averaging of these rudimentary models.

Stacking is another ensemble learning approach in which multiple classifiers from different

types are trained with the training set and the results of these classifiers are used by a Meta classifier to make the final decision. In this section, we describe our customized ensemble learning models and compare them with respect to their accuracy.

Bagging

Applying three customized bagging models including bag of decision trees, neural networks and random forests on the Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors forms the foundation of our bagging learning models. The classifier of each bagging method duplicates the specifications of the models discussed in the primitive learning models section. The accuracy of these models is measured using 10–fold cross validation technique. Figure 5.3 shows the accuracy of these bagging models.

Among the bagging models, bag of random forests on Fourier–Polynomial feature vector

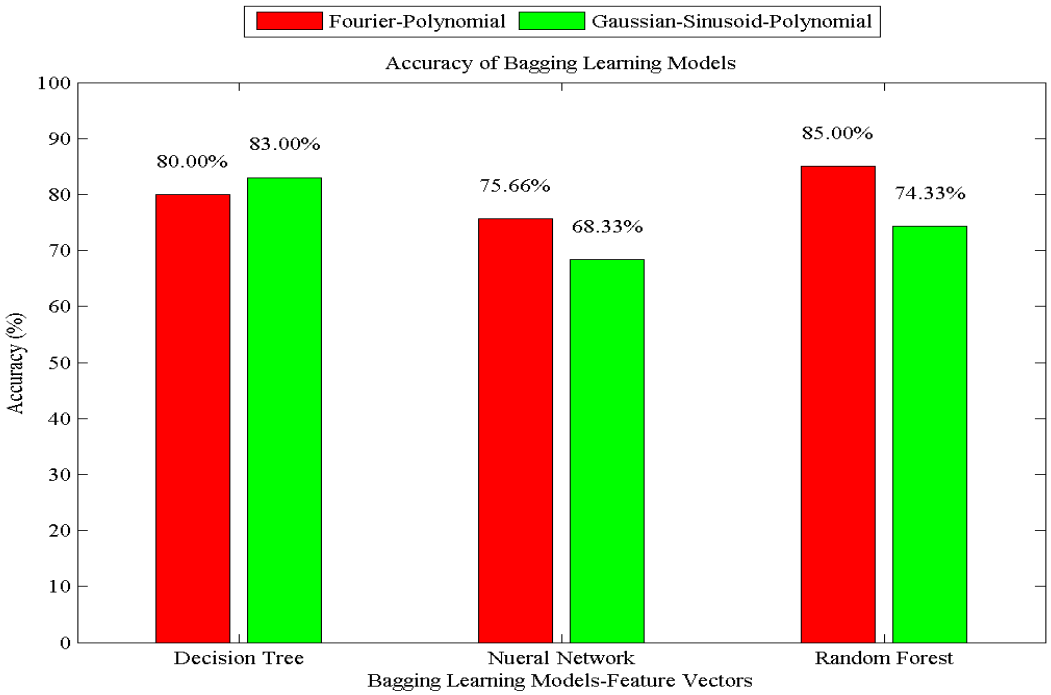

Figure 5.3: Accuracy of bagging models on Fourier–Polynomial and Gaussian–Sinusoid–Polynomial feature vectors

(85%) and bag of decision trees for Gaussian–Sinusoid–Polynomial descriptor (83%) hold the

highest precisions. However comparing the results of the bagging methods across the classifiers implies the highly reliability and consistency of the Fourier–Polynomial representation of the proposed features.

Boosting

Using the primitive learning models, we built another three customized boosting classifiers based on decision trees, neural networks, and random forests. These customized ensemble learning models are trained by the labeled Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors and are validated by 10–fold cross validation technique. Figure 5.4 displays the accuracy of the boosting techniques on the Gaussian–Sinusoid–Polynomial and Fourier–Polynomial feature vectors.

The highest precision among boosting methods belongs to boosting on decision trees

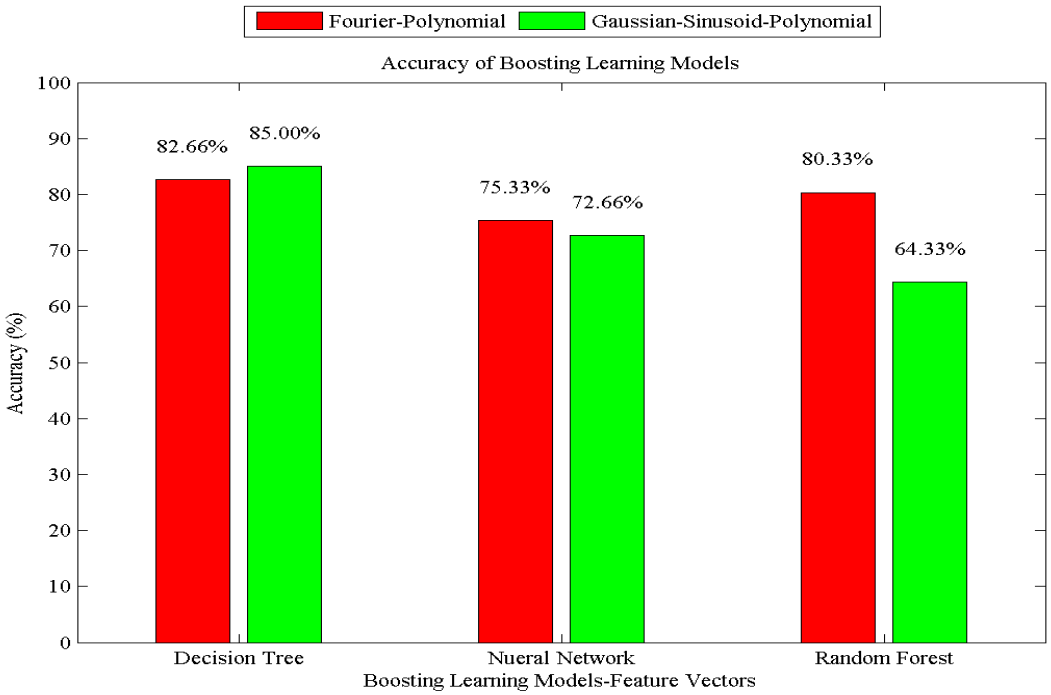

Figure 5.4: Accuracy of boosting models on Fourier–Polynomial and Gaussian–Sinusoid–Polynomial feature vectors

using Gaussian–Sinusoid–Polynomial description of the features (85%). Despite that, the

Fourier–Polynomial descriptor behaves consistently, as the accuracy of the learning models regarding several primitive classifiers stays relatively stable.

Stacking

The last advanced learning method to examine the feature vectors, in this study, is stacking. According to the two–layered definition of stacking method, the first layer consists of a stack including primitive learning models and the second contains a Meta classifier which classifies the input descriptor based on the output of the primitive models in the first layer. The customized stacking models, examined by this study, are built upon a fixed stack in the first layer (decision tree at the bottom of the stack, random forest in the middle and neural network at the top) and interchanging the Meta classifiers. Therefore, the stacking methods—disregarding the stack of classifiers in the first layer and according to their Meta classifier—are named decision tree, neural network and random forest. Figure 5.5 displays the precision of these classifiers.

Comparing the stacking models clarifies that the best stacking classifier is the stacking model with a neural network, as its Meta classifier, with 83% accuracy on Fourier–Polynomial descriptor. Analogous to the other advanced learning models and primitive classifiers, Fourier–Polynomial feature vector is more consistent and reliable descriptor in comparison with Gaussian–Sinusoid–Polynomial—because of its relatively robust behavior among the classifiers with same methodology but different classifiers.

Evaluation

Among the advanced learning models, bag of random forests on Fourier–Polynomial feature vector and decision trees with boosting method for Gaussian–Sinusoid–Polynomial vector provide the highest precision (85%). Moreover, stacking is considered the worst classifier among the advanced learning models.

Figure 5.6 and 5.7 display the bar chart accuracy of each classifier and the average area

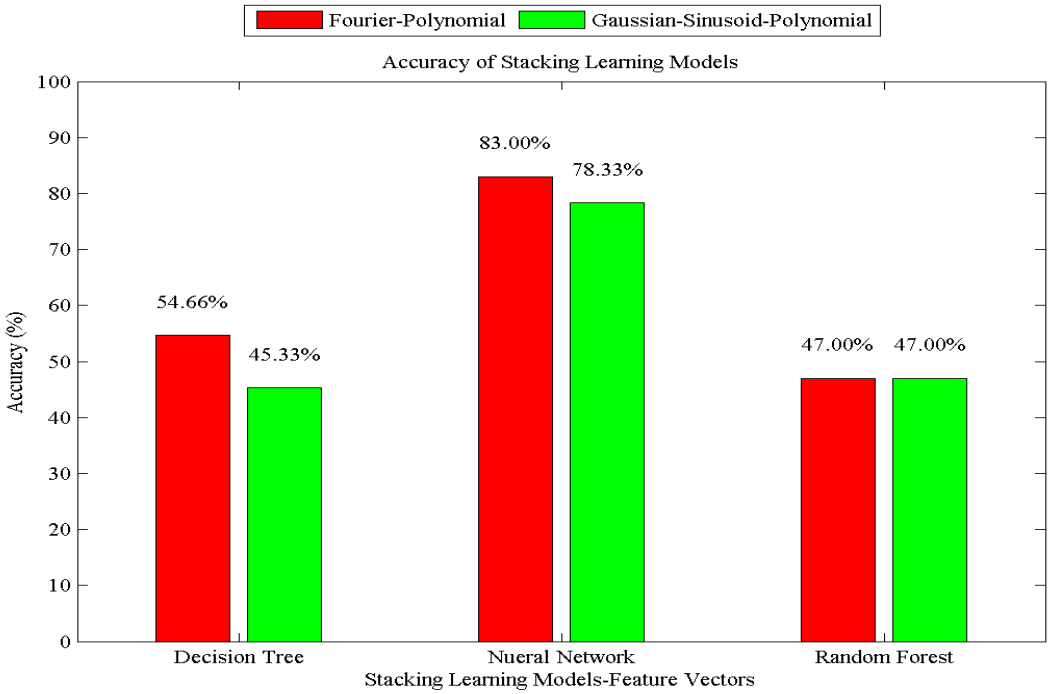

Figure 5.5: Accuracy of stacking models on Fourier–Polynomial and Gaussian–Sinusoid–Polynomial feature vectors

under the ROC curve of these systems, in order to compare the advanced models.

Comparing the primitive and advanced models, the best classifiers are bag of random forests and boosting of decision trees with 85% accuracy for Fourier–Polynomial and Gaussian–Sinusoid–Polynomial feature vectors, respectively. Moreover, Fourier–Polynomial descriptor is elected as the most consistent and reliable feature vector. In addition, among the learning techniques, bagging has the highest average accuracy either on Fourier–Polynomial or Gaussian–Sinusoid–Polynomial feature vectors. Figure 5.8 illustrates a pictorial accuracy representation of all the learning models.

Figure 5.9 shows the average area under the ROC curve of the studied classifiers including primitive and advanced models. With respect to the average area under the ROC curve of the learning models, it appears that along the objects in the validation set, bag

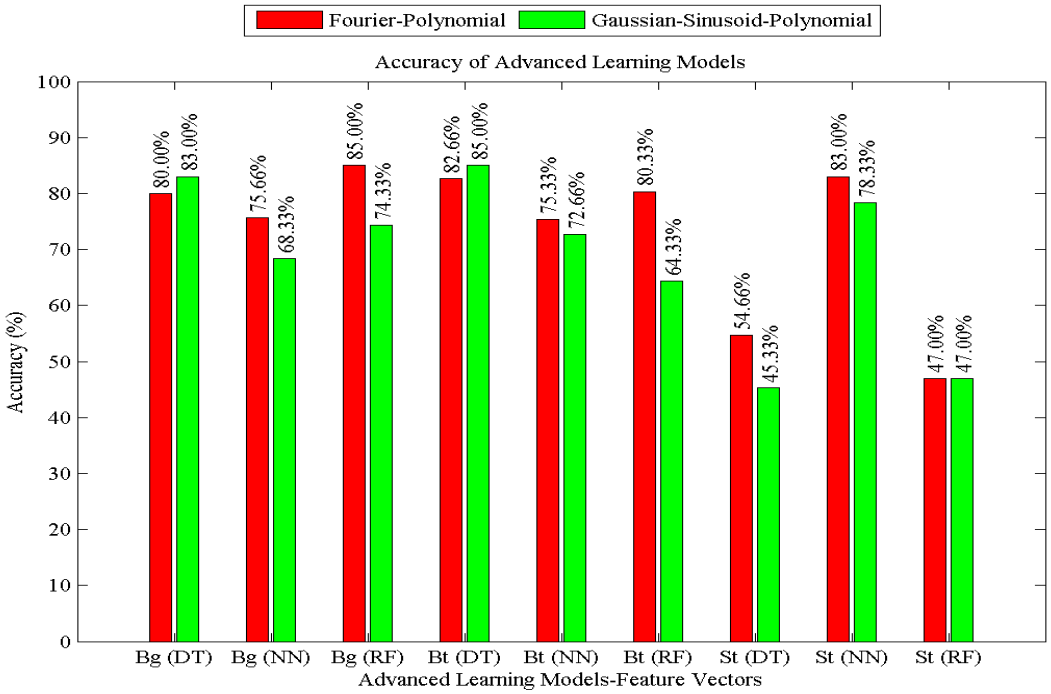

Figure 5.6: Accuracy of the advanced models with respect to the correctly classification rate

of decision trees for Gaussian–Sinusoid–Polynomial feature vector has the highest value for average ROC area; i.e., this classifier holds the lowest average rate of false–positive responses (0.5%). This position for Fourier–Polynomial descriptor belongs to boost of decision trees with highest average ROC area (0.988) and the lowest average false–positive classifications (0.6%). In other words, these models perform perfectly in almost all object classes except a few.

According to the experimental models and the acquired results, since there exists at least one learning model with an arbitrary precision that can distinguish objects in the dataset, the learnability of the features is proven. Moreover, as the objects in the dataset are in different size, orientation and position, with a similar analogy the features are scale–, rotation– and translation–invariant.

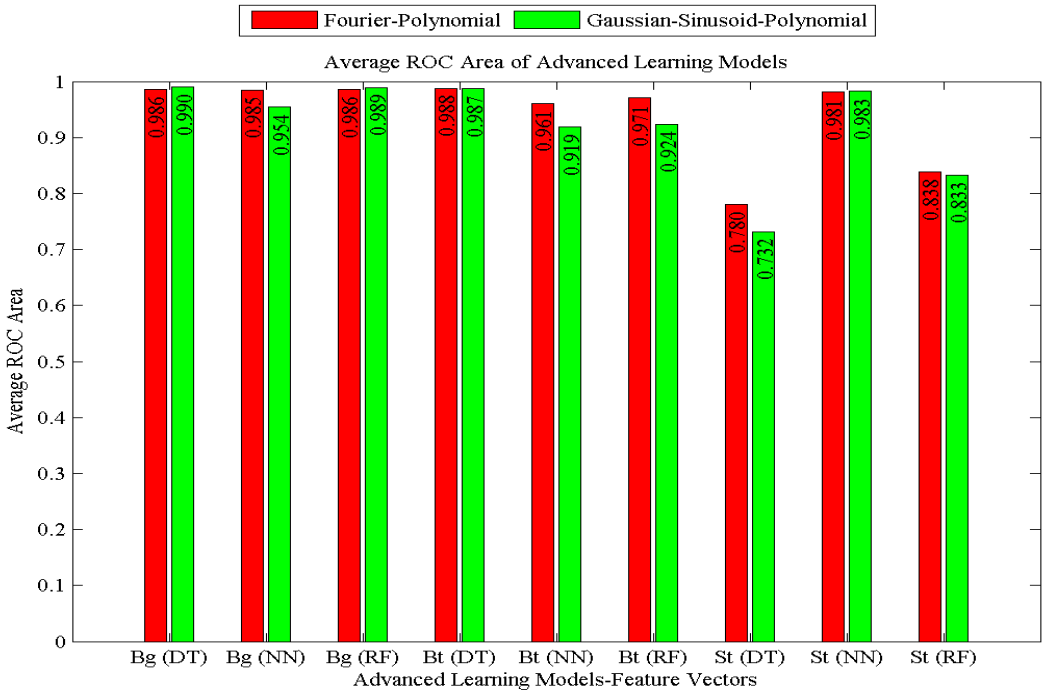

Figure 5.7: Average area under the ROC curve of advanced learning models

5.3 Features Impact Analysis

One way to measure the learnability of the features and descriptors in a learning model is to analyze how they affect the accuracy of the system. In order to estimate the impact of each single attribute discussed in Chapter 4, we take three techniques into the consideration: (1) Attribute Selection, (2) Attribute Subtraction and (3) Singleton Training. Following sections explain these methods and their results in details.

5.3.1 Attribute Selection

Attribute selection is used in decision tree–based learning models in a way that the most informative or discriminative attribute is selected as the root of the tree and its children are the ones with the next highest ranked in the list. Attribute selection is based on infor-

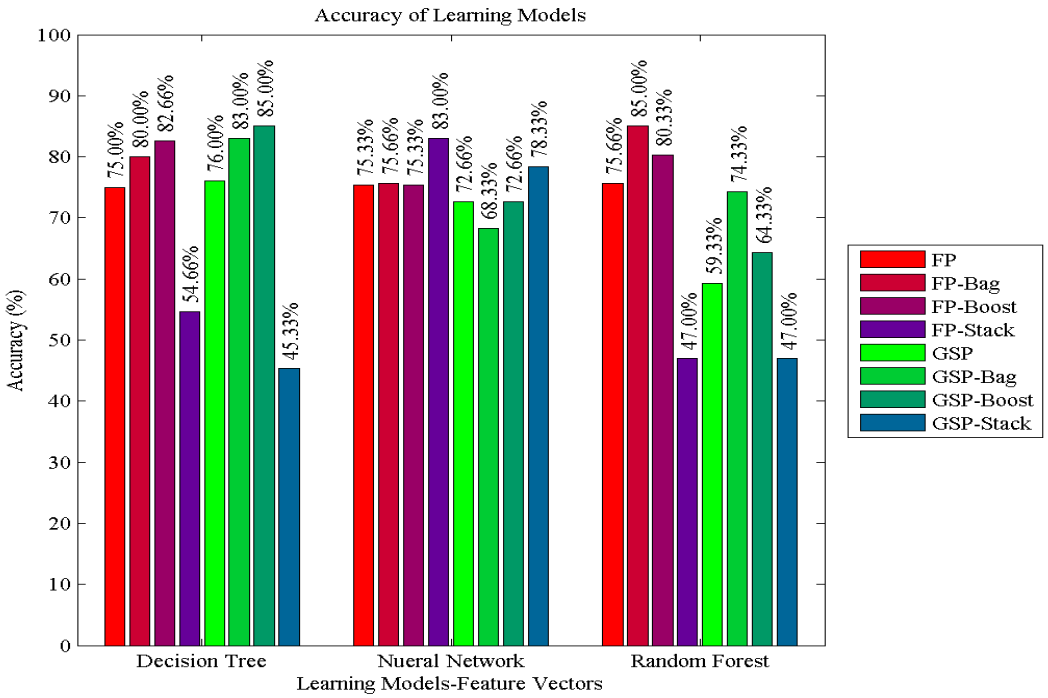

Figure 5.8: Accuracy of the all models with respect to the correctly classification rate

mation gain and entropy in information theory. Several methods exist to choose the best attributes to make the decision tree smaller and more efficient. We analyze our attributes in the two feature vectors with respect to the best decision tree classifiers that are obtained in Section 5.2. In addition, using a ranking algorithm on information gain of each attribute, the highly ranked attributes are selected as the most impactful features in each vector; i.e., the ones contain of more information about object properties which help the system to distinguish a particular object from others. Moreover, the lowest ranks belong to the least significant attributes in the feature vectors.

According to the decision tree classifiers, the most effective attributes in the learning process are appeared in the top levels of the tree. Regarding Section 5.2, the best decision tree for Gaussian–inusoid–Polynomial feature vector puts size class as the root of the tree and selects degree of symmetricity, A_2, T_2 and D_2 for the next level of the tree. That implies the

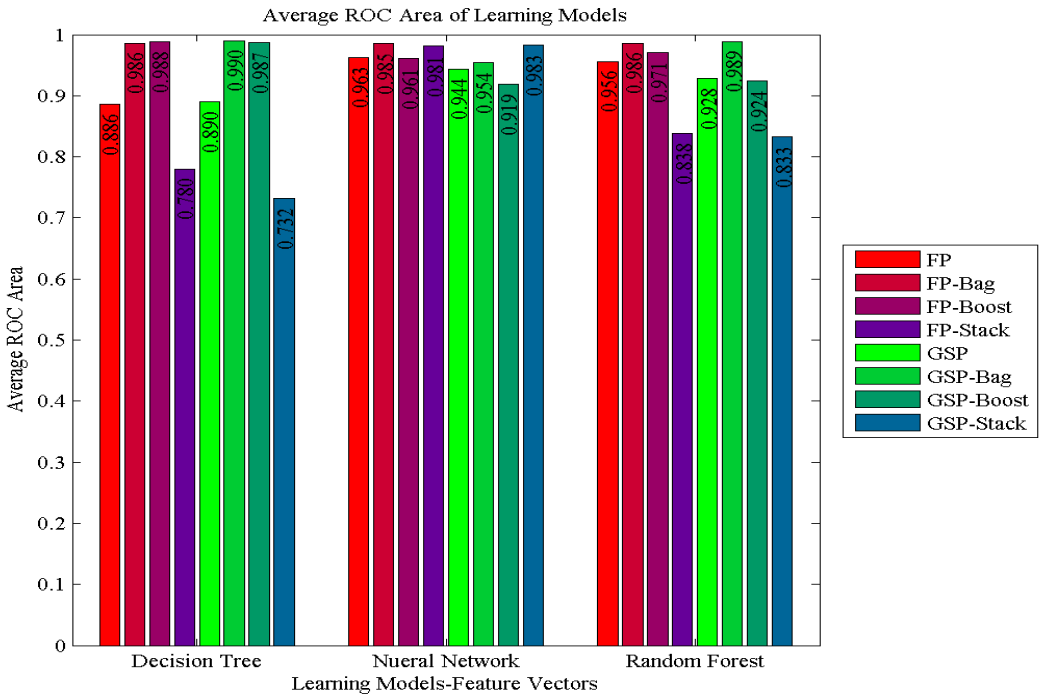

Figure 5.9: Average area under the ROC curve of all learning models

most impactful attribute of Gaussian–Sinusoid–Polynomial vector include size class, degree of symmetricity, A_2, T_2 and D_2. On the other hand, this classifier on Fourier–Polynomial feature vector chooses size class as the root of the tree and picks degree of symmetricity, A_2, Ar_2 and V_2 as the first level children of the root in the tree. That indicates the most significant attributes according to the decision tree classifiers are size class and degree of symmetricity.

With respect to information gain and ranking algorithm, the highly ranked attributes on Gaussian–Sinusoid–Polynomial vector are orderly size class, degree of symmetricity, T_2, A_1, T_1 and Ar_2; whereas the least significant features are A_2 and polynomial implicit reconstructed surface. However the most effective features on Fourier–Polynomial vector are size class, Ar_2, D_2, V_2, A_1 and degree of symmetricity, in order; whilst the lowest positions in the list belong to A_2 and polynomial implicit representation of the surface.

Therefore, considering the decision tree classifiers and ranking with regard to information gain of the features, size class, degree of symmetricity, Ar_2 and V_2 are the most impactful attributes on Fourier–Polynomial vector. Whereas for Gaussian–Sinusoid–Polynomial descriptor, these attributes are size class, degree of symmetricity and T_2.

5.3.2 Attribute Subtraction

Another way of indicating the effectiveness of an attribute, studied in this research, is how a learner performs without having the attribute in the feature vector. To do that, multiple feature vectors are made by removing a single attribute from the original feature vectors; in this case, we remove size class, degree of symmetricity, distance (D_1 and D_2), angle (A_1 and A_2), torsion (T_1 and T_2), area (Ar_1 and Ar_2), volume (V_1 and V_2) descriptors and polynomial surface attribute from Gaussian–Sinusoid–Polynomial and Fourier–Polynomial vectors, at a time. This generates 8 different feature vectors from each original descriptor. The labeled subtracted feature vectors are used as the training set for the best classifiers obtained from Section 5.2 to measure the negative and positive effects of these attributes on the learning process. In fact, attribute subtraction is a way to measure how positive or negative an attribute can affect the classifier.

The best classifier using Fourier–Polynomial feature vector, as selected in Section 5.2, is bag of random forests learning model. Figure 5.10 shows the effect of attribute subtraction for this classifier on Fourier–Polynomial descriptor in terms of accuracy and Figure 5.11 displays it with regard to average ROC area. According to the experimental results, removing distance descriptors from Fourier–Polynomial vector has the most negative effect on the accuracy of the system which means distance descriptors are the most impactful attributes in this vector. Comparing the average ROC area between the classifiers on subtracted Fourier–Polynomial vector implies that size class is the most effective attribute in this descriptor. The most negative effect on Fourier–Polynomial vector belongs to torsion descriptors with respect to

accuracy and volume according to average ROC area[1].

Boosting on decision trees provides the best classifier on Gaussian–Sinusoid–Polynomial

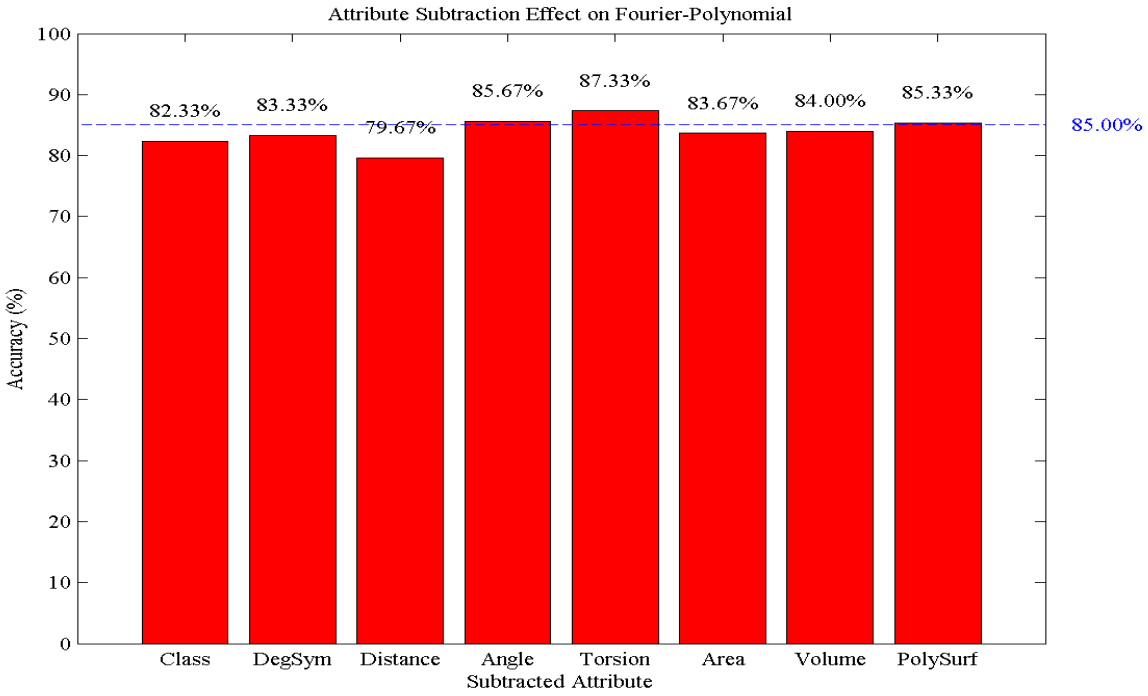

Figure 5.10: Attribute subtraction effect on accuracy of Fourier–Polynomial descriptor

vector according to Section 5.2. Figure 5.12 and 5.13 illustrate the impact of attribute removal on this classifier and descriptor with respect to accuracy and average ROC area. The experimental results show that the most positive effect belongs to torsion descriptor considering accuracy and average ROC area. Removing this descriptor from Gaussian–Sinusoid–Polynomial vector drops the accuracy of the learning system by 10%. Among all the attributes in the feature vector, angle descriptor has the most negative effect according to accuracy measurement; while polynomial implicit surface representation holds this position regarding average ROC area.

[1]Average ROC area shows the distribution of false positive classifications across the object classes

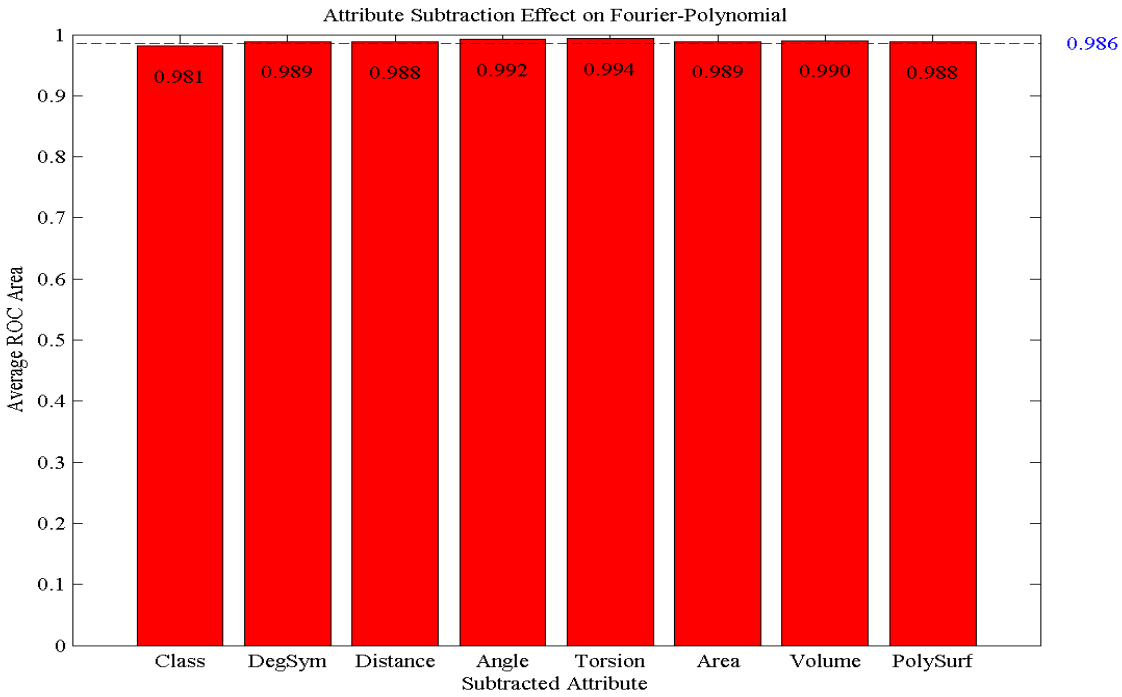

Figure 5.11: Attribute subtraction effect on average ROC area of Fourier–Polynomial descriptor

5.3.3 Singleton Training

The last but not least effectiveness measurement method for the features in the descriptors is called singleton training. Singleton training is the reversal of attribute subtraction. In this approach the best qualified classifier of each feature vector is trained using only one attribute or a set of similar descriptors of the original feature vectors. This technique evaluates how much an attribute or a set of descriptors in the same family are able to classify the dataset. Singleton training, in fact, shows the classification power of each attribute in the vector.

Several bagging models with random forests are trained using single attribute originated from Fourier–Polynomial feature vector. Figure 5.14 shows the recognition ability of these attributes according to accuracy and Figure 5.15 displays it with respect to average ROC

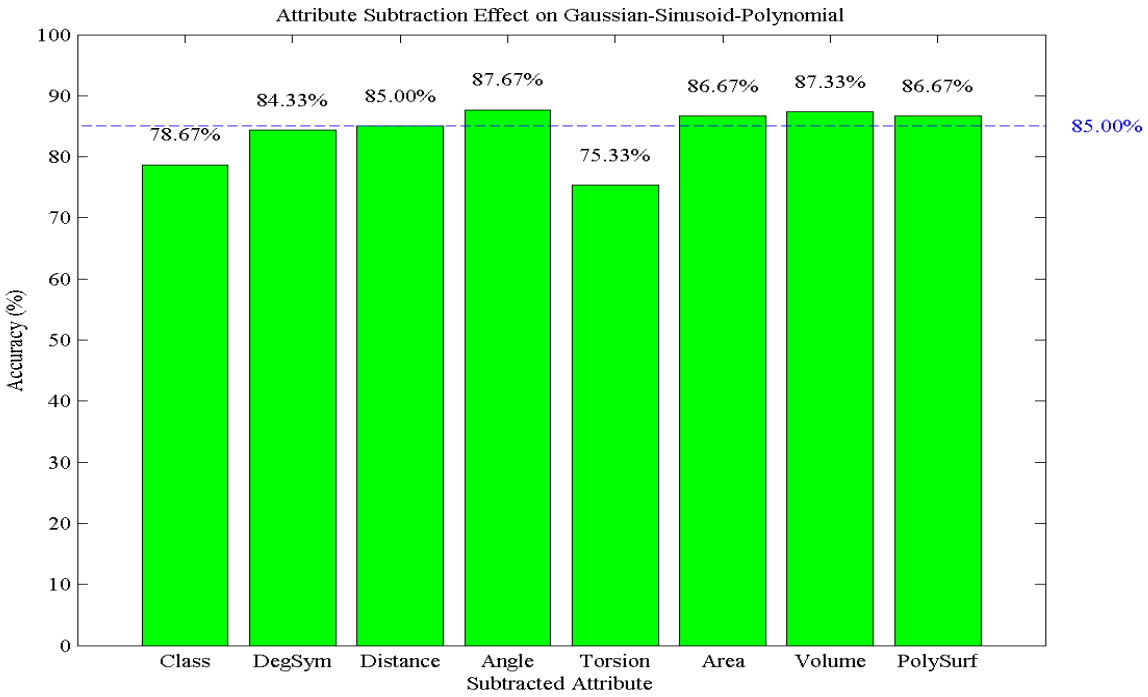

Figure 5.12: Attribute removal effect on accuracy of Gaussian–Sinusoid–Polynomial descriptor

area. Based on the results of singleton training on Fourier–Polynomial vector, distance descriptors (D_1 and D_2 together) are able to correctly classify 75.67% of the objects in the dataset, individually. This attribute also owns the tallest bar in the average ROC area bar chart among others. That means distance descriptors have the most learning capability among other attributes. The least effective feature in this classifier becomes degree of symmetricity with 11.33% classification rate; i.e., degree of symmetricity, individually, has the lowest classifying capability.

Singleton training on Gaussian–Sinusoid–Polynomial, on the other hand, was performed using boosting method with decision trees. Figure 5.16 and 5.17 illustrate the classifying capability of each attribute in Gaussian–Sinusoid–Polynomial vector according to accuracy and average ROC area, respectively. The experimental results determine that the most ca-

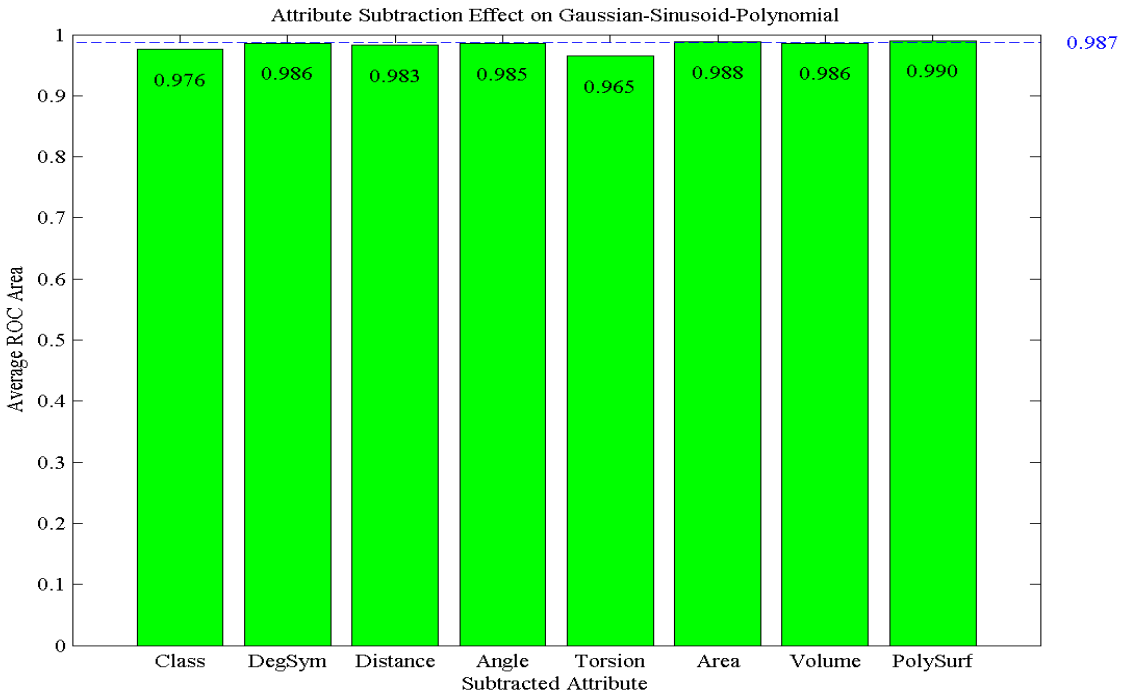

Figure 5.13: Attribute removal effect on average ROC area of Gaussian–Sinusoid–Polynomial descriptor

pable attribute in Gaussian–Sinusoid–Polynomial descriptor is torsion (T_1 and T_2). Torsion descriptors exclusively can classify 64.67% of the dataset, correctly. The average ROC area for this attribute is the highest among the others. Degree of symmetricity resides at the bottom of this list with 13% classification capability.

Considering all evaluation methods, it appears that torsion is the most impactful feature in Gaussian–Sinusoid–Polynomial descriptor; whilst in Fourier–Polynomial, distance provides useful classifying properties.

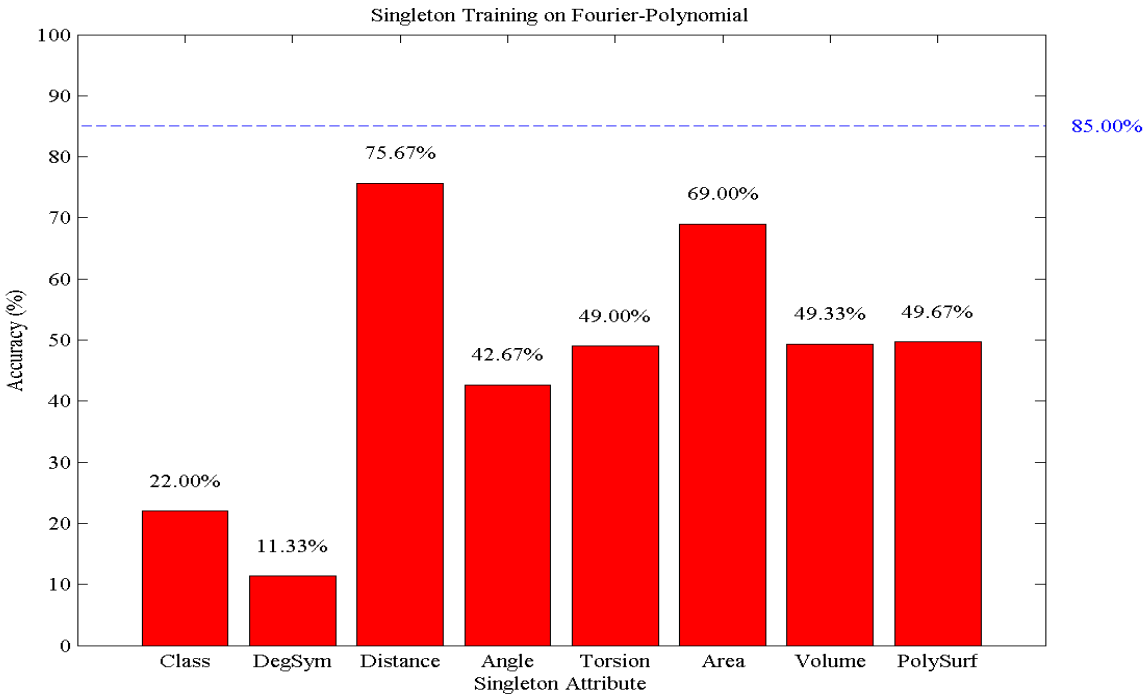

Figure 5.14: Accuracy of singleton training on Fourier–Polynomial descriptor

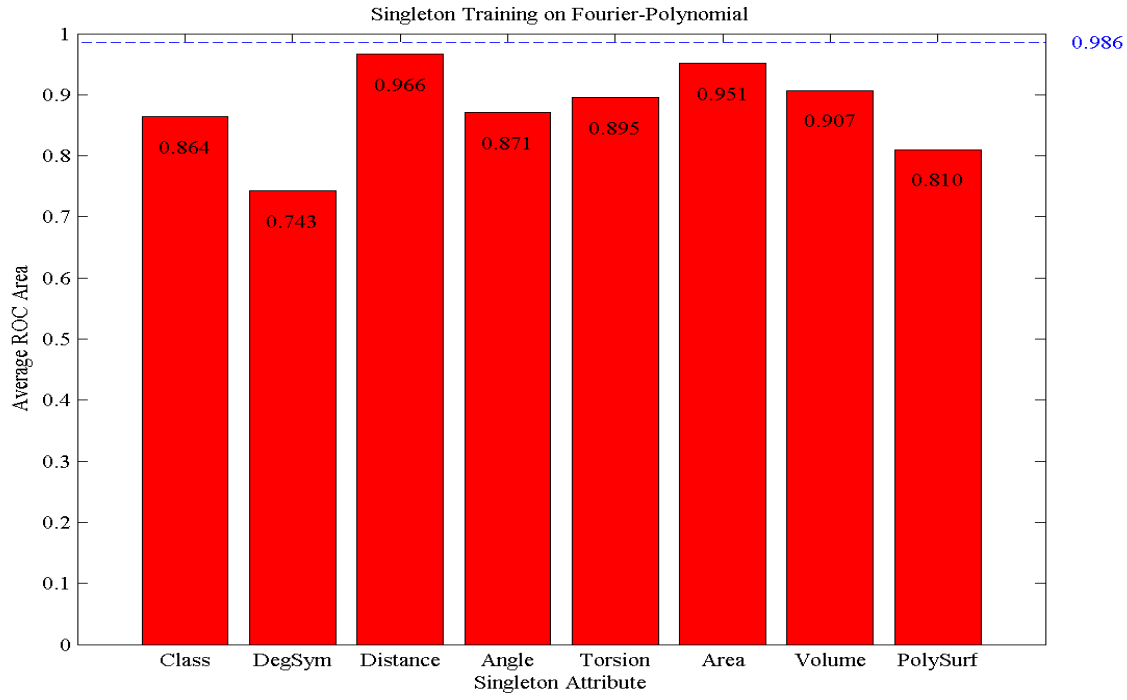

Figure 5.15: Average ROC area for singleton training on Fourier–Polynomial descriptor

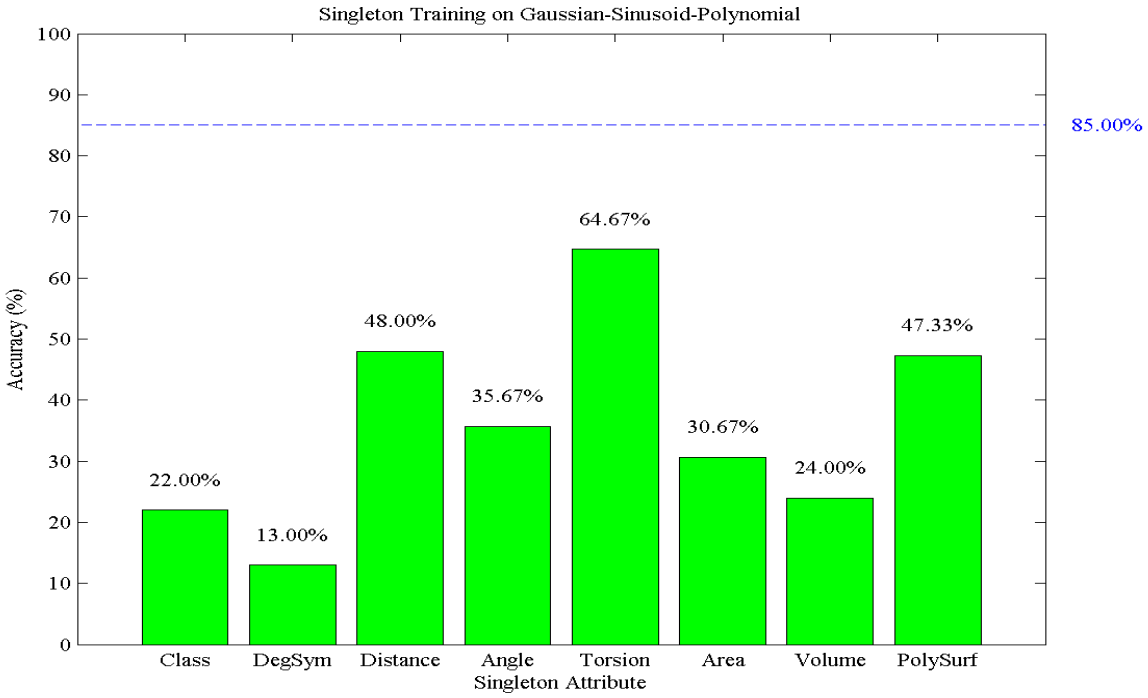

Figure 5.16: Accuracy of singleton training on Gaussian–Sinusoid–Polynomial descriptor

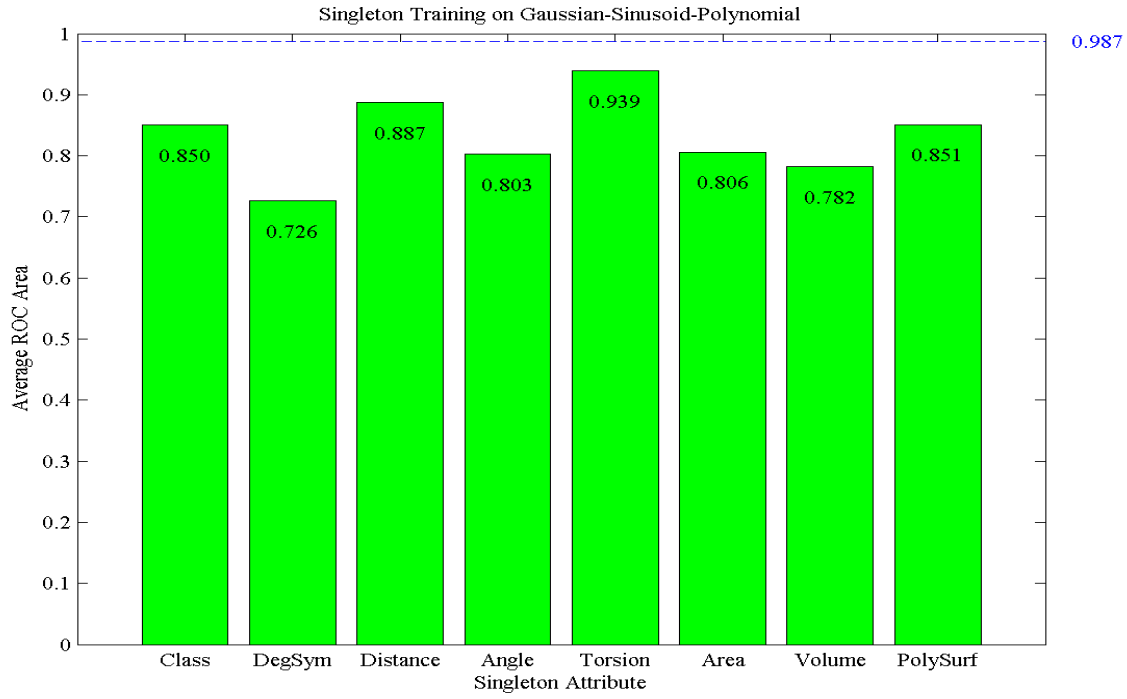

Figure 5.17: Average ROC area for singleton training on Gaussian–Sinusoid–Polynomial descriptor

5.4 Discussion

To summarize, in this chapter, we built several learning models based on the two labeled feature vectors (Gaussian–Sinusoid–Polynomial and Fourier–Polynomial) to compare the models, to find out the best classifier, and to pick the best representation. Then, we evaluated the internal attributes of each representation with several analytical techniaues. Our experimental results imply the two representations behave very similar but classifying the objects using Fourier–Polynomial shows stability and robustness. Bagging model of random forests classifier on Fourier–Polynomial representation and Boosting method of decision trees on Gaussian–Sinusoid–Polynomial descriptor outperform the other learning models. The analytical observations indicate distance descriptor in Fourier–Polynomial and torsion descriptor in Gaussian–Sinusoid–Polynomial are the most impactful attributes among others. Last but not least, we also observe that even the overal accuracy of a model on a feature vector is lower (e.g. neural network), that model classifies a good porsion of object classes correctly (average ROC area is higher).

5.5 Future Works

Learning phase of a recognition system is always open–ended. Other learning models or a combination of them may produce a system with higher accuracy. For future, other learning algorithms with different classification capabilities can be explored. In addition, other impact or effectiveness measurement methods need to be examined to define the best qualification approach and the most impactful feature(s).

In this chapter, we evaluated the learnability of the extracted features, the classification power of each attribute and the learning impact of these descriptors by building various classifiers. Furthermore, we provided a comparison of the two different presenta-

tions (Gaussian–Sinusoid–Polynomial and Fourier–Polynomial) of the features in terms of how learnable they are, and how impactful the representation can be for the recognition purposes.

CHAPTER 6

CONCLUSION

Our goal in this research was to propose a 3D point cloud–based feature extraction and object recognition methodology that eliminates the existing ambiguity in the recognition systems based on 2D projections, utilizes the missing dimensions in the recognition systems with regard to RGB–D approaches, and facilitates the depth data measurement by removing the parameters like capturing positions and orientations of cameras. This research focused on 3D point cloud representation, adequate descriptors assignment to 3D point clouds, and measurement of the learnability of the provided descriptors. Moreover, the ultimate goal of this study was to find out how 3D model or 3D point cloud processing may affect the object recognition procedure.

To achieve this goal, we followed a general object recognition architecture including three phases: (1) Data Collection, (2) Feature Extraction, and (3) Learning. Our contribution in each step includes collecting a dataset of 3D point cloud of objects containing 300 objects in 50 categories, proposing features and feature vector generation methods from 3D point clouds, and proving the learnability and usability of the features by finding the best object classifier and measuring the impact of each attribute in the proposed feature vectors with intuitive strategies.

To draw this dissertation into its close, we proved that there exists at least one learning model that can distinguish objects with a reasonable accuracy from their 3D point cloud

models using the proposed features. We also proved that our proposed features are invariant to scale, rotation and translation. Further analyses on the proposed features express that these features are learnable with different classification or discriminative capabilities. Moreover, we found out a single attribute has completely distinct distinguishing ability when it is represented in different ways.

Lastly, the recognition system that we proposed in this research proves that without any color or texture information and just using 3D points on the surface of objects, the class of the objects or at least the shape or topology of the objects are recognizable.

APPENDIX A

APPROXIMATION FUNCTIONS

Several approximation functions are used in this study to estimate fitting curves of probability distributions and best surface calculation of scattered 3D points. Curve fitting and surface reconstruction functions are including Gaussian, Sinusoid, Fourier and Polynomial.

A.1 Gaussian Estimation Function

First order Gaussian approximation function; which is used for 2D curve fitting, is defined with three coefficients as follows.

$$Gauss(1) = a_1 e^{-(\frac{X-b_1}{c_1})^2}$$

In order to generate higher orders of this estimation function, the first order Gaussian formulation with different coefficients are added, repetitively. For instance, Gaussian estimation function in order 8 is the summation of 8 first order Gaussian estimation with different coefficients. In this study, the following Gaussian approximation function; which is in 8^{th} order, is used to estimate the probability distribution of distance, area and volume descriptors.

$$Gauss(8) = a_1 e^{-(\frac{X-b_1}{c_1})^2} + a_2 e^{-(\frac{X-b_2}{c_2})^2} + a_3 e^{-(\frac{X-b_3}{c_3})^2} + ... + a_8 e^{-(\frac{X-b_8}{c_8})^2}$$

The general formulation of Gaussian estimation function is the following.

$$Gauss(n) = a_1 e^{-(\frac{X-b_1}{c_1})^2} + a_2 e^{-(\frac{X-b_2}{c_2})^2} + a_3 e^{-(\frac{X-b_3}{c_3})^2} + ... + a_n e^{-(\frac{X-b_n}{c_n})^2}$$

A.2 Sinusoid Estimation Function

Sinusoid curve fitting function is useful to estimate attributes like angle and torsion that have angular nature. The first order Sinusoid approximation function which is determined with three coefficients is the following.

$$Sinusoid(1) = a_1 \sin b_1 X + c_1$$

The higher orders of Sinusoid estimation function are created by adding the first order function with distinct coefficients, consecutively. Sinusoid estimation function of order 8 is used to calculate the best fitting curve of probability distribution of angle and torsion descriptors. This estimation function is as follows.

$$Sinusoid(8) = a_1 \sin b_1 X + c_1 + a_2 \sin b_2 X + c_2 + a_3 \sin b_3 X + c_3 + ... + a_8 \sin b_8 X + c_8$$

Here is the n^{th} order of this function.

$$Sinusoid(n) = a_1 \sin b_1 X + c_1 + a_2 \sin b_2 X + c_2 + a_3 \sin b_3 X + c_3 + ... + a_n \sin b_n X + c_n$$

A.3 Fourier Estimation Function

Similar to Gaussian and Sinusoid Estimation functions, basic Fourier transformation is its first order formulation. Unlike the Sinusoid and Gaussian, the first order Fourier transformation is defined with four distinct coefficients. This formulation is as follows.

$$Fourier(1) = a_0 + a_1 \cos(Xp) + b_1 \sin(Xp)$$

In this study, 8^{th} order of Fourier transformation are used to compute the best fitting curve of probability distributions of all the descriptors including distance, angle, torsion, area and volume. Analogous to the others, the higher orders of this estimation function are obtained by adding the first order with different coefficients, iteratively. The following is the 8^{th} order formulation of this approximation function.

$$Fourier(8) = a_0 + a_1 \cos(Xp) + b_1 \sin(Xp) + a_2 \cos(Xp) + b_2 \sin(Xp) + a_3 \cos(Xp) + b_3 \sin(Xp) + ... + a_8 \cos(Xp) + b_8 \sin(Xp)$$

The n^{th} order of Fourier transformation is as beneath.

$$Fourier(n) = a_0 + a_1 \cos(Xp) + b_1 \sin(Xp) + a_2 \cos(Xp) + b_2 \sin(Xp) + a_3 \cos(Xp) + b_3 \sin(Xp) + ... + a_n \cos(Xp) + b_n \sin(Xp)$$

A.4 Polynomial Estimation Function

Contrasting to the previous estimation functions, Polynomial function is used for surface fitting; i.e., this function is useful to compute the best fitting surface of 3D points. Therefore, the order of this function is defined by two digits. The following is the basic formulation of Polynomial estimation function in the order of $(1, 1)$.

$$Poly(1, 1) = p_{0,0} + p_{1,0}X + p_{0,1}Y$$

In this study, the higher order of the basic Polynomial function is used to represent the implicit reconstructed surface of the point cloud. This Polynomial estimation function is the following.

$$Poly(5,5) = p_{0,0} + p_{1,0}X + p_{0,1}Y + p_{2,0}X^2 + p_{1,1}XY + p_{0,2}Y^2 + p_{3,0}X^3 + p_{2,1}X^2Y +$$
$$p_{1,2}XY^2 + p_{0,3}Y^3 + p_{4,0}X^4 + p_{3,1}X^3Y + p_{2,2}X^2Y^2 + p_{1,3}XY^3 + p_{0,4}Y^4 + p_{5,0}X^5 + p_{4,1}X^4Y +$$
$$p_{3,2}X^3Y^2 + p_{2,3}X^2Y^3 + p_{1,4}XY^4 + p_{0,5}Y^5$$

The general formulation of $Poly(n, m)$ is as follows.

$$Poly(n,m) =$$
$$p_{0,0} + p_{1,0}X + p_{0,1}Y + p_{2,0}X^2 + p_{1,1}XY + p_{0,2}Y^2 + ... + p_{n,0}X^n + p_{n-1,1}X^{n-1}Y + ... + p_{0,m}Y^m$$

BIBLIOGRAPHY

[1] http://en.wikipedia.org/wiki/deep_learning on october 19, 2013.

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 9(1):3–15, 2003.

[3] Remi Allegre, Raphaëlle Chaine, and Samir Akkouche. Convection-driven dynamic surface reconstruction. In *Shape Modeling and Applications, 2005 International Conference*, pages 33–42. IEEE, 2005.

[4] Rémi Allègre, Raphaëlle Chaine, and Samir Akkouche. A dynamic surface reconstruction framework for large unstructured point sets. In *SPBG*, pages 17–26, 2006.

[5] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing*, volume 7, pages 39–48, 2007.

[6] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

[7] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421. ACM, 1998.

[8] Nina Amenta, Sunghee Choi, Tamal K Dey, and Naveen Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 213–222. ACM, 2000.

[9] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM, 2001.

[10] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2):127–153, 2001.

[11] Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.

[12] Chandrajit L Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118. ACM, 1995.

[13] Alan H Barr. Superquadrics and angle-preserving transformations. *IEEE Computer graphics and Applications*, 1(1):11–23, 1981.

[14] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[15] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[16] Fausto Bernardini, Chandrajit L Bajaj, Jindong Chen, and Daniel R Schikore. A triangulation-based object reconstruction method. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 481–484. ACM, 1997.

[17] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):349–359, 1999.

[18] Michael D Beynon, Tahsin Kurc, Umit Catalyurek, Chialin Chang, Alan Sussman, and Joel Saltz. Distributed processing of very large datasets with datacutter. *Parallel Computing*, 27(11):1457–1478, 2001.

[19] Eric Bittar, Nicolas Tsingos, and Marie-Paule Gascuel. Automatic reconstruction of unstructured 3d data: Combining a medial axis and implicit surfaces. In *Computer Graphics Forum*, volume 14, pages 457–468. Wiley Online Library, 1995.

[20] James F Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982.

[21] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics (TOG)*, 3(4):266–286, 1984.

[22] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232. ACM, 2000.

[23] Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Parallel poisson surface reconstruction. In *Advances in Visual Computing*, pages 678–689. Springer, 2009.

[24] Koen Buys, Cedric Cagniart, Anatoly Baksheev, Tinne De Laet, Joris De Schutter, and Caroline Pantofaru. An adaptable system for rgb-d based human body detection and pose estimation. *Journal of visual communication and image representation*, 25(1):39–52, 2014.

[25] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.

[26] Jonathan C Carr, Richard K Beatson, Bruce C McCallum, W Richard Fright, Tim J McLennan, and Tim J Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 119–ff. ACM, 2003.

[27] Frédéric Cazals and Joachim Giesen. Delaunay triangulation based surface reconstruction. In *Effective Computational Geometry for Curves and Surfaces*, pages 231–276. Springer, 2006.

[28] David Cohen-Steiner and Frank Da. A greedy delaunay-based surface reconstruction algorithm. *The visual computer*, 20(1):4–16, 2004.

[29] Antonino Crisafi, Daniela Giordano, and Concetto Spampinato. Griplab 1.0: Grid image processing laboratory for distributed machine vision applications. In *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2008. WET-ICE'08. IEEE 17th*, pages 188–191. IEEE, 2008.

[30] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In *Visualization'97., Proceedings*, pages 495–498. IEEE, 1997.

[31] Patricia Crossno and Edward Angel. Spiraling edge: Fast surface reconstruction from partially organized sample points. In *Proceedings of the conference on Visualization'99: celebrating ten years*, pages 317–324. IEEE Computer Society Press, 1999.

[32] Jelmer De Vries et al. Object recognition: a shape-based approach using artificial neural networks. *Department of Computer Science, University of Utrecht*, 2006.

[33] Douglas DeCarlo and Dimitri Metaxas. Blended deformable models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(4):443–448, 1996.

[34] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[35] Tamal K Dey and Joachim Giesen. Detecting undersampling in surface reconstruction. In *Discrete and Computational Geometry*, pages 329–345. Springer, 2003.

[36] Tamal K Dey, Joachim Giesen, Samrat Goswami, and Wulue Zhao. Shape dimension and approximation from samples. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 772–780. Society for Industrial and Applied Mathematics, 2002.

[37] Tamal K Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 127–134. ACM, 2003.

[38] Tamal K Dey and Samrat Goswami. Provable surface reconstruction from noisy samples. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 330–339. ACM, 2004.

[39] Tamal K Dey, Gang Li, and Jian Sun. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Point-based graphics, 2005. Eurographics/IEEE VGTC symposium proceedings*, pages 39–46. IEEE, 2005.

[40] Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando De Goes, and Mathieu Desbrun. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of mathematical imaging and vision*, 48(2):369–382, 2014.

[41] Huong Quynh Dinh, Greg Turk, and Greg Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 606–613. IEEE, 2001.

[42] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[43] Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In *Computer Vision-ECCV 2004*, pages 238–251. Springer, 2004.

[44] Rex A Dwyer. Higher-dimensional voronoi diagrams in linear expected time. *Discrete & Computational Geometry*, 6(1):343–367, 1991.

[45] H Edelsbrunner. Surface reconstruction by wrapping finite point set in space. rick pollack e eli goodman festschrift, a. aronov, s. basu, j. pach, m. sharir, eds, 2002.

[46] Herbert Edelsbrunner. Shape reconstruction with delaunay complex. In *LATIN'98: Theoretical Informatics*, pages 119–132. Springer, 1998.

[47] Herbert Edelsbrunner. Surface reconstruction by wrapping finite sets in space. In *Discrete and Computational Geometry*, pages 379–404. Springer, 2003.

[48] Herbert Edelsbrunner and Nimish R Shah. Triangulating topological spaces. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 285–292. ACM, 1994.

[49] Remondino Fabio et al. From point cloud to surface: the modeling and visualization problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(5):W10, 2003.

[50] Lian Fang and David C Gossard. Multidimensional curve fitting to unorganized data points by nonlinear minimization. *Computer-Aided Design*, 27(1):48–58, 1995.

[51] Steven Fortune. Voronoi diagrams and delaunay triangulations. *Computing in Euclidean geometry*, 1:193–233, 1992.

[52] Stefan Funke and Edgar A Ramos. Smooth-surface reconstruction in near-linear time. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 781–790. Society for Industrial and Applied Mathematics, 2002.

[53] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Transactions on Graphics (TOG)*, 22(1):83–105, 2003.

[54] Takahiko Furuya and Ryutarou Ohbuchi. Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *Proceedings of the ACM international conference on image and video retrieval*, page 26. ACM, 2009.

[55] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, 2010.

[56] Jacob Goldberger, Shiri Gordon, and Hayit Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 487–493. IEEE, 2003.

[57] M Gopi and Shankar Krishnan. A fast and efficient projection-based approach for surface reconstruction. In *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*, pages 179–186. IEEE, 2002.

[58] M Gopi, Shankar Krishnan, and Cláudio T Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Computer Graphics Forum*, volume 19, pages 467–478, 2000.

[59] Venkat N Gudivada and Vijay V Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems (TOIS)*, 13(2):115–144, 1995.

[60] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM transactions on graphics (TOG)*, 4(2):74–123, 1985.

[61] Mark O Güld, Christian J Thies, Benedikt Fischer, Daniel Keysers, Berthold B Wein, and Thomas M Lehmann. Platform for distributed image processing and image retrieval. In *Visual Communications and Image Processing 2003*, pages 1109–1120. International Society for Optics and Photonics, 2003.

[62] Stefan Gumhold, Xinlong Wang, and Rob MacLeod. Feature extraction from point clouds. In *Proceedings of 10th international meshing roundtable*, volume 2001. Citeseer, 2001.

[63] Andrew J Hanson. Hyperquadrics: smoothly deformable shapes with convex polyhedral bounds. *Computer vision, graphics, and image processing*, 44(2):191–210, 1988.

[64] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212. ACM, 2001.

[65] Christoph Hoffmann and John Hopcroft. The geometry of projective blending surfaces. *Artificial Intelligence*, 37(1):357–376, 1988.

[66] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[67] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Symposium on Geometry Processing*, pages 41–50. Citeseer, 2006.

[68] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. In *ACM transactions on graphics (TOG)*, volume 28, page 176. ACM, 2009.

[69] Matthew Johnson-Roberson, Jeannette Bohg, Mårten Björkman, and Danica Kragic. Attention-based active 3d point cloud segmentation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1165–1170. IEEE, 2010.

[70] Jyrki Katajainen. *Bucketing and filtering in computational geometry*. Turun Yliopisto. Matemaattisten Tieteiden Laitos. Tietojenkäsittelyoppi, 1987.

[71] Michael Kazhdan. Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing*, page 73. Eurographics Association, 2005.

[72] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[73] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[74] Daniel Keren and Craig Gotsman. Fitting curves and surfaces with constrained implicit polynomials. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(1):31–41, 1999.

[75] Gerhard Klimeck, Michael McAuley, Robert Deen, Fabiano Oyafuso, Gary Yagi, Eric M DeJong, and Thomas A Cwik. Near real-time parallel image processing using cluster computers. In *International Conference on Space Mission Challenges for Information Technology (SMC-IT), Pasadena, CA July*, pages 13–16, 2003.

[76] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21. ACM, 2004.

[77] Patrick Labatut, J-P Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. In *Computer Graphics Forum*, volume 28, pages 2275–2290. Wiley Online Library, 2009.

[78] Xiang Li, Wangen Wan, Xiao Cheng, and Bing Cui. An improved poisson surface reconstruction algorithm. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, pages 1134–1138. IEEE, 2010.

[79] XL Li, B Veeravalli, and CC Ko. Distributed image processing on a network of workstations. *International Journal of Computers and Applications*, 25(2):136–145, 2003.

[80] Ser-Nam Lim, Larry S Davis, and Ahmed Elgammal. Scalable image-based multi-camera visual surveillance system. In *Advanced Video and Signal Based Surveillance, 2003. Proceedings. IEEE Conference on*, pages 205–212. IEEE, 2003.

[81] Hong-Wei Lin, Chiew-Lan Tai, and Guo-Jin Wang. A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. *Computer-Aided Design*, 36(1):1–9, 2004.

[82] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[83] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[84] Ravikanth Malladi, James Sethian, Baba C Vemuri, et al. Shape modeling with front propagation: A level set approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(2):158–175, 1995.

[85] Elias S Manolakos, Demetris G Galatopoullos, and Andrew P Funk. Distributed matlab based signal and image processing using javaports. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–217. IEEE, 2004.

[86] David Marr and Herbert Keith Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London B: Biological Sciences*, 200(1140):269–294, 1978.

[87] Boris Mederos, Nina Amenta, Luiz Velho, and Luiz Henrique de Figueiredo. Surface reconstruction for noisy point clouds. In *Symposium on Geometry Processing*, pages 53–62. Citeseer, 2005.

[88] Robert Mencl. A graph–based approach to surface reconstruction. In *Computer Graphics Forum*, volume 14, pages 445–456. Wiley Online Library, 1995.

[89] Robert Mencl and Heinrich Müller. Interpolation and approximation of surfaces from three-dimensional scattered data points. In *Scientific Visualization Conference, 1997*, pages 223–223. IEEE, 1997.

[90] Robert Mencl and Heinrich Müller. Graph-based surface reconstruction using structures in scattered point sets. In *Computer Graphics International, 1998. Proceedings*, pages 298–311. IEEE, 1998.

[91] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[92] Doug Moore and Joe Warren. Approximation of dense scattered data using algebraic surfaces. In *System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on*, volume 1, pages 681–690. IEEE, 1991.

[93] Shigeru Muraki. Volumetric shape description of range data using "blobby model". In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 227–235. ACM, 1991.

[94] Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. Walrus: A similarity retrieval algorithm for image databases. In *ACM SIGMOD Record*, volume 28, pages 395–406. ACM, 1999.

[95] Thomas O'Donnell, Terrence E Boult, Xi-Sheng Fang, and Alok Gupta. The extruded generalized cylinder: A deformable model for object recovery. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 174–181. IEEE, 1994.

[96] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM SIGGRAPH 2005 Courses*, page 173. ACM, 2005.

[97] Yoshiyuki Okuyama, Tadaaki Kitamura, Yoshiki Kobayashi, Masakazu Yahiro, and Kazunori Fujiwara. Distributed image recognizing system and traffic flow instrumentation system and crime/disaster preventing system using such image recognizing system, November 14 1995. US Patent 5,467,402.

[98] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3d models with shape distributions. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 154–166. IEEE, 2001.

[99] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[100] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. In *Computer graphics forum*, volume 22, pages 281–289. Wiley Online Library, 2003.

[101] Antonio Plaza, David Valencia, Javier Plaza, and Pablo Martinez. Commodity cluster-based parallel processing of hyperspectral imagery. *Journal of Parallel and Distributed Computing*, 66(3):345–358, 2006.

[102] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[103] Harshad B Prajapati and Sanjay K Vij. Analytical study of parallel and distributed image processing. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[104] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. *ACM SIGGRAPH computer graphics*, 21(4):145–152, 1987.

[105] Jagadeesh Pujari and P Hiremath. Content based image retrieval based on color, texture and shape features using image and its complement. *International Journal of Computer Science and Security*, 1(4):25–28, 2007.

[106] Peter M Roth and Martin Winter. Survey of appearance-based methods for object recognition. *Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, Technical Report ICGTR0108 (ICG-TR-01/08)*, 2008.

[107] Daniel B Russakoff, Carlo Tomasi, Torsten Rohlfing, and Calvin R Maurer Jr. Image similarity using mutual information of regions. In *Computer Vision-ECCV 2004*, pages 596–607. Springer, 2004.

[108] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[109] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S Davis. Human detection using partial least squares analysis. In *Computer vision, 2009 IEEE 12th international conference on*, pages 24–31. IEEE, 2009.

[110] Frank J Seinstra, Jan-Mark Geusebroek, Dennis Koelma, Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. High-performance distributed video content analysis with parallel-horus. *IEEE multimedia*, (4):64–75, 2007.

[111] Chen Shen, James F O'Brien, and Jonathan R Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Siggraph 2005 Courses*, page 204. ACM, 2005.

[112] Young-Seok Sim, Chae-Seong Lim, Young-Shik Moon, and Sung-Han Park. Design and implementation of the visual programming environment for the distributed image processing. In *Image Processing, 1996. Proceedings., International Conference on*, volume 1, pages 149–152. IEEE, 1996.

[113] Piotr S Szczepaniak and Ryszard Tadeusiewicz. The role of artificial intelligence, knowledge and wisdom in automatic image understanding. *Journal of Applied Computer Science*, 18(1):75–85, 2010.

[114] Richard Szeliski, David Tonnesen, and Demetri Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pages 82–87. IEEE, 1993.

[115] Johan WH Tangelder and Remco C Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia tools and applications*, 39(3):441–471, 2008.

[116] Gabriel Taubin. An improved algorithm for algebraic curve and surface fitting. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 658–665. IEEE, 1993.

[117] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: Deformable superquadrics. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 606–615. IEEE, 1990.

[118] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial intelligence*, 36(1):91–123, 1988.

[119] Bharadwaj Veeravalli and Surendra Ranganath. Theoretical and experimental study on large size image processing applications using divisible load paradigm on distributed bus networks. *Image and Vision Computing*, 20(13):917–935, 2002.

[120] Dejan V Vranić and Dietmar Saupe. Description of 3d-shape using a complex function on the sphere. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 177–180. IEEE, 2002.

[121] Stefan Walk, Nikodem Majer, Konrad Schindler, and Bernt Schiele. New features and insights for pedestrian detection. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 1030–1037. IEEE, 2010.

[122] Andrew P Witkin and Paul S Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM, 1994.

[123] H Woo, E Kang, Semyung Wang, and Kwan H Lee. A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture*, 42(2):167–178, 2002.

[124] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2878–2890, 2013.

[125] Gary Yngve and Greg Turk. Creating smooth implicit surfaces from polygonal meshes. 1999.

[126] Meng Yu, Indriyati Atmosukarto, Wee Kheng Leow, Zhiyong Huang, and Rong Xu. 3d model retrieval with morphing-based geometric and topological feature maps. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–656. IEEE, 2003.

[127] HK Zhao, S Osher, B Merriman, and M Kang. Implicit, nonparametric shape reconstruction from unorganized points using a variational level set method. cam report 98-7, ucla. *UCLA, Los Angeles, CA*, 1998.

[128] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on*, pages 194–201. IEEE, 2001.