

ABSTRACT

Title of dissertation: A FEATURE-BASED SHAPE SIMILARITY
ASSESSMENT FRAMEWORK

Antonio Cardone, Ph.D., 2005

Directed by: Associate Professor Satyandra K. Gupta,
Department of Mechanical Engineering

The popularity of 3D CAD systems is resulting in a large number of CAD models being generated. Availability of these CAD models is opening up new ways in which information can be archived, analyzed, and reused. 3D geometric information is one of the main components of CAD models. Therefore shape similarity assessment is a fundamental geometric reasoning problem that finds several different applications. In many design and manufacturing applications, the gross shape of the 3D parts does not play an important role in the similarity assessment. Instead certain attributes of part features play a dominant role in determining the similarity between two parts.

Different feature-based models are usually created using their own coordinate systems. Therefore, feature-based shape similarity assessment involves finding the optimal alignment transformations for two sets of feature vectors. The optimal alignment corresponds to the minimum value of a distance function that is computed between the two sets of feature vectors being aligned. In order to compute the distance function the closest neighbor to each feature vector needs to be identified. We have developed optimal feature alignment algorithms based on the partitioning of the transformation space into regions such that the closest neighbors are invariant within each region. These algorithms

can work with customizable distance functions. We have shown that they have polynomial time complexity. For higher dimension transformation spaces it is harder to design algorithms based on the partitioning of transformation spaces because the data structures involved are very complex. In those cases, feature alignment algorithms based on iterative strategies have been developed. Iterative strategies make use of optimal feature alignment algorithms based on the partitioning of lower dimension transformation spaces. Extensive experiments have been carried out to provide empirical evidence that iterative strategies can find the optimal solution for feature alignment problems. A feature-based shape similarity analysis framework has been built based on the feature alignment algorithms. This framework has been demonstrated with the two following applications. A machining feature based alignment algorithm has been developed to automatically search databases for parts that are similar to a newly designed part in terms of machining features. We expect that the retrieved parts can be used as a basis to perform cost estimation of the newly designed part. A surface feature based alignment algorithm has been developed to automatically search databases for parts that are similar to a newly designed part in terms of surface features. We expect that the retrieved parts can be used as a basis to choose the most appropriate tool maker for the newly designed part.

We believe that the feature-based shape similarity assessment algorithms developed in this thesis will provide the foundations for designing new feature-based shape similarity algorithms that will enable designers to efficiently retrieve archived geometric information. We expect that these tools will facilitate information reuse and therefore decrease product development time and cost.

A FEATURE-BASED SHAPE SIMILARITY ASSESSMENT FRAMEWORK

By

Antonio Cardone

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Associate Professor Satyandra K. Gupta, Chairman / Advisor

Professor Davinder K. Anand

Professor David M. Mount

Associate Professor Jeffrey W. Herrmann

Associate Professor Linda C. Schmidt

© Copyright by
Antonio Cardone
2005

Acknowledgements

I would like to thank my advisor, Dr. Satyandra K. Gupta, for his help and support through these years. Working with Dr. Gupta has been an invaluable experience thanks to his hardworking spirit, skilled ways of analyzing and solving complex problems and ability to motivate me and guide me through challenging research work. Dr. Gupta's ability to create a diverse, hardworking and coordinated research group has proven an invaluable life experience that I will jealously keep in my heart and from which I am sure I will benefit in all my life.

I would like to give special thanks to Dr. David M. Mount, who is one of my dissertation committee members, for giving me invaluable advice on the algorithms that are designed in this thesis. I am sure my future research work will benefit from his advice as well.

I would like to thank my family and friends for their support during these rewarding but also difficult years. I would like to thank in particular my mother Dora, my father Michele and my brother Giuseppe for their continuous support, valuable advice, strength and patience though these years. Without them I would have never been able to overcome the difficult times and I would have never fully enjoyed my achievements.

I would also like to thank my dissertation committee members: Drs. Davinder K. Anand, Jeffrey W. Herrman and Linda C. Schmidt for serving in the committee and giving suggestions that will be useful not only for my dissertation but also to improve my research skills.

I would like to thank my past and present colleagues from Dr. Gupta's research group for their invaluable advice, help, support and friendship: Abhijit, Alok, Ashis, Brent, Changxin, Greg, Ira, Jun, Mukul, Sunil, Tao and Yao.

Finally, I am thankful to the Center for Energetic Concepts Development at the University of Maryland for supporting this research.

Table of contents

Chapter 1: Introduction	1
1.1 Background	1
1.2 Motivation	4
1.3 Research Issues	8
1.4 Thesis Outline	10
Chapter 2: Related Research	13
2.1 Overview of Techniques	14
2.2 Feature-Based Shape Signatures	18
2.3 Spatial Function Based Shape Signatures	23
2.4 Shape Histogram Based Shape Signatures	29
2.5 Section Image Based Shape Signatures	33
2.6 Topological Graph Based Shape Signatures	35
2.6.1 Model Signature Graphs	35
2.6.2 Multiresolutional Reeb Graphs	38
2.6.3 Graphs of Aligned Models	41
2.6.4 Skeletal Graphs	42
2.7 Shape Statistics	43
2.8 Point Pattern Alignment	46
2.9 Observations	48
Chapter 3: Optimal Attributed Point Alignment Algorithms Based On Partitioning Of Transformation Spaces	50
3.1 Motivation	50
3.2 Mathematical Foundations	53
3.3 Problem Formulation	55
3.4 Optimal Alignment Under 2 DOF Translations In \mathbb{R}^2	56
3.4.1 Step a: Building The Set Of Regions For The Attributed Points Of Set P	57
3.4.2 Step b: Minimization Of The Distance Function Within A Given Region	65
3.4.3 Steps c And d: Computing The Translation That Minimizes The Distance Over All The Regions	68
3.5 Optimal Alignment Under 1 DOF Rotations In \mathbb{R}^2	69
3.5.1 Step a: Building The Set Of Theta Intervals For The Attributed Points Of Set P	71
3.5.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval	77
3.5.3 Steps c and d: Computing The Rotation That Minimizes The Distance Over All The Theta Intervals	79
3.6 Optimal Alignment Under 3 DOF Translations In \mathbb{R}^3	79
3.6.1 Step a: Building The Set Of Regions For The Attributed Points Of Set P	80
3.6.2 Step b: Minimization Of The Distance Function Within A Given Region	83
3.6.3 Steps c and d: Computing The Translation That Minimizes The Distance Over All The Regions	84
3.7 Complexity Evaluation For Optimal Alignment Algorithms Based On Partitioning Of Transformation Space	84

3.7.1 Overview.....	84
3.7.2 Complexity Of The Overlapping Of Two Voronoi Diagrams In \mathbb{R}^2	86
3.7.3 Complexity Of The Overlapping Of m Spatial Arrangements $S(P_i)$ In \mathbb{R}^d	93
3.8 Summary	96
Chapter 4: Attributed Point Alignment Algorithms Based On Iterative Strategies.....	98
4.1 Motivation.....	98
4.2 Problem Formulation	99
4.3 Definition Of Iterative Strategies.....	100
4.4 Mathematical Foundations For Iterative Strategies In \mathbb{R}^2	102
4.5 Experimental Results	106
4.5.1 Tests On Iterative Strategies In \mathbb{R}^2	106
4.5.2 Tests On Iterative Strategies In \mathbb{R}^3 Using 6 DOFs	108
4.5.3 Tests On Iterative Strategies In \mathbb{R}^3 Using 3 Rotational DOFs.....	112
4.6 Summary	115
Chapter 5: Feature-Based Similarity Assessment Algorithms.....	118
5.1 Motivation.....	118
5.2 Background And Problem Formulation.....	122
5.2.1 Machining Features.....	122
5.2.2 Distance Function For Similarity Assessment.....	124
5.2.3 Problem Statement	128
5.3 Computing Similarity For Query Parts With Single Preferred Feature Interpretations	130
5.4 Finding The Optimal Alignment Under One Degree Of Freedom.....	133
5.4.1 Step a: Building The Set Of Theta Intervals For The RFVs Of Set P	134
5.4.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval	139
5.4.3 Steps c and d: Computing The Value Of Theta That Minimizes The Distance Over All The Theta Intervals	144
5.5 Experimental Results For Single Feature Interpretations	145
5.6 Similarity Assessment In Presence Of Multiple Feature Interpretations.....	151
5.7 Summary	160
Chapter 6: Surface Feature-Based Shape Similarity Assessment Algorithms.....	163
6.1 Motivation.....	163
6.2 Background And Problem Formulation.....	165
6.2.1 Surface Features.....	165
6.2.2 Distance Function For Similarity Assessment.....	168
6.2.3 Problem Statement	172
6.3 Computing Surface Feature-Based Similarity For Parts.....	173
6.4 Finding The Optimal Alignment Under One Degree Of Freedom Rotations.....	175
6.4.1 Step a: Building The Set Of Theta Intervals For The RSFVs Of Set P	176
6.4.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval	180
6.4.3 Steps c and d: Computing The Value Of Theta That Minimizes The Distance Over All The Theta Intervals	185

6.5 Iterative Schemes To Find Optimal Alignment Under Three Rotational Degrees Of Freedom	186
6.6 Experimental Results	189
6.6.1 Tests To Study The Performance Of Iterative Scheme	189
6.6.2 Tests On Mechanical Parts.....	191
6.7 Summary	196
Chapter 7: Conclusions.....	202
7.1 Intellectual Contributions.....	202
7.2 Anticipated Benefits.....	205
7.3 Directions For Future Work.....	207
References.....	211
Appendix.....	222
A. Calculation Of Partitioning Lines And Planes For Attributed Points In \mathbb{R}^2 And \mathbb{R}^3	222
B. Calculation Of Partitioning Curves For Attributed Points On The Unit Sphere.....	224
C. Calculation Of Partitioning Theta Values For Attributed Applied Vectors Under 1 DOF Rotations In \mathbb{R}^3	226

List of figures

Figure 1.1: An Example of Using a Similar Part for Cost Estimation.....	3
Figure 1.2: An Example of Using a Similar Part for Tool Maker Selection	5
Figure 1.3: Part C Is More Similar to Part A In Gross Shape, But Part B Is More Similar to Part A in Machining Cost	6
Figure 1.4: Part C Is More Similar to Part A In Feature Type And Count, But Part B Is More Similar to Part A in Machining Cost.....	7
Figure 1.5: Organization of the Thesis	12
Figure 2.1: An Example Depicting Different Ways of Representing Features	21
Figure 2.2: An Example Indicating the Low Discrimination Capability of Shape Distributions.....	31
Figure 2.3: An Example of Same Model Signature Graph for Two Different Parts	37
Figure 3.1: An Example of Aligning Two Parts Represented In Two Different Coordinate Systems	51
Figure 3.2: The Transformation Space of Point p_1 of Set P Is Partitioned Into Two Regions (a) Transformation-invariant Attributes Are the Same for Each Point (b) Transformation-invariant Attributes Are Different for Each Point, and Hence the Line L' Is Offset With Respect to the Line L	60
Figure 3.3: Example of Overlapping of Sets of Regions.....	65
Figure 3.4: Set of Theta-intervals for Point p_1 of Set P : (a) Case of Intersection Between Line L and Circle C_1 (b) Case of Non-intersection Between Line L and Circle C_1 .	73
Figure 3.5: Set of Theta-intervals for Point p_1 of Set P When Transformation Invariant Attributes of Points q_1 and q_2 of Set Q Are Different.....	75
Figure 3.6: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals	76
Figure 3.7: The Transformation Space of Point p_1 of Set P Is Partitioned Into Two Regions (a) Transformation-invariant Attributes Are the Same for Each Point (b) Transformation-invariant Attributes Are Different for Each Point, and Hence the Plane π'_{12} Is Offset With Respect to the Line π_{12}	82
Figure 3.8: Instance of Event $e(i,6)$ Occurring in \mathbb{R}^2 . Line L_1 Generated by Points p_{11} and p_{12} of Set A_1 and Line L_2 Generated by Points p_{21} and p_{22} of Set A_2 Intersect Within the Unit Square. Point p_{11} of Set A_1 Is the Farthest Point From the Intersection Point q and Lies in Ring R_i at Distance r_{max} From q . There Are Six Points in Total Contained in S_i	89
Figure 4.1: Histogram Showing the Number of Converging and Non-converging Instances Vs. The Number of Initial Conditions Used for Iterative Strategy in \mathbb{R}^2	109
Figure 4.2: Histogram Showing the Number of Converging and Non-converging Instances Vs. The Number of Initial Conditions Used for Iterative Strategy I_i^3 in \mathbb{R}^3	114
Figure 4.3: Histogram Showing the Number of Converging and Non-converging Instances vs. The Number of Initial Conditions Used for Iterative Strategies $I_{R_i}^3$ in \mathbb{R}^3	116
Figure 5.1: The Previously Machined Part (b) Can Be Potentially Used to Estimate the Cost of the Newly Designed Part (a)	120

Figure 5.2: Features Considered With Access and Orientation Vector: (a) Pocket (b) Slot (c) Notch (d) Through Slot (e) Step (f) Hole	124
Figure 5.3: Machined Parts With Access and Orientation Vectors	125
Figure 5.4: Access Vectors for the Parts of Figure 5.3	127
Figure 5.5: Equivalence Between Reduced Feature Vectors and Attributed Set of Points on Unit Sphere	129
Figure 5.6: Example of Composite Feature	131
Figure 5.7: Transformation-invariant Attributes Are The Same For Each Reduced Feature Vector: The Two Intersection Points Between Circle C_1 and Plane π_{12} Represent The Extreme Values Of The Theta Intervals	138
Figure 5.8: Transformation-invariant Attributes Are Different: The Two Intersection Points Between Circle C_1 and Plane π'_{12} Represent The Extreme Values Of The Theta Intervals	140
Figure 5.9: Set of Theta Intervals for Reduced Feature Vector p_1 of P	143
Figure 5.10: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals	144
Figure 5.11: (a) Initial Orientation of Part M_Q and Its Randomly Rotated Version Part M_P ; (b) Orientation of Part M_P After Step3b(iii) of the Algorithm COMPUTESIMILARITYMEASURE; (c) Final Orientation of Part M_P	147
Figure 5.12: Results Obtained For Query Part#A Used As Input to the System	149
Figure 5.13: Results Obtained for Query Part#B Used As Input To The System	150
Figure 5.14: Results Obtained for Query Part#A As Input Using a Shape Histogram Technique	152
Figure 5.15: Results Obtained for Query Part#B As Input Using a Shape Histogram Technique	153
Figure 5.16: Results Obtained for Query Part#C Used As Input To The System	154
Figure 5.17: Example Of Two Possible Feature Interpretations For A Machined Part..	155
Figure 5.18: Dominance Analysis For The Two Interpretations Of Feature 1 of part M_P With Respect To Their Closest Neighbor Feature 1 of Part M_Q	161
Figure 5.19: Results Obtained for Query Part#D Used As Input To The System	162
Figure 6.1: The Tool Maker of Part (b) Can Be A Potential Tool Maker for the Newly Designed Part (a)	165
Figure 6.2: Types Of Patches That Are Considered With Corresponding Location Point and Orientation Vector: (a) General (b) Cylindrical (c) Planar (d) Toroidal (e) Spherical	167
Figure 6.3: Equivalence Between Reduced Surface Feature Vectors and Set of Attributed Applied Vectors in \mathbb{R}^3	169
Figure 6.4: Set of Theta Intervals for Reduced Surface Feature Vector p_1 of P	183
Figure 6.5: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals	184
Figure 6.6: Histogram Representing Number of Converging and Non-converging Instances vs. Initial Conditions Used	191
Figure 6.7: (a) Initial Position of Part M_Q and Its Randomly Transformed Version Part M_P ; (b) Position of Part M_P Before Step3b(iii) of the Algorithm COMPUTESIMILARITYMEASURE_TWO; (c) Final Position of Part M_P	194
Figure 6.8: Results Obtained for Query Part#A Used As Input to the System	197

Figure 6.9: Results Obtained for Query Part#B Used As Input to the System.....	198
Figure 6.10: Results Obtained for Query Part#A As Input Using Fourier Transformation Based Technique.....	199
Figure 6.11: Results Obtained for Query Part#B As Input Using Fourier Transformation Based Technique.....	200
Figure 6.12: Results Obtained for Query Part#C As Input To The System; In Case (a) More Importance Is Given to Surface Patch Area and Location, in Case (b) More Importance Is Given to Surface Patch Type and Location	201

Chapter 1: Introduction

This chapter is organized in the following manner. Section 1.1 gives the necessary background to introduce the problem addressed in this thesis. Section 1.2 describes the motivation behind the research work described in this thesis. Section 1.3 identifies the major research issues addressed in this thesis. Section 1.4 describes how the remainder of the thesis is organized.

1.1 Background

Over the last fifteen years 3D CAD systems have become very popular in the industry. These CAD systems are being used to generate 3D models of parts. These models are used as a basis for engineering analysis and to generate manufacturing plans. 3D models also allow virtual prototyping and hence reduce the need for physical prototyping. Nowadays, organizations routinely set up databases of CAD models to enable all participants in the product development process to have access to 3D data to support their functions. Design, manufacturing, and service engineers are expected to greatly benefit from these databases. These databases are kept current by incorporating the latest versions of parts and hence significantly improve information dissemination. CAD databases for even moderate size companies are expected to be large in size.

Manufacturing companies are constantly looking for ways to reduce costs and the time-to-market. Intuitively, if two products are similar, it is possible to reuse information about one product to derive corresponding information about the other one. There are many possible applications where reuse of information can be of significant value. Representative examples include part-family formation, redesign suggestion generation, supplier selection, cost estimation, tooling design, machine selection, stock selection, and

design reuse. The following two examples illustrate in detail how shape similarity assessment can be used:

- **Machining Feature-Based Shape Similarity Assessment:** Nowadays, many job shops allow designers to submit a 3D model of the part to be machined over the Internet and provide a cost estimate based on the 3D part model. For some manufacturing domains such as rapid prototyping, reasonably accurate estimates of cost can be achieved by estimating volume or weight of the part. However, for some manufacturing domains such as machining, cost estimate depends on the geometric details of the object and automated procedures are not available for doing accurate cost estimation. Currently in such cases, humans perform cost estimation. In the Internet era, where designers solicit many quotes to make a decision, manual cost estimation is not economical. The cost of manufacturing a new part can be quickly estimated by finding previously manufactured parts that are similar in shape to the new part. If a sufficiently similar part can be found in the database of the previously manufactured objects, then the cost of the new part can be estimated by suitably modifying the actual cost of the previously manufactured similar part. Figure 1.1 shows an example of a previously manufactured part retrieved by a database search tool that can be used as a basis for providing a cost estimate for the new part.

- **Surface Feature-Based Shape Similarity Assessment:** Selecting a tool maker is an important step in molding of plastic parts. Many different kinds of tools exist that can be used to create plastic parts depending upon the shape of the part. Different tool

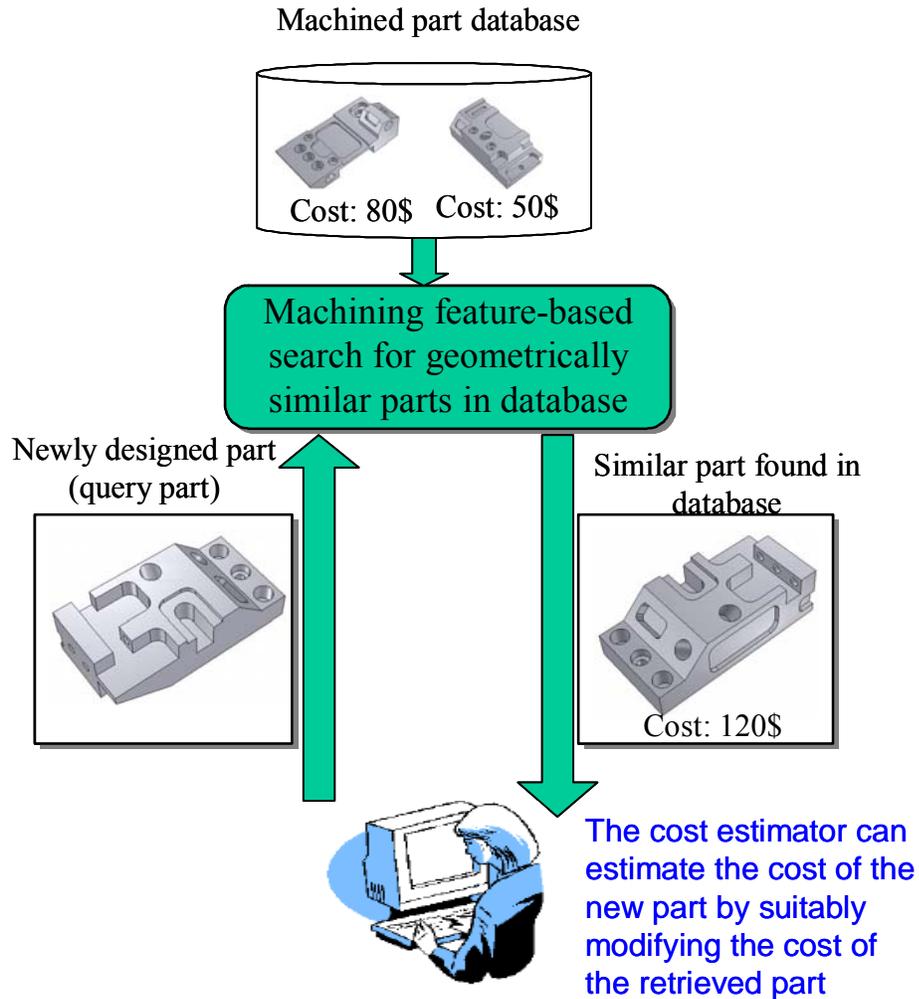


Figure 1.1: An Example of Using a Similar Part for Cost Estimation

makers specialize in different kinds of toolings. Therefore, one has to analyze the shape of the part to determine the most appropriate tool maker based on the type of tool needed for the part. Currently a fully generative method to determine the tool type based on the part shape does not exist. Therefore, another possible way to

identify potential tool makers is to find similar parts to the given part and identify tool makers for the similar parts. This method is currently being practiced by experienced part designers. However, they currently rely on their memory to locate the similar parts. Figure 1.2 shows an example of a previously molded part retrieved by a database search tool whose shape details are very similar to a new plastic part. Hence the same toolmaker that fabricated the mold for the retrieved plastic part can be approached to provide a mold for the new plastic part.

1.2 Motivation

The ability to search for similar products in a database by specifying a query product is expected to help companies in significantly reducing the associated time and cost compared with the manual methods of locating the similar products.

Currently, the following search tools are available to designers. First, if the part models are stored on computer hard drives, designers can use file name based search tools. These search tools work if a meaningful file naming convention based on part shape is adopted. However, developing and deploying a shape-based naming convention appears to be impractical in many large organizations. Second, designers can also attach text notations to parts and store them in the PDM database. This scheme only provides limited search capabilities and has a limited discrimination power. In the last few years many different part similarity based search tools have emerged. However existing shape similarity assessment techniques do not have a good performance in manufacturing applications. Shape similarity assessment techniques based on gross shape can only account for the overall shape of the parts and tend to ignore important shape details if they are relatively small in size. Figure 1.3 shows an example in which gross shape

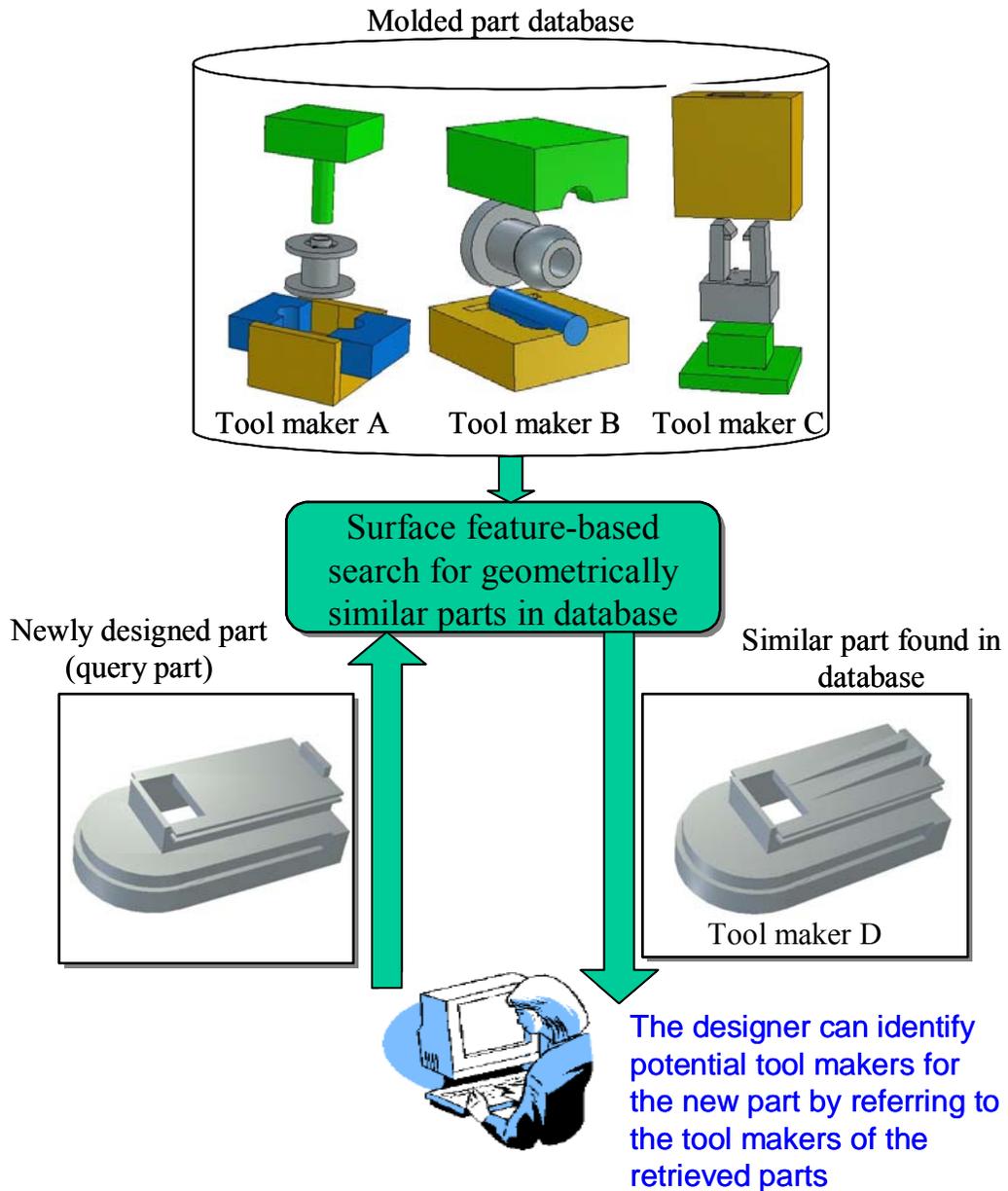


Figure 1.2: An Example of Using a Similar Part for Tool Maker Selection

similarity assessment techniques do not work from machining cost point of view. Part C in the figure would be ranked more similar to Part A than Part B if a gross shape similarity assessment technique was used. However machining cost of Part B is closer to Part A, as they both need one machining setup.

Most designer and engineers that use CAD system conceive the design in terms of shape features. In fact, in most modern CAD systems (e.g., Pro/Engineer, Unigraphics,

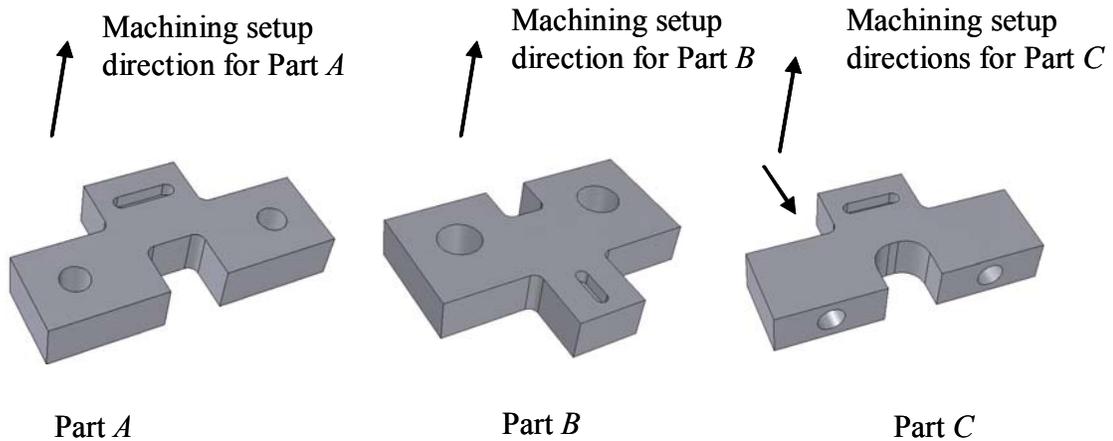


Figure 1.3: Part C Is More Similar to Part A In Gross Shape, But Part B Is More Similar to Part A in Machining Cost

etc.) features are the atomic elements using which parts and assemblies are defined. Based on our analysis of several applications, we believe that in order to be useful, the notion of similarity will have to be based on features. Furthermore, features are also used to define manufacturing and inspection operations. A feature can be viewed as a parameterized geometric object. Each feature has geometric (e.g., size, position, and orientation) and non-geometric (e.g., tolerance, surface finish) attributes associated with it. For a given application, not all feature attributes may play a role in determining the extent of similarity. For example, when looking for parts that have similar machining costs feature positions are not important. However, feature orientations are crucial as they affect the number of setups. On the other hand, in some other applications, feature positions may play an important role in determining similarity. Therefore, only

application-relevant feature attributes should be used in searching for similar parts. In summary, features provide a very convenient way of including critical details and filtering out irrelevant details in search for similar parts.

Existing feature based shape similarity assessment techniques also do not have a good performance in manufacturing applications. In fact they ignore relative positions and orientations of features, and hence cannot account for important issues such as feature interactions. Figure 1.4 shows an example in which a feature count based technique does not work from machining cost point of view. In fact Part C would be ranked more similar to Part A than Part B if a feature type and count similarity assessment technique was used. However the machining cost of Part B is closer to Part A, as they both need one

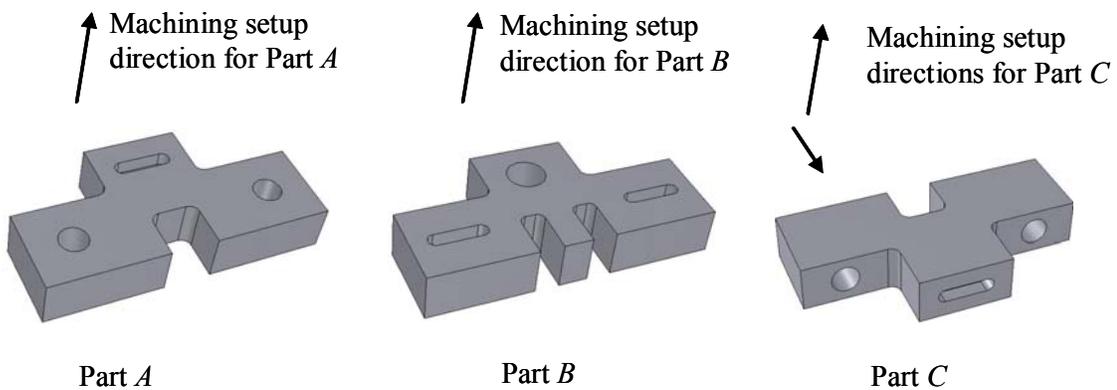


Figure 1.4: Part C Is More Similar to Part A In Feature Type And Count, But Part B Is More Similar to Part A in Machining Cost

machining setup.

Current search tools do not have a good performance on manufacturing applications. Hence currently designers locate parts by manually opening various files and browsing through them using a computer aided design system. This is a highly inefficient use of

designer's time, and is becoming a serious problem as the numbers of part models grow in the database.

1.3 Research Issues

To advance the field of shape similarity assessment for design and manufacturing applications, this thesis will focus on the following research issues:

- **Development Of A Feature-Based Shape Similarity Analysis Framework:** In many design and manufacturing applications the gross shape of the 3D parts does not play an important role in similarity assessment. Instead certain attributes of part features play a dominant role in determining the similarity between two parts. Therefore, we need a framework that uses feature information in assessing similarity. Typically, the degree of similarity between two parts can be measured using a distance function. Different applications typically require sometimes slightly (and sometimes significantly) different notions of similarities. Therefore, in order to be successful, the shape similarity search method will need to be able to work with user-specified distance functions. In addition to accounting for geometric attributes, this distance function will need to take into account non-geometric attributes such as tolerances and surface finish. Furthermore each feature characteristic can have different impact on similarity between parts, depending on the application. Hence the distance function must allow assigning a weight to each feature characteristic depending on its importance in the particular application addressed.
- **Optimal Alignment Of Feature-Based Models Based On Partitioning Of Transformation Space:** Feature-based similarity measures are defined using feature-based representations of the 3D parts. Hence a 3D part is represented by a set of

feature vectors. In order to assess similarity between two sets of feature vectors it is necessary to compute the distance between them. The distance depends on the relative position of the two sets of feature vectors belonging to two 3D parts, and on the closest neighbor to each feature vector. In general the closest neighbor to each feature vector and the distance value changes by applying a rigid body transformation to a set of feature vectors. So in order to assess similarity between two parts the transformation space is partitioned into regions within which the closest neighbor to each feature vector is invariant. Then the rigid body transformation that yields the minimum distance between the two sets of feature vectors needs to be computed for each region. Finally the rigid body transformation that yields the minimum distance over all the regions needs to be found. We will refer to such transformation as optimal alignment of feature-based models. Finding the optimal alignment is a computational task that involves a certain number of degrees of freedom, which depends on the transformation used and on the characteristics of the feature vectors being considered. In general finding the optimal alignment is harder if a higher dimension transformation is involved. For lower dimension transformations it is possible to design algorithms that can find the optimal alignment. Hence it is necessary to identify the classes of transformations for which algorithms to find the optimal alignment can be designed. Once the corresponding algorithms are designed it is also necessary to study their complexity in order to assess their efficiency.

- **Alignment of Feature-Based Models Based On Iterative Strategies:** It appears to be difficult to design algorithms to directly obtain the optimal alignment of feature-based models for higher dimension transformation spaces due to implementation

challenges in computing four and higher dimensional geometric entities. However, many applications involve finding the optimal alignment for higher dimension transformations. In these cases solutions can be found by iteratively solving many different alignment problems in lower dimension transformation spaces. However, iterative strategies can get stuck in local minima and they may take a long time to converge. Hence it is necessary to identify the classes of alignment problems for which iterative strategies can be used in a computationally efficient manner.

- **Applications Of Feature-Based Similarity Assessment Algorithms:** Feature-based similarity algorithms can be used to perform feature-based shape similarity assessment in many applications. However it is necessary for each application to choose a feature representation that characterizes each part based on its most significant characteristics. Then, based on the feature-representation chosen, it is necessary to study the performance of the feature-alignment algorithms for that particular application. In this thesis two applications will be used to demonstrate the possible use of the feature-based similarity assessment algorithms: part database search based on machining features and part database search based on surface features. In general there might be multiple possible interpretations of the machining features characterizing a part. Hence the machining feature-based similarity algorithms must account for multiple possible interpretations of features.

1.4 Thesis Outline

This thesis is organized as follows.

Chapter 2 describes literature survey related to shape similarity assessment and alignment problems.

Chapter 3 describes optimal alignment algorithms based on partitioning of transformation spaces.

Chapter 4 describes alignment algorithms based on iterative strategies.

Chapter 5 describes a machining feature-based shape similarity assessment algorithm that can be used to search part databases.

Chapter 6 describes a surface feature-based shape similarity assessment algorithm that can be used to search part databases.

Chapter 7 identifies the main research contributions of this thesis and describes the anticipated industrial benefits from this research work.

Figure 1.5 shows the general organization of the thesis.

Feature-based shape similarity assessment

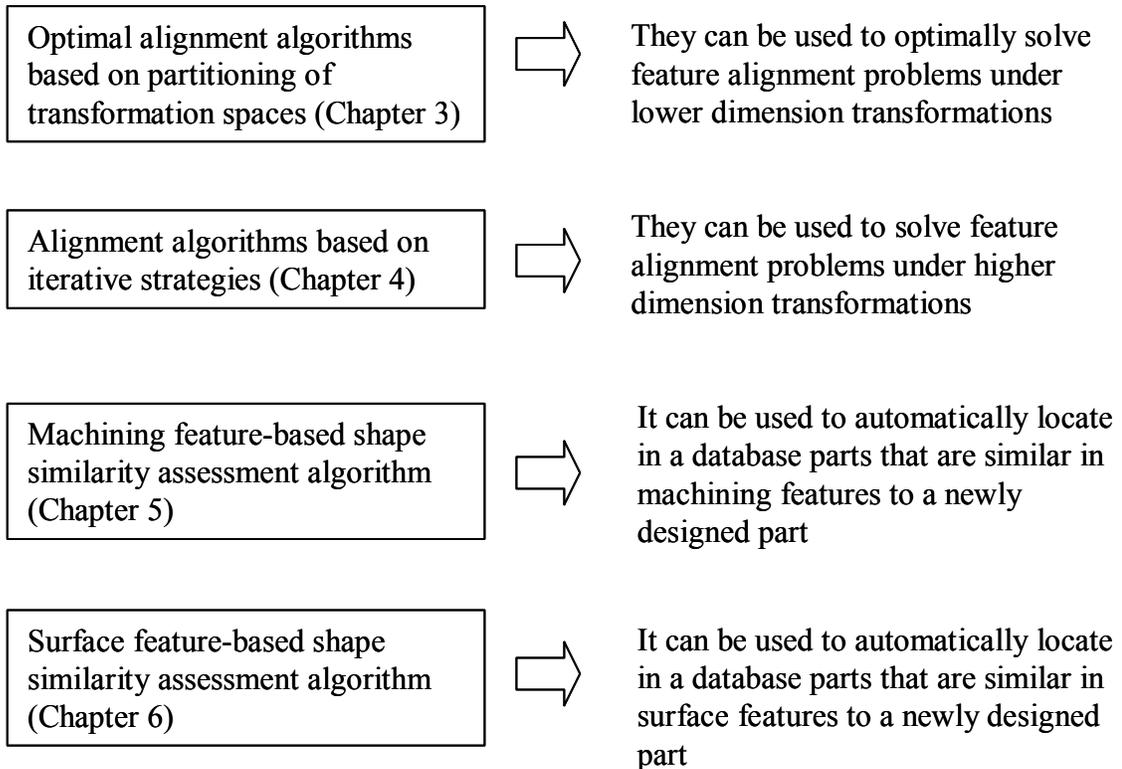


Figure 1.5: Organization of the Thesis

Chapter 2: Related Research

The popularity of 3D models poses new challenges in managing databases of increasing size. With more and more 3D models being added to databases, a need to organize and index databases of 3D models is imminent. This will provide a systematic and efficient way of retrieving similar models from the database. One of the main criteria used to organize and index databases is shape similarity of the 3D models.

Over the last few years several papers have been written that describe algorithms for shape similarity assessment [Card03, Camp01, Tang04]. Specialized algorithms for medical [Keim99, Youn74] and computer vision [Arma93, Belo01, From04, Mori01, Sidd99, Thac95, Zhan99] applications have also been developed. However the main body of work can be divided into two different categories: (1) similarity assessment of 2D shapes, and (2) similarity assessment of 3D shapes. Representative work in 2D category includes shape signatures based on Fourier descriptor [Arbt90], turning functions [Arki91], bending functions [Youn74], and arch height functions [Lin92]. A comprehensive discussion of 2D shape signatures can be found in [Alt96], [Lonc98], [Velt01]. 2D geometry and 3D geometry have several fundamental differences. Unfortunately in most cases methods for computing and matching signatures of 2D shapes cannot be easily extended to 3D shapes. Hence it is often necessary to build new algorithms that deal with 3D shapes. We will mainly focus on algorithms that deal with 3D shapes.

This chapter is organized in the following manner. Section 2.1 provides an overview of the various techniques that are being used to perform similarity assessment along with a classification scheme. Sections 2.2 to 2.7 describe various approaches in detail,

summarizing their advantages as well as limitations. Section 2.8 addresses the point alignment problem, describing some of the techniques used. Finally, Section 2.9 concludes this chapter with a few observations.

2.1 Overview of Techniques

Various techniques have been developed to perform similarity assessment of 3D solid models. A computationally efficient way to solve this problem is to first abstract 3D shapes into shape signatures and use them to perform similarity assessment. Shape signatures are abstractions of the actual shapes that completely characterize the 3D object. For instance a 3D object can have a matrix, a set of vectors or a graph as shape signature. Similarity assessment between two 3D parts involves two main steps. The first step is to compute the shape signature of the object. The second step is to compare the shape signatures by a suitable distance function. Most papers in literature argue that the distance function should satisfy certain properties. Some of them are listed as follows and will be used to evaluate the shape similarity techniques. Positivity requires that the distance function be non-negative. Identity requires that, if the distance function is equal to 0, the two parts compared be the same and vice versa. Symmetry requires that the distance function be symmetric. Triangle inequality can be defined as follows: consider three solid models x , y and z . Let $\delta(S(y),S(w))$ be the distance between the shape signatures $S(y)$ and $S(w)$ of two solid models y and w . Triangle inequality is satisfied if $\delta(S(x),S(y)) + \delta(S(y),S(z)) \geq \delta(S(x),S(z))$. Even shape signatures should satisfy certain properties. For instance, they should be invariant with respect to the representation of the solid model (CSG, B-rep etc.) and to the transformations applied to it. They should also be robust and sensitive with respect to changes in shape. Majority of the techniques used

in the shape similarity assessment area can be classified on the basis of the type of shape signatures being used. The following types of shapes signatures are currently being used.

- **Features:** Feature-based techniques compute the shape signature of an object based on the type, size, orientation, number and other properties of the features and their interactions. Once the features are extracted and their significant characteristics are determined, the comparison is carried out using a suitable distance function. For example, feature-graph signatures are compared by performing graph isomorphism. These techniques discriminate the 3D models based on the features and their characteristics. Hence, they do not consider the gross shape of the object. Feature interactions and multiple interpretations still pose significant challenges to successful extraction of features. Many different types of approaches have been developed [Gupt99, Karn05]. Some of these techniques appear to be promising for the cost estimation domain. Particularly techniques described in [Rame01] can be used as a filter to quickly prune dissimilar machined parts. Section 2.2 describes representative feature-based techniques in detail.
- **Spatial Functions:** These techniques use shape signatures that are spatial functions. An example of a spatial function is the Gaussian map that maps the set of normal vectors of a solid onto a unit sphere. The problem of matching and comparing 2D spatial functions defined over a unit sphere involves manipulating three degrees of freedom (the three angles needed to align the surface of a sphere). The main challenge in these techniques is to identify the characteristics to be represented using spatial functions and to determine an efficient matching procedure to compare two

shape signatures. Section 2.3 describes representative spatial function based techniques in detail.

- **Shape Histograms:** These techniques are based on sampling of points on the surface of the 3D models. Several significant characteristics can be extracted from the set of points obtained. Once these characteristics are determined, they are organized in the form of histograms that store the frequency of occurrence of their values. Then, these histograms are compared using a suitable distance function. The accuracy of these signatures depends on the number of points used. Large numbers of points result in higher accuracy. However, the efficiency of these signatures is inversely proportional to the number of points. Thus with an increase in the accuracy, the efficiency decreases. Section 2.4 describes representative shape histogram based techniques in detail.
- **Section Images:** These techniques use sections of the solids as shape signatures. Solids are sectioned at various places and the sections are then analyzed for similarity. This analysis can be carried out using neural network or by using 2D similarity assessment techniques. As these techniques involve sections, they are not invariant to scaling, translation and rotation and can compare objects only with known orientations. They are well suited for rotational parts due to their rotational symmetry. Techniques that use neural networks do not actually compare the two solids but classify the solids into groups based on group technology codes. Based on the images of the sections they determine the group code to which the part belongs. They are robust but involve training of the network to improve the classification and hence require significant time to implement. Also the number of sections affects the

accuracy of comparison. If number of sections being considered is small, then small features on the objects may not be recorded. Section 2.5 describes representative section image based techniques in detail.

- **Topological Graphs:** These techniques use topological graphs as shape signatures to perform similarity analysis. These graphs usually represent the connectivity information of the boundary of the solid such as the adjacency between faces. The nodes and edges of the graph may carry additional information related to the solid model. The comparison can then be carried out by matching the graphs based on relevant characteristics or by graph isomorphism algorithms. However, comparing graphs is not trivial and requires considerable computational time if a graph isomorphism algorithm is used. In order to have sufficient discrimination capability, the graphs need to store as much information as possible. But storing excessive information further increases the computational time. Hence there exists a tradeoff between accuracy of comparison and computational time. In some cases, graph properties such as degree of nodes, number of nodes, number of edges, eigenvalues etc. have been used for comparison. Section 2.6 describes representative topological graph based techniques in detail.
- **Shape Statistics:** Many shape comparison techniques use basic geometric properties in order to perform coarse comparison between solids. They may also be used to reduce the search space. Commonly used properties include volume, surface area, convex hull volume etc. These numerical values representing statistical properties of the shape form the signature of the solid. Such signatures do not carry any topological information. These methods cannot provide sufficient discrimination power for

detailed comparison but are useful as quick and efficient filters. The approaches in this category are explained in Section 2.7.

- **GT Codes:** Group Technology has traditionally been used to categorize parts having similarities in design and manufacturing. Group Technology (GT) involves classifying similar products into groups in order to achieve economies of scale normally associated with high-volume production [Burb75]. In order to implement GT, one must have a concise coding scheme for describing products and a method for grouping (or classifying) similar products, such as the popular Opitz, DCLASS, and MICLASS schemes. In each case the basic idea is for the users to use various tables and rules to capture critical design and manufacturing attributes of a part in an alphanumeric string, or GT code, that is assigned to that part. However, as the classification is done manually, it is subject to individual interpretation. It has been shown that human perception of similarity is subjective [Sant95]. Thus, there are possibilities of errors in such classifications.

2.2 Feature-Based Shape Signatures

The first step in the technique described in [Rame01] consists of extracting the features from a B-rep model. This is achieved by constructing cells that are portions of space resembling machining features. Once these cells are obtained following a series of rules, they are mapped to machining features. Then, relevant feature characteristics are used to perform the comparison. For this, a feature class is defined as a group of geometrically similar features (i.e. identical topology and relative angles between faces). A T-group is a group of features in which the features differ from each other only by translation. Similarly an S-group is such that the features belonging to it have the same critical

dimensions. Seven characteristics are used for comparison. These include feature existence, feature count, feature direction, feature size, directional distribution, size distribution and relative orientation. Feature existence represents the number of different classes of features present in the object and is expressed as a binary vector of dimension n , where n is the total number of feature classes in the two objects being compared. Each element in the vector assumes a value of 1 if the corresponding type of feature is present in the object or else it is 0. Feature count represents the number of instances for every class of feature in a given 3D object. It is expressed as a vector of dimension n . Each element denotes the number of instances of the corresponding feature. Feature direction represents the number of T-groups for every class and is expressed as a vector of dimension n . Each element indicates the number of T-groups for the corresponding class. Feature size is similar to feature direction and represents the number of S-groups. For every class of features, directional distribution represents the number of instances of features within a T-group belonging to the class considered. Size distribution is similar to directional distribution and is defined for S-groups. Finally, relative orientation represents the relative orientation between T-groups over all the different classes of features. Two objects P_A and P_B are compared by a weighted distance using the following formula.

$$d(P_A, P_B) = \left(\sum_{i=0}^n w_i [d_i(c_i(A), c_i(B))]^r \right)^{1/r}$$

where n is the number of characteristics chosen for the comparison, d_i is the distance between the two compared objects relative to the i th characteristic and w_i is the weight assigned to the i th characteristic. The characteristics considered in the comparison have to be independent of each other. Only planar and cylindrical surfaces are considered.

Objects where the cylindrical features intersect other faces non-orthogonally are also ruled out. This technique also does not account for local feature interactions.

Another technique described in [Cici00, Cici01, Cici02], involves feature extraction and comparison to determine similarity between mechanical parts. It defines a Model Dependency Graph for each of the two objects being compared and determines the largest common sub-graph between them to assess similarity. The feature extraction is carried out using FBMach System consisting of a library of machining features. After performing feature extraction, the Model Dependency Graph representing the features and their interactions is defined. The nodes of this graph correspond to features and store attributes of the features as ‘labels’ at the nodes. Thus model dependency graph $G = (V, E)$ comprises of a set of nodes $V = \{f_0, \dots, f_n\}$ where f_i is a machining feature of the solid. An edge between the two nodes exists if the corresponding features f_i and f_j have non-zero intersection between them. Thus $E = \{\{f_i, f_j\} \text{ such that } \text{vol}(f_i) \cap \text{vol}(f_j) \neq \emptyset\}$. To compare the two solids, the largest common sub-graph (LCS) between the two model dependency graphs needs to be determined. The problem of exactly determining the largest common sub-graph, however, is NP-complete and hence a hill climbing/ gradient descent algorithm is used to obtain a large enough sub-graph. The proposed algorithm involves assigning random mapping between the nodes of the two graphs initially, and then swapping the mappings such that evaluation function assumes the lowest value. The evaluation function H is the count of the number of mismatched edges. The measure of similarity is given as $H^* = \frac{\min\{H_1, \dots, H_n\}}{|E_1|}$ where H_1, \dots, H_n are the final values of H from up to n random restarts of the algorithm and $|E_1|$ is the number of edges in the smaller of the two graphs.

This technique provides means for determining objects having similar machining features. However, the Model Dependency Graph generated using this method is not unique for a given solid. This is because the features can be constructed in different order and in multiple different ways. In Figure 2.1 an example of features that can be constructed in different ways is shown. In fact, the issue of multiple feature interaction is a common problem to all existing feature-based similarity assessment methods. This technique considers only feature interaction and does not account for feature size and orientation.

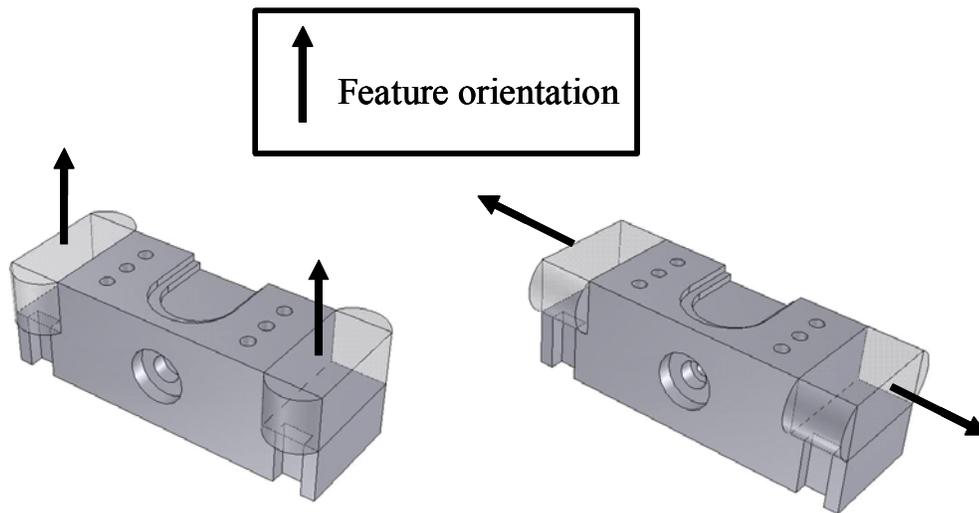


Figure 2.1: An Example Depicting Different Ways of Representing Features

The technique described in [Kim03] is based on convex decomposition of 3D solid models and their form feature decomposition (FFD) and negative feature decomposition (NFD). These decompositions result in a tree yielding a hierarchical representation of a 3D model. The convex hull is the root of the tree, i.e. the most abstract representation of the 3D model. In FFD the features are detected and divided into positive features (i.e. volume added to part) and negative features (i.e. volume subtracted from part). In NFD

only the negative features are considered along with the machining precedence among features and some feature characteristics such as feature type, union and intersection of accessibility cones and features number. Hence it is possible to identify groups of features using the information contained in the tree. Each group corresponds to a branch of the tree. The groups of features are compared based on the characteristics of the features listed previously. So 3D models are compared by matching pairs of branches of the corresponding trees. The branches are matched using optimization algorithms such as best-first search ones. Furthermore the matching is refined using feature machining directions. This technique's performance is also affected by multiple feature interpretations.

The technique described in [Elin97] is based on a graph representation of the input 3D models. This graph representation is used as the shape signature for the model. Let us consider two objects, m and m' . Then $c(m)$ and $c(m')$ will be the value of a characteristic for the object m and m' . The equivalence relation $E_i(m, m')$ is true if and only if $c(m) = c(m')$ (i.e., the two objects are equivalent with respect to the characteristic considered). So, depending on the number of properties or characteristics considered, many different equivalence relations can be defined. Let R_i be an equivalence relation. R_i is valid if R_{i+1} is valid, as the former is contained in the latter. To compare the two objects it is necessary to define $M(m, m')$ as the biggest value of i for which $R_i(m, m')$ holds true. So, if we consider three objects m , m' and m'' , and if $M(m, m') > M(m, m'')$ then m is closer to m' than m is to m'' . Given the previous definitions, a tree can be built whose leaf nodes represent the parts in the database being compared and the rest of the nodes represent the equivalence relations. Once the tree is obtained, it is possible to obtain the degree of

similarity between two objects by calculating the value of M . In [Elin97], the main focus is on the manufacturing aspects of the object represented by the 3D model (see also [Elin96]). For each model a graph called *design signature* is constructed. The nodes represent some characteristics of the design, while the edges represent the relationships among these characteristics. An application is provided in [Elin97], where nodes represent features and edges their interactions. The nodes are labeled with a number of parameters, such as type of feature and machining direction. The edges are labeled depending on the type of intersection occurred: a description of the types of intersections is provided in [Elin97]. The equivalence relation considered in this application is isomorphism between two graphs. It is usually an expensive task, but in this case it is made easier from the labeling of nodes and edges. In fact the labeling allows matching sub graphs more easily.

In the technique described in [Srin98], different attributes of features such as feature type, machining type etc. are stored in Attribute Type table. A qualitative matrix is used to record all the feature interactions. By searching through the Attribute Type table and qualitative matrix similar models can be retrieved.

Techniques described in this section perform similarity assessment based on the features of the parts and their characteristics. Hence the techniques have been primarily developed for product design and manufacturing.

2.3 Spatial Function Based Shape Signatures

In [Hebe95] a spherical representation that stores the curvature distribution of 3D surfaces of an object is used as signature. The solids to be compared must have a genus of zero. To generate the representation, a tessellated sphere is deformed such that it closely

approximates the shape of the object. Each node of the tessellated sphere has three adjacent nodes. A local regularity constraint is introduced during the deformation to ensure that each mesh is similar to others in area. According to this condition, the projection of each node on the triangle formed by the adjacent three nodes should coincide with the centroid of that triangle. This representation then yields the shape signature of the object. Two types of forces are used to perform the deformation. One type of force tries to bring the mesh nodes closer to the surface, while the other helps in maintaining the local regularity constraint. The algorithm for deforming the sphere is based on combining these two forces between the solid model and the spherical mesh. After a uniform surface mesh is obtained, the curvature at every node of the mesh is computed using three discrete nodes in the neighborhood of that node. Once the curvature function is defined, one of the two objects is rotated such that it aligns with the other. Let S_A and S_B be the mesh representations of the shapes A and B , and $k_I(S_A)$ and $k_R(S_B)$ be the local curvature functions, where I and R are identity and rotation matrix respectively. Then the distance between the shapes A and B can be computed using L_p norm as shown below. Alternatively Hausdorff distance may be used to compute the difference [Hutt90b].

$$d_p(S_A, S_B, R) = (\int |k_I(S_A) - k_R(S_B)|^p dS)^{1/p}$$

where $d_p(S_A, S_B, R)$ is the sum of the curvature differences over the sphere [Shum96].

The distance between A and B is then expressed as follows.

$$D_p(A, B) = \min_R d_p(S_A, S_B, R)$$

which is d_p minimized over all possible rotations R .

The shape similarity assessment in this case is invariant with respect to translation, rotation and scaling as the curvature depends only on the relative locations of the four nodes that are used to define the local curvature function k . As the distance is computed using L_p norm, it obeys the positivity property. It also satisfies the identity, symmetry, and triangle inequality properties [Shum96]. The distance between the two shapes can be computed in time $O(n^2)$ where n is the number of nodes on the sphere. However, this technique is restricted to solids having zero genus (i.e. solids without holes). This is a serious restriction considering that holes are a common feature in CAD models. Also, the mesh is an approximate representation of the solid and the accuracy depends on the number of tessellations. As the mesh becomes finer, accuracy increases but so does the computational time. A technique similar to this one is described in [Schw87].

In [Ko03a] a technique to match two free-form solids is described. The matching is performed by using the distribution of Gaussian and mean curvature over the 3D models and minimizing a distance function defined in [Ko03a]. The distance function is based on Euclidean distance between points. The Interval Projected Polyhedron (IPP) algorithm is used. It finds correspondences among the intersection points between iso-curvature lines of the two solids that are being matched. These correspondences are found by solving a non-linear polynomial equation system. The constraints for this equation system are obtained from the distance function being minimized. The equation system yields a solution, which is the translation and rotation to be applied in order to match the two solids. If tight tolerances and good curvature estimations are used the accuracy of the technique is high but the efficiency decreases. In [Ko03b] the matching of solids is

performed by referring also to the umbilical points on their surface. A possible application is copyright preservation.

The technique described in [Tuzi00] uses the slope diagram representation [Ghos96] of convex polyhedra [Grun67] and uses mixed volumes and volumes based on Minkowski addition [Ghos93] to define the similarity measure. The definition and mathematical representation of mixed volume can be found in [Tuzi00]. A *slope diagram representation* (SDR) is one where a face is represented on the unit sphere by a spherical point, which is an intersection of its normal with the unit sphere. An edge is represented by a spherical arc, which is an arc of a great circle joining the points representing the two faces that share the edge. A vertex is represented by a region of the sphere known as a spherical polygon bounded by the spherical arcs corresponding to the edges sharing the vertex. Let P and Q be the two objects to be compared. Then a rotation r is applied to the SDR of Q while the SDR of P is kept fixed. Such a rotation r can be determined by identifying a set of finite number of critical rotations. Such critical rotations include situations where spherical points of the rotated SDR of Q intersect spherical arcs or points of the SDR of P . These rotations minimize the objective functions defined based on volumes and mixed volumes. This technique, which is defined for convex shapes, is invariant with respect to translation and rotation. However, a considerable computational effort is needed to determine the set of finite critical angles. The technique described in [Tuzi00] is restricted to convex polyhedra. Hence, it is limited in scope.

The techniques described previously have not been developed for product design and manufacturing, but they could be applied to this domain. In particular, these techniques could be used in applications where curvature plays a major role.

In [Novo03] 3D Zernike descriptors are used to represent 3D models. The 3D models are voxelized in order to obtain the Zernike descriptors. An object function that defines the object is obtained using the Zernike functions and the Zernike moments. This object function is projected onto a set of orthonormal Zernike functions. The formulae and theory behind it are presented in detail in [Novo03]. Zernike functions are combinations of monomial up to a given order. So Zernike descriptors (i.e. Zernike functions, moments and object function) are invariant with respect to scaling and affine transformations. Their performance in similarity assessment has been compared in [Novo03] to other techniques such as spherical harmonic descriptors [Funk03]. Zernike descriptors have a better performance in similarity assessment. In fact they are able to detect topological and geometrical details that spherical harmonic descriptors cannot when the complexity of the 3D models increases.

The accuracy of Zernike descriptors increases with the number of Zernike moments considered and with the number of voxels used in voxelization. In fact this way higher frequencies are considered and the discrimination capability increases [Novo03]. On the other hand, with higher frequencies the Zernike moments are unstable and not robust with respect to geometry and topology. Hence it is necessary to trade off between accuracy of similarity assessment and robustness with respect to topology and geometry. Even efficiency decreases if the number of Zernike moments considered is increased.

In [Dey03] a topological approach is used to represent the 3D models. The 3D models are represented initially by point samples. A flow discretization is used and applied to the set of input points. The technique uses tools such as Voronoi diagrams and Delaunay triangulations [Dey03]. The part is finally divided into a number of Delaunay tetrahedra.

These tetrahedra are grouped: each group corresponds to a feature of the part. A feature is represented by a weighted point. The point is the weighted average of all the centroids of the tetrahedra forming each feature. The weight used is the volume of the tetrahedra that form the feature. Hence each part is represented by a set of weighted points. Similarity assessment is performed by aligning the sets of points representing the two parts being compared and evaluating the distance between them. In this technique the features extracted do not necessarily correspond to the intuitive ones [Dey03].

In [Elad01] 3D models in VRML format are represented by moments calculated on the model surfaces. A weighted Euclidean distance is used to compare two models. An interactive and iterative database search procedure is used to retrieve similar parts from a database. After each application of the database search the user identifies the relevant and irrelevant top matches by his/her criteria. Then a quadratic optimization problem is solved such that the weights of the distance function are modified to fit user preferences. This iterative process ends when the user is satisfied with the outcome of the database search.

In [Vran01] and [Yu03] two techniques that rely on spherical harmonics are described. In both of them a preliminary alignment of the parts is performed to obtain invariance with respect to translation, rotation and scaling. Then spherical harmonics are obtained by shooting rays from the origin and detecting the distance of the origin from the intersections of the rays with the model surface. The similarity assessment is performed by evaluating Euclidean distance on the Fourier transforms of the spherical harmonics obtained.

2.4 Shape Histogram Based Shape Signatures

The technique described in [Osad01, Osad02] computes shape distributions of solid models using shape functions and then compares these shape distributions to assess similarity. Initially random points are generated on the surface of a triangulated solid. For creating a single random point, a triangle is randomly selected from the set of triangles that make up the solid. A point P on the surface is then obtained by generating two random numbers r_1 and r_2 and evaluating the following expression.

$$P = (1 - \sqrt{r_1})p_1 + r_1(1 - r_2)p_2 + \sqrt{r_1r_2}p_3$$

where p_1 , p_2 , and p_3 are the points representing the vertices of the triangle under consideration.

Once a set of random points is obtained on the surface of the solid model, different shape functions are used to compute shape distributions for the solid model. The shape functions include

- D1: Computes the distance between a fixed point and a random point. This shape function is not suitable as the chosen fixed point is usually not invariant to rotation or translation.
- D2: Computes the distance between two random points. This function is invariant to rotation and translation and is robust.
- D3: Computes the square root of the area of triangle generated by three random points. This function is also invariant to translation and rotation but not as efficient as D2.

- D4: Computes the cube root volume of the tetrahedron generated by selecting four random points. This method is computationally inefficient even for lesser number of points.
- A3: Computes the angle between three random points. This function is invariant to translation, rotation and scaling but it is not very robust.

Out of these the D2 shape function has been found to be most suitable for computing shape distributions due to its robustness and efficiency along with invariance to rotation and translation. After calculating the distances between random points, they are normalized using the mean distance. The shape distribution is the histogram that measures the frequency of occurrence of distances within a specified range of distance values. Once the shape distributions are generated the distance between the two solid models is computed using L_N norm. Thus the distance can be expressed as follows.

$$D(f, g) = (\int |f - g|^N)^{1/N}$$

where f and g are the shape distributions. Usually L_2 norm is used for comparison. Other distances such as Earth Mover's distance [Rubn98] or Match distances [Shen83], [Werm85] can also be used.

This technique is robust and efficient. Also there is no restriction on the type of solid models that can be compared. However, as this method involves generating random points on the surface of the solid, it fails to satisfy conditions of identity and symmetry. As the number of points increase the comparison is more robust, but the computational time increases. Furthermore as objects become more and more complex, the shape distributions tend to assume similar shape. This results in inaccurate comparison of solid models. Figure 2.2 shows three parts and their corresponding D2 shape signatures. Based

on our implementation of the algorithm it can be seen that *heat_exchanger2* is more similar to a *grip* than to *heat_exchanger1*. Thus this technique has limited discrimination capability.

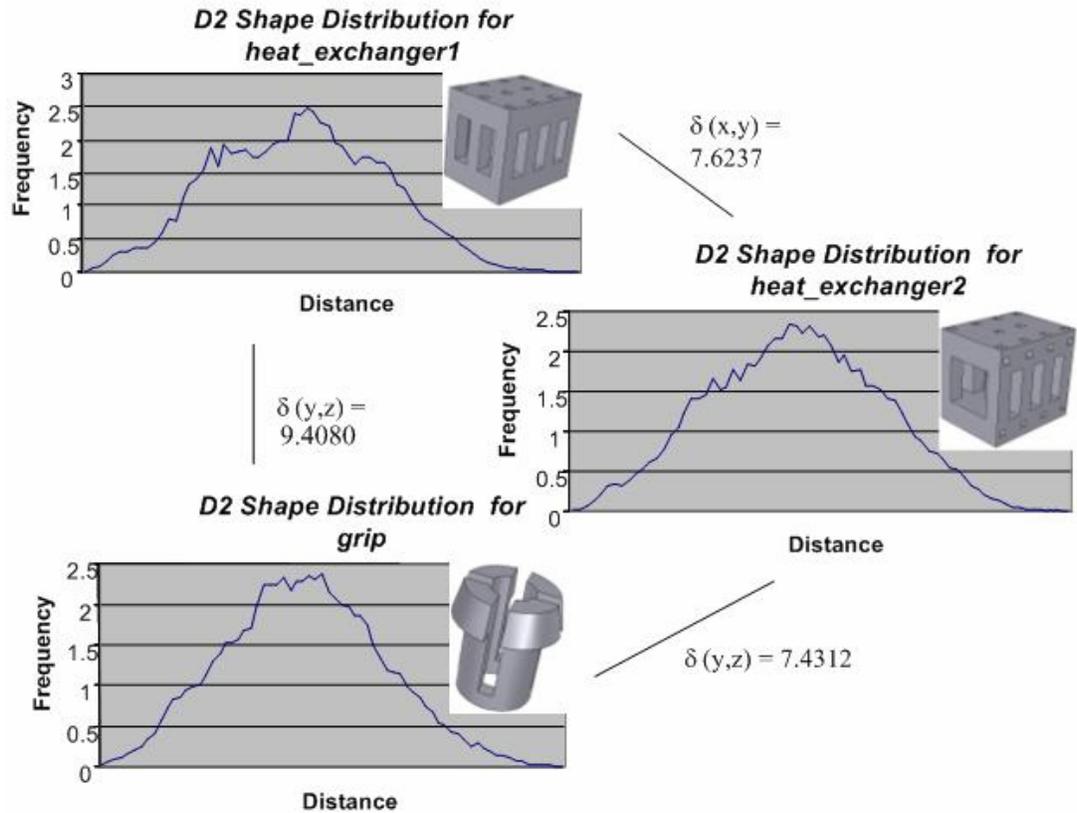


Figure 2.2: An Example Indicating the Low Discrimination Capability of Shape Distributions

An extension of the previous technique is described in [Ip02]. The procedure for generating random points on the surface as well as the shape function used is the same. However, instead of computing a single shape distribution for each solid model, this method computes four different shape distributions based on in/out classification of the line joining the random points whose length is the distance measure. The first distribution is the same as in the previous method. The second distribution takes into consideration

all the lines joining the random points that lie inside the solid model. The frequency of occurrence of the length of these lines is also measured. The third distribution accounts for all the lines that lie outside the solid. Finally, the fourth distribution includes those lines that lie partially inside the solid and partially outside. The distributions are then compared using L_2 norm. This technique aims at improving the ability of the previous one to distinguish between complex parts having detailed features. However, it fails to satisfy the properties of identity and symmetry for the same reasons as the previous method. Moreover its computational efficiency is low, as it involves determining whether a line lies inside, outside or partially inside a solid. In [Ip03] this technique is used with an automated learning system based on k closest neighborhood learning algorithm.

In [Ohbu03a] another extension of the technique described in [Osad01, Osad02] is described. The shape distribution is based both on the distance between random points generated as described previously and on the angle between the normal vectors to the triangles to which the random points belong. The corresponding histogram is called Absolute Angle-Distance histogram. In [Ohbu03b] a further improvement is obtained by defining a number of alpha-shapes representing the 3D models. Each alpha shape represents the model at a different resolution, from convex hull to detailed representation. Then for each alpha-shape the corresponding Absolute Angle-Distance histogram is created. This technique performs better than the one described in [Ohbu03a] but it is less efficient [Ohbu03b].

The similarity assessment techniques described in this section can detect gross shape similarity. Hence these techniques could be used to perform a pruning on the database to search for similar parts in design and manufacturing applications.

2.5 Section Image Based Shape Signatures

Manual classification and coding of parts for group technology applications is time-consuming and prone to errors. In [Kapa91] a neural network system has been proposed for classifying parts based on bitmaps of the part drawings. The neural network consists of number of layers of neurons, which include an input layer, some hidden layers and an output layer. The theory of neural network systems is described in [Khan90]. The input to the neural network system is a vector I containing bit data that represents the image of a part drawing. For every input i , there is a neuron with a weight vector W^i attached to it. The input to each neuron is the dot product of these two vectors. The output of the neuron is a vector O_i corresponding to the input and activation function f_s . The algorithm for determining the output is described in [Lipp89]. The maximum number of neurons that can be used in any given layer is defined using Kolmogorov's theorem [Ande88]. The vector element of O_i with the highest value represents the group to which the part belongs. The output in this case is an Opitz code used to classify rotational parts based on characteristics such as length to diameter ratio. At the beginning, random values are assigned to the weight vectors. The network is then trained using standard inputs for which target outputs have been identified. If the difference between the actual output and target output is above the threshold value then the weight vectors are adjusted such that the error is reduced.

This technique involves classification of part drawings and hence it does not account for rotation or translation of the solid model. It classifies the part drawings using group technology. There is no direct comparison between the part drawings: they are classified based on their characteristics such as L/D ratio, presence of holes etc. However, the solid

models available in the databases or the Internet have arbitrary orientation and hence this method will require manual intervention to identify the part drawing with desired orientation.

The technique described previously involves classification of rotational parts using neural network system. In [Chun94] this classification has been extended to include 3D parts based on their binarized part drawing image. A back-propagation neural network system has been proposed to classify the 3D parts into a number of predetermined part families. The theory of back-propagation neural network is explained in [Hech89]. Also some concepts related to the neural network such as learning rate, number of neurons in the hidden layer and number of hidden layers are discussed in [Chun94]. The modified technique described in [Chun94] is similar to the one discussed before and uses gradient search procedure to determine the weight vectors such that they reduce the error between the target value and actual value. The learning rate should allow the learning algorithm to converge to minimum error solution without oscillation of the network and without getting trapped in a local minimum. The local minimum can be avoided by adjusting the value of the momentum. The momentum is similar to physical momentum and allows the network to bounce from a local position and seek a better solution. The formation of part families depends on the predetermined number of part families. Also the learnability of the group increases as the number of hidden neurons increases. This technique involves classification of parts using neural network and hence suffers from the same problems associated with the previous technique. However, it provides an insight into the various parameters that affect the performance of the neural network system used to classify the part drawings.

In [Herr00] a technique for variant fixture planning is defined. The two dimensional projection of the part to be manufactured is considered. The attributes used for assessing shape similarity include the maximum inter vertex distance (MID), the maximum vertex edge distance (MVED) and the total enclosed area (TEA) [Herr00]. Three similarity measures are defined to compare the parts, each one corresponding to one of the parameters mentioned previously.

In [Chen03] a 3D part retrieval system that is based on similarity between 2D views is proposed. 2D images are compared using both contour shape descriptors (based on Zernike moment descriptors) and region shape descriptors (based on Fourier descriptors). This technique is invariant with respect to translation, rotation and scaling. In choosing the number of 2D views to be considered it is necessary to trade-off between efficiency and accuracy.

These techniques have been applied to product design and manufacturing applications. Most of them are specifically used to classify the parts in a database to reuse design information.

2.6 Topological Graph Based Shape Signatures

2.6.1 Model Signature Graphs

In [McWh01a] Model Signature Graphs have been proposed for topological comparison of solid models. They are an extension of Attribute Adjacency Graphs, mentioned in [Josh88], and are introduced in order to consider curved surfaces. Model Signature Graphs are constructed from the boundary representation of the solid. Each node in the Model Signature Graph represents a face of the solid model. There exists an edge between two nodes of the graph if the corresponding faces are adjacent. This graph forms

the shape signature of the solid model. Along with the connectivity information between the faces, the identifier for the face (planar, conical, etc.), mathematical representation of the surface, surface area and set of surface normals can also be stored at the nodes. The edge of a Model Signature Graph represents the edge between two adjacent faces of the solid model. Identifier for the edge, concavity/convexity of the edge, mathematical representation of the edge and length of the curve can also be stored at the edge [McWh01a]. This additional information helps in more accurate comparison of the solid models. However, in the current implementation the edge angle information is not stored. Thus two simple objects may have the same Model Signature Graph, as shown in Figure 2.3. Once a Model Signature Graph is constructed, the solid models are compared using spectral graph theory. The eigenvalues of the Laplacian matrix [Chun97] are used in the comparison. A ‘normalized’ form of Laplacian is defined as follows.

$$L_G(u, v) = \left\{ \begin{array}{ll} 1 & : \text{if } u = v \text{ and } d_v \neq 0 \\ \frac{-1}{\sqrt{d_u d_v}} & : \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & : \text{otherwise} \end{array} \right\}$$

where u and v are nodes of the graph and d_u and d_v are the degrees of the nodes. The eigenvalues of the Laplacian are strongly related to other graph properties such as the graph diameter. The *graph diameter* is the largest number of vertices, which must be traversed, in order to travel from one vertex to another in the graph.

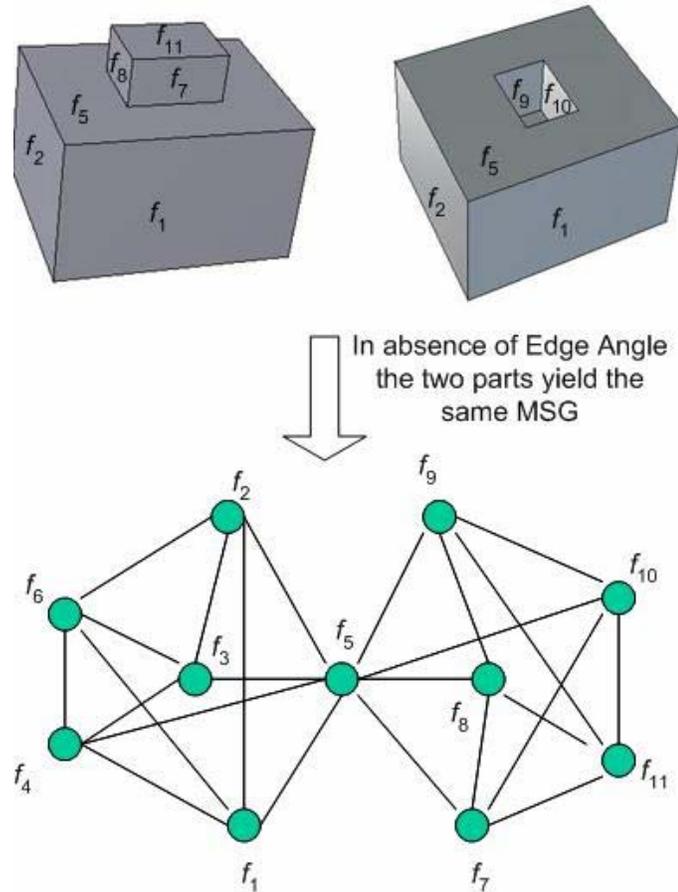


Figure 2.3: An Example of Same Model Signature Graph for Two Different Parts

Another technique proposed for comparing the graphs is the use of graph invariance vectors [McWh01b, McWh01c]. Graph invariance vectors are vectors whose elements are graph invariants. The vectors are then compared using L_2 norm to determine similarity between the graphs and hence the solid models. The graph invariants that form the graph invariance vectors include node and edge count, minimum and maximum degree of the nodes, median and mode degree of the nodes, and diameter of the graph. The use of graph invariance vectors improves the efficiency of the method. However it decreases the accuracy of comparison.

In [ElMe03a] a graph representation similar to Model Signature Graphs is obtained. The attributed graph used has the same topological information as the MSG. Additional information on the mathematical formulation, type and orientation of faces and edges is attached to the nodes and the edges of the attributed graph. The graph is obtained from the STEP representation of the 3D parts through an algorithm described in [ElMe03a]. The STEP representation contains information about surface and edge equations that is directly transferred to the corresponding attributed graph. The attributed graphs are then used to assess similarity between the corresponding 3D parts [ElMe03b]. A coarse indexing and comparison of graphs is performed based on the number of nodes of the graphs and their attributes such as corresponding surface type or number of incident edges. A more precise comparison is performed by using graph comparison. In this case it is necessary to trade-off between efficiency and accuracy. Graph comparison is computationally expensive. Disregarding some of the information stored in the graph improves the efficiency of graph comparison, but decreases the accuracy. Hence both an exact algorithm and an inexact, but more efficient algorithm are given in [ElMe03b].

These techniques have been applied to mechanical parts and are applicable to product design and manufacturing domain.

2.6.2 Multiresolutional Reeb Graphs

In this technique, the skeletal and topological structure of the 3D model is defined by Multiresolutional Reeb Graphs [Reeb46], which are used to compare the 3D objects. Reeb graphs have already been used in applications such as modeling 3D shapes [Laza99], [Taka97]. First, the Reeb Graph is defined on the input object, which is a triangulated solid. It is obtained by defining a suitable function over the 3D object

considered. An example of a suitable function is geodesic curvature. In general, the similarity function can be chosen depending on the particular topological properties selected. Then the function value range over the object is split into a number of sub ranges. This number is chosen depending on the desired level of resolution. A part of the object will correspond to each sub range. This part will be made of several connected regions. Every connected region will correspond to a node of the Reeb graph, and the adjacent nodes will be connected by edges. The corresponding nodes and connecting edges are also shown. The Reeb graph for the two models is created in $O(V \log(V))$ time, where V is the number of vertices in the mesh of the solid. Now the two corresponding graphs need to be matched. Corresponding nodes are matched in such a way so as to maximize a similarity function. In fact, the function is chosen such that the similarity between the two objects increases with its value. Thus, the best possible matching among the pair of nodes of the two graphs will maximize the value of the similarity function. Once this best matching is found, the value of similarity function between the two objects will yield the degree of similarity. At this stage the similarity function values corresponding to the best matching found are computed for every matched pair of nodes. They are then summed over the two objects, yielding a similarity value for the two objects being compared. Higher the value, more similar are the objects. Self-comparison of an object yields a value of 1, which is the maximum possible value. It takes $O(M(N+M))$ to match and assess similarity, where N and M are the number of nodes in the two graphs with $M < N$. So, with the increase in the accuracy of mesh and in the resolution of the Reeb graph, the efficiency decreases. Furthermore, if the function used to define Reeb graph is based on geodesic distance it is not very robust with respect to

small deformations on the surface. It is necessary to choose both a robust and efficiently computable function, which is not a trivial task. Finally, from the experimental results reported in [Hila01] it can be observed that this method is not invariant to Euclidean transformations (e.g., rotation, translation, scaling). Thus a given model when compared with its scaled, translated or rotated version will not yield a similarity value of 1.

In [Besp03a] the Multiresolutional Reeb Graphs defined in [Hila01] are used to assess similarity between 3D models of increasing geometric complexity. The experimental results show that the similarity assessment is insensitive to topology with the increase in geometric complexity of the parts compared. Hence an open issue is to define the function used to define Reeb Graphs in such a way that the similarity assessment is more sensitive to topology changes in the 3D parts.

This technique has been applied to product design and manufacturing domain. However, the choice of the function used to construct the Reeb graph obviously affects the resulting graph. In fact in [Bias05] the Reeb Graph is built by using two different functions over the 3D object: geodesic curvature and distance from the center of mass of the object. Then shape similarity is assessed by using an error tolerant algorithm for graph isomorphism. The experimental results presented in [Bias05] show that the performance of the two Reeb graphs defined previously by using two distinct distance functions is different. Hence Reeb graph is a flexible tool that can be used to assess similarity in several applications of product design and manufacturing, by choosing an appropriate function.

A technique that uses some of the tools and concepts developed in the previous two approaches is described in [Besp03b]. It uses space-scale decomposition to extract the

features from a 3D mechanical part model in VRML format. The geodesic distance is used as a distance function between the points of the 3D model, and the matrix of the distances is built for all the points of the 3D model. Then a singular value decomposition (SVD) of the matrix is performed, and using it k sets of points are extracted from the 3D model. Each set of points is a feature of the 3D model [Besp03b]. In [Besp03b] it is $k = 2$. The decomposition algorithm is applied recursively splitting each obtained feature into 2 features until the desired resolution is reached. This way a binary tree is obtained, and a recursive algorithm is used to match the binary trees of the two 3D models being compared. The algorithm finds the best match between nodes of the trees that are at the same level. From the perspective of the application to design and manufacturing domain, the feature extraction proposed in [Besp03b] does not necessarily obtain machining features of the part.

2.6.3 Graphs of Aligned Models

In [Sun95] a similarity assessment technique has been described based on the information provided by B-rep model and CSG tree termed as T0 tree. T0 tree is a specialized linear tree whose primitives are all sweeps obtained by sweeping a face in space along a profile. Initially, in the preprocessing stage, the T0 tree is used to determine the major sweep directions. Each of the sweep directions is expressed as a double (v_1, v_2) , where v_1 is the normal vector of a set of parallel faces and v_2 is the vector indicating the direction by which these parallel faces are organized in space. The set of parallel faces having normal vector v_1 is called layer faces F . Initially, for each pair of matched major sweep directions, the layer faces are matched using their normalized areas and their offsets along v_2 . If p_1 is a point on plane P_1 and p_2 a point on P_2 such that $p_2 = p_1 + dv$, where d is

a real number and v is a unit vector, then d is called offset from P_1 to P_2 along v . Once the layer faces of major sweep directions are matched such that there exists a one-to-one mapping, the objects are rotated so that the unit vectors v_2 match. Additional pair of faces, which do not have normal along v_2 are matched by attributed string matching algorithm [Tsai85], to completely align the two models. After rotating the layer faces to the correct orientation, initial matched sub-graphs of the layer faces are obtained. The nodes in the graph represent faces while the edges represent the intersection between those two faces. Once the layer faces are matched faces adjacent to matched layer faces are analyzed. If they match then they are included in the matched sub-graphs by expanding the sub-graphs. All possible matching sub-graphs are generated for all the major sweep directions and the B-rep matching coefficient is computed. This technique has the following restrictions on the models it can compare. All the solid models should have at least one or more major sweep directions. Also the models must be polyhedral.

This technique has been applied to product design and manufacturing on models that comply with the restrictions mentioned previously.

2.6.4 Skeletal Graphs

In [Sund03] a technique is described that uses skeletal graphs of the 3D models to assess their similarity. The 3D models are first voxelized with a certain resolution. Then, using a distance function described in detail in [Sund03], a skeleton is obtained that represents the structure of the model. The thickness of the skeleton and its level of detail can be modified as needed. The skeleton obtained consists of segments of the desired thickness and of joints. Using a minimum spanning tree (MST) based algorithm an undirected graph is obtained. Finally the graph is directed by orienting the edges. Each node

corresponds to a segment in the skeleton of the 3D model, and carries information on the local shape of the 3D model. On the other hand each edge corresponds to a joint of the skeleton and carries information on the flexibility of the 3D model. In order to match the obtained skeletal graphs efficiently the larger isomorphic subgraph problem is not solved. Instead the nodes are matched using the eigenvalue information stored at each node and obtained from the adjacency matrix of the graph. Hence a one-to-one mapping among the nodes of the two skeletal graphs is created. The outcome of the match is not guaranteed to comply with the hierarchical structure of the skeletal graphs that are being matched. In order to achieve it, a depth-first search algorithm is used. It is necessary to design algorithms that improve the matching process with different levels of resolutions in the voxelization and in the skeletal graph extraction [Sund03]. Furthermore, machining features are not guaranteed to be accurately detected by a skeletal graph.

2.7 Shape Statistics

The technique described in [Rea01], uses global shape metrics such as surface area/volume ratio, number of holes, compactness, and crinkliness to perform similarity assessment. These metrics are orientation independent and are extracted from a STL file. Compactness is the non-dimensional ratio of the square of the volume over the cube of the surface area while crinkliness is the surface area of the model divided by surface area of a sphere having the same volume. They are calculated for all the solid models and are stored as searchable entries in a database. To analyze the performance of the search engine, similarity matrices based on human perception of similarity have been generated. In [Sung02] and [Corn03] new filters for shape matching have been proposed. These are based on the coefficient of surface area and convex hull of the solid model. The convex

hull based filters include hull crumpliness, hull packing and hull compactness. Hull crumpliness is the ratio of surface area of object to surface area of its convex hull. Hull packing is the percentage of the convex hull volume not occupied by the original object.

The filters proposed in this technique are useful for pruning out parts from a large database. They do not have a high discrimination power. These filtering techniques have been applied to large databases of mechanical parts.

In [Iyer03] and [Lou04] the 3D models are voxelized with different resolutions (i.e. voxel sizes) depending on the desired approximation level. The geometric characteristics considered are moment invariants, geometric parameters and principal moments. The moment invariants are derived from the second order moments. Their analytical expression can be found in [Iyer03]. In order to calculate the moment invariants, the 3D model needs to be translated so that its centroid corresponds to the origin of the coordinate system. Because of the described translation the moment invariants are invariant with respect to translation, scaling and rotation. The geometric parameters are the ratio of overall surface area to normalized volume of the 3D models, the factor used to normalize the volume and the overall volume of the 3D models. The principal moments reflect the distribution of the models in the coordinate system. The principal moments can be very sensitive to noise if they are calculated taking into account higher order moments. Hence in [Iyer03] and [Lou04] only the second order moments are considered. In order to take into account topologic characteristics of the 3D models as well, a thinning algorithm is used to obtain the skeleton of the voxel model. The thinning algorithm preserves topology but not geometry in general. Then the skeletal graph is obtained, and the eigenvalues of its adjacency matrix are extracted. So finally each 3D

model is represented by a vector whose components are moment invariants, geometric parameters, principal moments and eigenvalues. The performance of this technique in shape similarity assessment can be improved. The geometric characteristics have a good discrimination capability. However the skeletal graph eigenvalues do not show a good discrimination capability and more information is needed. Furthermore, when applying this technique to machined parts, there is not direct relationship between machining features and skeletal graphs.

The technique introduced in [Ohbu02] uses a combination of three vectors to characterize a polygonal-mesh model. The first vector contains the moments of inertia of the model surfaces around its principal axes, the second vector contains the average distances of the model surfaces from its principal axes and the third vector contains the standard deviation of the average distances of the model surfaces from its principal axes. The similarity between models is assessed by computing Euclidean distance between the corresponding vectors. In some cases an elastic-matching distance is used instead in order to give a less rigid similarity measure than Euclidean distance [Ohbu02].

The technique introduced in [Anke99] is based on the partition of the space into regions. Each region (i.e. circular sector, shell) contains a certain fraction of the volume of the 3D model. Some regions may not contain any volume of the 3D model. With the help of these regions of space a histogram is built. The histogram measures the fraction of volume contained in each of the space regions considered. The distance between shape distributions is calculated using a quadratic distance function. As observed in [Anke99] the performance of this technique is affected both by the number of space regions and their geometric form. Increasing the number of space regions decreases the efficiency of

the technique while improving the discrimination capability. Also, the histograms obtained in the technique do not carry any specific information on the features of the 3D model. Hence this technique cannot be used for machined parts similarity assessment, but as a quick filter.

2.8 Point Pattern Alignment

As explained in Chapter 1, feature-based similarity assessment involves alignment of sets of feature vectors. This problem is directly related to point pattern alignment problems. A large number of papers have been written on the point pattern-alignment problem in the field of computer vision, pattern recognition, and computational geometry [Alt96], [Hutt90a]. Some of the formulations focus on exact alignments [Atki87], [Alt88], [Spri94]. However, in an attempt to circumvent the high complexity of point pattern matching, a number of approximation algorithms have been proposed.

Some of the approximate alignment techniques proposed perform 1-1 alignments [Alt88], [Heff94], in case the two sets being aligned have the same number of points. A hybrid approach combining branch-and-bound search of the transformation space with point-to-point alignments was proposed by Mount et al. [Moun99] in the context of image registration. Experimental studies have shown these methods to be quite efficient and accurate [Gavr99]. In case of different cardinalities of the two sets some of the approximate alignment techniques use the assumption that every point in one set has a close match in the other set in terms of the (standard) Hausdorff distance [Chew99], [Hutt92], [Hutt93c]. Efficient constant factor approximation algorithms have been proposed in [Good94] and [Indy99]. In the latter case the running times are sensitive to the ratio between the farthest and the closest points in the set. The fundamental

combinatorial issue is bounding the number of possible aligning elements. Another interesting approach is based on using the speed of a graphics coprocessor to accelerate the search [Agar03a]. These techniques are not suitable for object similarity applications where models may fail to share some features in common. Even under these relatively restrictive assumptions, the computational complexity can be quite high. Generalizations of these techniques to match more complex shapes such as segments, disks and balls have been proposed in [Chew97], [Agar94] and [Agar03b].

Robust similarity measures have been introduced to account for the fact that models may fail to share some common features. The best-known approach is based on the *partial Hausdorff distance* [Hutt93a], [Hutt93b], which allows some fraction of the points to be unmatched by minimizing the k th largest distance rather than the maximum distance. Another approach is the symmetric difference measure, which is based on the number of common features between the two sets [Velt01].

An important class of alignment methods for searching in large object databases is geometric hashing [Lamd88a], [Lamd88b], [Lamd88c], [Wolf97]. Geometric hashing was originally proposed as an approach to geometric object recognition. A small number of points are chosen from the object, which together define a local coordinate frame. The remaining points are then stored in a hash table according to their relation to this local frame, where each hash entry stores the index of the object and the defining frame. This is repeated for all object and all frames. In order to search for a given query object, a frame is selected from the query object, and its points are then hashed relative to this frame. The resulting entries of the hash table then “vote” as to the most likely choice of the frame and object that would give rise to this combination of hashes. Transformation

invariance is therefore achieved by storing points relative to a local frame. Geometric hashing has been successfully used in a wide variety of applications, and has been shown to quite efficient in some of them [Iran96].

2.9 Observations

Based on the literature survey given in this chapter, the following observations can be made.

- Shape signatures are abstractions of parts that capture only the 3D shape characteristics that are considered relevant. In manufacturing applications shape features rather than gross shape determine similarity between parts. However existing feature-based similarity assessment approaches do not consider feature relative positions and orientations. Therefore they may not be able to account for feature interactions that are dependent on these attributes. Furthermore they do not account for multiple feature interpretations.
- As mentioned before, GT coding schemes have been used primarily for classification and retrieval of mechanical parts. Although the GT approach has been used with some success in past, it has several limitations. Describing designs as short strings creates a coarse classification scheme, which limits the kinds of real-world retrieval problems for which the approach can be useful. Moreover, these techniques were developed prior to the advent of inexpensive computer technology; hence, they are not rigorously defined and are intended for human, not machine interpretation. This can cause difficulty in automating the generation of GT codes.
- Many previous approaches have favored symmetric distance measures. However, distance measures that are not symmetric in nature are of interest as well because of

the following reasons. Let A and B be two objects. Let A contain subset of features of B . In this case, B can be used to estimate cost of A by simply deleting extra processing steps (i.e., steps corresponding to features that are not present in A) from B . So distance of A from B should be small. On the other hand A cannot be used to estimate cost for B . So distance of B from A should be very large. Therefore use of asymmetric distance measures should be explored.

- The choice of the distance function depends on the field of application for shape similarity assessment. Consider for instance cost estimation of machined parts. Two machined parts may have in general different number and types of features. Hence it is critical to choose a distance function that can be applied to two sets of features of different cardinality and types.

Chapter 3: Optimal Attributed Point Alignment Algorithms Based On Partitioning Of Transformation Spaces

This chapter is organized as follows. Section 3.1 describes the motivation behind the research work described in this chapter. Section 3.2 presents a result that will be the foundation for the algorithms described in this chapter. In Section 3.3 the problem formulation is given. In Section 3.4 an optimal alignment algorithm in \mathbb{R}^2 under 2 DOF translations is presented. In Section 3.5 an optimal alignment algorithm in \mathbb{R}^2 under 1 DOF rotations is described. In Section 3.6 an optimal alignment algorithm in \mathbb{R}^3 under 3 DOF translations is presented. In Section 3.7 the complexity of the algorithms described is evaluated. Then Section 3.8 summarizes the chapter.

3.1 Motivation

For feature-based shape similarity assessment 3D parts are represented by sets of feature vectors. A distance function that is evaluated between the two sets of feature vectors yields the similarity degree between the two parts being compared. In general the parts to be compared are represented in different coordinate systems. Therefore in order to assess similarity between two parts it is necessary to align the two parts such that the distance between the two corresponding sets of feature vectors is minimized. Figure 3.1 shows an example of aligning two parts that are initially represented in two different coordinate systems. We will refer to the aligning transformation that minimizes the distance between two sets of features as *optimal feature alignment*. Part features can be represented as attributed points in the space. Part features can be represented as attributed points or vectors in the space. Attributed points are points that also carry parameters other than

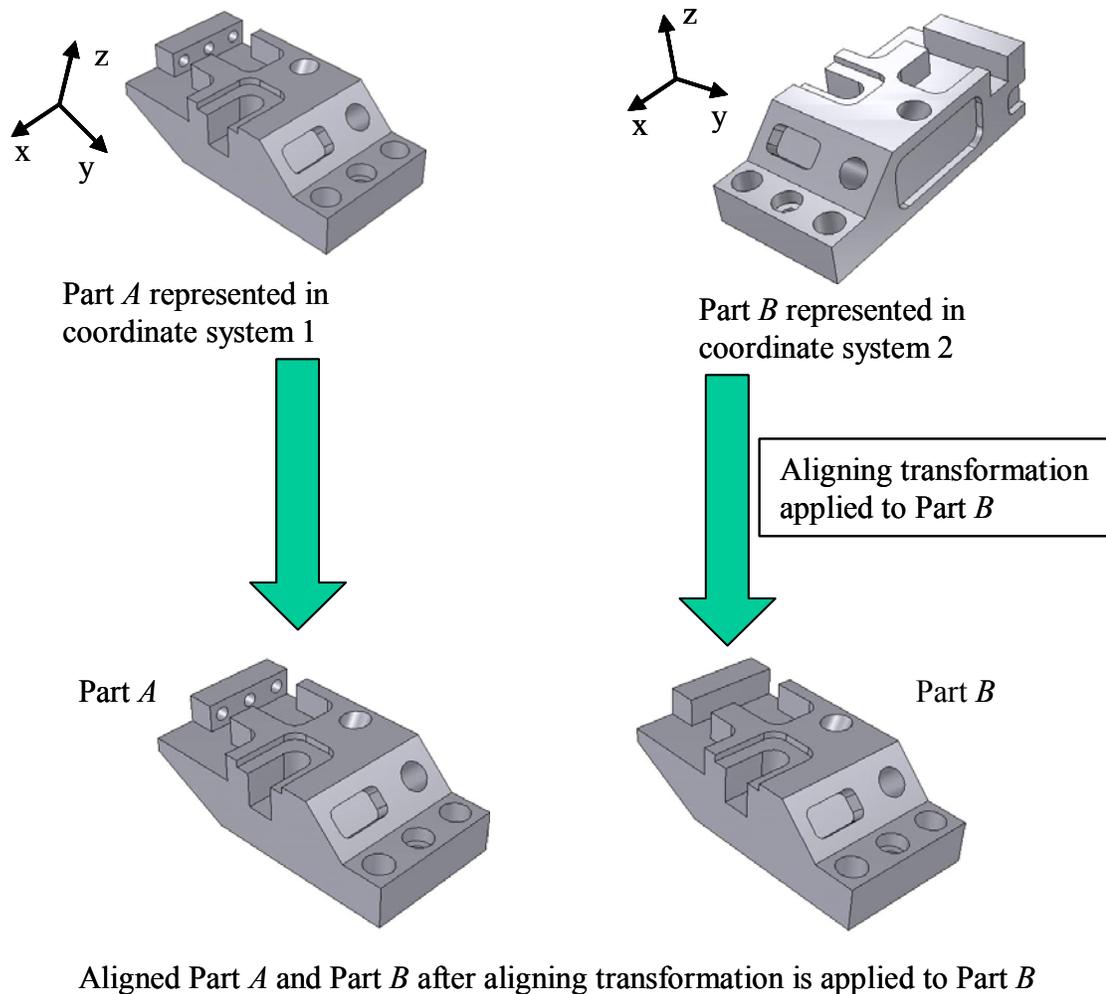


Figure 3.1: An Example of Aligning Two Parts Represented In Two Different Coordinate Systems

their coordinates. Point coordinates represent the feature position in the space, while other parameters represent the other significant feature characteristics of interest. Therefore attributed point optimal alignment problems will be addressed in this chapter.

Most of the distance functions that are used to compare sets of points involve computing the closest neighbor to each point of one set among the points of the other set. The distance function that will be used to compare two sets of attributed points in this

chapter is defined as follows. In general an attributed point will have some components that change with the transformation and some components that remain invariant with the transformation. We will refer to the former components as *transformation-dependent* attributes and to the latter components as *transformation-invariant* attributes. Consider an attributed point p in \mathbb{R}^3 that is represented by using four components. The first three components are the transformation-dependent coordinates x^p , y^p and z^p of point p . The fourth component w^p represents the transformation-invariant attribute that is assigned to point p . For the sake of simplicity each point carries a transformation-invariant attribute. The transformation-invariant attribute can be seen as the combination of any number of transformation-invariant attributes without affecting the generality of the problem. Let P and Q be sets of attributed points in \mathbb{R}^3 . Then, P and Q are compared using the following distance function.

$$\bar{d}(P, Q) = \frac{\sum_{i=1}^n \min_{q \in Q} d(p_i, q)}{n} \quad (3.1)$$

Depending on the form of distance function chosen, properties such as positivity, identity, symmetry and triangle inequality may or may not be satisfied. The form of the distance function defined in Equation (3.1) is such that positivity and identity properties are satisfied. It is asymmetric because this property is often desirable in manufacturing applications, as observed in Chapter 2. Also, the distance function consists of a summation of quadratic terms and hence it does not satisfy triangle inequality. However this particular form of distance function is easy to differentiate, which is a highly desirable property. Furthermore it can be observed that the distance function consists of the summation of single attributed point distances. Thus, the attributed points belonging

to the two sets that are aligned by minimizing the distance function will be distributed in a similar way in the space.

In order to minimize the distance function described in Equation (3.1) it is necessary to know the closest neighbor to each attributed point $p_i \in P$ among the attributed points of set Q . The closest neighbor to each point depends on the relative position of the two sets. Hence the closest neighbors change for specific values of the aligning transformation applied to one of the two sets. Therefore in order to find the optimal aligning transformation that minimizes the distance function it is necessary to know how the closest neighbors change with the aligning transformation.

The number of DOFs that is involved in the optimal alignment problem depends on the dimension of the points and on the aligning transformations used, and it is referred to as *dimension of the optimal alignment problem*. In this chapter optimal alignment algorithms are designed by using the distance function defined by Equation (3.1).

3.2 Mathematical Foundations

As previously explained in order to minimize the distance function defined in Equation (3.1) it is necessary to know the closest neighbors for each aligning transformation applied. Unfortunately, closest neighbors change throughout the transformation space and hence it is difficult to apply classical optimization techniques to compute the transformation that leads to the minimum distance. Given two sets of features, there are exponentially many closest neighbor combinations. Therefore solving this problem by enumeration is not likely to work. In practice, it turns out that actually a significantly lower number of combinations are geometrically feasible. Hence the transformation space needs to be partitioned into regions or intervals such that within each region the

closest neighbors do not change. Within each region the problem is solved by using the analytical techniques and then the minimum over all the regions is found. The following theorem provides a basis for a spatial partitioning approach to work.

Theorem 1: *Given a partitioning of the transformation space \mathbf{T} into regions such that the closest neighbors are invariant in each region, the transformation \mathbf{T}_{min} corresponding to the minimum value \bar{d}_{min} of the distance function over all the regions is guaranteed to lie within the region c^* whose corresponding closest neighbors have been used to compute it.*

Proof. Suppose by way of contradiction that the transformation \mathbf{T}_{min} does not lie within region c^* , but it lies within region c' . In that case compute the distance function in correspondence of the transformation \mathbf{T}_{min} , but this time using the closest neighbors corresponding to region c' . Let us denote the corresponding distance value by \bar{d}'_{min} . Observe that, by definition of closest neighbors, for each translation belonging to region c' to compute the distance by using the closest neighbors corresponding to region c' is guaranteed to yield a distance value smaller than by using the closest neighbors corresponding to any other region. Hence as the transformation \mathbf{T}_{min} lies within region c' we are guaranteed that $\bar{d}'_{min} < \bar{d}_{min}$. This leads to a contradiction, as the initial hypothesis was that \bar{d}_{min} is the minimum value of the distance function over all the regions. Hence the transformation \mathbf{T}_{min} is guaranteed to lie within region c^* , which proves Theorem 1.

Theorem 1 ensures that the transformation that minimizes the distance function lies within the region whose closest neighbors have been used to minimize the distance function. Please note that the phrase “within the region” means that the point lies either in

the interior or on the boundary of the region. This result is the basis of the optimal alignment algorithms described in this chapter.

3.3 Problem Formulation

Consider the distance function defined in Equation (3.1). The distance function between points $p \in P$ and $q \in Q$ is defined as follows.

$$d(p, q) = (x^p - x^q)^2 + (y^p - y^q)^2 + (z^p - z^q)^2 + (w^p - w^q)^2 \quad (3.2)$$

Consider a rigid body transformation applied to the set P in \mathbb{R}^3 . The most general rigid body transformation applied to a set of points in \mathbb{R}^3 involves six DOFs. Given a Cartesian coordinate system, six DOFs are represented by the three components of a translation Δx , Δy and Δz along the three coordinate axis and the three rotations $\Delta \theta$, $\Delta \phi$ and $\Delta \psi$ about the three coordinate axis. Hence the corresponding transformation matrix \mathbf{T} will be function of the six DOFs involved. The distance function defined in Equation (3.1) can then be written as:

$$\bar{d}(\mathbf{TP}, Q) = \frac{\sum_{i=1}^n \min_{q \in Q} d(\mathbf{Tp}_i, q)}{n} \quad (3.3)$$

Imagine applying a transformation \mathbf{T} to set P . The distance function between sets P and Q can be evaluated, for every possible transformation \mathbf{T} , by using Equation (3.3). We refer to the problem of finding the transformation \mathbf{T} applied to set P that minimizes the distance function defined in Equation (3.3) between attributed point sets P and Q as *attributed point alignment under the transformation \mathbf{T}* . The definitions and notations introduced in this section will be modified in order to refer to attributed points in \mathbb{R}^2 . In \mathbb{R}^2 the most general rigid body transformation \mathbf{T} will involve three DOFs, that is the two

components of translation Δx and Δy and the rotation $\Delta \theta$ about the origin. In this section the algorithms for the following three attributed point set optimal alignment problems are presented: (1) optimal alignment under 2 DOF translations in \mathbb{R}^2 , (2) optimal alignment under 1 DOF rotations in \mathbb{R}^2 and (3) optimal alignment under 3 DOF translations in \mathbb{R}^3 .

The most general alignment problem in \mathbb{R}^3 involves a 6 DOF transformation and hence its dimension is six. As the three alignment problems solved in this chapter have lower dimension we refer to them as lower dimension alignment problems. Their solution will be the basis to solve higher dimension alignment problems.

The alignment algorithm presented in this chapter can be used in many different applications, as the points carry a transformation-invariant attribute that can be obtained by combining any number of transformation-invariant attributes for a given feature. The distance function defined in Equations (3.1), (3.2) and (3.3) is a very general one. It is not sensitive to outliers like Hausdorff distance, and its mathematical form has been chosen so that mathematical operations such as differentiation can be easily performed.

3.4 Optimal Alignment Under 2 DOF Translations In \mathbb{R}^2

The algorithm TWODOFALIGNMENT finds the translation $(\Delta x, \Delta y)$ that minimizes the distance function given by Equation (3.1). Given a Cartesian coordinate system, the transformation space in this case is represented by the two components of the translation $(\Delta x, \Delta y)$ in the coordinate plane XY. The general Equation (3.1) can be specified for two sets of attributed points in \mathbb{R}^2 and for the two degrees of freedom translation $(\Delta x, \Delta y)$. The overall algorithm that solves the two-degree of freedom problem is given below.

Input:

- Sets P and Q of attributed points in \mathbb{R}^2 .

Output:

- Translation $(\Delta x_{\min}, \Delta y_{\min})$ that minimizes the distance function defined in Equation (3.1).

Steps:

- a. Partition the transformation space into regions such that the closest neighbor $q_j \in Q$ to each attributed point $p_i \in P$ is invariant in each region using the algorithm FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL.
- b. Within each region c obtained from Step a compute the value of the translation $(\Delta x(c), \Delta y(c))$ that minimizes the distance function defined in Equation (3.1) for region c .
- c. Find region c^* such that the distance function defined in Equation (3.1) reaches the minimum value over all the regions obtained in Step a.
- d. Return the corresponding value $(\Delta x_{\min}, \Delta y_{\min}) = (\Delta x(c^*), \Delta y(c^*))$ of the translation for the region c^* found in Step c.

In the Subsection 3.4.1, 3.4.2 and 3.4.3 the steps of the previously described algorithm and the algorithm FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL will be described.

3.4.1 Step a: Building The Set Of Regions For The Attributed Points Of Set P

To compute the distance value in Equation (3.1), the closest neighbor $q_j \in Q$ to each $p_i \in P$ needs to be determined. The closest neighbor $q_j \in Q$ to each $p_i \in P$ changes with the

translation of set P with respect to set Q . Thus, the closest neighbor to each $p_i \in P$ needs to be obtained by taking into account the translation $(\Delta x, \Delta y)$. As anticipated before, the transformation space in this case is the plane representing each possible translation being applied to the points of set P . It is necessary to know for each value of the translation $(\Delta x, \Delta y)$ the closest attributed point $q_j \in Q$ to each attributed point $p_i \in P$. The closest neighbor to each attributed point of P changes at specific values of $(\Delta x, \Delta y)$. Therefore the transformation space can be partitioned into a set of regions within which the closest neighbor to each attributed point of P is known and invariant. The following algorithm is used for this purpose.

Algorithm: FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL

Input:

- Sets P and Q of attributed points in \mathbb{R}^2 .

Output:

- Set of regions and for each region the closest neighbor to every attributed point of P from set Q .

Steps:

1. For each attributed point p_i of P , do the following.
 - a. For each possible pair of distinct attributed points q_k and q_l of Q do the following. Partition the transformation space into regions within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. The partitioning is performed by intersecting the transformation space on which p_i , q_k and q_l are located
-

with a line whose locus is such that $d(p_i, q_k) = d(p_i, q_l)$, where d is the distance function defined in Equation (3.2). This step will be described in more detail after the description of the overall algorithm.

- b. Overlap the intersecting regions obtained in Step 1.a so that the transformation space is further partitioned into a set of regions.
 - c. For each region obtained in Step 1.b, do the following. Using the closest neighbors obtained in Step 1.a, find the attributed point q_j of Q such that $d(p_i, q_j)$ is minimum over all the attributed points of Q .
 - d. Merge the adjacent regions that have the same closest neighbor into one single region.
2. Overlap the set of intersecting regions being obtained in Step 1 for each attributed point p_i of P . Within the set of intervals being obtained the closest neighbor to every attributed point of P from set Q is invariant and known.

The algorithm described above yields the set of regions for the attributed points of P .

In the next paragraphs Step 1.a and Step 2 will be explained in detail.

L and L' are lines whose points represent translations that bring point p_1 of set P at the same distance between points q_1 and q_2 of set Q

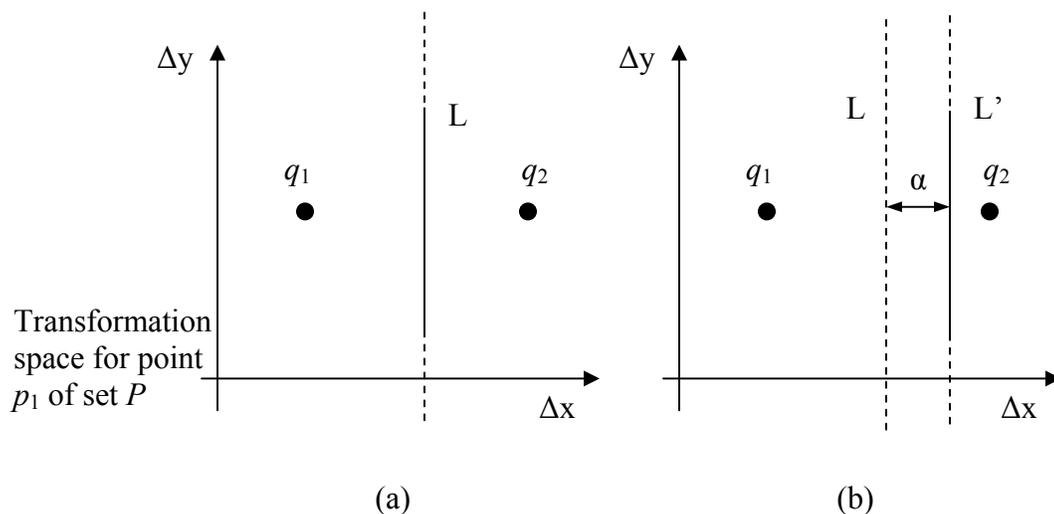


Figure 3.2: The Transformation Space of Point p_1 of Set P Is Partitioned Into Two Regions (a) Transformation-invariant Attributes Are the Same for Each Point (b) Transformation-invariant Attributes Are Different for Each Point, and Hence the Line L' Is Offset With Respect to the Line L

In Step 1.a, the closest neighbor to each attributed point $p_i \in P$ needs to be obtained by using the distance function defined in Equation (3.2). The distance function accounts for the transformation-invariant attribute. Hence the transformation-invariant attribute needs to be considered in obtaining the closest neighbors. First let us consider the case where the attributed points have identical transformation-invariant attributes. As shown in Figure 3.2(a), the two attributed points q_1 and q_2 of set Q are represented on the transformation space of attributed point p_1 of P . As their transformation-invariant attributes have the same value, the locus of points L of the transformation space whose distance defined in Equation (3.2) from point q_1 is the same as the distance from point q_2 is the line through the midpoint between q_1 and q_2 and perpendicular to the segment joining q_1 and q_2 . Now let us consider the case where the attributed points have different

transformation-invariant attributes. Let Δw_{11} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_1 of Q . Similarly let Δw_{12} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_2 of Q . Let $\Delta w_{11} > \Delta w_{12}$ and $\Delta w^2 = \Delta w_{11}^2 - \Delta w_{12}^2$. In this case it is necessary to locate the locus of points L' such that $d(p_1, q_1) = d(p_1, q_2)$ using the distance function defined in Equation (3.2). Because of the presence of transformation-invariant attributes, the locus of points L' will no longer be the line through the midpoint between q_1 and q_2 and perpendicular to the segment joining q_1 and q_2 . As shown in Figure 3.2(b), the line will be offset by α in the direction of the point having the higher value Δw_{ij} , in this case q_1 . The value of α is defined as follows.

$$\alpha = \frac{\Delta w^2}{2H} \quad (3.4)$$

where H is equal to the Euclidean distance between q_1 and q_2 . The value of the offset α depends on the value of Δw and H . In Appendix A the value of α defined in Equation (3.4) will be derived.

After Step 1.c the closest neighbor for each region of the transformation space of point p_i is known and invariant. However in general there might be adjacent regions of the transformation space whose correspondent closest neighbors are coincident. In those cases it is necessary to merge those regions into one by eliminating the common edges.

Observe that Step 1 of the algorithm FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL yields the closest neighbors for each attributed point of P separately. A set of regions is built for a particular attributed point $p_i \in P$ such that in each region the closest attributed point of Q to p_i is known. Thus several

sets of regions are obtained, one for each member of P . The overlapping of the sets of regions being performed in Step 2 yields the set of regions for P . Within each of the regions the distance given by Equation (3.1) can be minimized using closed form mathematical formulae. The only independent variables in the formulae are the components of the translation $(\Delta x, \Delta y)$. The single sets of regions for each attributed point of P are combined into the set of regions for the attributed points of P by overlapping so that the transformation space is further partitioned into regions.

Each of the resulting regions is obtained from the intersection of the regions of the initial sets of regions. Figure 3.3 shows two sets of regions that are overlapped. One set of regions is the set of regions of attributed point p_1 of set P (see Figure 3.3(a)), the other one is the set of regions of attributed point p_2 of set P (see Figure 3.3(b)). The region c , indicated in Figure 3.3(c) by an arrow point, is clearly contained in one of the regions of each of the two sets of regions that have been overlapped. As shown in Figure 3.3(a) and Figure 3.3(b), the regions c_1 and c_2 overlap to generate region c . Thus, region c represents a region in the set of regions for the attributed points of P . Within c , q_1 is the closest

neighbor to p_1 and q_2 is the closest neighbor to p_2 . Each point of c corresponds to a transformation applied to the set of attributed points P while Q is fixed. Thus, within any region of the set of regions for the attributed points of P , the closest attributed point of Q to each attributed point in P is known. The distance function defined in Equation (3.1) can now be computed for each region. The distance function defined in Equation (3.1) for each region can be expressed as a function of the coordinates (x, y) of the attributed points of P and Q . Coordinates of P and Q can be expressed as a function of $(\Delta x, \Delta y)$,

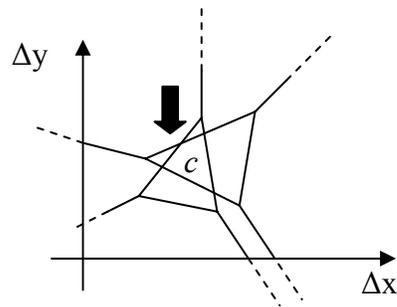
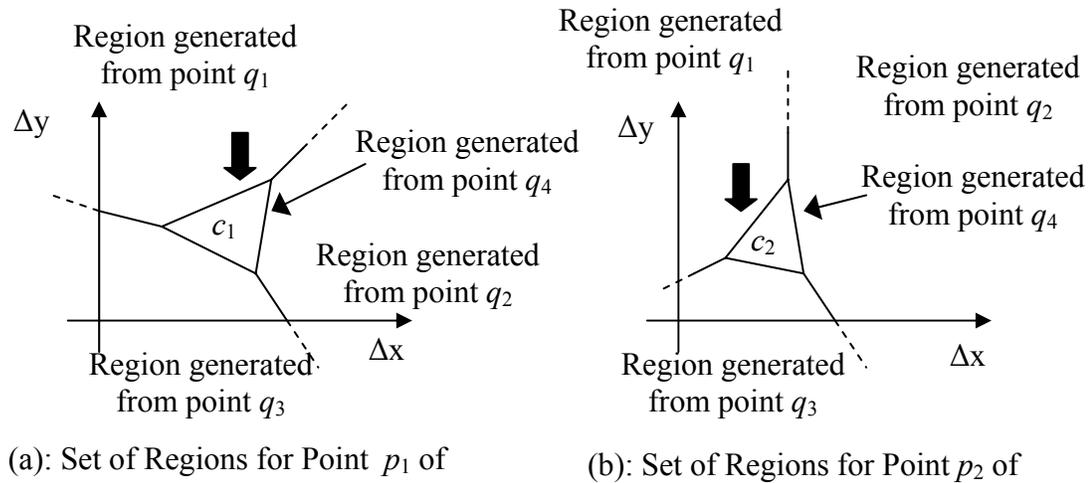


Figure 3.3: Example of Overlapping of Sets of Regions

which are the components of translation. Thus the distance function defined in Equation (3.1) is expressed as a function of $(\Delta x, \Delta y)$ as explained in the next subsection.

3.4.2 Step b: Minimization Of The Distance Function Within A Given Region

The location of an attributed point p in the planar transformation space can be represented by the coordinates (x^p, y^p) . Let $x_o^{p_i}$ and $y_o^{p_i}$ be the coordinates of the known initial position for attributed point $p_i \in P$.

In the previous subsection the set of regions for all the attributed points of P was built by overlapping the single sets of regions of each attributed point. The transformation

space is thus partitioned into a number of regions. Within each region the closest attributed point in Q to each of the attributed point in P is known. The following definitions, valid within each single region, will be used:

$$\begin{cases} x^{q_j}(i) = x \text{ coordinate of the closest attributed point } q_j(i) \in Q \text{ to } p_i \in P \\ y^{q_j}(i) = y \text{ coordinate of the closest attributed point } q_j(i) \in Q \text{ to } p_i \in P \\ w^{q_j}(i) = \text{transformation-invariant attribute of the closest point } q_j(i) \in Q \text{ to } p_i \in P \end{cases} \quad (3.5)$$

Consider a single region and an attributed point $p_i \in P$. Let $(\Delta x, \Delta y)$ be the translation applied to the attributed points of set P . Then,

$$\begin{cases} x^{p_i}(\Delta x) = x_o^{p_i} + \Delta x \\ y^{p_i}(\Delta y) = y_o^{p_i} + \Delta y \end{cases} \quad \forall \text{ point } p_i \in P \quad (3.6)$$

Within a single region, it is necessary to compute $\bar{d}(P, Q)$ as a function of the transformation $(\Delta x, \Delta y)$. The term accounting for Z coordinate in the distance function defined in Equation (3.2) is not considered in this case as the alignment problem involves attributed points in \mathbb{R}^2 .

$$\bar{d}(\mathbf{TP}, Q)(x^{p_i}(\Delta x), y^{p_i}(\Delta y)) = \frac{\sum_{i=1}^n \left\{ [x^{p_i}(\Delta x) - x^{q_j}(i)]^2 + [y^{p_i}(\Delta y) - y^{q_j}(i)]^2 + (w^{p_i} - w^{q_j}(i))^2 \right\}}{n} \quad (3.7)$$

Using the notations introduced in Equations (3.5) and (3.6), Equation (3.7) can be simplified to,

$$\bar{d}(\Delta x, \Delta y) = \frac{\sum_{i=1}^n \left\{ [x_o^{p_i} + \Delta x - x^{q_j}(i)]^2 + [y_o^{p_i} + \Delta y - y^{q_j}(i)]^2 + (w^{p_i} - w^{q_j}(i))^2 \right\}}{n} \quad (3.8)$$

In order to minimize $\vec{d}(\Delta x, \Delta y)$ its derivative with respect to Δx and Δy must be set to zero. By doing this and simplifying, we get the following expression for the translation components.

$$\begin{cases} \Delta x = \frac{\sum_{i=1}^n \{x^{q_i}(i) - x_o^{p_i}\}}{n} \\ \Delta y = \frac{\sum_{i=1}^n \{y^{q_i}(i) - y_o^{p_i}\}}{n} \end{cases} \quad (3.9)$$

Observe that the distance function defined in Equation (3.8) is a continuous function, and it is also bounded. The values of Δx and Δy resulting from Equations (3.9) identify a local minimum of the distance function if and only if the corresponding Hessian matrix is positive definite, that is its eigenvalues are positive. As $\frac{\partial^2 \vec{d}}{\partial(\Delta x)\partial(\Delta y)} = 0$ and

$\frac{\partial^2 \vec{d}}{\partial^2(\Delta x)} = \frac{\partial^2 \vec{d}}{\partial^2(\Delta y)} = 2$ the Hessian has two coincident positive eigenvalues whose value is

2. Hence the values of Δx and Δy resulting from Equations (3.9) identify a local minimum of the distance function.

Equations (3.9) yield the translation $(\Delta x, \Delta y)$, applied to the set of attributed points P , which minimizes the distance between the sets of attributed points P and Q . This value of the translation is valid only within a single region of the set of regions for all the attributed points of P . In general the value of $(\Delta x, \Delta y)$ that is found is not guaranteed to lie in the region where the distance function is defined. Values of $(\Delta x, \Delta y)$ that lie outside the corresponding region have no physical meaning and should be discarded. In fact, by referring to Theorem 1, values of the translation $(\Delta x, \Delta y)$ that lie outside the region whose closest neighbors have been used to compute them will not correspond to the global

minimum of the distance function. By not considering those regions the computation of the translation that minimizes the distance function over all the regions may be speeded up, as several regions will not be considered.

Equations (3.9) have been obtained by differentiating the distance function with respect to Δx and Δy , which is a standard minimization technique in the continuous domain. Thus, the translation value obtained for a region c of the set of regions for all the attributed points of P yields the best possible alignment between the two attributed point sets for all permissible translations within the region c .

3.4.3 Steps c And d: Computing The Translation That Minimizes The Distance Over All The Regions

The values of $\Delta x(c)$ and $\Delta y(c)$ obtained in the Equations (3.9) yield the translation that minimizes the distance between the two attributed point sets P and Q within a single region c of the set of regions for all the attributed points of P . To obtain the corresponding value of the distance $\vec{d}(c)$ it is sufficient to substitute the value of Δx and Δy obtained from Equations (3.9) into Equation (3.8). Hence, for each region, $\vec{d}(c)$ is the minimum distance. Finally Steps c and d of the algorithm TWODOFALIGNMENT involve finding the values of Δx and Δy corresponding to the minimum distance over all the regions. The minimum distance over all the regions is obtained as follows:

$$\vec{d}_{\min} = \min_{c \in C} \vec{d}(c) \quad (3.10)$$

where C is the set of all the regions c of the partitioned transformation space. The value given by the Equation (3.10) is the minimum distance between sets P and Q . The corresponding translation $(\Delta x_{\min}, \Delta y_{\min})$ is found as follows: let c^* be the region in which

the minimum distance was found (refer to Equation (3.10)). Then $(\Delta x_{min}, \Delta y_{min})$ is obtained as follows:

$$\begin{cases} \Delta x_{min} = \Delta x(c^*) \\ \Delta y_{min} = \Delta y(c^*) \end{cases} \quad (3.11)$$

The Equations (3.11) yield the translation to apply to P in order to minimize the distance between P and Q . Equation (3.10) provides the minimum distance between two sets of attributed points in \mathbb{R}^2 under two degree of freedom translation.

3.5 Optimal Alignment Under 1 DOF Rotations In \mathbb{R}^2

The second optimal alignment algorithm that is designed in this chapter is ONEDOFALIGNMENT. It finds the rotation θ that minimizes the distance function given by Equation (3.1) in \mathbb{R}^2 . The distance function defined in Equation (3.1) can be specified for two sets of attributed points in \mathbb{R}^2 and for the one degree of freedom rotation θ . From now on in this thesis the range of rotations $[0, 2\pi]$ will be referred to as *theta range*, and any interval contained in this range as *theta interval*. Given a Cartesian coordinate system, consider the rotation θ about a coordinate axis Z and the coordinate plane XY perpendicular to it. Each attributed point can only move along a circle lying on the coordinate plane XY . The initial position of the attributed point must belong to the circle. The center of the circle corresponds to the center of rotation being used. In this case the center of mass of the rotating set of points P computed without considering the transformation-invariant attributes will be used as center of rotation. Each point of the previously defined circle corresponds to one and only one value of the rotation θ about the coordinate axis Z . The transformation space in this case is represented by the closed

interval of real numbers representing all the possible rotations $\theta \in [0, 2\pi]$. The transformation space will be referred to as the theta range $[0, 2\pi]$. The overall algorithm that solves the one degree of freedom problem is given below.

Algorithm: ONEDOFALIGNMENT

Input:

- Sets P and Q of attributed points in \mathbb{R}^2 .

Output:

- Angle θ_{min} that minimizes the distance function defined in Equation (3.1).

Steps:

- a. Partition the theta range $[0, 2\pi]$ into theta intervals such that the closest neighbor $q_j \in Q$ to each attributed point $p_i \in P$ is invariant in each interval using the algorithm FINDINVARCLOSESTNEIGHBORSFOR1DOFRROT.
- b. Within each theta interval c obtained from Step a compute the value of the rotation $\theta(c)$ that minimizes the distance function defined in Equation (3.1) for interval c .
- c. Find interval c^* such that the distance function defined in Equation (3.1) reaches the minimum value over all the intervals obtained in Step a.
- d. Return the corresponding value $\theta_{min} = \theta(c^*)$ of the rotation for the interval c^* found in Step c.

In the Subsections 3.5.1, 3.5.2 and 3.5.3 the steps of the previously described algorithm and the algorithm FINDINVARCLOSESTNEIGHBORSFOR1DOFRROT will be described.

3.5.1 Step a: Building The Set Of Theta Intervals For The Attributed Points Of Set P

As in Section 3.4, in order to compute the distance value in Equation (3.1), the closest neighbor $q_j \in Q$ to each $p_i \in P$ needs to be determined. The closest neighbor $q_j \in Q$ to each $p_i \in P$ changes with the rotation of set P with respect to set Q . Thus, the closest neighbors for each $p_i \in P$ need to be obtained by taking into account the rotation θ around the fixed axis that has been defined in the previous subsection. It is necessary to know, for each value of the rotation θ , the closest attributed point $q_j \in Q$ to each attributed point $p_i \in P$. The closest neighbor to each attributed point of P changes only at specific values of θ . Thus, the theta range $[0, 2\pi]$ can be partitioned into a set of theta intervals within which the closest neighbor to each attributed point of P is known and invariant. The following algorithm is used for this purpose.

Algorithm: FINDINVARCLOSESTNEIGHBORSFOR1DOFROT

Input:

- Sets P and Q of attributed points.

Output:

- Set of theta intervals and for each interval the closest neighbor to every attributed point of P from set Q .

Steps:

1. For each attributed point p_i of P do the following.
 - a. For each possible pair of distinct attributed points q_k and q_l of Q do the following. Partition the theta range $[0, 2\pi]$ into intervals within which either
-

$d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. The partitioning is performed by intersecting the circle representing the trajectory of p_i with a line whose locus is such that $d(p_i, q_k) = d(p_i, q_l)$, where d is the distance function defined in Equation (3.2). This step will be described in more detail after the description of the overall algorithm.

- b. Overlap the intersecting subintervals obtained in Step 1.a so that the range $[0, 2\pi]$ is further partitioned into a set of intervals.
 - c. For each interval obtained in step 1.b do the following. Using the closest neighbors being obtained in Step 1.a, find the attributed point q_j of Q such that $d(p_i, q_j)$ is minimum over all the attributed points of Q .
 - d. Merge the adjacent intervals that have the same closest neighbor into one single interval.
2. Overlap the set of intersecting intervals obtained in Step 1 for each attributed point p_i of P . Within the set of intervals being obtained the closest neighbor to every attributed point of P from set Q is invariant and known.

The algorithm described above yields the set of theta intervals for the attributed points of P . In the next paragraphs Step 1.a and Step 2 will be explained in detail.

In Step 1.a, the closest neighbors for each attributed point $p_i \in P$ need to be obtained by using the distance function defined in Equation (3.2). The transformation-invariant attributes need to be considered. As we did in Subsection 3.4.1, let us consider a case where the attributed points have identical transformation-invariant attributes. As shown in Figure 3.4, the dotted circle C_1 centered on the rotation center represents the trajectory of p_1 of P as it is rotated. Consider two attributed points q_1 and q_2 of Q in \mathbb{R}^2 . In general

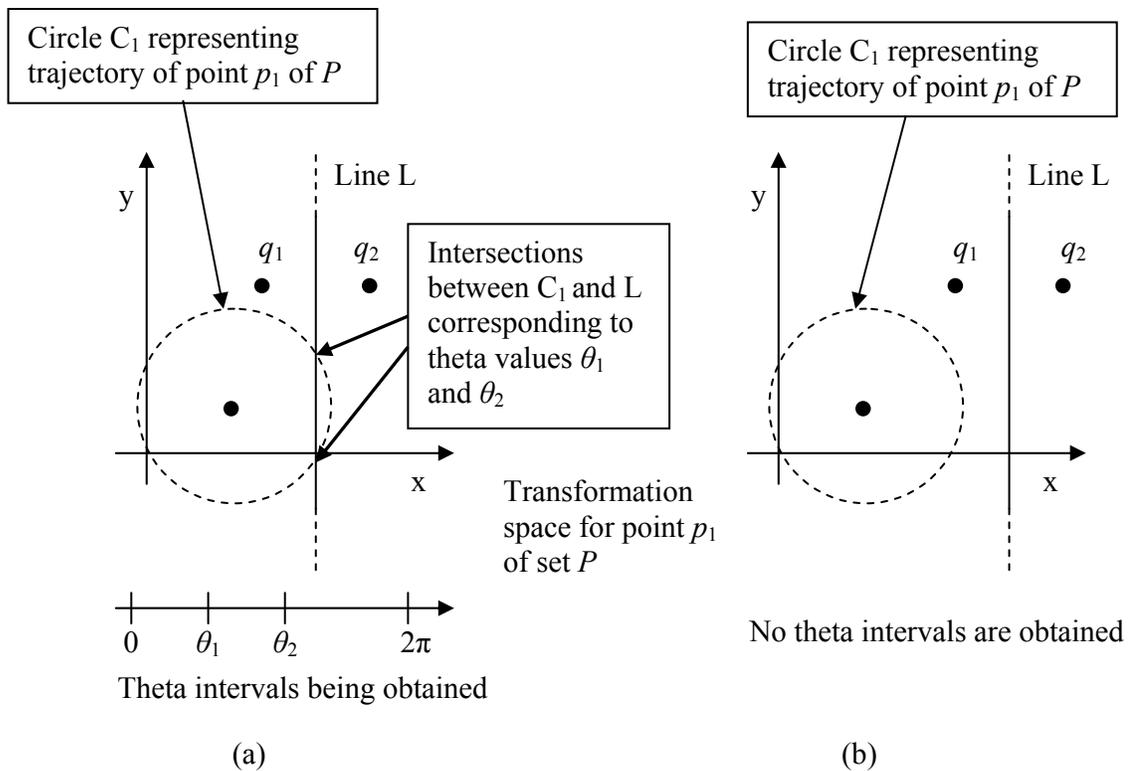


Figure 3.4: Set of Theta-intervals for Point p_1 of Set P : (a) Case of Intersection Between Line L and Circle C_1 (b) Case of Non-intersection Between Line L and Circle C_1

along a portion of the trajectory $d(p_1, q_1) < d(p_1, q_2)$ and along the remaining portion $d(p_1, q_1) > d(p_1, q_2)$. The procedure to obtain the theta intervals such that the closest neighbors are invariant is as follows. As the transformation-invariant attributes have the

same value, the locus of points L of the transformation space whose distance defined in Equation (3.2) from point q_1 is the same as the distance from point q_2 is the line through the midpoint between q_1 and q_2 and perpendicular to the segment joining q_1 and q_2 . In Figure 3.4(a) the line L and the circle C_1 are intersected, obtaining two points on the circle. Each point of the circle corresponds to a value of theta within the theta range $[0,2\pi]$. Hence the two intersection points correspond to the extreme values of the theta intervals being obtained, as shown in Figure 3.4(a). Within each interval either $d(p_1,q_1) < d(p_1,q_2)$ or $d(p_1,q_1) > d(p_1,q_2)$ and the closest neighbor to p_1 is known. In Figure 3.4(b) the line L and the circle C_1 do not intersect. That means that for the entire theta range $[0,2\pi]$ either $d(p_1,q_1) < d(p_1,q_2)$ or $d(p_1,q_1) > d(p_1,q_2)$: the closest neighbor does not change. Now let us consider the case where the attributed points have different transformation-invariant attributes. Let Δw_{11} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_1 of Q . Let Δw_{12} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_2 of Q . Let $\Delta w_{11} > \Delta w_{12}$ and $\Delta w^2 = \Delta w_{11}^2 - \Delta w_{12}^2$. In this case it is necessary to locate the locus of points L' such that $d(p_1,q_1) = d(p_1,q_2)$ using the distance function defined in Equation (3.2). As in Subsection 3.4.1 the locus of points L' will no longer be the line through the midpoint between q_1 and q_2 and perpendicular to the segment joining q_1 and q_2 . As shown in Figure 3.5, the line will be offset by α in the direction of the point having the higher value Δw_{ij} , in this case q_1 . The value of α is defined in Equation (3.4). Again, there are two possibilities: L' can either intersect the circle or not. The same conclusions can be drawn as in the case of points having the same transformation-invariant attributes.

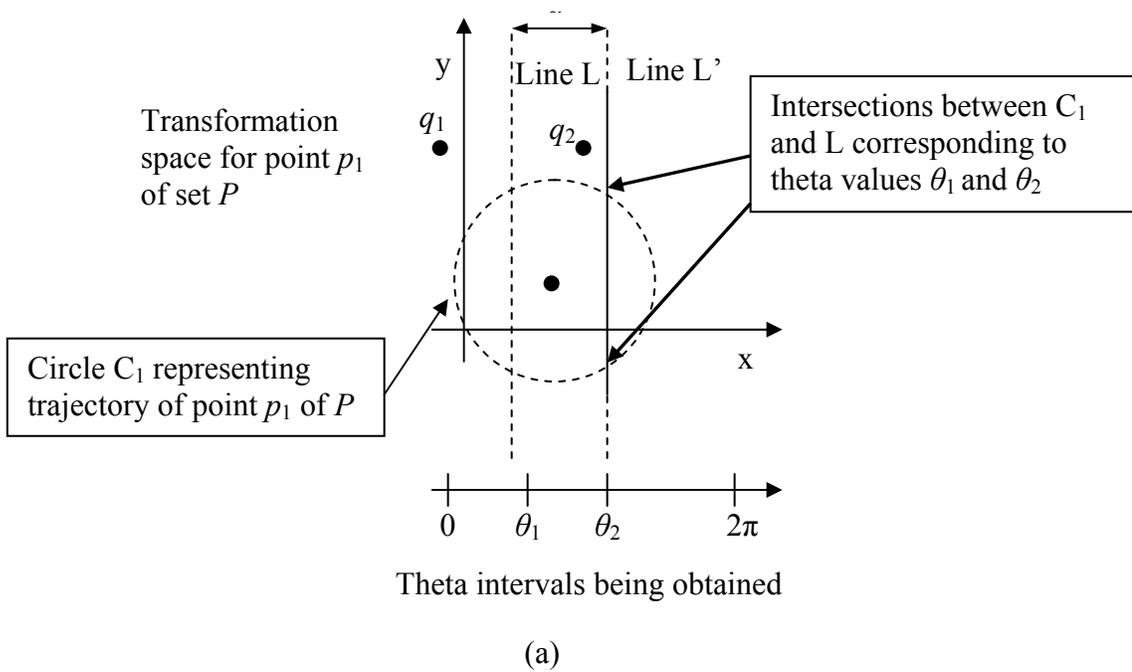


Figure 3.5: Set of Theta-intervals for Point p_1 of Set P When Transformation Invariant Attributes of Points q_1 and q_2 of Set Q Are Different

Step 2 of the algorithm `FINDINVARCLOSESTNEIGHBORSFOR1DOFROT` can be explained by using the same arguments as for Step 2 of the algorithm `FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL` presented in Subsection 3.4.1. For a particular attributed point $p_i \in P$ a set of theta intervals is available such that in each interval the closest attributed point of Q to p_i is known. The sets of theta intervals for each single point of P are overlapped. This yields the set of theta intervals for the attributed points of P . Figure 3.6 shows an example of two sets of intervals that are overlapped. One set of intervals is the set of theta intervals of attributed point p_1 of set P (see Figure 3.6(a)), the other one is the set of theta intervals of attributed point p_2 of set P (see Figure 3.6(b)). The interval c , indicated in Figure 3.6(c) by an arrow point, is clearly contained in one of the intervals of each of the two sets of theta intervals that have been

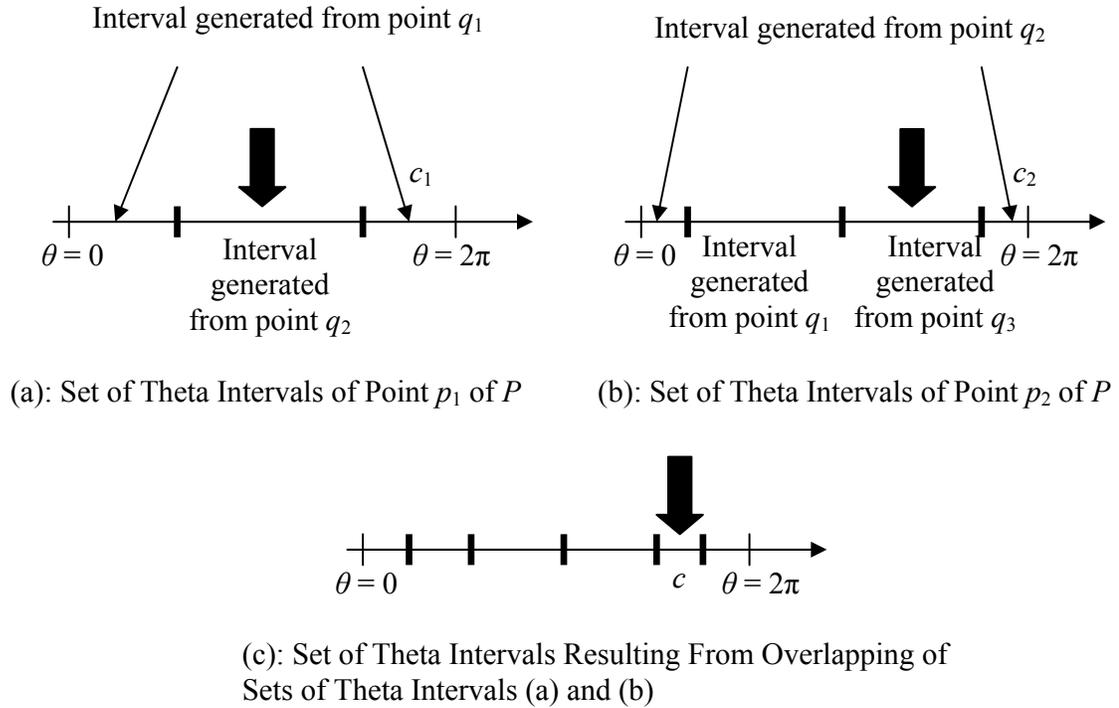


Figure 3.6: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals

overlapped. As shown in Figure 3.6(a) and Figure 3.6(b), the intervals c_1 and c_2 overlap to generate interval c . Thus, interval c represents a region in the set of theta intervals for the attributed points of P . Within c , q_1 is the closest neighbor to p_1 and q_2 is the closest neighbor to p_2 . Each point of c corresponds to a rotation applied to the set of attributed points P while Q is fixed. Within any obtained interval the closest attributed point of Q to each attributed point in P is known.

Within each of the obtained intervals the distance function defined in Equation (3.1) can now be computed and minimized using closed form mathematical formulae. It can be expressed as a function of the coordinates (x,y) of the attributed points of P and Q . Coordinates of P and Q can be expressed as functions of θ , which is the angle of rotation. Thus the distance function defined in Equation (3.1) is expressed as a function of θ as

explained in the next subsection. Rotation θ will be the only independent variable in the formulae.

3.5.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval

Some of the notations introduced in Subsection 3.4.2 hold for the algorithm being presented here. In particular, the location of an attributed point p in the planar transformation space can be represented by the coordinates (x^p, y^p) . Equations (3.5) hold as well. In this case it is necessary to define also the center of rotation (x_B, y_B) . The angle $\theta_o^{p_i}$ determines the initial position of point p_i . The quantity d_i represents the Euclidean distance between each attributed point $p_i \in P$ and the center of rotation.

Focus on a single interval and a moving attributed point $p_i \in P$. Let θ be the translation applied to the attributed points of set P . Then,

$$\begin{cases} x^{p_i}(\theta) = x_B + d_i \cos(\theta_o^{p_i} + \theta) \\ y^{p_i}(\theta) = y_B + d_i \sin(\theta_o^{p_i} + \theta) \end{cases} \quad \forall \text{ point } p_i \in P \quad (3.12)$$

Within a single interval, it is necessary to compute $\bar{d}(P, Q)$ as a function of the rotation θ . The term accounting for Z coordinate in the distance function defined in Equation (3.2) is not considered as the alignment problem addressed involves points in \mathbb{R}^2 .

$$\bar{d}(\theta) = \frac{\sum_{i=1}^n \left\{ [x^{p_i}(\theta) - x^{q_j}(i)]^2 + [y^{p_i}(\theta) - y^{q_j}(i)]^2 + (w^{p_i} - w^{q_j}(i))^2 \right\}}{n} \quad (3.13)$$

Using the notations introduced in Equations (3.5) and (3.12), Equation (3.13) can be simplified to,

$$\bar{d}(\theta) = \frac{\sum_{i=1}^n \left\{ [x_B + d_i \cos(\theta_o^{p_i} + \theta) - x^{q_j}(i)]^2 + [y_B + d_i \sin(\theta_o^{p_i} + \theta) - y^{q_j}(i)]^2 + (w^{p_i} - w^{q_j}(i))^2 \right\}}{n} \quad (3.14)$$

In order to minimize $\bar{d}(\theta)$ its derivative with respect to θ must be set to zero. It is important to remember that (x_B, y_B) is the center of mass of set P computed without considering the transformation-invariant attributes. By setting the derivative to zero and simplifying, we get the following expression for the rotation angle:

$$\text{tg}(\theta) = \frac{\sum_{i=1}^n d_i \left([y_B - y^{q_j}(i)] \cos \theta_o^{p_i} - [x_B - x^{q_j}(i)] \sin \theta_o^{p_i} \right)}{\sum_{i=1}^n d_i \left([x_B - x^{q_j}(i)] \cos \theta_o^{p_i} + [y_B - y^{q_j}(i)] \sin \theta_o^{p_i} \right)} \quad (3.15)$$

The distance function defined in Equation (3.14) is a continuous function, and it is also bounded. The values of θ resulting from Equation (3.15) can identify a local minimum or a local maximum of the distance function, depending on the sign of the second derivative. Hence, we also check the sign of the second derivative in order to choose the right value of θ .

Equation (3.15) yields the transformation θ , applied to the set of attributed points P , which minimizes the distance between the sets of attributed points P and Q . This value of the transformation is valid only within a single interval of the set of theta intervals for all the attributed points of P . In case θ does not lie in the interval whose closest neighbors have been used in computing the distance function, referring to Theorem 1 the interval in question should not be considered.

The transformation value obtained for an interval c of the set of theta intervals for all the attributed points of P yields the optimal alignment between the two attributed point sets for all permissible rotations within the interval c .

3.5.3 Steps c and d: Computing The Rotation That Minimizes The Distance Over All The Theta Intervals

The value of $\theta(c)$ obtained in Equation (3.15) yields the rotation that minimizes the distance between the two attributed point sets P and Q within a single interval c of the set of theta intervals for all the attributed points of P . To obtain the corresponding value of the distance $\bar{d}(c)$ the value of θ obtained from Equation (3.15) should be substituted into Equation (3.14). Hence, for each interval, $\bar{d}(c)$ is the minimum distance. Finally in Step d of the algorithm ONEDOFALIGNMENT the value of θ corresponding to the minimum distance over all the intervals is found. The minimum distance over all the intervals is obtained as follows.

$$\bar{d}_{\min} = \min_{c \in C} \bar{d}(c) \quad (3.16)$$

where C is the set of all the intervals c of the partitioned theta range $[0, 2\pi]$. The value given by the Equation (3.16) is the minimum distance between sets P and Q . The corresponding rotation θ_{\min} is found as follows: let c^* be the interval in which the minimum distance was found (refer to Equation (3.16)). Then θ_{\min} is obtained as follows:

$$\theta_{\min} = \theta(c^*) \quad (3.17)$$

Equation (3.17) yields the rotation to apply to P in order to minimize the distance between P and Q . Equation (3.16) provides the minimum distance between two sets of attributed points P and Q .

3.6 Optimal Alignment Under 3 DOF Translations In \mathbb{R}^3

The third optimal alignment algorithm THREEDOFALIGNMENT finds the translation $(\Delta x, \Delta y, \Delta z)$ that minimizes the distance function given by Equation (3.1). The general

Equation (3.1) can be specified for two sets of attributed points in \mathbb{R}^3 and for three degrees of freedom translations $(\Delta x, \Delta y, \Delta z)$. Given a Cartesian coordinate system, the transformation space in this case is represented by the three components of a translation $(\Delta x, \Delta y, \Delta z)$ in the space. The overall algorithm that solves the three-degree of freedom problem can be obtained from the algorithm `TWODOFALIGNMENT` presented in Section 3.4. It is only necessary to substitute \mathbb{R}^2 by \mathbb{R}^3 and to consider the third coordinate Z . The algorithm `FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL` needs to be substituted by the algorithm `FINDINVARCLOSESTNEIGHBORSFOR3DOFTRANSL`. Hence the algorithm `THREEDOALIGNMENT` can be analyzed referring to the algorithm `TWODOFALIGNMENT` defined in Section 3.4 with the only changes described previously. In the next subsections the steps of the algorithm `THREEDOALIGNMENT` and the algorithm `FINDINVARCLOSESTNEIGHBORSFOR3DOFTRANSL` will be described.

3.6.1 Step a: Building The Set Of Regions For The Attributed Points Of Set P

Step a of the algorithm `THREEDOALIGNMENT` can be described referring to Subsection 3.4.1 with the only difference that in this case the algorithm is defined in \mathbb{R}^3 and hence the third coordinate of the attributed points needs to be considered. The transformation space in this case is the three-dimensional space \mathbb{R}^3 representing each possible translation being applied to the points of set P . Even in this case it is necessary to obtain the closest neighbors for each of the attributed points $p_i \in P$.

The algorithm `FINDINVARCLOSESTNEIGHBORSFOR3DOFTRANSL` will be used to obtain the closest neighbors. Even this algorithm is very similar to the one defined in

Subsection 3.4.1. It is only necessary to substitute \mathbb{R}^2 by \mathbb{R}^3 and to consider the third coordinate Z . Furthermore, as the algorithm is defined in \mathbb{R}^3 , in Step 1.a the transformation space is intersected with a plane rather than a line. Hence the algorithm `FINDINVARCLOSESTNEIGHBORSFOR3DOFTRANSL` can be analyzed referring to the algorithm `FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL` defined in Subsection 3.4.1 with the only changes described previously. In the next paragraphs Step 1.a and Step 2 of the algorithm `FINDINVARCLOSESTNEIGHBORSFOR3DOFTRANSL` will be explained.

Step 1.a is similar to the corresponding step described in Subsection 3.4.1 for algorithm `FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL`. The only difference is that in this case planes need to be used instead of lines. Figure 3.7(a) shows the case in which the two attributed points q_1 and q_2 of set Q have identical transformation-invariant attributes. The locus of points π_{12} of the transformation space whose distance defined in Equation (3.2) from point q_1 is the same as the distance from point q_2 is the plane through the midpoint between q_1 and q_2 and perpendicular to the segment joining q_1 and q_2 . On the other hand Figure 3.7(b) shows the case in which the attributed points have different transformation-invariant attributes. Let Δw_{11} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_1 of Q . Similarly let Δw_{12} be the difference between the transformation-invariant attributes of attributed point p_1 of P and attributed point q_2 of Q . Let $\Delta w_{11} > \Delta w_{12}$ and $\Delta w^2 = \Delta w_{11}^2 - \Delta w_{12}^2$. In this case the locus of points π_{12} of the transformation space whose distance defined in Equation (3.2) from point q_1 is the same as the distance from point q_2 is a plane perpendicular to the segment joining q_1 and q_2 , but offset with respect to the

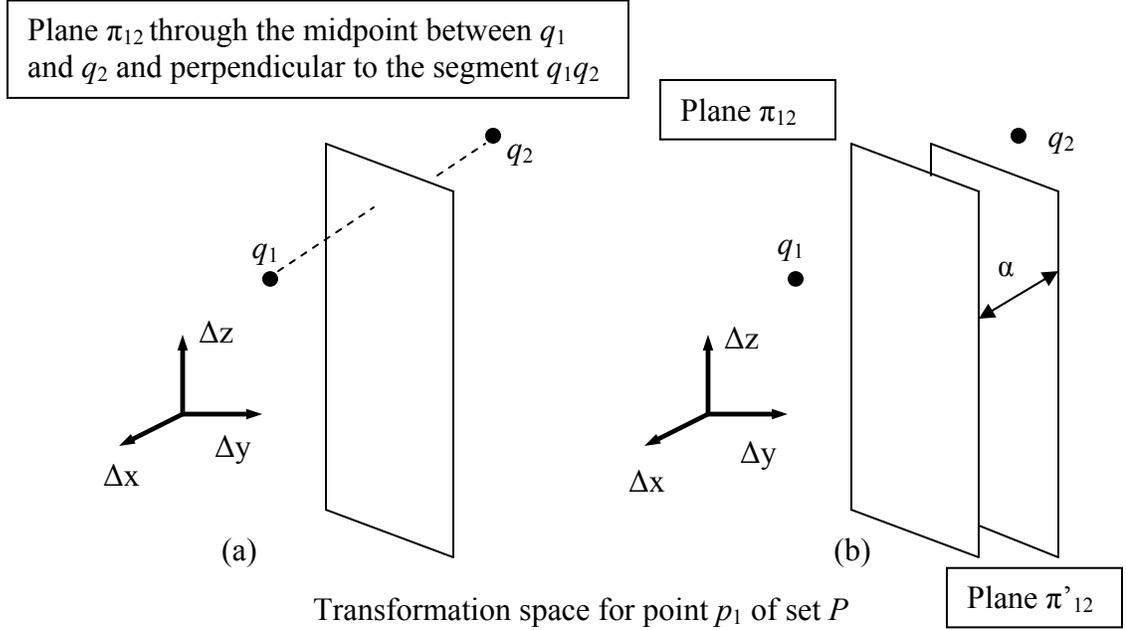


Figure 3.7: The Transformation Space of Point p_1 of Set P Is Partitioned Into Two Regions (a) Transformation-invariant Attributes Are the Same for Each Point (b) Transformation-invariant Attributes Are Different for Each Point, and Hence the Plane π'_{12} Is Offset With Respect to the Line π_{12}

midpoint between q_1 and q_2 . The offset value is the same as the one given by Equation (3.4), and it is derived in Appendix A as well.

In Step 2 the set of regions for the attributed points of P are obtained by overlapping the sets of regions obtained in Step 1. Within each region the distance defined in Equation (3.3) can be minimized. The observations made in Subsection 3.4.1 about Step 2 of algorithm FINDINVARCLOSESTNEIGHBORSFOR2DOFTRANSL apply to this subsection as well.

At this stage it is possible to compute the distance function defined in Equation (3.3) for each region. It can be expressed as a function of the coordinates (x,y,z) of the attributed points of P and Q . Coordinates of P and Q can be expressed as a function of

$(\Delta x, \Delta y, \Delta z)$, which are the components of translation. Thus the distance function defined in Equation (3.3) is expressed as a function of $(\Delta x, \Delta y, \Delta z)$ as explained in the next subsection.

3.6.2 Step b: Minimization Of The Distance Function Within A Given Region

The formulae and notations presented in this subsection are very similar to the ones presented in Subsection 3.4.2, as both the alignment problems solved involve translations, one in \mathbb{R}^2 and the other one in \mathbb{R}^3 . All the definitions and notations presented in Subsection 3.4.2, in particular Equations (3.5), (3.6), (3.7) and (3.8), can be easily modified to take into account the third coordinate Z in \mathbb{R}^3 . Hence only the final values of the translation, obtained in the same way as in Subsection 3.4.2, are shown as follows.

$$\left\{ \begin{array}{l} \Delta x = \frac{\sum_{i=1}^n \{x^{q_j}(i) - x_o^{p_i}\}}{n} \\ \Delta y = \frac{\sum_{i=1}^n \{y^{q_j}(i) - y_o^{p_i}\}}{n} \\ \Delta z = \frac{\sum_{i=1}^n \{z^{q_j}(i) - z_o^{p_i}\}}{n} \end{array} \right. \quad (3.18)$$

All the considerations on the distance function and the values of translations given by Equations (3.9) made in Subsection 3.4.2 can be extended to Equations (3.18).

3.6.3 Steps c and d: Computing The Translation That Minimizes The Distance Over All The Regions

The formulae derived in Subsection 3.4.3 can be extended to the case of \mathbb{R}^3 attributed points. So Equation (3.10) that yields the minimum distance over all the regions is reported again as follows:

$$\bar{d}_{\min} = \min_{c \in C} \bar{d}(c)$$

where C is the set of all the regions c of the transformation space. The corresponding translation $(\Delta x_{\min}, \Delta y_{\min}, \Delta z_{\min})$ is found as follows: let c^* be the interval in which the minimum distance was found (refer to Equation (3.10)). Then $(\Delta x_{\min}, \Delta y_{\min}, \Delta z_{\min})$ is obtained as follows:

$$\begin{cases} \Delta x_{\min} = \Delta x(c^*) \\ \Delta y_{\min} = \Delta y(c^*) \\ \Delta z_{\min} = \Delta z(c^*) \end{cases} \quad (3.19)$$

Equations (3.19) correspond to Equations (3.11) which were obtained for alignment problems under two DOF translations in \mathbb{R}^2 .

3.7 Complexity Evaluation For Optimal Alignment Algorithms Based On Partitioning Of Transformation Space

3.7.1 Overview

The three optimal alignment algorithms presented in the previous section are based on partitioning the transformation space into regions or intervals for which the closest neighbors remain invariant. Then a distance function is minimized within each interval or region, and finally the minimum value of the distance function over all the regions or intervals obtained is found. Therefore the complexity of the algorithms depends on the

number of intervals or regions obtained. Hence in order to evaluate the complexity of the optimal algorithms it is necessary to evaluate the number of intervals or regions the transformation space is partitioned into. Observe that the spatial arrangement that partitions the transformation space into regions or intervals is obtained by overlapping a number of spatial arrangements of the same dimension. Hence in order to evaluate the complexity of the final spatial arrangement it is necessary to evaluate the number of regions or intervals resulting from the overlapping of several spatial arrangements. A formal definition of the problem is given as follows.

Consider a set P of attributed points $p = (x_1^p, x_2^p, \dots, x_d^p, w^p)$ in \mathbb{R}^d and the following distance function, generalization of the one defined in Equation (3.2).

$$d(p, q) = \sum_{i=1}^d (x_i^p - x_i^q)^2 + (w^p - w^q)^2 \quad (3.20)$$

The quantity w^p represents the transformation-invariant attribute of point p . Consider the partitioning of \mathbb{R}^d into convex regions. Each region contains only one point p and all the points of the regions are closer to p than to any other point of P . The distance is measured by using the distance function defined in Equation (3.20). We will refer to the previously defined partitioning as *spatial arrangement* $S(P)$. Consider the overlapping of m spatial arrangements $S(P_i)$ that are built from m different sets P_i of attributed points. We would like to evaluate the complexity of the resulting spatial arrangement.

Observe that if the transformation-invariant attributes of each point p are not considered each spatial arrangement $S(P_i)$ corresponds to the Voronoi diagram of the set of points P_i [deBe97]. In order to simplify the problem, in the next subsection Voronoi diagrams in \mathbb{R}^2 are considered. In particular the complexity of the overlapping of two

Voronoi diagrams in \mathbb{R}^2 is evaluated. This result will then be used to solve the problem previously defined.

3.7.2 Complexity Of The Overlapping Of Two Voronoi Diagrams In \mathbb{R}^2

Consider a problem involving two random sets of point sites in \mathbb{R}^2 , denoted by A_1 and A_2 .

We assume that each of the sets consists of n points that have been sampled from a uniform distribution over a square of side length \sqrt{n} in \mathbb{R}^2 . Thus, the expected number of points in each unit square within this region is 1. Let $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ denote the respective Voronoi diagrams of these point sets. The number of vertices, edges and cells of a Voronoi diagram determine its complexity. A Voronoi diagram in \mathbb{R}^2 has complexity $O(n)$ [Aure91]. The question that we wish to consider is the complexity of the two dimensional arrangement resulting from the overlapping of two Voronoi diagrams $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$.

Unfortunately, this problem is complicated by the presence of boundary effects. To simplify matters, we will consider a different formulation, which captures the essential elements of the problem, without the boundary issues. A set generated by a *Poisson process* [Grim85] in \mathbb{R}^2 with rate $\delta \geq 0$ has the property that, for any measurable region R of area $A(R)$, if we let $n(R)$ denote the random variable of the number of point sites that the process generates in R , then for all $k \geq 0$, it is well known that

$$\Pr(n(R) = k) = \frac{(\delta A(R))^k \exp(-\delta A(R))}{k!} \quad (3.21)$$

It follows that the expected value, $E(n(R))$, is $\delta A(R)$.

Let A_1 and A_2 be two sets of points in \mathbb{R}^2 that have been generated by a Poisson process with rate 1. For $i \geq 1$, let S_i denote an axis-parallel square of side length i centered at the origin, and let S_0 be the empty set. Clearly $A(S_i) = i^2$. For $i \geq 1$, define a *ring* R_i to be the set-theoretic difference of two concentric squares $S_i - S_{i-1}$.

Let us consider the problem in the Poisson context. Let $m = \lfloor \sqrt{n} \rfloor$, and consider an $m \times m$ square S_m centered at the origin. From Equation (3.21) the expected number of point sites of A_1 and A_2 lying within this square is $O(n)$. Let $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ denote the respective Voronoi diagrams, restricted to lie within S_m . Our main result is as follows.

Theorem 2: *Consider two random point sets A_1 and A_2 generated independently from a Poisson process with rate 1 in \mathbb{R}^2 . Then the expected complexity of the two dimensional arrangement resulting from the overlapping of $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ is $O(n)$.*

Proof. The complexity of the two dimensional arrangement resulting from the overlapping of $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ is determined by the number of intersections between the edges of $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$. In fact the number of new edges, cells and vertices resulting from the overlapping is proportional to the number of edge intersections occurring. Because the Poisson process is stationary, the random variables that evaluate the number of edge intersections occurring in any unit square contained within S_m are identical. Thus, it suffices to show that the expected number of edge intersections occurring within the unit square S_1 centered at the origin is $O(1)$, and it will follow immediately by the linearity of expectation that the total number of intersections in S_m is

$O(m^2) = O(n)$. Let I_1 be a random variable denoting the number of intersections resulting from the overlapping of $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ that lie in the square S_1 . Our goal is to show that $E(I_1)$ is $O(1)$.

Let i and k be nonnegative integer values. Consider an intersection that occurs within S_1 . This intersection is generated by the intersection of two Voronoi edges, one from $\text{Vor}(A_1)$ and the other from $\text{Vor}(A_2)$. Let $\{p_{11}, p_{12}\} \in A_1$ and $\{p_{21}, p_{22}\} \in A_2$ denote the points that gave rise to the intersecting edges, and let r_{max} denote the farthest distance of any of these points from the intersection point. Let $e(i,k)$ denote the random event that the edge intersection lies in S_1 , where the point at distance r_{max} lies in the ring R_i , and there are exactly k points of $A_1 \cup A_2$ lying within S_i . In Figure 3.8 an instance of the event $e(i,6)$ is shown. Every edge intersection can uniquely be associated with some event $e(i,k)$, and since the complexity of a Voronoi diagram generated by k points is $O(k)$, it follows that at most $O(k^2)$ intersections between the $O(k)$ edges can be associated with each such event. Let $\Pr(e(i,k))$ denote the probability of this event occurring. Thus, up to a constant factor c , the expected complexity in S_1 satisfies:

$$E(I_1) \leq c \sum_{i \geq 1} \sum_{k \geq 0} k^2 \Pr(e(i,k)).$$

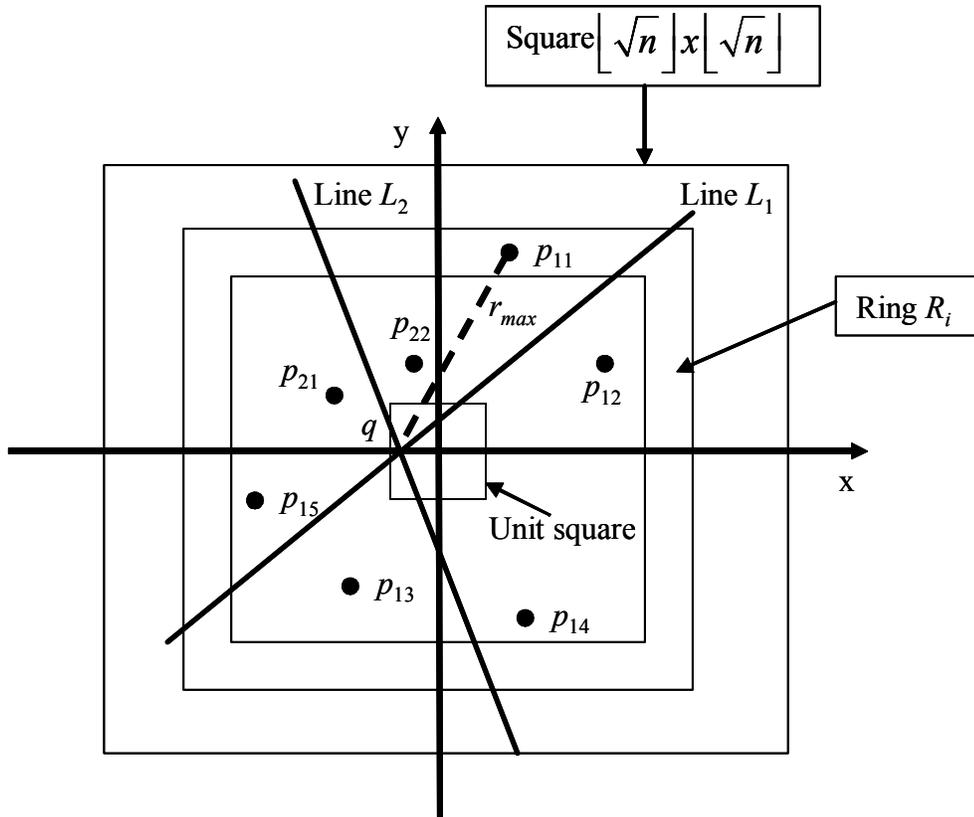


Figure 3.8: Instance of Event $e(i,6)$ Occurring in \mathbb{R}^2 . Line L_1 Generated by Points p_{11} and p_{12} of Set A_1 and Line L_2 Generated by Points p_{21} and p_{22} of Set A_2 Intersect Within the Unit Square. Point p_{11} of Set A_1 Is the Farthest Point From the Intersection Point q and Lies in Ring R_i at Distance r_{max} From q . There Are Six Points in Total Contained in S_i

For $e(i,k)$ to occur, two necessary events must occur. First, let q denote the intersection point of the two Voronoi edges. It follows from Voronoi diagram properties that there must be a circle centered at q whose radius has length r_{max} that contains no points of either A_1 or A_2 . Because the point at distance r_{max} from point q lies in R_i , its distance from the origin is at least $(i-1)/2$. Since the distance from q to the origin is at most $\sqrt{2}/2$, it follows that there is a circle of radius $(i-3)/2$ centered at the origin that either contains no points of A_1 or no points of A_2 . For concreteness, let us assume the former. Because the two point sets are drawn from the same distribution, the other case will give rise to the

same expected number of intersections, and so this will only increase the total number of intersections by a factor of 2. For $i > 3$, let $e_1(i)$ denote the event that there is a circle of radius $(i-3)/2$ centered at the origin that contains no point of A_1 . To simplify notation, for $i \leq 3$, let $e_1(i)$ denote an event that occurs with probability 0. Second, there must be k points lying within S_i . Call this event $e_2(i,k)$.

Let $P_1(i)$ and $P_2(i,k)$ denote the respective probabilities of events $e_1(i)$ and $e_2(i,k)$. From Equation (3.21), the fact that the circle has area $\pi(i-3)^2/4$ and the fact that the circle must be empty, that is $k = 0$, we have

$$P_1(i) = \exp\left(-\frac{\pi(i-3)^2}{4}\right) \leq \exp\left(-\frac{\pi(i/C)^2}{4}\right),$$

where the constant $C > 2$ and $i > \frac{3C}{C-2}$. Values of $i < \frac{3C}{C-2}$ can be discarded. The rationale behind this will be explained below.

From Equation (3.21), the fact that by independence the union $A_1 \cup A_2$ is a Poisson process with rate 2 and the fact that the square S_i has area i^2 we have

$$P_2(i,k) = \frac{(2i^2)^k \exp(-2i^2)}{k!}.$$

Thus to provide an upper bound on the expected complexity $E(I_1)$, it suffices to bound the following quantity:

$$X = \sum_{i \geq 1} \sum_{k \geq 0} k^2 (P_1(i) \wedge P_2(i,k)).$$

Here is a quick outline of the analysis. Observe that when i is large, the probability $P_1(i)$ decreases rapidly, because it is very unlikely that there can be a large circle with no

points. On the other hand, when i is small, it is unlikely that k will be much larger than its expected value, which is $O(i^2)$, since the probability $P_2(i,k)$ decreases rapidly as k increases above this quantity. Thus, for an intersection to lie in S_1 we expect i to be small and so we expect k to be small as well. Thus, we expect number of intersections occurring within S_1 to be small on average.

In order to separate these two cases, we define k to be *small* if it is less than $(ei)^2$ (where e is the base of the natural logarithm), and large otherwise. We break the analysis of X into two parts, depending on the size of k . Henceforth, let $w = ei$.

Using the fact that $P_1(i) \wedge P_2(i,k) \leq \min(P_1(i), P_2(i,k))$ the following formulae are obtained.

$$\begin{aligned}
X &\leq \sum_{i \geq 1} \sum_{k \geq 0} k^2 \min(P_1(i), P_2(i,k)) \\
&= \sum_{i \geq 1} \left[\sum_{0 \leq k \leq w^2} k^2 \min(P_1(i), P_2(i,k)) + \sum_{k \geq w^2} k^2 \min(P_1(i), P_2(i,k)) \right] \\
&\leq \sum_{i \geq 1} \left[\sum_{0 \leq k \leq w^2} k^2 P_1(i) + \sum_{k \geq w^2} k^2 P_2(i,k) \right] \\
&\leq \sum_{i \geq 1} \sum_{0 \leq k \leq w^2} k^2 P_1(i) + \sum_{i \geq 1} \sum_{k \geq w^2} k^2 P_2(i,k).
\end{aligned}$$

Let $X_1 = \sum_{i \geq 1} \sum_{0 \leq k \leq w^2} k^2 P_1(i)$ and $X_2 = \sum_{i \geq 1} \sum_{k \geq w^2} k^2 P_2(i,k)$. We will show that each of these is $O(1)$. First we consider X_1 . By applying our bound on $P_1(i)$ in X_1 and ignoring the terms for $i < C$, which can be always bounded, we have

$$X_1 \leq \sum_{i \geq C} \sum_{0 \leq k \leq w^2} k^2 \exp\left(-\frac{\pi(i/C)^2}{4}\right) \leq \sum_{i \geq C} w^6 \exp\left(-\frac{\pi(i/C)^2}{4}\right) \leq \sum_{i \geq C} (ei)^6 \exp\left(-\frac{\pi(i/C)^2}{4}\right).$$

By making the variable substitution $j = \pi i^2 / 4C^2$ we have

$$X_1 \leq \frac{64(Ce)^6}{\pi^3} \sum_{j \geq 0} \frac{j^3}{e^j}.$$

It is well known [Corm01] that last summation converges and so $X_1 = O(1)$.

Next we consider X_2 . First observe that because $k \geq w^2 = (ei)^2$ and so $i \leq \sqrt{k}/e$, we

can alter the order of the summation to obtain

$$X_2 = \sum_{i \geq 1} \sum_{k \geq w^2} k^2 P_2(i, k) \leq \sum_{k \geq 0} \sum_{1 \leq i \leq \sqrt{k}/e} k^2 P_2(i, k).$$

Now, by applying our bound on $P_2(i, k)$ in X_2 we have

$$X_2 \leq \sum_{k \geq 0} \sum_{1 \leq i \leq \sqrt{k}/e} k^2 \left(\frac{(2i^2)^k \exp(-2i^2)}{k!} \right),$$

and making the substitution $j = 2i^2$ we have

$$X_2 \leq \sum_{k \geq 0} \sum_{2 \leq j \leq 2k/e^2} k^2 \left(\frac{(j)^k \exp(-j)}{k!} \right).$$

Recall that by Stirling's approximation [Corm01], there exists a constant c such that

$k! \geq c\sqrt{k} \left(\frac{k}{e}\right)^k$. Clearly $2 \leq j \leq 2k/e^2$ and $\exp(-j) \leq 1$, and so we have

$$X_2 \leq \sum_{k \geq 0} k^2 \left(\frac{(2k/e^2)^{k+1}}{k!} \right) \leq \sum_{k \geq 0} k^2 \left(\frac{(2k/e^2)^{k+1}}{c\sqrt{k} (k/e)^k} \right) \leq \frac{2}{e^2 c} \sum_{k \geq 1} \frac{k^{5/2}}{\left(\frac{e}{2}\right)^k} \leq \sum_{k \geq 1} \frac{k^3}{\left(\frac{e}{2}\right)^k}.$$

As before, this summation converges, and so $X_2 = O(1)$. Therefore, the expected number of intersections is at most $X = X_1 + X_2 = O(1)$, as desired. As the number of edge intersection occurring is $O(n)$, the complexity of the two dimensional arrangement resulting from the overlapping of the two Voronoi diagrams $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ is $O(n)$.

The important result obtained in Theorem 2 will be extended to d dimensional Voronoi diagrams in the next subsection. Then the complexity of the overlapping of m spatial arrangements defined in Subsection 3.7.1 will be evaluated.

3.7.3 Complexity Of The Overlapping Of m Spatial Arrangements $S(P_i)$ In \mathbb{R}^d

The result of Theorem 2 can be extended to d dimensional Voronoi diagrams by using the following theorem. Note that the constant hidden in the O notation of the Voronoi diagram complexity $O(n)$ grows exponentially with the Voronoi diagram dimension d .

Theorem 3: *Consider two random point sets A_1 and A_2 generated independently from a Poisson process with rate 1 in \mathbb{R}^d for a fixed constant d . Then the expected complexity of the d dimensional arrangement resulting from the overlapping of $Vor(A_1)$ and $Vor(A_2)$ is $O(n)$.*

Proof. In order to prove Theorem 3 let us focus on the complexity of d dimensional Voronoi diagrams. For random point sets generated independently from a Poisson process with rate 1 in \mathbb{R}^d the expected complexity is $O(n)$. In fact Voronoi diagrams for any reasonable probabilistic distribution are combinatorially simple and their complexities are linear with respect to the number of points n with constant of proportionality increasing exponentially with d [Bien05, Dwyer91]. By using this result the proof for Theorem 2 can be used to prove Theorem 3 as well. In the case of d dimensional Voronoi diagrams the concept of Voronoi entities needs to be introduced. Voronoi-entities can be defined as i -faces, that is the i -dimensional faces forming the Voronoi diagram. In particular d -faces are the d -dimensional spatial regions into which \mathbb{R}^d is partitioned by the Voronoi diagram. The $(d-1)$ -faces are called *facets* and $(d-2)$ -

faces are called *ridges*. In \mathbb{R}^3 2-faces are the 3D spatial regions of the Voronoi diagram, facets (i.e. 2-faces) are the faces forming each region, ridges (i.e. 1-faces) are the edges bounding each facet and 0-faces are vertices. Therefore in \mathbb{R}^d Voronoi entity intersections need to be counted rather than Voronoi edge intersections. Therefore the proof of Theorem 2 can be extended to d dimensional space by substituting the concept of edge intersection with the more general one of entity intersection. Therefore the estimated number of intersections will be $O(n)$ in the d dimensional space as well. Hence the complexity of the d dimensional arrangement resulting from the overlapping of $\text{Vor}(A_1)$ and $\text{Vor}(A_2)$ is $O(n)$.

Now the complexity of the overlapping of m spatial arrangements $S(P_i)$ in \mathbb{R}^d defined in Subsection 3.7.1 can be evaluated referring to two corollaries that follow immediately from Theorem 3. They are stated and proved as follows.

Corollary 1: *Consider m random point sets A_i for $i=1,2,\dots,m$ of cardinality n generated independently from a Poisson process with rate 1 in \mathbb{R}^d for a fixed constant d . Then the expected complexity of the d dimensional arrangement resulting from the overlapping of $\text{Vor}(A_i)$ is $O(m^2n)$.*

Proof. The complexity of the d -dimensional arrangement resulting from the overlapping of the m Voronoi diagrams $\text{Vor}(A_i)$ is determined by the total complexity of these m diagrams plus the total number of intersections between each pair of distinct Voronoi diagrams $\text{Vor}(A_i)$ and $\text{Vor}(A_j)$. Each individual diagram has an expected complexity of $O(n)$ and hence their total complexity is $O(mn)$. In order to bound the number of intersections, we observed previously that the number of new entities resulting

from the overlapping is proportional to the number of intersections occurring. From Theorem 3 it follows that for each pair of Voronoi diagrams, their overlap has $O(n)$ intersections in expectations. As there are $O(m^2)$ possible distinct pairs of the m Voronoi diagrams being overlapped, the total number of new entities in the resulting d -dimensional arrangement is $O(m^2n)$. Therefore the complexity of the d dimensional arrangement resulting from the overlapping of m Voronoi diagrams $\text{Vor}(A_i)$ is $O(m^2n)$.

Corollary 2: *Consider m random attributed point sets A_i for $i=1,2,\dots,m$ of cardinality n generated independently from a Poisson process with rate 1 in \mathbb{R}^d . The attributes are assumed to be generated independently from a Poisson process of rate 1 over \mathbb{R} . Then the expected complexity of the d -dimensional arrangement resulting from the overlapping of $S(A_i)$ is $O(m^2n)$.*

Proof. A spatial arrangement $S(A_i)$ has been defined in Subsection 3.7.1. Observe that a set P of attributed points $p = (x_1^p, x_2^p, \dots, x_d^p, w^p)$ in \mathbb{R}^d can be seen as a set P' of non-attributed points $p = (x_1^p, x_2^p, \dots, x_d^p, x_{d+1}^p)$ in \mathbb{R}^{d+1} , where $x_{d+1}^p = w^p$. Hence let us consider the $d+1$ -dimensional arrangement resulting from the overlapping of m Voronoi diagrams $\text{Vor}(A_i)$ of non-attributed point sets in \mathbb{R}^{d+1} . From Corollary 1 it can be inferred that the complexity of the $d+1$ -dimensional arrangement is $O(m^2n)$. In order to address attributed points in \mathbb{R}^d it is necessary to consider the intersection of the $d+1$ -dimensional arrangement with the hyperplane $x_{d+1}^p = c$ in \mathbb{R}^{d+1} where c is a constant real number. The resulting d -dimensional spatial arrangement in \mathbb{R}^d will have complexity $O(m^2n)$.

Corollary 2 addresses the problem stated in Subsection 3.7.1 in the case of attributed point sets generated from a Poisson process and yields the complexity evaluation for the lower dimension algorithms. In fact Corollary 2 guarantees that the number of regions resulting from the overlapping of m spatial arrangements of complexity $O(n)$ generated independently from a Poisson process is $O(m^2n)$. The evaluated low order polynomial complexity is smaller than the exponential complexity $O(n^m)$, which is obtained in case each entity of each overlapping arrangement intersects with all the entities of all the other arrangements. Hence, considering the fact that some of the regions will also be discarded by using Theorem 1, the optimal algorithms based on partitioning of transformation spaces that have been developed can efficiently solve attributed point alignment problems.

3.8 Summary

In this chapter, we have shown that it is possible to partition the transformation space into a set of regions such that the closest neighbors remain invariant in each region. Using this partitioning, we have designed new algorithms to perform attributed point optimal alignment by searching for optimal solutions in each region. We have also shown that the resulting numbers of partitions are bounded by low order polynomials in the case of well-behaved uniform attributed point distributions and hence it is possible to use these algorithms in practice.

Theoretically, the method used in this chapter can work for all types of rigid body transformations. However, in higher dimensions, the data structures needed to perform partitioning and hold results are very complex and difficult to implement. Therefore, in this chapter we have focused on lower dimension transformations. In Chapter 4 we

describe how these algorithms can be used in iterative strategies to perform alignment using higher dimensional transformations.

The optimal alignment corresponds to the global minimum of a distance function that is computed between the two sets of attributed points being aligned. The distance function that is used needs to be differentiable. It accounts for transformation dependent and transformation invariant attributes. Distance function can include desired numbers of attributes and their weights, hence it is customizable. Furthermore the distance function is asymmetric, a property that may be desirable in manufacturing applications. The algorithms can be easily modified in order to be extended to symmetric distance functions.

The complexity of the algorithms designed depends on the complexity of the spatial arrangements used to partition the transformation space. A low order polynomial upper bound complexity has been obtained in the general case of d dimensional arrangements for well-behaved uniform attributed point distributions.. The result obtained is valid for sets of attributed points generated by using a Poisson process. This assumption is reasonable as the optimal alignment algorithms in this thesis are used to perform feature-based shape similarity assessment for manufacturing applications. In manufacturing field part features are expected to be uniformly distributed attributed points or vectors. Hence the theoretical result should be applicable to manufacturing applications.

Chapter 4: Attributed Point Alignment Algorithms Based On Iterative Strategies

This chapter is organized as follows. In Section 4.1 the motivation behind the research work described in this chapter is described. Section 4.2 gives the problem formulation. In Section 4.3 iterative strategies are formally defined. In Section 4.4 properties of a particular class of iterative strategies are analyzed. In Section 4.5 experimental results are presented. In Section 4.6 the main results are summarized.

4.1 Motivation

In theory, the partitioning scheme described in Chapter 3 can be used to handle any arbitrary transformation space. For example, if we have a six dimensional transformation space, then this space can be partitioned into spatial regions that are six dimensional entities. However, implementing direct partitioning of transformation spaces that involve more than three dimensions appears to be a very challenging task for the following reason. Years of research in solid modeling community has established excellent foundations for representing and computing three dimensional geometric entities with adequate precision. In fact, numerous commercial and academic libraries are available for constructing two and three dimensional geometric entities. A typical library for constructing and querying three dimensional geometric entities consists of tens of thousands of lines of codes. Currently, libraries are not available for constructing and querying four or higher dimensional geometric entities. Data structures and algorithms involved in implementing four dimensional entities are significantly more complex than three dimensional entities. Therefore, at least in the near term, robust implementation of partitioning of transformation spaces involving more than three dimensions appears to be impractical.

We are interested in exploring strategies in which a higher dimension problem is transformed into a sequence of lower dimension problems by fixing certain dimensions in each of the lower dimensional problems. We refer to these problem solving strategies as *iterative strategies*. These strategies involve use of sequential application of optimal alignment algorithms based on partitioning of lower dimension transformation spaces. This corresponds to searching for the optimal alignments in certain projections of the transformation space in an iterative manner.

Building iterative strategies that can lead to the optimal solution of higher dimension alignment problems is a challenging task. In fact iterative strategies can get stuck in local minima rather than leading to the optimal solution. Hence it is necessary to identify characteristics and properties of iterative strategies such that optimal solutions of higher dimension alignment problems can be found.

4.2 Problem Formulation

Consider two sets of attributed points P and Q . P and Q are compared using the distance function defined in Equation (3.1). The distance function between the attributed points $p \in P$ and $q \in Q$ was defined in Equation (3.2). A more general definition is the following.

$$d(p, q) = \sum_{i=1}^d (x_i^p - x_i^q)^2 + (w^p - w^q)^2 \quad (4.1)$$

In Equation (4.1) d is the dimension of the attributed points $p \in P$ and $q \in Q$, x_i^p and x_i^q are the i -th coordinates of points $p \in P$ and $q \in Q$ and finally w^p and w^q are the transformation-invariant attributes associated to points $p \in P$ and $q \in Q$. The transformation \mathbf{T} applied to one set with respect to the other such that distance between the two sets is minimized is sought. The distance function defined in Equation (3.1) can

be written as in Equation (3.3) as it is function of the transformation \mathbf{T} . The global minimum of the distance function is the optimal solution of the alignment problem.

4.3 Definition Of Iterative Strategies

In order to solve problems involving higher degree of freedom transformations iterative strategies that use optimal alignment algorithms based on partitioning of lower dimension transformation spaces are defined as follows.

Consider two sets of points P and Q . P needs to be aligned with respect to Q using a transformation $\mathbf{T} = (t_1, t_2, \dots, t_m)$ belonging to a transformation space Γ . Assume that a set of algorithms that can perform optimal alignment between P and Q based on partitioning of lower dimension transformation spaces Γ^s is available. Every $\mathbf{T}^s \in \Gamma^s$ is of the form such that one or more of its components is zero (e.g., $\mathbf{T}^s = (t_1, t_2, t_3, 0, 0, 0)$). Therefore $\Gamma^s \subset \Gamma$.

Let the given set of optimal alignment algorithms based on partitioning of lower dimension transformation spaces be

$$\{\text{ALIGN-}\mathbf{T}_1^s, \text{ALIGN-}\mathbf{T}_2^s, \dots, \text{ALIGN-}\mathbf{T}_n^s\} \quad (4.2)$$

where $\text{ALIGN-}\mathbf{T}_i^s$ performs the optimal alignment of P with respect to Q using a \mathbf{T}_i^s transformation. The following notation describes the effect of alignment.

$$P' = \text{ALIGN-}\mathbf{T}_i^s(P, Q) \quad (4.3)$$

where P' is transformed P after applying the transformation that results in the optimal alignment of P with respect to Q using \mathbf{T}_i^s .

Assume that the transformation set $\{\mathbf{T}_1^s, \mathbf{T}_2^s, \dots, \mathbf{T}_n^s\}$ that corresponds to the optimal alignment algorithms described previously is such that for every component t_i of general transformation \mathbf{T} , there exists a lower dimension transformation with a corresponding

non-zero component. If this condition is met then set $\{\mathbf{T}_1^s, \mathbf{T}_2^s, \dots, \mathbf{T}_n^s\}$ is said to span the dimension of \mathbf{T} .

Now consider the following sequence of application of algorithms.

$$(P_1 = \text{ALIGN}_1(P, Q), P_2 = \text{ALIGN}_2(P_1, Q), \dots, P_k = \text{ALIGN}_k(P_{k-1}, Q)) \quad (4.4)$$

where $\text{ALIGN}_i \in \{\text{ALIGN-}\mathbf{T}_1^s, \text{ALIGN-}\mathbf{T}_2^s, \dots, \text{ALIGN-}\mathbf{T}_n^s\}$

This sequence terminates when the following condition is met.

$$|\bar{d}(P_k, Q) - \bar{d}(P', Q)| < \varepsilon \quad (4.5)$$

where $P' = \text{ALIGN}(P_k, Q)$ and $\text{ALIGN} \in \{\text{ALIGN-}\mathbf{T}_1^s, \text{ALIGN-}\mathbf{T}_2^s, \dots, \text{ALIGN-}\mathbf{T}_n^s\}$.

Two general observations can be made about iterative strategies. The first one is that different iterative strategies applied to the same alignment problem can lead to different outcomes and can have significantly different performances. From now on the initial position of the point sets being aligned will be referred to as *initial condition*. The second observation is that the outcome of an iterative strategy depends also on the initial condition. The same iterative strategy applied to the same sets of attributed points starting from two different initial conditions can in general have different outcomes. In light of these observations, in order to choose the most appropriate iterative strategy to optimally solve a higher dimension alignment problem the following two pieces of information are needed. The first one is whether the outcome of a particular iterative strategy is guaranteed to be at least a local minimum. In fact an iterative strategy that is guaranteed to reach a local minimum is expected to perform better than an iterative strategy that is not guaranteed to reach a local minimum. This is because the performance of an iterative strategy would depend only on the average number of local minima expected for the alignment problem. The other piece of information needed is the number of initial

conditions that are needed to reach the global minimum. Observe that this piece of information is related to the average number of local minima expected for the alignment problem addressed. This is true especially if the iterative strategy being used is guaranteed to reach a local minimum.

In the next section the mathematical foundations for a class of iterative strategies in \mathbb{R}^2 are given. These mathematical foundations guarantee that this particular class of iterative strategies in \mathbb{R}^2 always leads to a local minimum of the distance function defined in Equation (3.1).

4.4 Mathematical Foundations For Iterative Strategies In \mathbb{R}^2

Consider the alignment problem in \mathbb{R}^2 under 3 DOF transformations. In this case, referring to the two sets of attributed points P and Q , P needs to be aligned with respect to Q using the transformation $\mathbf{T} = (\Delta x, \Delta y, \theta)$ belonging to the transformation space Γ . The first two components of the transformation \mathbf{T} represent the translation components and the third one represents the rotation around an axis perpendicular to \mathbb{R}^2 . The optimal alignment algorithms described in Section 3.4 and Section 3.5 can be used to provide partial solutions to the alignment problem defined above. Call the former algorithm ALIGN- \mathbf{T}^1 and the latter algorithm ALIGN- \mathbf{T}^2 . Consider the lower dimension transformation spaces Γ^1 and Γ^2 . The transformations $\mathbf{T}^1 \in \Gamma^1$ and $\mathbf{T}^2 \in \Gamma^2$ are such that one or more components are zero: $\mathbf{T}^1 = (\Delta x, \Delta y, 0)$ and $\mathbf{T}^2 = (0, 0, \theta)$. Algorithm ALIGN- \mathbf{T}^1 can perform optimal alignment by using the transformations $\mathbf{T}^1 \in \Gamma^1$ and algorithm ALIGN- \mathbf{T}^2 can perform optimal alignment by using the transformations $\mathbf{T}^2 \in \Gamma^2$.

The two optimal alignment algorithms ALIGN-T^1 and ALIGN-T^2 provide partial solutions for the alignment problem defined above. However it is possible to define an iterative strategy I^2 by using these two algorithms as follows.

$$(P_1 = \text{ALIGN}_1(P, Q), P_2 = \text{ALIGN}_2(P_1, Q), \dots, P_k = \text{ALIGN}_k(P_{k-1}, Q)) \quad (4.6)$$

where referring to Equation (4.2) and to the previous paragraph the following is valid.

$$\begin{cases} \text{ALIGN}_i(P, Q) = \text{ALIGN-T}^1 & \text{if } i \text{ is even} \\ \text{ALIGN}_i(P, Q) = \text{ALIGN-T}^2 & \text{if } i \text{ is odd} \end{cases} \quad (4.7)$$

The rotation θ that is used in algorithm ALIGN-T^2 is performed around the center of mass of the point set P that is being rotated. The center of mass is computed without considering the transformation-invariant attribute of each point. This paragraph and in particular Equations (4.6) and (4.7) define the iterative strategy I^2 . An important theorem on iterative strategy I^2 is proved as follows.

Theorem 4: *Consider the attributed point alignment problem in \mathbb{R}^2 under 3 DOF transformation formulated in this section. If the iterative strategy I^2 defined by Equations (4.6) and (4.7) is applied to that alignment problem, then the resulting distance value is guaranteed to be a local minimum for the distance function defined in Equation (3.1).*

Proof. In order to prove Theorem 4, let us refer to the notations introduced in Subsections 3.4.2 and 3.5.2 and corresponding equations. As in this case both translations and rotations are being considered, the following representation of points of transformed set P can be used.

$$\begin{cases} x^{p_i}(\theta) = x_B + \Delta x + d_i \cos(\theta_o^{p_i} + \theta) \\ y^{p_i}(\theta) = y_B + \Delta y + d_i \sin(\theta_o^{p_i} + \theta) \end{cases} \quad \forall \text{ point } p_i \in P \quad (4.8)$$

The distance function defined in Equation (3.1) can be adapted for the notations being used and the transformation $\mathbf{T} = (\Delta x, \Delta y, \theta)$. Theorem 4 can be proved by checking the following conditions in correspondence of the outcome of the iterative strategy I^2 .

1. The first partial derivatives of the distance function with respect to each of the transformation component must be equal to 0
2. The Hessian of the distance function must be positive definite.

Observe that the first partial derivatives of the distance function defined in Equation (3.1) are equal to 0 by definition of the iterative strategy I^2 itself. In fact the outcome of the iterative strategy corresponds to a minimum of the distance function with respect to both translations and rotations. Hence the first derivatives of the distance function with respect to translations and rotations are both equal to 0. So it is only necessary to check the second condition. In order for the second condition to be valid it is necessary that all the eigenvalues of the Hessian matrix be strictly positive. Consider the Hessian matrix corresponding to the distance function used.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta^2} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta \partial x} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta \partial y} \\ \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x \partial \theta} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x^2} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x \partial y} \\ \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial y \partial \theta} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial y \partial x} & \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial y^2} \end{bmatrix} \quad (4.9)$$

The elements of the Hessian have the following expression.

$$\left\{ \begin{array}{l}
\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x^2} = \frac{\partial^2 \bar{d}(x, y, \theta)}{\partial y^2} = 2 \\
\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta^2} = \frac{-2 \sum_{i=1}^n [\Delta x + x_B - x^{q_j}(i)] d_i \cos(\theta + \theta_o^{p_i}) - 2 \sum_{i=1}^n [\Delta y + y_B - y^{q_j}(i)] d_i \sin(\theta + \theta_o^{p_i})}{n} \\
\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x \partial y} = 0 \\
\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial x \partial \theta} = \frac{-2 \sum_{i=1}^n d_i \sin(\theta + \theta_o^{p_i})}{n} \\
\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial y \partial \theta} = \frac{2 \sum_{i=1}^n d_i \cos(\theta + \theta_o^{p_i})}{n}
\end{array} \right. \quad (4.10)$$

By definition of the iterative strategy I^2 and in particular by using the fact that the rotations take place around the center of mass of the set being rotated, the third and the fourth Equations (4.10) are equal to 0. Hence the characteristic equation of the Hessian matrix defined in Equation (4.9) has the following simple expression.

$$(2 - \lambda)^2 \left(\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta^2} - \lambda \right) \quad (4.11)$$

So two eigenvalues are coincident and equal to 2, which is a positive number. The sign of the third eigenvalue depends on the sign of the second derivative $\frac{\partial^2 \bar{d}(x, y, \theta)}{\partial \theta^2}$.

Observe that both algorithm ALIGN- \mathbf{T}^1 and algorithm ALIGN- \mathbf{T}^2 defined in this section are optimal. Hence in correspondence of the outcome of iterative strategy I^2 both the algorithms reach the global minimum with respect to the transformations that they use. In particular ALIGN- \mathbf{T}^2 will reach the global minimum with respect to rotation θ . Therefore

it immediately follows that $\frac{\partial^2 \vec{d}(x, y, \theta)}{\partial \theta^2} > 0$ in correspondence of the outcome of the iterative scheme I^2 . This proves that the Hessian is positive definite. Hence both the conditions stated previously are met and so the value of distance represents a local minimum for the distance function, which proves the theorem.

4.5 Experimental Results

A series of experiments has been run to assess the performance of the previously described iterative strategies. In this section these experiments are presented. In Subsection 4.5.1 experimental results by using iterative strategies in \mathbb{R}^2 are presented. In Subsection 4.5.2 experimental results by using iterative strategies in \mathbb{R}^3 using 6 DOFs are presented. In Subsection 4.5.3 experimental results by using iterative strategies in \mathbb{R}^3 using 3 rotational DOFs are presented.

4.5.1 Tests On Iterative Strategies In \mathbb{R}^2

Consider the alignment problem in \mathbb{R}^2 under 3 DOF transformations formulated in Section 4.4 and the iterative strategy I^2 defined by Equations (4.6) and (4.7). By Theorem 4 the distance value corresponding to the two sets being aligned is guaranteed to be a local minimum for the distance function defined in Equation (3.1). Hence the solution is in general a local minimum. The local minimum reached depends on the initial condition of the point sets being aligned. The same iterative scheme applied to the same sets of points starting from different initial conditions may lead to two different local minima.

Hence more initial conditions are considered, more likely the global minimum is reached by iterative strategy I^2 .

In order to assess the performance of iterative strategy I^2 , in the first two experiments it has been applied to the attributed point alignment problem in \mathbb{R}^2 under 3 DOF transformations.

In the first experiment, 200 initial sets of 20, 40 and 80 attributed points inside a circle of a fixed size were randomly generated for a total of 600 initial sets. The transformation-invariant attributes were also generated randomly. Then a random transformation was applied to each of the 600 sets, creating 600 more sets of attributed points. Hence finally 600 pairs of attributed points were obtained. Consider all the pairs of sets consisting of one initial set and one corresponding additional sets created as explained previously. The iterative strategy I^2 was applied to each pair of attributed point sets, evaluating a total of 600 instances. The expected minimum distance corresponding to the optimal alignment computed among the sets of each pair is 0. Cases in which the optimal alignment was not found were handled using the following procedure. A random transformation was applied to the initial set of the pair in order to create a different initial condition. Then the experiment was repeated with the different initial condition that had been obtained for those instances. This procedure was repeated until the optimal alignment was found or the limit of ten different initial conditions was reached. Hence out of the 600 instances, the optimal alignment (i.e. distance = 0) was found in all of them.

In the second experiment, 200 initial sets of 20 attributed points were randomly generated and then again a random transformation was applied to each of them, creating

200 more sets of attributed points. This time before applying the iterative scheme 5%, 10%, 20% and 40% of the attributed points were erased from one of the sets being compared. We expected the erasing of the points not to affect the alignment of each pair of sets, as the distance function used is defined also for sets of different cardinality. Out of the 800 instances (i.e. 200 instances for each distinct number of erased points), the optimal alignment was found in all of them.

In Figure 4.1 a histogram representing the number of converging and non-converging instances vs. the number of initial conditions used for both the first and the second experiment is shown.

4.5.2 Tests On Iterative Strategies In \mathbb{R}^3 Using 6 DOFs

Consider the alignment problem in \mathbb{R}^3 under 6 DOF transformations. In this case, referring to the two sets of attributed points P and Q , P needs to be aligned with respect to Q using the transformation $\mathbf{T} = (\Delta x, \Delta y, \Delta z, \psi, \varphi, \theta)$. The first three components of the transformation \mathbf{T} represent the translation components. The second three components represent the rotations around the coordinate axis X, Y, and Z respectively. The optimal alignment algorithms described in Section 3.4 and Section 3.5 can be used to provide partial solutions to the alignment problem defined above. Call ALIGN- \mathbf{T}_{XY} the algorithm described in Section 3.4 if it uses translations in the coordinate plane XY. Similarly call ALIGN- \mathbf{T}_θ the algorithm described in Section 3.5 if it uses rotations about the coordinate axis Z. Observe that the Z coordinate of each attributed point remains constant when the two optimal alignment algorithms ALIGN- \mathbf{T}_{XY} and ALIGN- \mathbf{T}_θ are applied to attributed

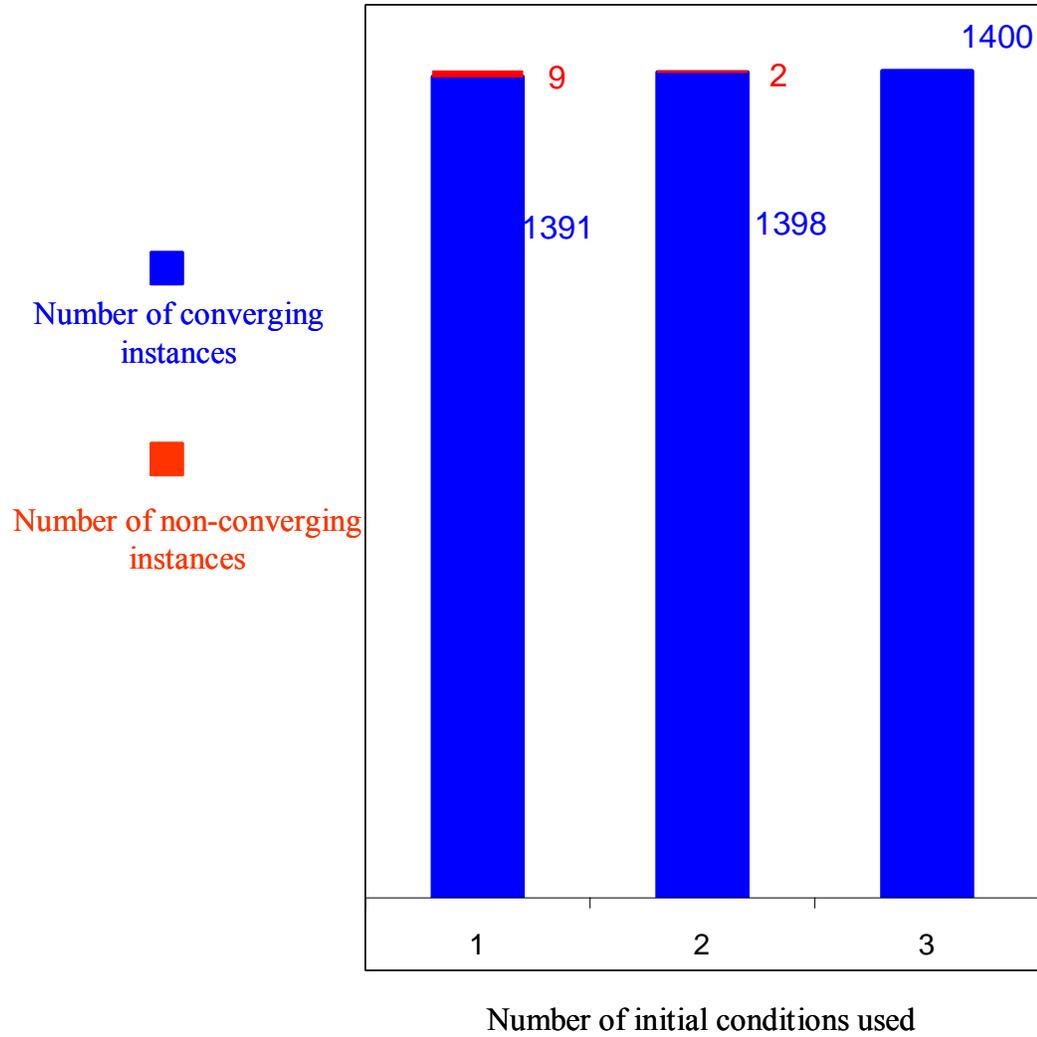


Figure 4.1: Histogram Showing the Number of Converging and Non-converging Instances Vs. The Number of Initial Conditions Used for Iterative Strategy in \mathbb{R}^2

points in \mathbb{R}^3 . However it is necessary to account for it in the distance function as a transformation-invariant attribute.

It is possible to define an iterative strategy I_{XY} in \mathbb{R}^3 by using the two algorithms previously defined as follows.

$$(P_1 = \text{ALIGN-T}_{\mathbf{XY}}(P, Q), P_2 = \text{ALIGN-T}_{\theta}(P_1, Q), P_3 = \text{ALIGN-T}_{\mathbf{XY}}(P_2, Q), P_4 = \text{ALIGN-T}_{\theta}(P_3, Q), \dots) \quad (4.12)$$

The rotation θ that is used in algorithm ALIGN-T_{θ} is performed around the center of mass of the point set P that is being rotated. The center of mass is computed without considering the transformation-invariant attribute of each point.

Iterative strategy $I_{\mathbf{XY}}$ can be used as basis to solve the optimal alignment problem in \mathbb{R}^3 . In fact iterative strategy $I_{\mathbf{XY}}$ involves 3 degree of freedom transformations: translation in a coordinate plane and rotation around the axis perpendicular to the coordinate plane. Iterative strategy $I_{\mathbf{xy}}$ provides a partial solution to the optimal point alignment problem in \mathbb{R}^3 because it uses only three out of the six degrees of freedom involved. Similarly the iterative strategies $I_{\mathbf{xz}}$ and $I_{\mathbf{yz}}$ can be defined. The same observations made for iterative strategy $I_{\mathbf{xz}}$ apply to iterative strategies $I_{\mathbf{xz}}$ and $I_{\mathbf{yz}}$. Now consider the following six possible distinct sequences of application of the iterative strategies previously defined.

$$\left\{ \begin{array}{l} I_1^3 = (I_{\mathbf{xy}}, I_{\mathbf{xz}}, I_{\mathbf{yz}}, \dots) \\ I_2^3 = (I_{\mathbf{xy}}, I_{\mathbf{yz}}, I_{\mathbf{xz}}, \dots) \\ I_3^3 = (I_{\mathbf{xz}}, I_{\mathbf{xy}}, I_{\mathbf{yz}}, \dots) \\ I_4^3 = (I_{\mathbf{xz}}, I_{\mathbf{yz}}, I_{\mathbf{xy}}, \dots) \\ I_5^3 = (I_{\mathbf{yz}}, I_{\mathbf{xy}}, I_{\mathbf{xz}}, \dots) \\ I_6^3 = (I_{\mathbf{yz}}, I_{\mathbf{xz}}, I_{\mathbf{xy}}, \dots) \end{array} \right. \quad (4.13)$$

Each of the I_i^3 uses all the six degrees of freedom involved in the optimal alignment problem in \mathbb{R}^3 . Hence each of the iterative strategies I_i^3 can be applied to the optimal

alignment problem in \mathbb{R}^3 . In order to assess the performance of the iterative strategies I_i^3 defined by Equations (4.12) and (4.13), in the first and second experiments they are applied to the optimal alignment problem in \mathbb{R}^3 .

In the first experiment, 1000 initial sets of 20, 40 and 80 attributed points inside a sphere of a fixed size were randomly generated. The transformation-invariant attributes were also generated randomly. Then a random transformation was applied to each of the 1000 sets, creating 1000 more sets of attributed points. Hence finally 3000 pairs of sets of attributed points were obtained. Consider all the pairs of sets consisting of one initial set and one corresponding additional sets created as explained previously. The iterative strategies I_i^3 were applied to each pair of attributed point sets, for $i = 1, 2, \dots, 6$, until convergence was reached. A total of 3000 instances were evaluated. The expected minimum distance corresponding to the optimal alignment computed among the sets of each pair is 0. Cases in which the optimal alignment was not found were handled using the same procedure as previously. A random transformation was applied to the initial set of the pair in order to create a different initial condition. Then the experiment was repeated with the different initial condition that had been obtained for those instances. This procedure was repeated until the optimal alignment was found or the limit of ten different initial conditions was reached. Out of the 3000 instances, the optimal alignment (i.e. distance = 0) was found in all of them.

In the second experiment, 1000 initial sets of 20 points were randomly generated and then again a random transformation was applied to each of them, creating 1000 more sets of points. This time, as it was done previously, before applying the iterative scheme 5%,

10%, 20% and 40% of the points were erased from one of the sets being compared. The erasing of the points was not expected to affect the alignment, as the distance function can be applied to sets of different cardinality. Out of the 4000 instances (i.e. 1000 instances for each distinct number of erased points), the optimal alignment was found in all of them.

In Figure 4.2 a histogram representing the number of converging and non-converging instances vs. the number of initial conditions used for both the first and the second experiment is shown.

For the point alignment problem in \mathbb{R}^3 under 6 DOF transformations we have investigated whether the iterative strategies I_i^3 are guaranteed to lead to a local minimum or not. For this purpose the Hessian of the distance function has been evaluated in correspondence of the outcomes that did not corresponded to the optimal alignment (i.e. distance = 0). Its positive definiteness cannot be guaranteed. Experiments have been carried out in order to assess the performance of this class of iterative strategies on the optimal point alignment problem in 3D. Out of the 757 evaluations of the Hessian, 6.87% of the cases were found not positive definite. Hence for the point alignment problem in \mathbb{R}^3 under 6 DOF transformations the Hessian is not guaranteed to be positive definite.

4.5.3 Tests On Iterative Strategies In \mathbb{R}^3 Using 3 Rotational DOFs

Consider the point alignment problem in \mathbb{R}^3 for 3 rotational DOFs. In this case, referring to the two sets of attributed points P and Q , P needs to be aligned with respect to Q using the transformation $\mathbf{T} = (\psi, \varphi, \theta)$. The three components of the transformation \mathbf{T} represent

the rotations around the coordinate axis X, Y, and Z respectively. The optimal alignment algorithm described in Section 3.5 can be used to provide partial solutions to the alignment problem defined above. Using the notations introduced in the previous subsection call ALIGN-T_θ the algorithm described in Section 3.5 if it uses rotations about the coordinate axis Z. Observe that the Z coordinate of each attributed point remains constant when the optimal alignment algorithm ALIGN-T_θ is applied to attributed points in \mathbb{R}^3 . However it is necessary to account for it in the distance function as a transformation-invariant attribute. The algorithms ALIGN-T_ψ and ALIGN-T_ϕ are defined in a similar way.

There are six possible distinct sequences I_{Ri}^3 of application of the lower dimension optimal alignment algorithms ALIGN-T_θ , ALIGN-T_ψ and ALIGN-T_ϕ . They are defined as follows.

$$\left\{ \begin{array}{l} I_{R1}^3 = (\text{ALIGN-T}_\theta, \text{ALIGN-T}_\phi, \text{ALIGN-T}_\psi, \dots) \\ I_{R2}^3 = (\text{ALIGN-T}_\theta, \text{ALIGN-T}_\psi, \text{ALIGN-T}_\phi, \dots) \\ I_{R3}^3 = (\text{ALIGN-T}_\phi, \text{ALIGN-T}_\theta, \text{ALIGN-T}_\psi, \dots) \\ I_{R4}^3 = (\text{ALIGN-T}_\phi, \text{ALIGN-T}_\psi, \text{ALIGN-T}_\theta, \dots) \\ I_{R5}^3 = (\text{ALIGN-T}_\psi, \text{ALIGN-T}_\theta, \text{ALIGN-T}_\phi, \dots) \\ I_{R6}^3 = (\text{ALIGN-T}_\psi, \text{ALIGN-T}_\phi, \text{ALIGN-T}_\theta, \dots) \end{array} \right. \quad (4.14)$$

Each sequence I_{Ri}^3 identifies an iterative strategy. In order to assess the performance of iterative strategies I_{Ri}^3 , they have been applied to the optimal point alignment problem in \mathbb{R}^3 under three 1 DOF rotations.

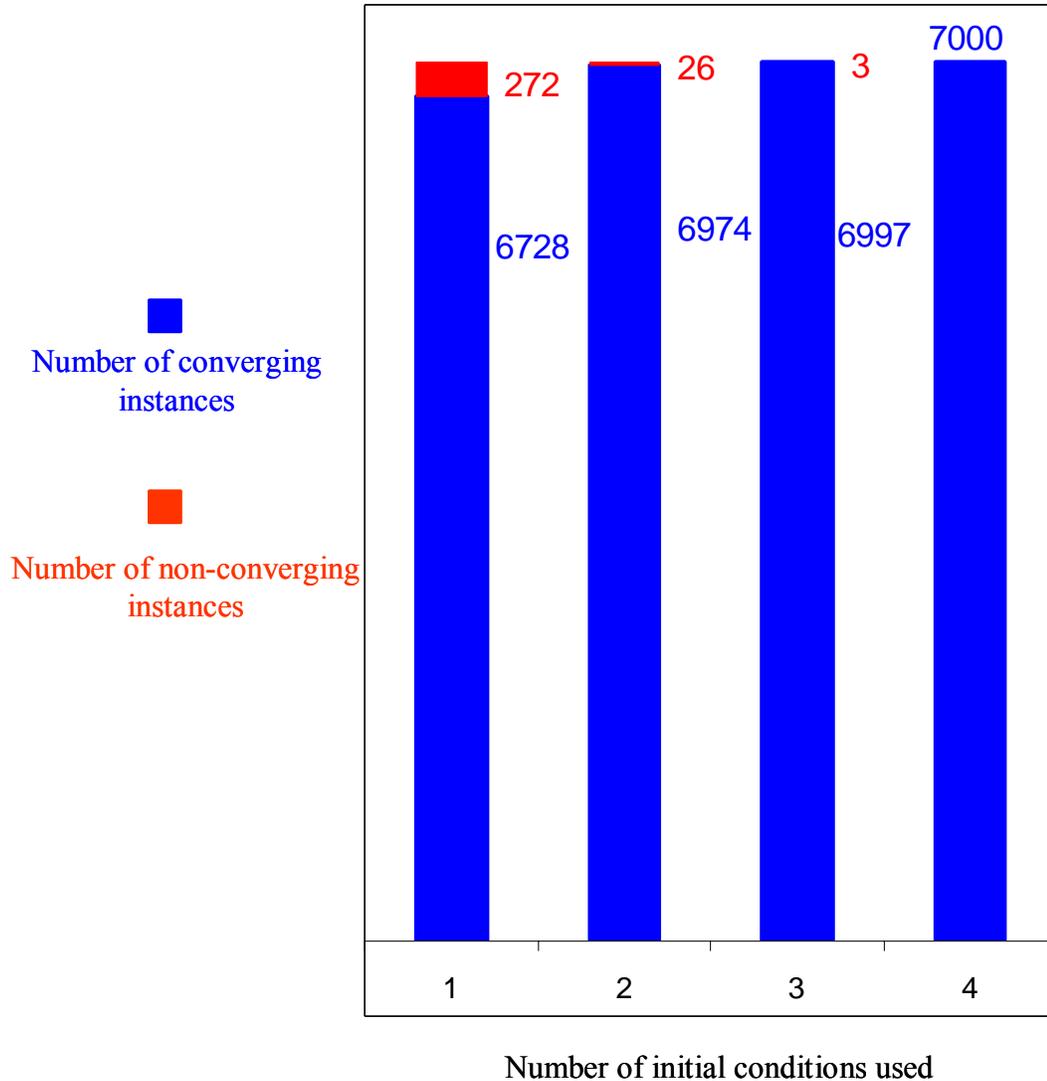


Figure 4.2: Histogram Showing the Number of Converging and Non-converging Instances Vs. The Number of Initial Conditions Used for Iterative Strategy I_i^3 in \mathbb{R}^3

In the experiment 1000 initial sets of 20 attributed points inside a sphere of a fixed size were randomly generated. The transformation-invariant attributes were also generated randomly. Then a random transformation was applied to each of the 1000 sets, creating 1000 more sets of attributed points. Hence finally 1000 pairs of sets of attributed points were obtained. Consider all the pairs of sets consisting of one initial set and one corresponding additional sets created as explained previously. For each pair of attributed

point sets two corresponding points were matched using a translation. Then the iterative strategies I_{Ri}^3 were applied to each pair of point sets, for $i = 1, 2, \dots, 6$, until convergence was reached. A total of 1000 instances were evaluated. The expected minimum distance corresponding to the optimal alignment computed among the sets of each pair is 0. Cases in which the optimal alignment was not found were handled using the same procedure as previously. A random transformation was applied to the initial set of the pair in order to create a different initial condition. Then the experiment was repeated with the different initial condition that had been obtained for those instances. This procedure was repeated until the optimal alignment was found or the limit of ten different initial conditions was reached. Out of the 1000 instances, the optimal alignment (i.e. distance = 0) was found in all of them. In Figure 4.3 a histogram representing the number of converging and non-converging instances vs. the number of initial conditions used for the third experiment is shown.

4.6 Summary

This chapter describes alignment algorithms based on iterative strategies in \mathbb{R}^2 and \mathbb{R}^3 .

The iterative strategies use optimal alignment algorithms based on partitioning of lower dimension transformation spaces. Extensive experiments have been carried out in order to evaluate the performance of the iterative strategies and to identify some of their characteristics. We have shown that iterative strategies are capable of producing optimal solutions in \mathbb{R}^2 and \mathbb{R}^3 .

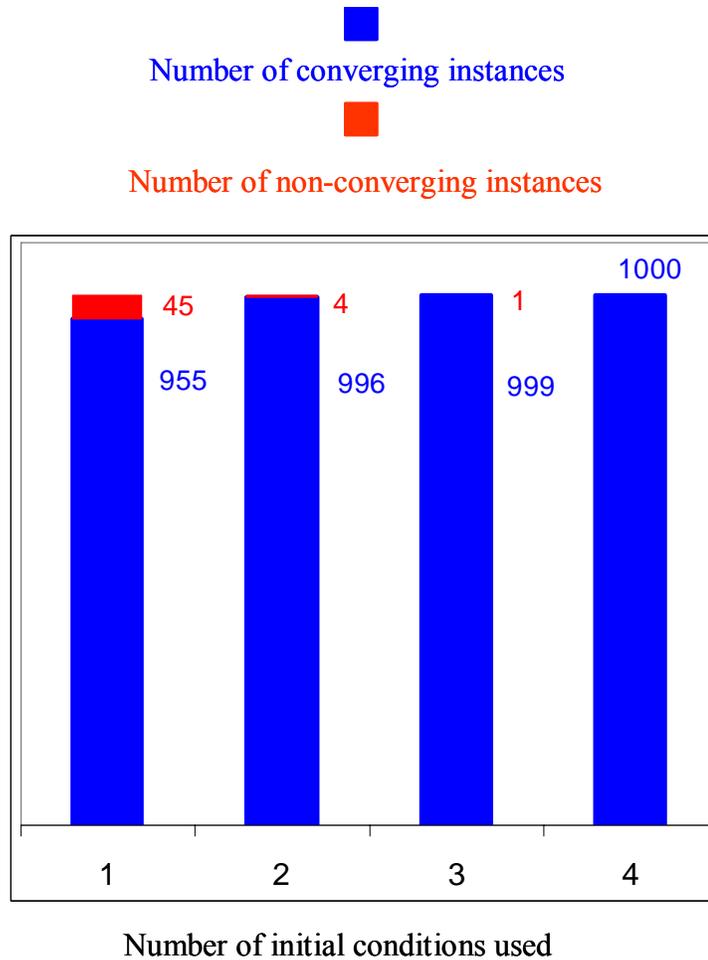


Figure 4.3: Histogram Showing the Number of Converging and Non-converging Instances vs. The Number of Initial Conditions Used for Iterative Strategies $I_{R_i}^3$ in \mathbb{R}^3

We have shown that an iterative strategy in \mathbb{R}^2 leads to a local minimum. We have designed iterative strategies and conducted experiments in \mathbb{R}^3 using 6 DOF and 3 DOF transformations. The experiments show that the number of local minima is low and hence few initial conditions are sufficient to find the optimal solution. Experiments were carried out by using randomly generated sets of attributed points. In theory, there could be pathological cases that are not represented in our experiments. In such pathological cases,

the performance of the iterative strategy might be significantly different from the performance observed in our experiments. However, we do not expect such pathological cases to be encountered in manufacturing applications. Hence, we believe that reasonable empirical evidence has been provided that iterative strategies can be used to find the optimal solution for point alignment problems.

Chapter 5: Feature-Based Similarity Assessment Algorithms

This chapter is organized as follows. Section 5.1 gives the motivation and identifies the goal of the chapter. Section 5.2 provides the necessary definitions and presents the problem formulation. Section 5.3 describes the machining feature-based similarity assessment algorithm in the case of single feature interpretation. The part of this algorithm that performs optimal alignment under one degree of freedom rotations is described in Section 5.4. Section 5.5 gives the results of the computational experiments that have been performed by considering single preferred feature interpretations. Section 5.6 presents an extension of the algorithm to a class of multiple feature interpretations. Finally, Section 5.7 presents the concluding remarks.

5.1 Motivation

For some manufacturing domains such as rapid prototyping, reasonably accurate estimates of cost can be generated by estimating the volume or weight of the part. However, for 3-axis machining, the accurate cost estimation is much more difficult. Cost for a machined part can be defined as a summation of material costs, setup costs, tooling costs, and operation costs. Material costs depend upon the cost of stock being used. Setup costs depend on how many setups are needed and fixturing methods used in each setup. Therefore, setup costs depend on how features are oriented in space and how they interact with each other to affect fixturing and introduce precedence constraints. Tooling costs depend on the tools being used. Therefore, tooling cost depends on machining feature types. Operation costs depend on the time taken to machine various features. Therefore, operation costs depend on feature types, dimensions, and tolerances.

Given a set of machining features belonging to a part, it is easy to estimate operation costs. However, it is difficult to estimate setup costs from the description of machining features alone. Setup costs not only depend upon the total number of feature access directions but also on the type of precedence constraints that exist among features and how each setup is fixtured. Currently there is no automated method for performing setup planning in commercially available process planning systems. Therefore, unless a detailed setup plan is manually developed, machining cost cannot be estimated accurately from the description of machining features alone. Therefore, currently cost estimation is done manually for machined parts if high accuracy is desired in cost estimates.

Accurate cost estimation can take anywhere from a few minutes to a few hours depending upon the expertise of the cost estimator and the complexity of the part. Based upon our conversations with human cost estimators, it appears that many of them implicitly use estimates from previously completed tasks to generate new quotes.

Manual cost estimation is inefficient, especially when the designer submits the 3D model over the Internet for getting quotes. One way to perform cost estimation is to search a database of previously machined parts and automatically locate parts similar to the newly designed part, so that the machining cost of the retrieved parts can be potentially used to estimate the cost of the new part. Figure 5.1(a) shows a newly designed part and Figure 5.1(b) shows a previously machined part that can be potentially used to estimate the cost of the new part. Thus, there is a need to develop a system that can assist the human cost estimators by quickly finding previously machined parts similar to the query part.

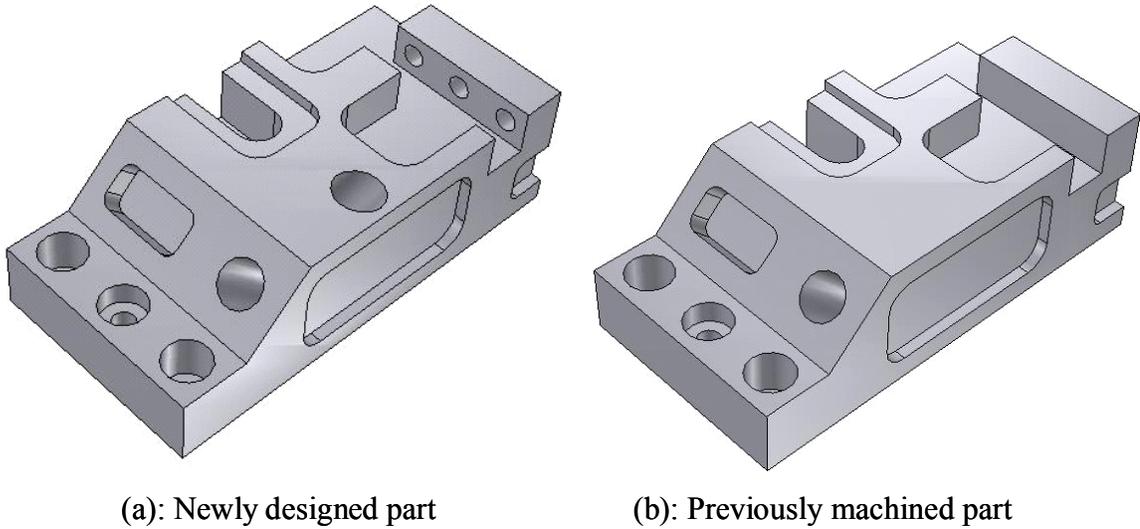


Figure 5.1: The Previously Machined Part (b) Can Be Potentially Used to Estimate the Cost of the Newly Designed Part (a)

In order to be considered similar, two sets of machining features not only need to be of similar number and of similar type but also the features need to be distributed in space in a similar manner. The reason behind this requirement is as following. If two sets of machining features are distributed in a similar way in the space, then they will have similar interactions from a setup point of view. Feature interactions that need to be considered during cost estimation are the spatial interactions that influence setups. If two sets of given machining features can be aligned in the space such that for every feature in the first set there is a corresponding feature in the second set, that is of the same type and with similar parameters and situated at the same place in the space, then we can implicitly ensure features in the two sets will have the same spatial interactions. This implicit similarity in spatial interactions ensures that two parts will have similar setups both in terms of operations and fixturing methods.

The cost of machined parts is determined by their machining features rather than their overall shapes. Hence feature-based techniques are the most suitable to estimate the cost of machined parts. Not all components of feature vectors play an equal role in determining similarity between the two parts. Based on the nature of the application, some components contribute significantly to similarity measures while others have virtually no effect on the similarity. Therefore, we utilize reduced feature vectors in determining the degree of similarity between two parts. Reduced feature vectors are defined in such a way that they only include feature components having large influence on similarity. In this chapter we will only focus on machining features defined for 3-axis machining centers. Most modern 3D CAD/CAM systems (e.g., Pro/Engineer, Unigraphics, etc.) allow users to define machining features.

This chapter introduces reduced feature vector sets that are suitable for a cost estimation application and describes the algorithms for the alignment of the reduced feature vectors of a database part and the query part. Reduced feature vectors (RFVs) for a part are usually defined using a specific coordinate system. In order to correctly measure the distance between two given sets of reduced feature vectors, we need to transform one set with respect to the other using rigid body transformations such that we get the minimum distance between the two sets. We refer to this step as the alignment step in this chapter.

The output of the algorithms is a rank ordering of the machined parts in a database based on the degree of similarity with respect to the query part. Each retrieved part will have a distance value with respect to the query part. The larger the distance, the less similar the retrieved part is to the query part. The cost of machining the query part can be

estimated by using the cost of previously machined parts that have a very small distance value. The features of the query part may be interpreted differently corresponding to different possible machining operations. The algorithms presented in this chapter have been extended to account for existence of multiple interpretations of machining features. These extensions are applicable to parts for which individual feature interpretations are independent of each other.

5.2 Background And Problem Formulation

5.2.1 Machining Features

The key drivers for the machining cost of a prismatic part are the number of setups, the number of tool changes and the machining operation cost. Setup is any changeover activity that is necessary to change the part orientation. For 3-axis machining, the number of setups depends on the relative orientation of the feature access vectors. The *access vector* of a machining feature is a unit vector that gives the direction along which the tool moves in order to machine the desired feature. The *orientation vector* of a machining feature is a unit vector that gives the direction along which the tool moves in order to give the desired orientation to the feature. For some features, such as holes, this vector has no technical meaning. In this chapter we make the assumption that feature positions do not play an important role in determining the machining cost of a prismatic part. This is a reasonable assumption as long as the parts being considered do not have thin sections. Feature positions play an important role in determining fixturing plans for part with thin sections.

Access vectors are modeled using unit vectors. A part having two differently oriented features will require two setups while a part having two features with the same access

direction will require only a single setup. Tool changes are determined by the type of feature that has to be machined. A part having a hole and a pocket with the same access direction will require a tool change: the drill used to machine the hole needs to be replaced by a mill to machine the pocket. The machining operation cost increases with the volume of the feature to be machined. It also depends on the machining tolerance of features.

The features that have been considered include pockets, open slots, steps and holes. Figure 5.2 shows all of the features considered. Open slots could be of three types: slot, notch and through slot. They are shown in figures 5.2(b), 5.2(c) and 5.2(d) respectively with their access and orientation vectors. The hole is an example of a feature where orientation vector does not need to be defined because of symmetry. Each of the previously listed features can be completely characterized by providing the values of certain parameters such as height, width, length, and radius.

Figures 5.3(a), 5.3(b), 5.3(c), 5.3(d) and 5.3(e) show parts A, B, C, D and E with their corresponding feature access and orientation vectors. In Figure 5.4, the access vectors of the parts are shown. Parts A and C are considered dissimilar from the cost estimation point of view, and so are parts A and E, because the feature access directions cannot be aligned. The feature access directions of parts A and B and parts A and D can be aligned exactly. However part A is more similar to part B than to part D from a cost estimation point of view, because the types of features of parts A and B match.

RFVs of a feature of a part consist of those feature components that are important from a machining effort point of view. RFVs are mathematically equivalent to attributed points on the unit sphere. Figure 5.5 shows an example of RFVs of a part and their

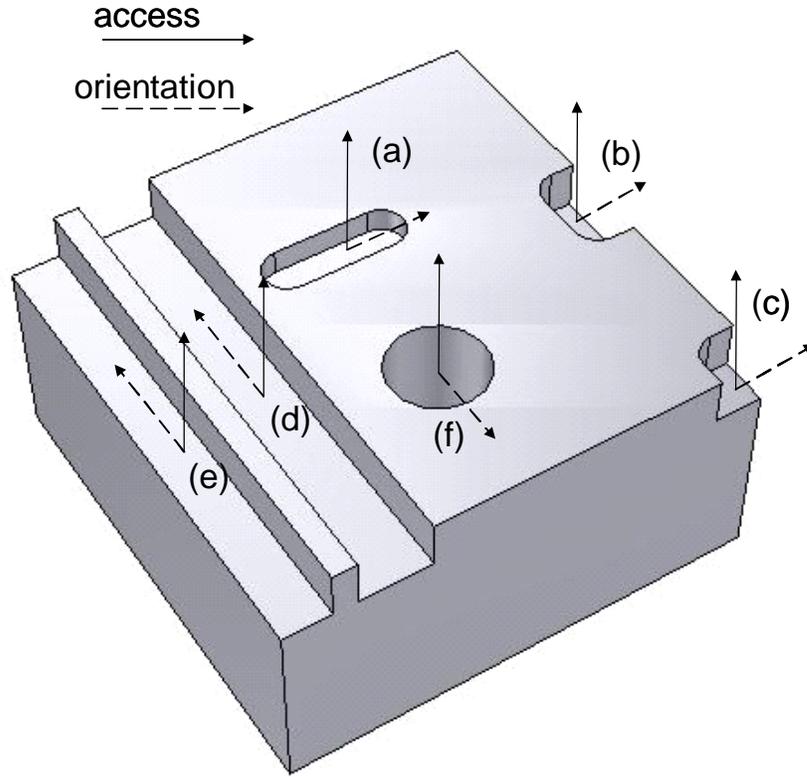


Figure 5.2: Features Considered With Access and Orientation Vector: (a) Pocket (b) Slot (c) Notch (d) Through Slot (e) Step (f) Hole

equivalent attributed points on the unit sphere. Therefore, the problem of aligning two sets of RFVs is equivalent to the problem of aligning attributed points on the unit sphere. Hence in this chapter we will use terms RFVs and attributed points on unit sphere interchangeably.

5.2.2 Distance Function For Similarity Assessment

Let $p \in P$ and $q \in Q$ be the two sets of RFVs corresponding to parts M_P and M_Q . Then, P and Q are compared using the following distance function, which has the same expression as the one defined in Equation (3.1).

$$\bar{d}(P, Q) = \frac{\sum_{i=1}^n \min_{q \in Q} d(p_i, q)}{n} \quad (5.1)$$

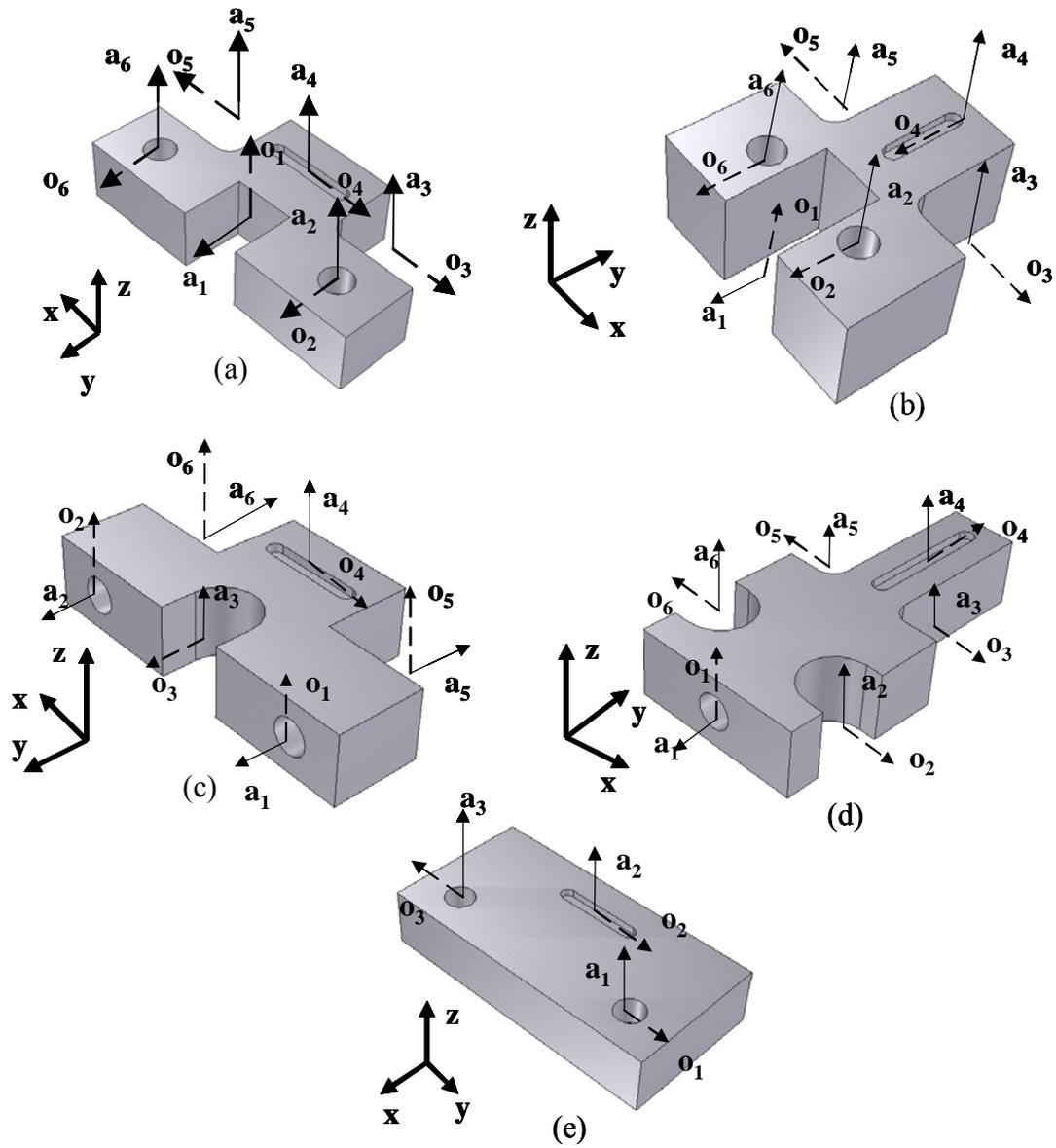


Figure 5.3: Machined Parts With Access and Orientation Vectors

As observed previously, the key drivers for the machining cost of a prismatic part are the number of setups, the number of tool changes and the machining operation cost. The distance function between two RFVs $p \in P$ and $q \in Q$ needs to account for them. Each RFV is represented by using six components. Specific components are x^p , y^p , z^p , $V(p)$,

$\varepsilon(p)$, $n(p)$. The first three components x^p , y^p and z^p represent the orientation of the RFV p , and are transformation-dependent. The other three components $V(p)$, $\varepsilon(p)$ and $n(p)$ are transformation-invariant. The fourth component $V(p)$ represents the normalized volume of the RFV. The volumes are normalized using the average volume of all the features of the parts being compared. The fifth component $\varepsilon(p)$ represents the normalized dimensional tolerance. In this chapter only dimensional tolerances are taken into account. The dimensional tolerances are normalized using the dimensional tolerance value occurring most often in the database. The sixth component $n(p)$ is referred to as the group cardinality of p . The reason behind including this component is the following. Two different features may have the identical first five components in the reduced feature vectors, making it difficult to distinguish between them. This may cause a problem in the use of asymmetric distance functions. Therefore, we group such features into a single composite feature. In order to handle composite features, we have introduced the group cardinality as the sixth component. If no grouping has been performed then the value of $n(p)$ is set to 1. Figure 5.6 shows an example of a composite feature.

To increase the efficiency of comparison and avoid the problem described above, parts that do not have a comparable value to the number of features of the query part are discarded. This pruning step ensures that parts with a comparable number of features are assessed for similarity, so that the retrieved parts have a cost comparable to the query part.

The distance function between RFVs $p \in P$ and $q \in Q$ is defined as follows.

$$d(p, q) = (x^p - x^q)^2 + (y^p - y^q)^2 + (z^p - z^q)^2 + (1 - \delta(p, q)) [w_V (V(p) - V(q))^2 + w_\varepsilon (\varepsilon(p) - \varepsilon(q))^2 + w_C (n(p) - n(q))^2] + w_T \delta(p, q) \quad (5.2)$$

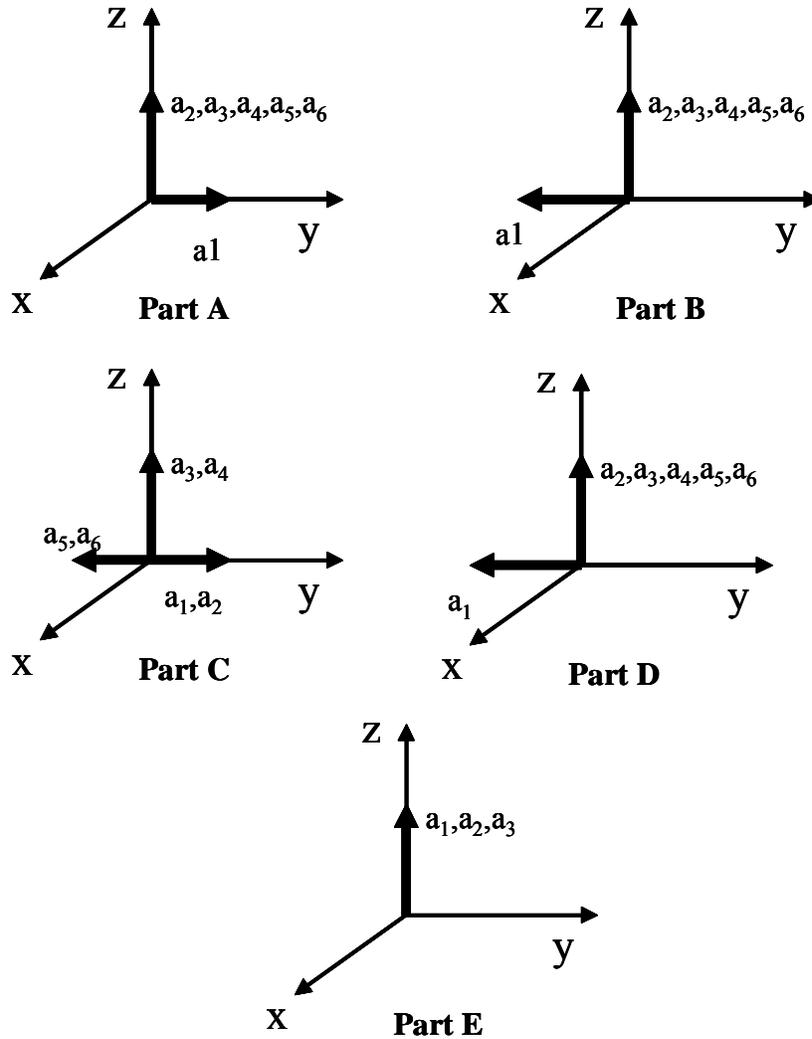


Figure 5.4: Access Vectors for the Parts of Figure 5.3

The first three terms account for the difference in position between p and q and relate to the number of tool setups. The last four terms account for the difference in the transformation-invariant attributes that are considered. Specifically, the fourth term accounts for the difference in volume between the corresponding features and relates to the machining operation cost. The fifth term accounts for the difference in dimensional tolerance between the corresponding features and relates to machining operation cost. The sixth term of the distance function accounts for the difference in group cardinality

between the RFVs corresponding to p and q and relates to the machining operation cost. Finally, the seventh term accounts for the difference in type between the corresponding RFVs and relates to the number of tool changes. The term δ has the following expression.

$$\begin{cases} \delta(p, q) = 0 & \text{if type of } p \text{ is equal to type of } q \\ \delta(p, q) = 1 & \text{if type of } p \text{ is different from type of } q \end{cases}$$

So all the key drivers for the machining cost of a prismatic part are accounted for. The volume, tolerance and group cardinality terms are multiplied by the quantity $(1 - \delta(p, q))$, so that when the types of features p and q do not match, volume and tolerance terms are not considered. The quantities w_V , w_ϵ , w_C and w_T represent the weights given by the user to the volume, tolerance, group cardinality and type terms respectively. The distance function can be customized by: (a) changing the weight associated with each of the terms in the distance function, (b) considering additional transformation-invariant feature parameters as needed.

The distance function defined in Equation (5.1) is the measure of similarity between parts M_P and M_Q , represented by two sets of RFVs; the smaller the value of the distance given by Equation (5.1), the more similar are the parts M_P and M_Q .

5.2.3 Problem Statement

The input to the system described in this chapter is a database of previously machined parts whose cost is already known and a newly designed part whose machining cost is to be estimated. The system outputs previously machined parts similar to the query part.

Each part has been modeled in its own coordinate system. Therefore, we need to align the parts using rigid body transformations before computing the distance. The parts are represented by using two sets of RFVs. Hence, as stated previously, the problem of

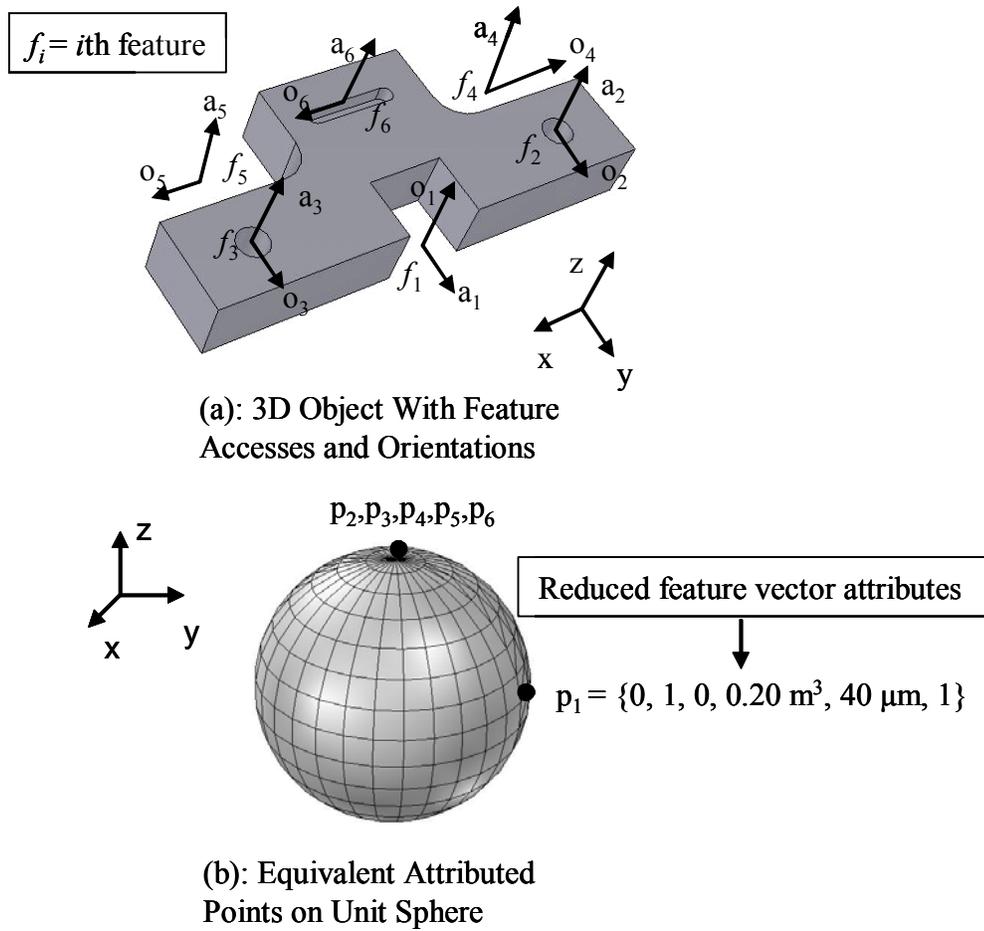


Figure 5.5: Equivalence Between Reduced Feature Vectors and Attributed Set of Points on Unit Sphere

aligning two sets of RFVs is equivalent to the problem of aligning attributed points on the unit sphere. To align the two sets of attributed points on the unit sphere, one set has to be moved with respect to the other set. Rigid body transformation of a set of attributed

points on the surface of the unit sphere involves three degrees of freedom. The distance function has to be minimized over all of the possible configurations of the moving attributed point set with respect to the stationary one. The transformation matrix for the three degrees of freedom transformation is given by $\mathbf{R} = \mathbf{R}(\theta, \varphi, \psi)$ where θ , φ , and ψ are the three degrees of freedom considered. Assuming that P is the moving set, the transformed set P can be written as $\mathbf{R}P$. The distance function defined in Equation (5.1) can then be written as

$$\vec{d}(\mathbf{R}P, Q) = \vec{d}(\mathbf{R}P, Q)(\theta, \varphi, \psi) \quad (5.3)$$

This chapter introduces an algorithm to find the best alignment between two sets of attributed points on the unit sphere by transforming one attributed point set such that the distance function is minimized.

In general, a query part can have multiple feature interpretations based on how access directions for machining features are selected [Gupt95]. Sections 5.3, 5.4, and 5.5 describe an algorithm for feature-based shape similarity assessment of parts having a single preferred interpretation. Section 5.6 extends this algorithm to deal with query parts with multiple possible interpretations.

5.3 Computing Similarity For Query Parts With Single Preferred Feature Interpretations

As mentioned previously, aligning two sets of attributed points on the unit sphere is a three degree of freedom problem. For estimating the cost of machining the new part based on an existing part, the two parts should have at least one feature of the same type. If the two parts have no common features then one part cannot be used to estimate the cost of the other and hence the part needs to be pruned. Thus, two degrees of freedom in

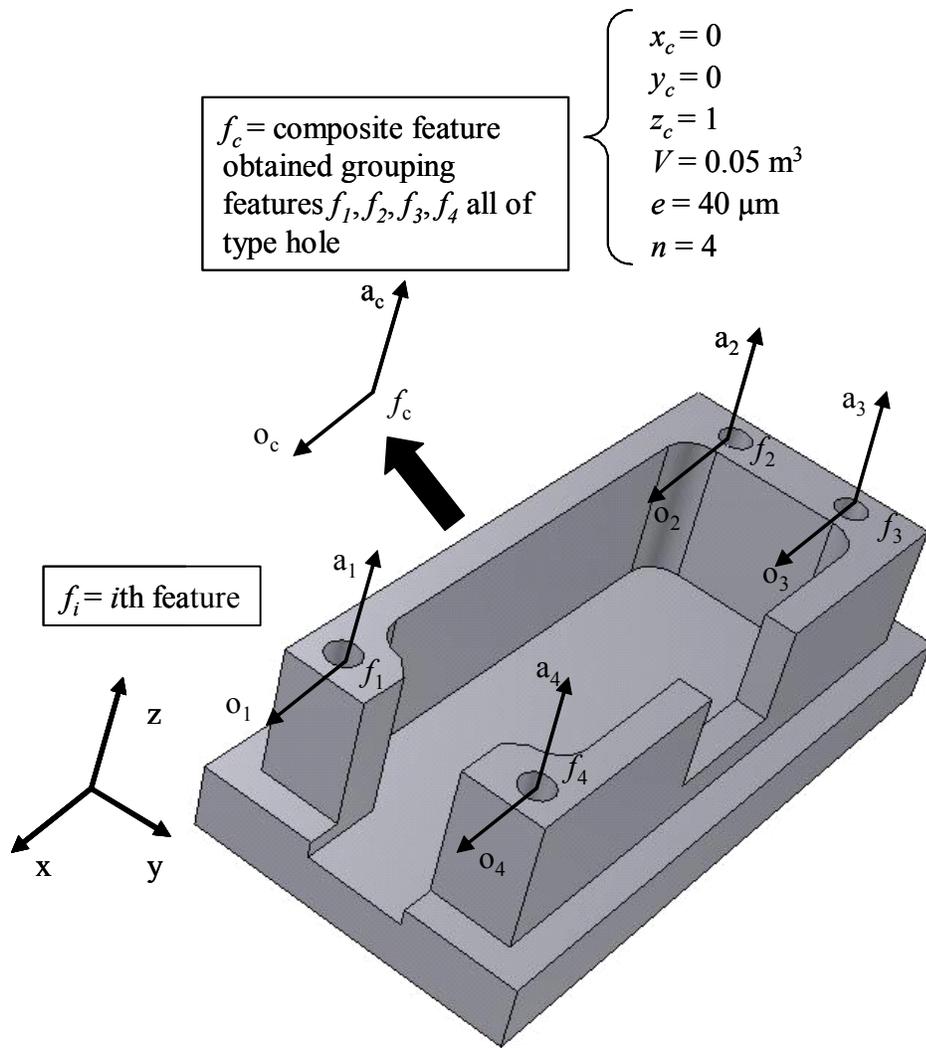


Figure 5.6: Example of Composite Feature

this problem can be constrained by considering combinations of features. Each feature of M_P is aligned with every feature of M_Q having the same type. The total number of alignments that need to be performed is not large. This is because the number of combinations of features of the two parts of the same type is not significantly large, as most of the reasonably complex mechanical parts have fewer than 100 instances of composite features.

Consider a pair of RFVs $p_i \in P$ and $q_j \in Q$ of the same type equivalent to two attributed points on the unit sphere. Initially, the rotation represented by the matrix \mathbf{R}_{ij} is applied to the two sets P and Q such that $p_i \in P$ and $q_j \in Q$ are aligned. Then the two sets P and Q are rotated again such that $p_i \in P$ and $q_j \in Q$ are aligned with the Z axis. Finally the set P is rotated with respect to Q about the Z axis. The rotation value θ for which the distance function computed between P and Q is minimized is found using the algorithm COMPUTETHETA (described in Section 5.4). The value of the distance function corresponding to the value θ is the minimum value of the distance function for a particular RFV pair alignment. Now, the next alignment is considered and the procedure is repeated. The output is the minimum value of the distance over all the RFV pair alignments. The overall algorithm is given below.

Algorithm: COMPUTESIMILARITYMEASURE

Input:

- Parts M_P and M_Q .

Output:

- Degree of similarity between M_P and M_Q based on the distance function defined in Equation (5.1).

Steps:

1. Let P and Q be the RFV sets corresponding to M_P and M_Q .
2. Initialize $d_{min} = \text{Infinity}$.
3. For each RFV p_i of P , do the following.
 - a. Initialize $(d_{min})_i = \text{Infinity}$.
 - b. For each RFV q_j of Q , do the following.

-
- i. If $p_i \in P$ and $q_j \in Q$ are of the same type, rotate P using the transformation matrix $\mathbf{R}_{i,j}$ such that p_i aligns with q_j .
 - ii. Else go to next value of j in Step 3b.
 - iii. Rotate P and Q using transformation matrix \mathbf{R}_z such that p_i and q_j align with Z axis.
 - iv. Compute $(\theta_{min})_{i,j}$ rotation about Z axis that minimizes the distance function and the corresponding distance function value $(d_{min})_{i,j}$ using the algorithm COMPUTETHETA.
 - v. If $(d_{min})_i$ is greater than $(d_{min})_{i,j}$ then $(d_{min})_i = (d_{min})_{i,j}$.
- c. If d_{min} is greater than $(d_{min})_i$ then $d_{min} = (d_{min})_i$.
2. Return d_{min} .
-

5.4 Finding The Optimal Alignment Under One Degree Of Freedom

The algorithm COMPUTETHETA finds the angle θ that minimizes the distance function given by Equation (5.1) between two sets of RFVs on the unit sphere. The angle θ represents a rotation around a fixed axis: the algorithm solves the one degree of freedom problem. The one independent variable of the problem is the rotation θ applied to one of the two sets. The overall algorithm is given below.

Algorithm: COMPUTETHETA

Input:

- Sets P and Q of RFVs.

Output:

- Angle θ_{min} that minimizes the distance function defined in Equation (5.1).
-

Steps:

- a. Partition the theta range $[0, 2\pi]$ into theta intervals such that the closest neighbor $q_j \in Q$ to each RFV $p_i \in P$ is invariant in each interval using the algorithm FINDINVARIANTCLOSESTNEIGHBORS.
- b. Within each theta interval c obtained from Step a compute the value of the rotation $\theta(c)$ that minimizes the distance function defined in Equation (5.1) for interval c .
- c. Find interval c^* such that the distance function defined in Equation (5.1) reaches the minimum value over all the intervals obtained in Step a.
- d. Return the corresponding value $\theta_{min} = \theta(c^*)$ of the rotation for the interval c^* found in Step c.

Note that many steps of algorithms COMPUTETHETA and FINDINVARIANTCLOSESTNEIGHBORS defined in this chapter are very similar to the steps of algorithms ONEDOALIGNMENT and FINDINVARCLOSESTNEIGHBORSFOR1DOFRROT described in Chapter 3. However in this chapter RFVs on the unit sphere are aligned, while in Chapter 3 attributed points in \mathbb{R}^2 were aligned. Hence there are some substantial differences.

In the next subsections the steps of algorithms COMPUTETHETA and FINDINVARIANTCLOSESTNEIGHBORS will be described.

5.4.1 Step a: Building The Set Of Theta Intervals For The RFVs Of Set P

To compute the distance value in Equation (3.1), the closest neighbor $q_j \in Q$ to each $p_i \in P$ needs to be determined. The closest neighbor $q_j \in Q$ to each $p_i \in P$ changes with the rotation of set P with respect to set Q . Thus, the closest neighbors for each $p_i \in P$ need to

be obtained by taking into account the rotation θ around the fixed axis as explained in the previous section. It is necessary to know, for each value of the rotation θ , the closest RFV $q_j \in Q$ to each RFV $p_i \in P$. The closest neighbor to each RFV of P changes only at specific values of θ . Thus, the theta range $[0, 2\pi]$ can be partitioned into a set of theta intervals within which the closest neighbor to each RFV of P is known and invariant. The following algorithm is used for this purpose.

Algorithm: FINDINVARIANTCLOSESTNEIGHBORS

Input:

- Sets P and Q of RFVs.

Output:

- Set of theta intervals and for each interval the closest neighbor to every RFV of P from set Q .

Steps:

1. For each RFV p_i of P , do the following.
 - a. For each possible pair of distinct RFVs q_k and q_l of Q , do the following. Partition the theta range $[0, 2\pi]$ into subintervals within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. The partitioning is performed by intersecting the unit sphere on which p_i, q_k and q_l are located with a plane whose locus is such that $d(p_i, q_k) = d(p_i, q_l)$, where d is the distance function defined in Equation (3.2). This step will be described in more detail after the description of the overall algorithm.
 - b. Overlap the intersecting subintervals obtained in Step 1.a so that the range $[0, 2\pi]$ is further partitioned into a set of intervals.

-
- c. For each interval being obtained in Step 1.b, do the following. Using the closest neighbors being obtained in Step 1.a, find the RFV q_j of Q such that $d(p_i, q_j)$ is minimum over all the RFVs of Q .
2. Overlap the set of intersecting intervals being obtained in Step 1 for each RFV p_i of P . Within the set of intervals being obtained the closest neighbor to every RFV of P from set Q is invariant and known
-

The algorithm described previously yields the set of theta intervals for the RFVs of P . In the next paragraphs Step 1.a and Step 2 will be explained in detail.

In Step 1.a, the closest neighbors for each RFV $p_i \in P$ need to be obtained by using the distance function defined in Equation (5.2). The distance function accounts for relevant feature attributes. The transformation-invariant attributes need to be considered. First let us consider a case where the RFVs have identical transformation-invariant attributes. As shown in Figure 5.7, the dotted circle represents the trajectory of p_1 of P . Consider two RFVs q_1 and q_2 of Q on the unit sphere. Along a portion of this trajectory $d(p_1, q_1) < d(p_1, q_2)$ and along the remaining portion $d(p_1, q_1) > d(p_1, q_2)$. The procedure to obtain the theta intervals such that the closest neighbor is invariant is as follows. Consider a plane π_{12} through the center of the unit sphere which represents the locus of the points whose distance from RFV q_1 of Q is the same as the one from RFV q_2 of Q . Consider also the circle C_1 representing the trajectory of RFV p_1 of P around the fixed axis. The plane and the circle are intersected, obtaining two points on the circle. The circle corresponds to the theta range $[0, 2\pi]$ and the two intersection points correspond to the extreme values of the two theta intervals being obtained, as shown in Figure 5.7. Within each interval either $d(p_1, q_1) < d(p_1, q_2)$ or $d(p_1, q_1) > d(p_1, q_2)$ and the closest

neighbor to p_1 is known. Now let us consider the case where reduced feature vectors have different transformation-invariant attributes. Let Δw_{11}^2 be the difference between the transformation-invariant attributes in $d(p_1, q_1)$ and let Δw_{12}^2 be the difference between the transformation-invariant attributes in $d(p_1, q_2)$. Let $\Delta w_{11}^2 < \Delta w_{12}^2$ and $\Delta w^2 = \Delta w_{12}^2 - \Delta w_{11}^2$. In this case it is necessary to locate a plane π'_{12} such that $d(p_1, q_1) = d(p_1, q_2)$ using the distance function defined in Equation (5.2). Because of the presence of transformation-invariant attributes, the plane π'_{12} will no longer be the plane that is located at the same distance from RFVs q_1 and q_2 of Q . As shown in Figure 5.8, the plane will be offset by α in the direction of the point having the smaller value Δw_{ij} , in this case q_2 . The value of α is defined as follows.

$$\alpha = \frac{\Delta w^2}{2H} \quad (5.4)$$

where H is equal to the Euclidean distance between q_1 and q_2 . Depending on the value of Δw and H , it is possible that the value of the offset α is such that the plane does not intersect the circle at all. In this case, the theta range $[0, 2\pi]$ will not be divided into intervals, and it will be either $d(p_1, q_1) < d(p_1, q_2)$ or $d(p_1, q_1) > d(p_1, q_2)$ throughout the theta range $[0, 2\pi]$. In Appendix B the value of α defined in Equation (5.4) will be derived.

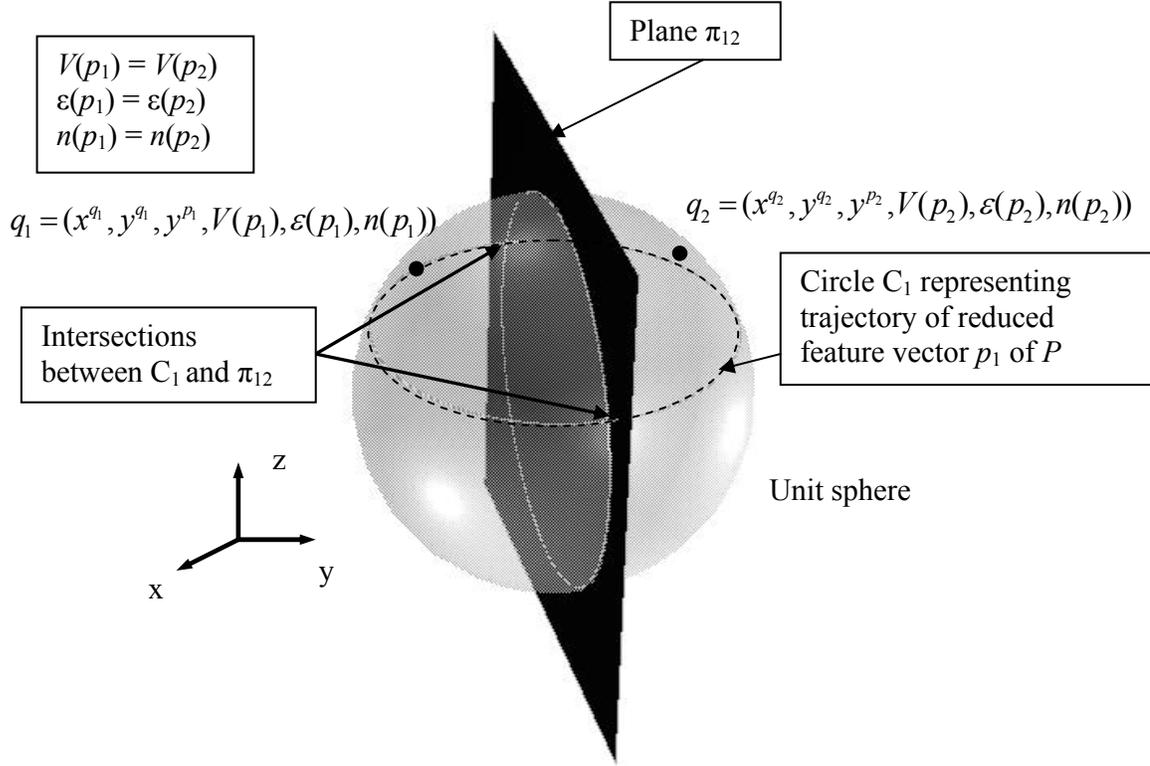


Figure 5.7: Transformation-invariant Attributes Are The Same For Each Reduced Feature Vector: The Two Intersection Points Between Circle C_1 and Plane π_{12} Represent The Extreme Values Of The Theta Intervals

Observe that Step 1 of the algorithm `FINDINVARIANTCLOSESTNEIGHBORS` yields the closest neighbors for each RFV of P separately. A set of theta intervals is built for a particular RFV $p_i \in P$ such that in each interval the closest RFV of Q to p_i is known. In Figure 5.9 the set of theta intervals within the range $[0, 2\pi]$ for the RFV $p_i \in P$ is shown. Thus several sets of theta intervals are obtained, one for each RFV of P . The overlapping of the sets of theta intervals being performed in Step 2 yields the set of theta intervals for the RFVs of P . Within each of the intervals the distance given by Equation (5.1) can be minimized using closed form mathematical formulae. The only independent variable in the formulae is rotation θ . The single sets of theta intervals for each RFV of P are combined into the set of theta intervals for the RFVs of P by overlapping so that the

resulting range $[0, 2\pi]$ is further partitioned into intervals. Each of the resulting intervals is obtained from the intersection of the intervals of the initial sets of intervals. Figure 5.10 shows two sets of intervals that are overlapped. One set of intervals is the set of theta intervals of RFV p_1 of set P (see Figure 5.10(a)), the other one is the set of theta intervals of RFV p_2 of set P (see Figure 5.10(b)). The interval c , indicated in Figure 5.10(c) by an arrow point, is clearly contained in one of the intervals of each of the two sets of theta intervals that have been overlapped. As shown in Figure 5.10(a) and Figure 5.10(b), the intervals c_1 and c_2 overlap to generate interval c . Thus, interval c represents a region in the set of theta intervals for the RFVs of P . Within c , q_1 is the closest neighbor to p_1 and q_2 is the closest neighbor to p_2 . Each point of c corresponds to a transformation applied to the set of RFVs P while Q is fixed. Thus, within any interval of the set of theta intervals for the RFVs of P , the closest RFV of Q to each RFV in P is known. The distance function defined in Equation (5.1) can now be computed for each interval. The distance function defined in Equation (5.1) for each interval can be expressed as a function of the coordinates (x, y, z) of the RFVs of P and Q . Co-ordinates of P and Q can be expressed as a function of θ , which is the angle of rotation. Thus the distance function defined in Equation (5.1) is expressed as a function of θ as explained in the next subsection.

5.4.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval

The location (x^p, y^p, z^p) of an attributed point p on the unit sphere can be represented by two angles: θ and ϕ . Let θ^{p_i} and ϕ^{p_i} be the known angle values for RFV $p_i \in P$ before applying algorithm COMPUTETHETA. Similarly, let ϕ^{q_j} and θ^{q_j} be the known angle values for RFV $q_j \in Q$ before applying algorithm COMPUTETHETA. These angle values

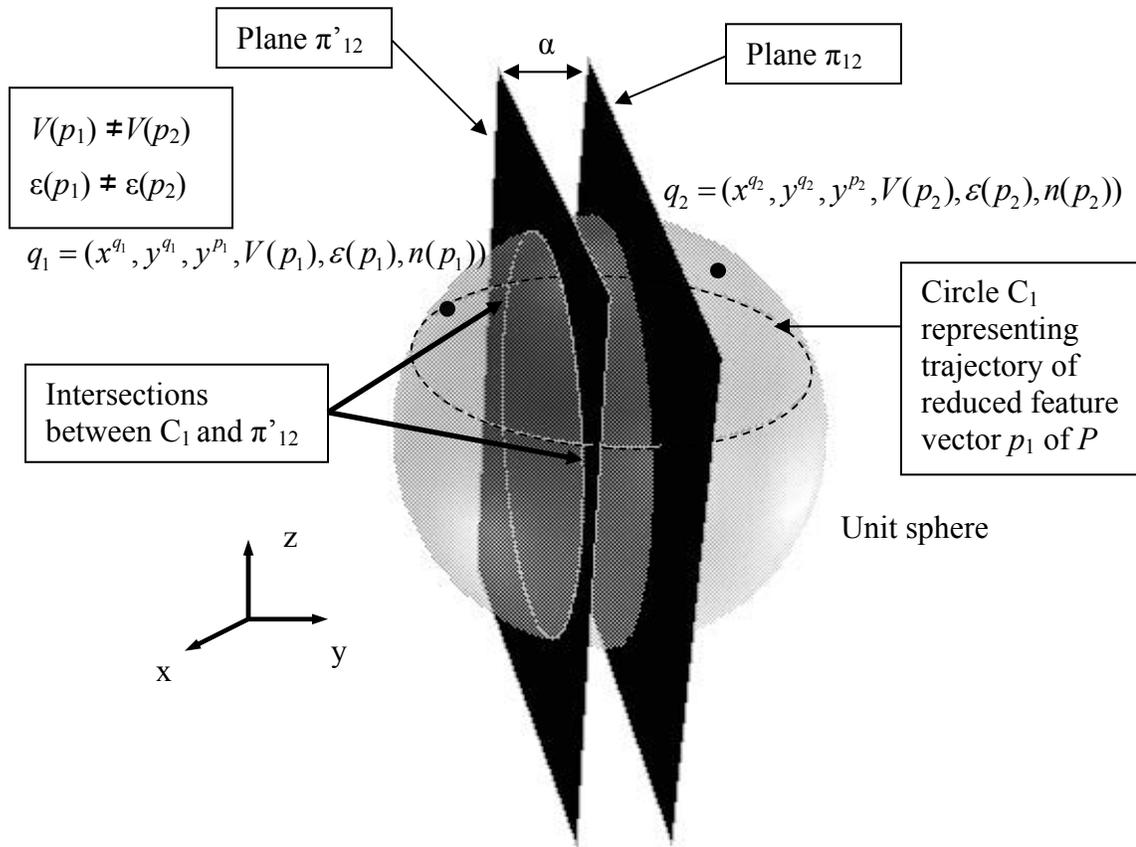


Figure 5.8: Transformation-invariant Attributes Are Different: The Two Intersection Points Between Circle C_1 and Plane π'_{12} Represent The Extreme Values Of The Theta Intervals

refer to the positions of RFVs of P and Q after the initial alignment described in Section 5.3.

In the previous subsection, the set of theta intervals for all the RFVs of P was built by overlapping the single sets of theta intervals of each RFV. The range $[0, 2\pi]$ is thus partitioned into a number of intervals. Within each interval the closest RFV in Q to each of the RFV in P is known. The following definitions, valid within each single interval, will be used.

$$\left\{ \begin{array}{l} x^{q_j}(i) = x \text{ coordinate of the closest RFV } q_j(i) \in Q \text{ to the RFV } p_i \in P \\ y^{q_j}(i) = y \text{ coordinate of the closest RFV } q_j(i) \in Q \text{ to the RFV } p_i \in P \\ z^{q_j}(i) = z \text{ coordinate of the closest RFV } q_j(i) \in Q \text{ to the RFV } p_i \in P \\ \theta^{q_j}(i) = \theta \text{ angle of the closest RFV } q_j(i) \in Q \text{ to the RFV } p_i \in P \\ \varphi^{q_j}(i) = \varphi \text{ angle of the closest RFV } q_j(i) \in Q \text{ to the RFV } p_i \in P \end{array} \right. \quad (5.5)$$

Consider a single theta interval and a moving RFV $p_i \in P$. Let θ be the rotation applied to the RFVs of set P , θ^{p_i} and φ^{p_i} the angles of p_i previously defined. Then,

$$\left\{ \begin{array}{l} x^{p_i}(\theta) = \cos(\theta^{p_i} + \theta) \cos(\varphi^{p_i}) \\ y^{p_i}(\theta) = \sin(\theta^{p_i} + \theta) \cos(\varphi^{p_i}) \\ z^{p_i} = \sin(\varphi^{p_i}) \end{array} \quad \forall \text{ point } p_i \in P \right. \quad (5.6)$$

On the other hand, for a fixed RFV $q_j(i) \in Q$ closest to $p_i \in P$:

$$\left\{ \begin{array}{l} x^{q_j}(i) = \cos(\theta^{q_j}(i)) \cos(\varphi^{q_j}(i)) \\ y^{q_j}(i) = \sin(\theta^{q_j}(i)) \cos(\varphi^{q_j}(i)) \\ z^{q_j}(i) = \sin(\varphi^{q_j}(i)) \end{array} \quad \forall \text{ point } q_j(i) \in Q \right. \quad (5.7)$$

Within a single interval, it is necessary to compute $\bar{d}(P, Q)$ as a function of the transformation θ .

$$\bar{d}(\mathbf{R}P, Q)(x^{p_i}(\theta), y^{p_i}(\theta), z^{p_i}) = \frac{\sum_{i=1}^n \left\{ \begin{array}{l} \{ [x^{p_i}(\theta) - x^{q_j}(i)]^2 + [y^{p_i}(\theta) - y^{q_j}(i)]^2 + \\ + [z^{p_i} - z^{q_j}(i)]^2 \} + w_T \delta(p_i, q_j(i)) + \\ + (1 - \delta(p_i, q_j(i))) [w_V (V(p_i) - V(q_j(i)))^2 + \\ + w_\varepsilon (\varepsilon(p_i) - \varepsilon(q_j(i)))^2 + w_C (n(p_i) - n(q_j(i)))^2] \end{array} \right\}}{n} \quad (5.8)$$

Using the notations introduced in (5.5), (5.6), and (5.7), Equation (5.8) can be simplified to,

$$\bar{d}(\theta) = \frac{\sum_{i=1}^n \left\{ \begin{aligned} & \left[\left(\cos(\theta^{p_i} + \theta) \cos(\varphi^{p_i}) - \cos(\theta^{q_j(i)}) \cos(\varphi^{q_j(i)}) \right)^2 + \right. \\ & \left. \left(\sin(\theta^{p_i} + \theta) \cos(\varphi^{p_i}) - \sin(\theta^{q_j(i)}) \cos(\varphi^{q_j(i)}) \right)^2 + \right. \\ & \left. \left(\sin(\varphi^{p_i}) - \sin(\varphi^{q_j(i)}) \right)^2 \right] + \\ & + w_r \delta(p_i, q_j(i)) + (1 - \delta(p_i, q_j(i))) [w_v (V(p_i) - V(q_j(i)))^2 + \\ & + w_\varepsilon (\varepsilon(p_i) - \varepsilon(q_j(i)))^2 + w_c (n(p_i) - n(q_j(i)))^2] \end{aligned} \right\}}{n} \quad (5.9)$$

In order to minimize $\bar{d}(\theta)$ its derivative with respect to θ must be set to zero. By doing this and simplifying, we get the following expression.

$$tg(\theta) = \frac{\sum_{i=1}^n \left[\sin(\theta^{q_j(i)} - \theta^{p_i}) \cos(\varphi^{q_j(i)}) \cos(\varphi^{p_i}) \right]}{\sum_{i=1}^n \left[\cos(\theta^{q_j(i)} - \theta^{p_i}) \cos(\varphi^{q_j(i)}) \cos(\varphi^{p_i}) \right]} \quad (5.10)$$

Observe that the distance function defined in Equation (5.9) is a continuous function, and it is also bounded. The values of θ resulting from Equation (5.10) can identify local minima or local maxima of the distance function, depending on the sign of the second derivative. Hence it is necessary to check the sign of the second derivative by substituting the values of θ resulting from Equation (5.10) in the second derivative of the distance function defined in Equation (5.9). The values of θ that yield a positive value for the second derivative are local minima. Among them the θ value corresponding to the global minimum will be chosen.

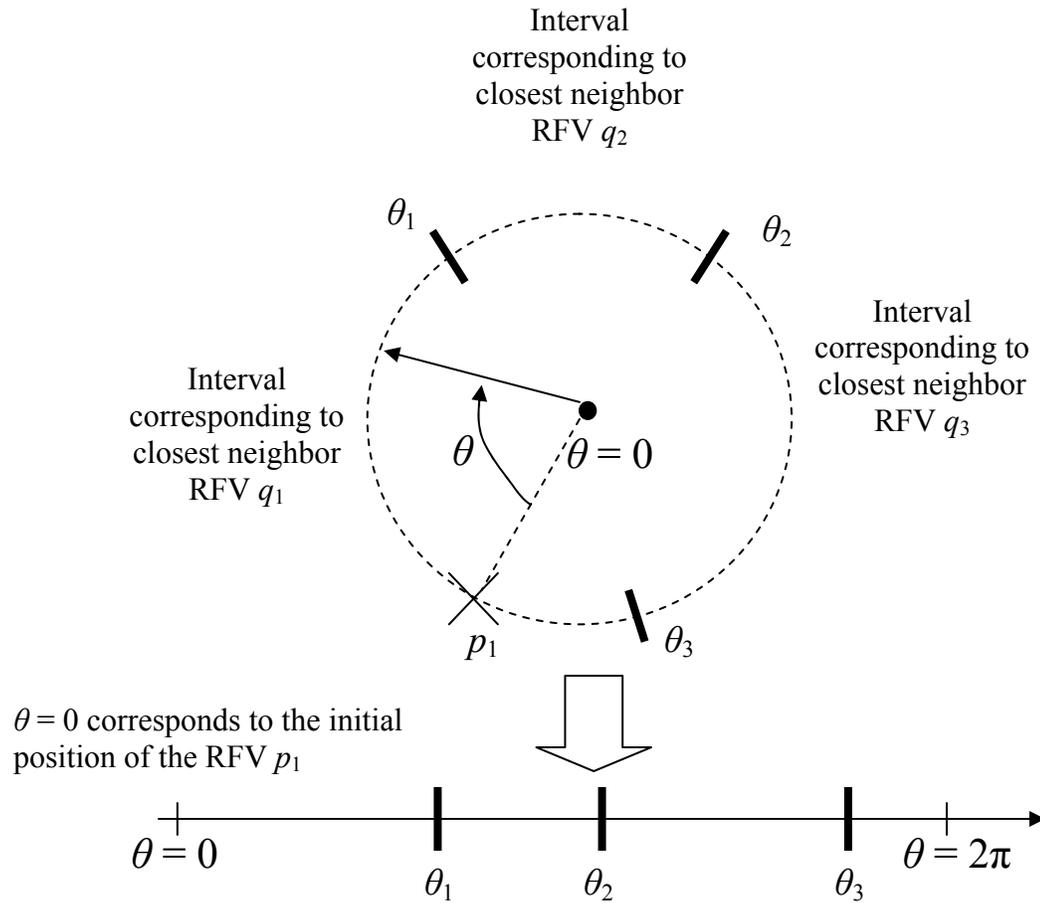


Figure 5.9: Set of Theta Intervals for Reduced Feature Vector p_1 of P

Equation (5.10) yields the transformation θ , applied to the set of RFVs P , which minimizes the distance between the sets of RFVs P and Q . This value of the transformation is valid only within a single interval of the set of theta intervals for all the RFVs of P . In general the value of θ that is found is not guaranteed to lie in the interval where the distance function is defined. Values of θ that lie outside the corresponding interval have no physical meaning and should be discarded. In fact Theorem 1 guarantees that none of them will be the θ value corresponding to the global minimum over all the intervals.

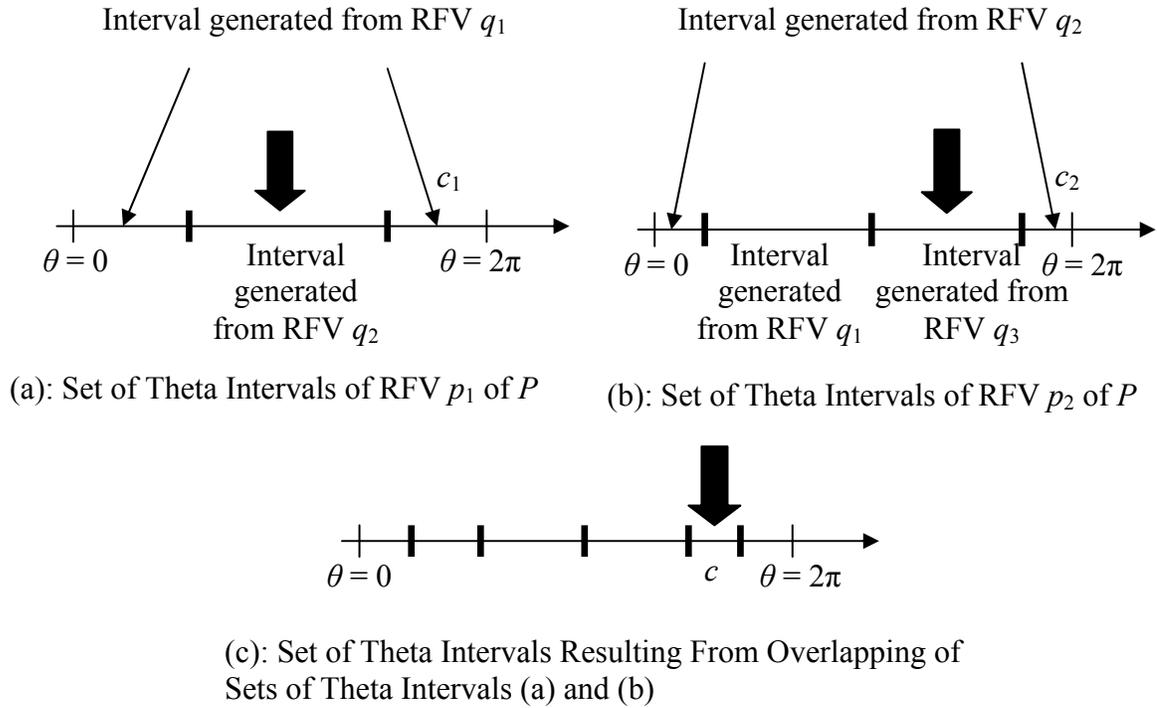


Figure 5.10: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals

Equation (5.10) has been obtained by differentiating the distance function with respect to θ , which is a standard minimization technique in the continuous domain. Thus, the transformation value obtained for an interval c of the set of theta intervals for all the RFVs of P yields the best possible alignment between the two RFV sets for all permissible transformations within the interval c .

5.4.3 Steps c and d: Computing The Value Of Theta That Minimizes The Distance Over All The Theta Intervals

The value of $\theta(c)$ obtained in the Equation (5.10) yields the rotation that minimizes the distance between the two RFV sets P and Q within a single interval c of the set of theta intervals for all of the RFVs of P . To obtain the corresponding value of the distance $\bar{d}(c)$

it is sufficient to substitute the value of θ obtained from Equation (5.10) into Equation (5.9). Hence, for each interval, $\vec{d}(c)$ is the minimum distance. Finally Step d of the algorithm COMPUTETHETA involves finding the value of θ corresponding to the minimum distance over all of the intervals. The minimum distance over all of the intervals is obtained as in Subsection 3.5.3. The same formulae can be used and the same considerations are valid. They are reported for clarity as follows.

$$\vec{d}_{\min} = \min_{c \in C} \vec{d}(c) \quad (5.11)$$

where C is the set of all the intervals c of the partitioned theta range $[0, 2\pi]$. Equation (5.11) yields the minimum distance between sets P and Q . The corresponding rotation θ_{\min} is found as follows.

$$\theta_{\min} = \theta(c^*) \quad (5.12)$$

where c^* is the interval in which the minimum distance was found.

Equation (5.12) yields the rotation to apply to P in order to minimize the distance between P and Q and Equation (5.11) yields the minimum distance between two sets of RFVs equivalent to attributed points on the unit sphere under one degree of freedom rotation.

5.5 Experimental Results For Single Feature Interpretations

A software system has been implemented based on the algorithms presented in this chapter in C++ programming language. The input to the system is the query part that the designer has newly designed and the directory in which all the previously machined parts exist. The system performs the alignment using the algorithms described previously and outputs those previously machined parts that are similar to the query part based on the

distance function described in Section 5.2. The output models are rank ordered based on this distance function starting with the one having the smallest distance value.

The parts have been defined using our own feature-based design system implemented in C++ programming language using ACIS geometric kernel libraries. The user needs to define the dimensional parameters, type, location, orientation and dimensional tolerance of all the features of the part. For each part the features are listed in a text file along with their parameters. We also generate its boundary representation to visually verify its correctness. Our information model is consistent with the Pro/Manufacture feature information model. We have tested this consistency on several different examples. In order to use our algorithm, one needs to either directly define features using features in our system, model parts using Features in Pro/Manufacture, or use a feature recognition system to identify machining features.

The procedure for aligning the two parts used as input to the system is illustrated using the example shown in Figure 5.11. Figure 5.11(a) shows the initial orientations of two parts M_P and M_Q that are to be compared. Part M_P is obtained by randomly rotating part M_Q . The system, initially, orients part M_P such that one of its features aligns with a feature of the same type of part M_Q as shown in Figure 5.11(b). The system then computes the angle of rotation θ such that the distance function is minimized. The final orientations of the two parts are shown in Figure 5.11(c).

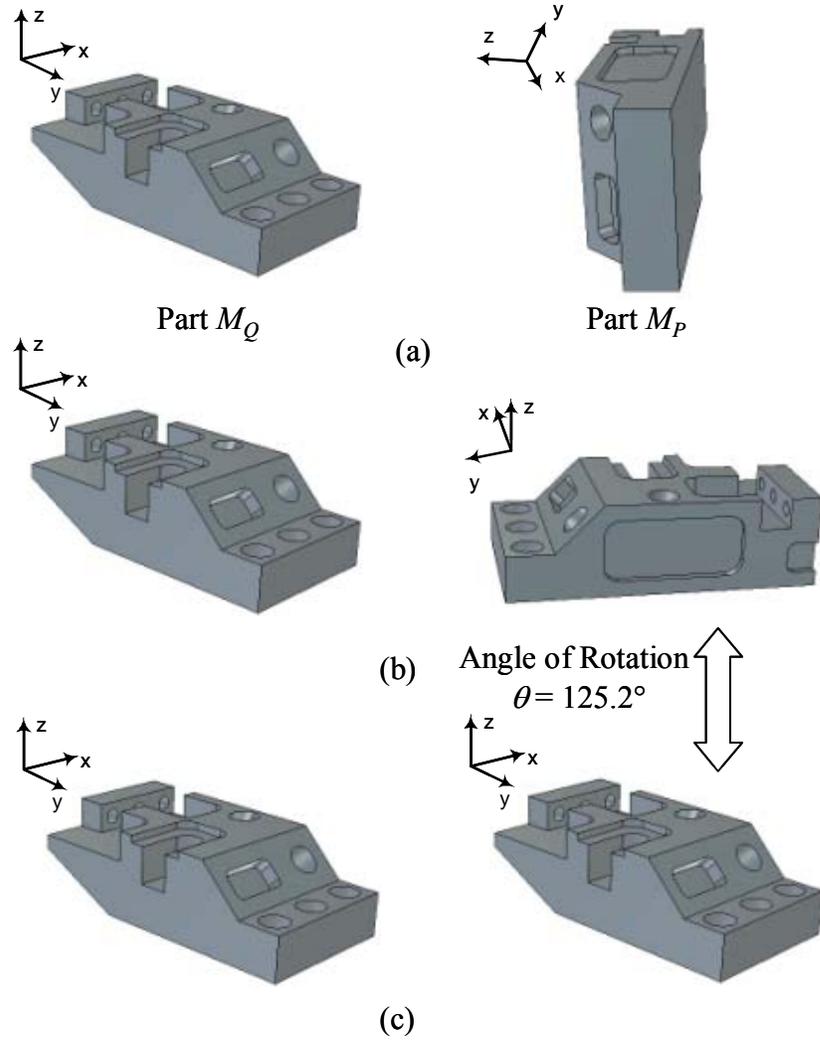


Figure 5.11: (a) Initial Orientation of Part M_Q and Its Randomly Rotated Version Part M_P ; (b) Orientation of Part M_P After Step3b(iii) of the Algorithm COMPUTESIMILARITYMEASURE; (c) Final Orientation of Part M_P

The database used for all the experiments consists of 120 parts having 20 to 30 features each. The dimensional tolerance value for all the features of 116 out of 120 parts has been set to $50\mu\text{m}$. The features of the remaining four parts have dimensional tolerance values of $10\mu\text{m}$, $25\mu\text{m}$, $75\mu\text{m}$ and $100\mu\text{m}$. The weights of the volume (w_V) and group cardinality (w_C) terms have been set to 1. The weight of the type term (w_T) has been set to 10 so that if two features of different type are aligned, the distance value is magnified.

The weight of tolerance term (w_ϵ) has been set to 10 in order to increase its influence on the distance function. The weights can be modified by the user to increase/decrease the influence of feature attributes on the distance function. In computing the distance function the volume of each feature has been normalized using the average value of the volumes of all the features of the parts being compared. The dimensional tolerance values have been normalized by dividing the dimensional tolerance value for each feature by $50\mu\text{m}$.

The first two experiments test the algorithms performance by focusing on feature volume, orientation, group cardinality and type. All of the features of the two query parts Part#A and Part#B being used have been assigned a dimensional tolerance value of $50\mu\text{m}$, so that almost all of the parts in the database have the same dimensional tolerance values as the query parts. Hence the parts being retrieved from the database will be the ones that are more similar to the query parts in feature volume, orientation, group cardinality and type. Figures 5.12 and 5.13 show the two query parts and those previously machined parts from the database of existing parts that are similar to the query parts. For each experiment the top three matches will be shown. The value of the distance between the parts is also indicated. Let us consider Part#118 in Figure 5.13. The distance value between Part#118 and the query Part#B is $d = 1.3107$. The contribution of the transformation-dependent term to the distance defined in Equation (5.2) is 0.3745. Among the transformation-invariant terms, the contribution of the volume term is 0.1670 and the contribution of the type term is 0.7692. Both the tolerance term and the group cardinality term do not give contribution to the distance.

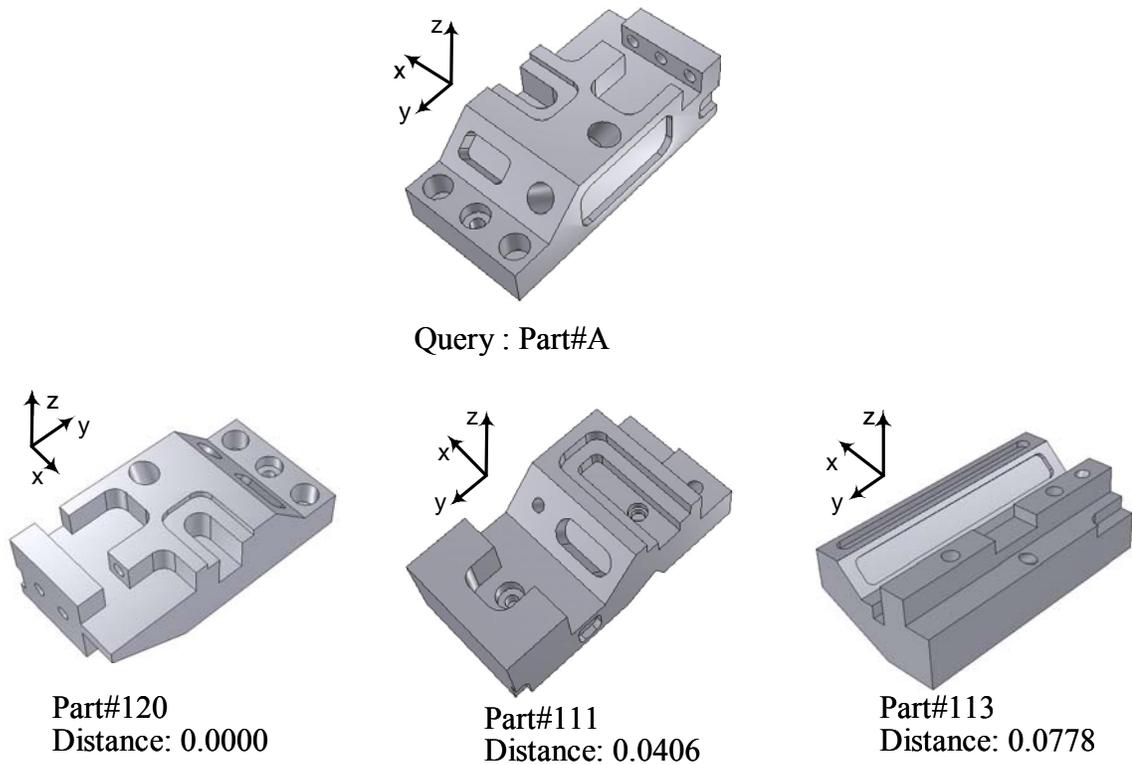


Figure 5.12: Results Obtained For Query Part#A Used As Input to the System

The performance of this algorithm was compared with a representative gross-shape based technique based on D2 shape function for similarity assessment between 3D parts [Osad01, Osad02]. Figures 5.12 and 5.13 show the results of our feature-based algorithm. Figures 5.14 and 5.15 show the results for the shape histogram based technique when applied to the same query parts. Part#120 and Part#119 are the only common retrieved parts in both cases. They have exactly the same features as respectively the query Part#A and Part#B, and their gross shape is also very similar to Part#A and Part#B.

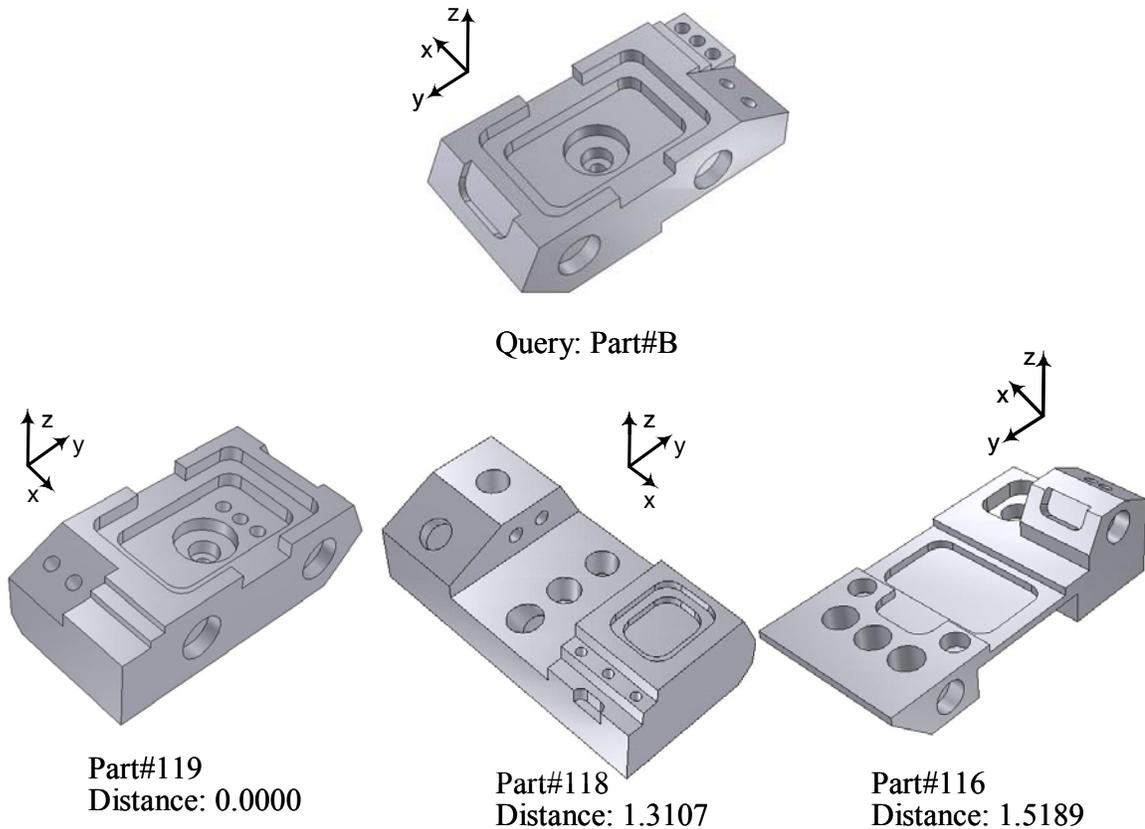


Figure 5.13: Results Obtained for Query Part#B Used As Input To The System

As shown in Figure 5.12, Part#113 is different in gross shape from the query Part#A, and hence it is not retrieved by the shape histogram based technique but is retrieved by our feature-based algorithm. The features of the two parts are similar in orientation and types, and hence have potentially similar machining costs. Therefore the cost of Part#A can be potentially estimated by using Part#113. As shown in Figure 5.14, not all of the parts have features similar to the query Part#A. For instance, Part#561 has all the features oriented along one of the coordinate axis, and the number and type of features does not match with the query Part#A. Part#561 is retrieved by the shape histogram based

technique but not by our feature-based algorithm. The cost of Part#A cannot be estimated by using Part#561 as the two parts are very different in feature orientation, types and volume. Hence, our feature-based algorithm is more suitable for cost estimation of machined parts. Similar conclusions can be drawn from examples in Figures 5.13 and 5.15.

The third experiment assesses the performance of the algorithm by focusing on feature dimensional tolerances. The query part used in this case is Part#C with a feature dimensional tolerance value of $10\mu\text{m}$. As described previously, most of the parts of the database have dimensional tolerance values of $50\mu\text{m}$, while only four of them have different dimensional tolerance values. Figure 5.16 shows those previously machined parts from the database of existing parts that are similar to the query part, along with their distance from the query part and their dimensional tolerance value. In this case, the retrieved parts are the ones that have dimensional tolerance values closest to the query part.

5.6 Similarity Assessment In Presence Of Multiple Feature Interpretations

In order to compute the degree of similarity using feature-based algorithms between pairs of 3D machined parts correctly, sometimes it is necessary to account for multiple possible interpretations of features [Gupt95]. Each feature interpretation corresponds to a different way of machining the feature. Figure 5.17 shows an example of multiple feature interpretations.

It is reasonable to assume that the database parts that will be used to estimate the cost of the newly designed part have already been machined and their machining cost is known. Hence, for each database part, we can safely assume that the preferred

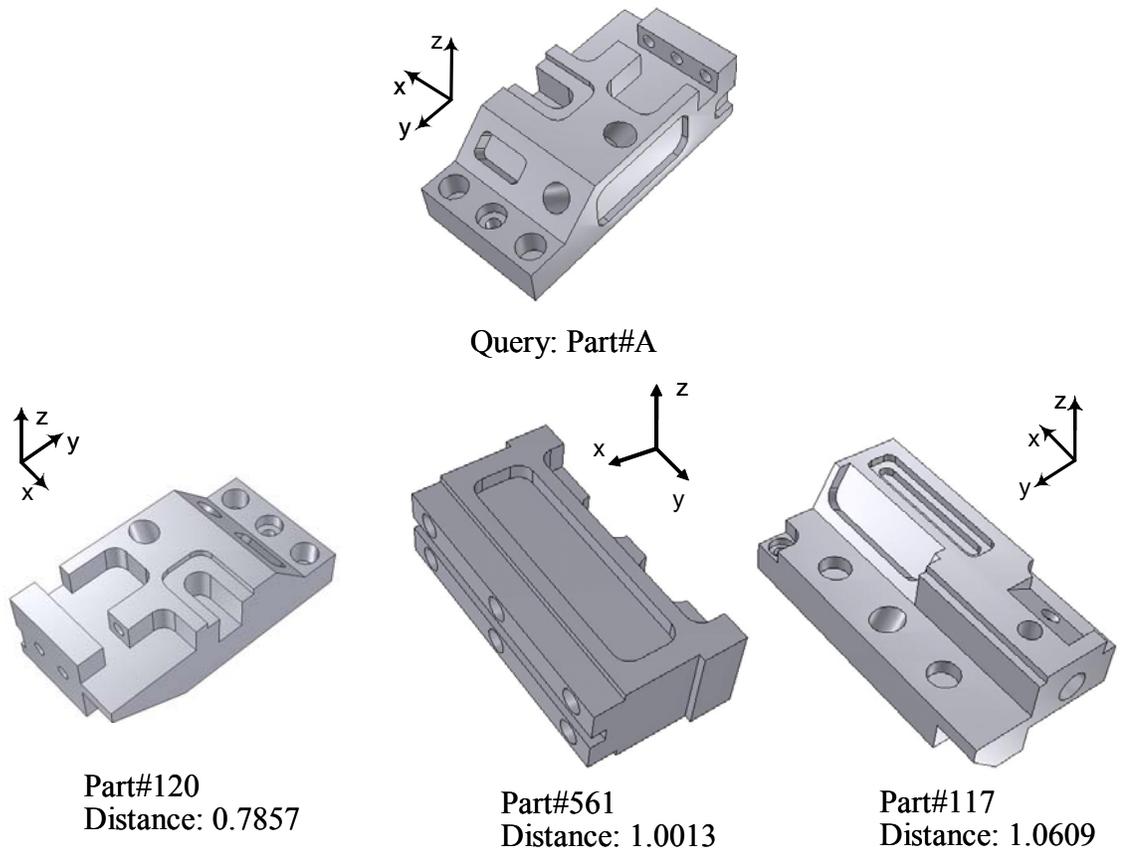


Figure 5.14: Results Obtained for Query Part#A As Input Using a Shape Histogram Technique

interpretation is known. On the other hand, the newly designed part whose cost needs to be estimated may have multiple possible interpretations, each corresponding to a different way of machining it and the preferred interpretation may not be known. For these reasons it will be assumed that only the query part has the possibility of multiple feature interpretations, while the database parts have unique preferred feature interpretations.

We generate multiple feature interpretations for each feature of the query part by changing the access direction and suitably transforming size parameters for certain types of features. The permissibility of a particular access direction is tested by performing accessibility analysis by sweeping the feature in the access direction and testing its

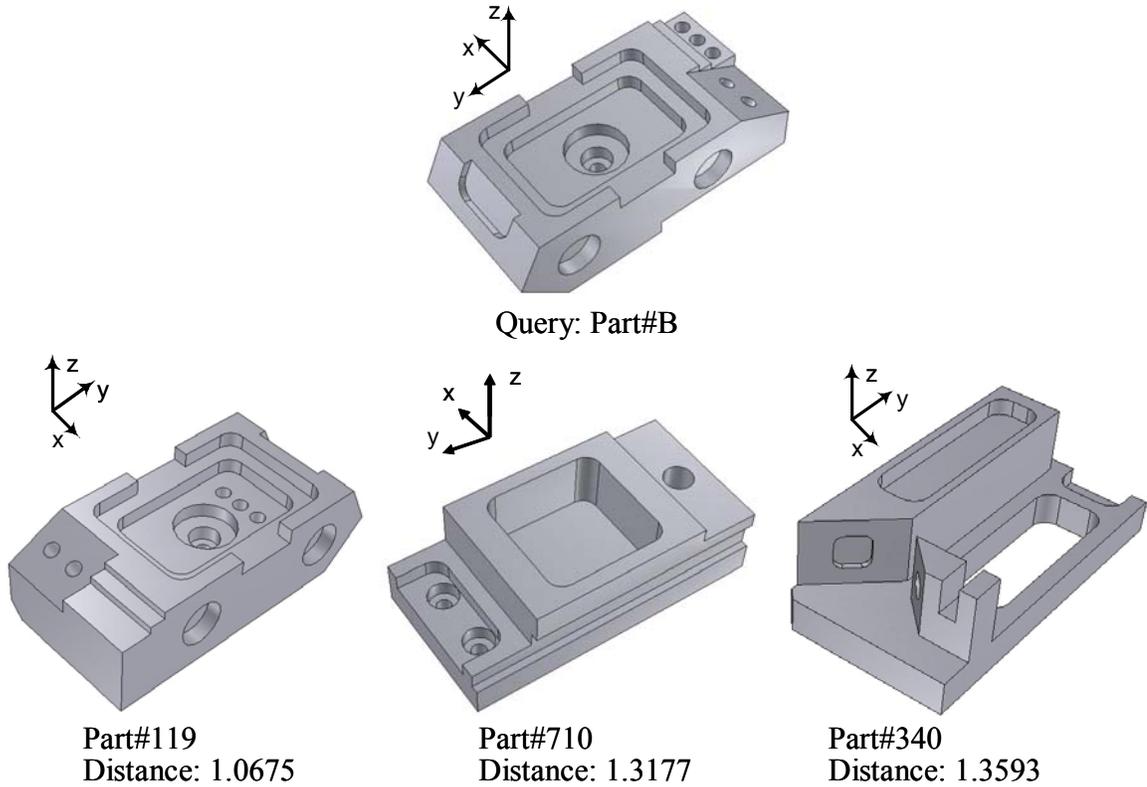


Figure 5.15: Results Obtained for Query Part#B As Input Using a Shape Histogram Technique

intersection with the part. If the swept body intersects with the part, then the corresponding access direction is not permitted. Each permissible interpretation is used for further analysis in the algorithm.

In order to handle multiple feature interpretation, it is necessary to modify the distance function defined in Equation (5.1). Refer to the definitions given in Section 5.2. Imagine that part M_Q is a database part whose cost is known. Hence the interpretation of the features for part M_Q is unique. On the other hand, part M_P is a newly designed part whose cost needs to be estimated. Hence its feature interpretation has not been yet defined. Denote the set of RFVs for these parts as P and Q . Let $A_i = (p_i^1, p_i^2, \dots, p_i^{k_i})$ be the set of possible k_i interpretations for RFV p_i of set P . Let

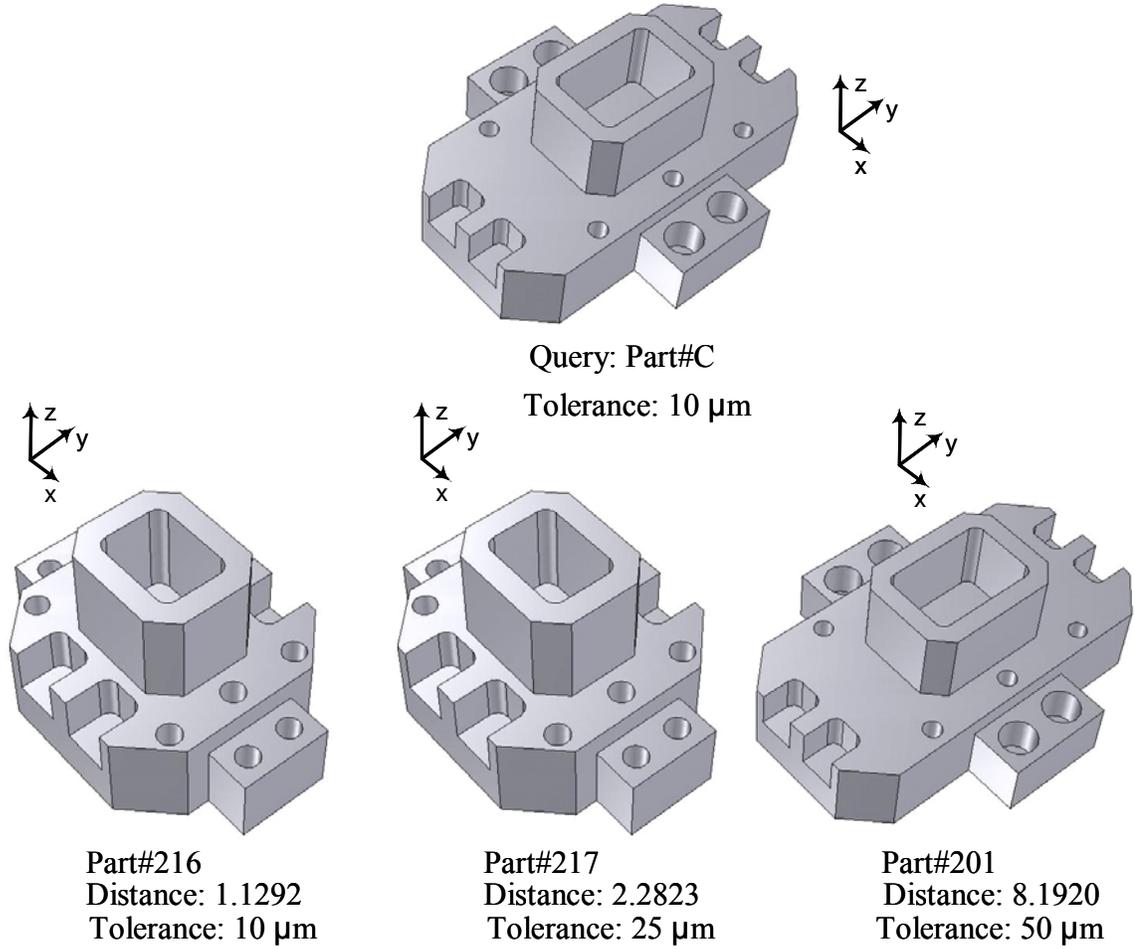


Figure 5.16: Results Obtained for Query Part#C Used As Input To The System

$p_i^k = (x^{p_i^k}, y^{p_i^k}, z^{p_i^k}, V(p_i^k), \varepsilon(p_i^k), n(p_i^k)) \in A_i \subset P$ be the k th interpretation of RFV p_i . In order to account for multiple interpretations of features, the sets P and Q are compared using the following distance function.

$$\bar{d}(P, Q) = \frac{\sum_{i=1}^n \min_{p_i^k \in A_i} \min_{q \in Q} d(p_i^k, q)}{n} \quad (5.13)$$

The overall algorithm described in Section 5.3 constrains two out of the three degrees of freedom involved by aligning all the possible pairs of RFVs of the same type. The

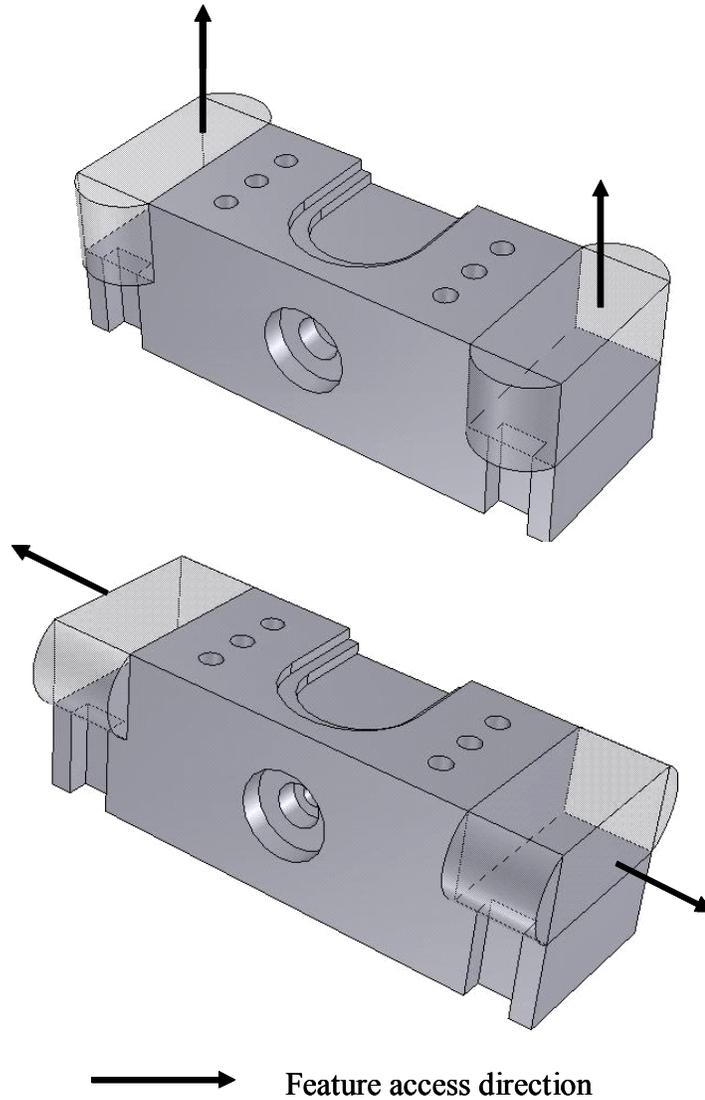


Figure 5.17: Example Of Two Possible Feature Interpretations For A Machined Part

same reasoning can be used in case of multiple interpretations of features. In this case pairs of RFVs of the same type are identified, but then all the possible interpretations of each RFV of P need to be aligned to the corresponding RFV of part Q .

In order to compute the distance value in Equation (5.13), we need to know the closest RFV $q_j \in Q$ to each interpretation $p_i^k \in P$. The closest neighbor change with the

rotation θ around the fixed axis identified through the alignment of a pair of RFVs of the same type. It is necessary to divide the theta range $[0,2\pi]$ into theta intervals such that the closest neighbor to each RFV interpretation is invariant. In order to use the algorithm defined in Section 5.4, it is necessary to consider each RFV interpretation for part M_P as a regular RFV. By using this assumption it is possible to apply the algorithm defined in Section 5.4 without any modification, to obtain a set of theta intervals. Within each interval the closest RFV $q_j \in Q$ to each RFV interpretation $p_i^k \in P$ is invariant. Observe that in this case the pruning based on number of features will not be performed, as each feature interpretation of part M_P will be considered as a feature and hence the number of features being considered will be higher than the actual one. Also, the features of part M_P will not be grouped, as the grouping may result in considering simultaneously two or more interpretations of the same feature in computing the distance function defined in Equation (5.13).

Once the set of theta intervals is obtained, a further step is necessary in order to compute the distance function defined in Equation (5.13). It is necessary to choose the RFV interpretation whose distance from its closest neighbor is minimum over all the interpretations. Such RFV interpretation will be referred to as *preferred interpretation*. This choice needs to be made within each theta interval for each RFV of part M_P . Let us focus on the set $A_i = (p_i^1, p_i^2, \dots, p_i^{k_i})$ of possible k_i interpretations for point p_i of set P corresponding to a feature of part M_P . Observe that the distance function defined in Equation (5.13) is a linear combination distance function. This property allows to focus on each term $\min_{p_i^k \in A_i} \min_{q \in Q} d(p_i^k, q)$ independently and minimize it. Within each interval, the

closest neighbor $q_j(k,i)$ of Q to each interpretation p_i^k is constant. Hence each term of the distance function can be written as $\min_{p_i^k \in A_i} d(p_i^k, q(k,i))$. So it is necessary to perform a dominance analysis on the distance functions $d(p_i^k, q(k,i))$ between each interpretation p_i^k and its closest neighbor $q_j(k,i)$ within the interval being considered. The dominance analysis consists of identifying the distance function that has the minimum value over all the distance functions corresponding to the RFV interpretations. The RFV interpretation corresponding to the minimum distance function will be the preferred one. The dominance analysis in some cases might result in splitting the theta interval into subintervals within which different RFV interpretations are the preferred ones. In order to visualize the described procedure, refer to the example in Figure 5.18. The Z axis is the rotation axis being considered, and the theta interval is assumed to be the entire $[0,2\pi]$ range. In Figure 5.18(a), the access vectors to the two interpretations of feature 1 of part M_P are shown. Also the access vector to feature 1 of part M_Q , that is the closest neighbor to both the feature interpretations, is shown. In Figure 5.18(b) the representation of the corresponding RFVs on the unit sphere is shown. In Figure 5.18(c) the corresponding distance functions are plotted. As shown in the figure, the distance functions are both constant in the interval being considered and the distance function corresponding to RFV p_i^1 is the minimum one. Hence the corresponding RFV interpretation is the preferred one. So the described dominance analysis needs to be performed for each theta interval. It yields a set of theta intervals, in general but not necessarily different from the one previously obtained. Within each interval the preferred RFV interpretation and its closest neighbor are invariant. The described procedure yields

the theta intervals for a single feature of M_p . It needs to be repeated for each RFV of part M_p . So finally a set of theta intervals for each feature of M_p is available. Overlapping the intersecting intervals will further split the range $[0,2\pi]$ into theta intervals, within which the preferred RFV interpretation and its closest neighbor are invariant for all the features of part M_p . Now it is possible to compute the distance function defined in Equation (5.13) and minimize it.

The minimization of the distance function defined in Equation (5.13) can be performed following the same steps as in Subsection 5.4.2. The following notations need to be introduced. Let $\theta^{p_i^k}$ and $\varphi^{p_i^k}$ be the known angle values for RFV interpretation $p_i^k \in A_i \subset P$ at the end of the RFV pair alignment described previously. Similarly let φ^{q_j} and θ^{q_j} be the known angles for RFV $q_j \in Q$ at the end of the RFV pair alignment described previously. Within each interval, for each set of interpretations $A_i = (p_i^1, p_i^2, \dots, p_i^{k_i})$, let $p_i^{\bar{k}_i}$ be the preferred one. The definitions (5.5) need to be modified as follows.

$$\left\{ \begin{array}{l} x^{q_j}(\bar{k}_i, i) = x \text{ coordinate of the closest RFV } q_j(\bar{k}_i, i) \in Q \text{ to the RFV } p_i^{\bar{k}_i} \in A_i \subset P \\ y^{q_j}(\bar{k}_i, i) = y \text{ coordinate of the closest RFV } q_j(\bar{k}_i, i) \in Q \text{ to the RFV } p_i^{\bar{k}_i} \in A_i \subset P \\ z^{q_j}(\bar{k}_i, i) = z \text{ coordinate of the closest RFV } q_j(\bar{k}_i, i) \in Q \text{ to the RFV } p_i^{\bar{k}_i} \in A_i \subset P \\ \theta^{q_j}(\bar{k}_i, i) = \theta \text{ angle of the closest RFV } q_j(\bar{k}_i, i) \in Q \text{ to the RFV } p_i^{\bar{k}_i} \in A_i \subset P \\ \varphi^{q_j}(\bar{k}_i, i) = \varphi \text{ angle of the closest RFV } q_j(\bar{k}_i, i) \in Q \text{ to the RFV } p_i^{\bar{k}_i} \in A_i \subset P \end{array} \right. \quad (5.14)$$

Following the same steps as in Subsection 5.4.2 with the new notations introduced previously, again $\bar{d}(\theta)$ is minimized by setting its derivative with respect to θ to zero.

$$tg(\theta) = \frac{\sum_{i=1}^n \left[\sin(\theta^{q_j}(\bar{k}_i, i) - \theta^{p_i^{\bar{k}_i}}) \cos(\varphi^{q_j}(\bar{k}_i, i)) \cos(\varphi^{p_i^{\bar{k}_i}}) \right]}{\sum_{i=1}^n \left[\cos(\theta^{q_j}(\bar{k}_i, i) - \theta^{p_i^{\bar{k}_i}}) \cos(\varphi^{q_j}(\bar{k}_i, i)) \cos(\varphi^{p_i^{\bar{k}_i}}) \right]} \quad (5.15)$$

The same observations made in Subsection 5.4.2 are valid here. The last step consists of minimizing the distance over all of the theta-intervals. The same reasoning and formulae presented in Subsection 5.4.3 can be used for this task. So by modifying the algorithms and using slightly different notations than in the previously described way, it is possible to account for multiple interpretations of features. In defining the previously described algorithms the assumption that each group of RFV interpretations A_i is independent from the others has been used.

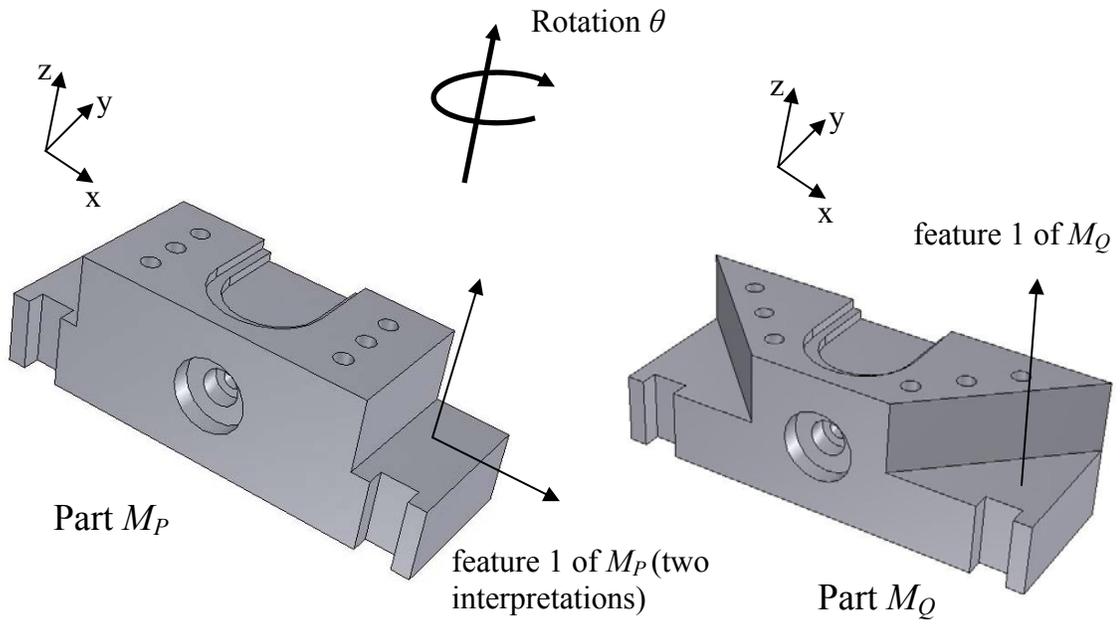
An experiment that has been carried out to assess the performance of our approach is described. Both the database and the distance function weights are the same as the ones used for the experiments in Section 5.5. In this case the query Part #D has two features with two possible interpretations each, as shown in Figure 5.19. The top three matches to the query part from the database are shown in Figure 5.19. Part#211 is the first retrieved part. It has exactly the same feature orientation, type and characteristics as the query part, and the feature interpretations suggested are the ones corresponding to access vectors a_1^1 and a_2^1 . Part#212 and Part#213 are the second and third retrieved parts. They have slightly different feature characteristics from the query part. The feature interpretations suggested are the ones corresponding to access vectors a_1^2 and a_2^2 . Part#211 can be potentially used for estimating the cost of the query part, as it has both similar features and it suggests the most convenient feature interpretation. In fact, machining the features shown in Figure 5.19 along Z axis is more appropriate than along X axis. Thus, even

though Part#212 and Part#213 have features similar to the query part, Part#211 can be potentially used for cost estimation.

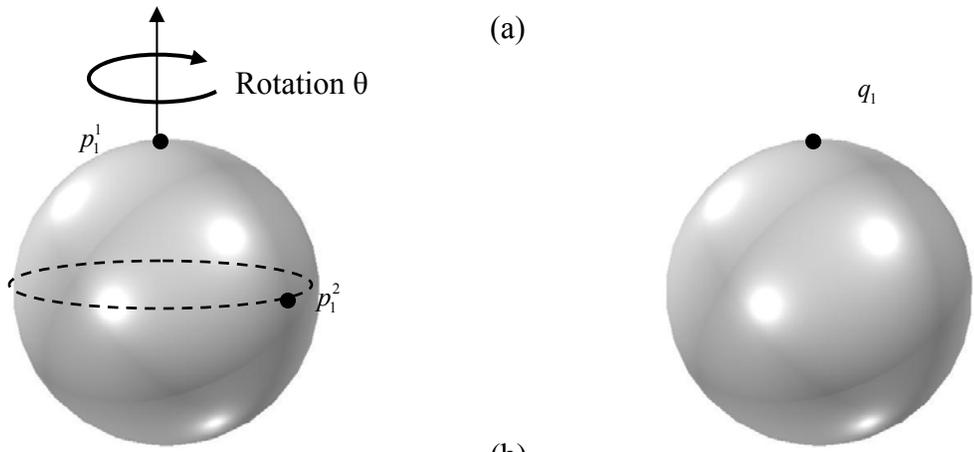
5.7 Summary

This chapter provides algorithms for identifying those parts in a database that are similar to a given query part in machining features and hence can be potentially used as a basis for estimating the machining cost of the query part. We have developed a distance function to account for the key drivers for the machining features of a prismatic part. We have developed an algorithm that performs feature alignment to minimize this function. We have implemented the algorithm to show proof of the concept. We have tested the algorithm on some examples in order to assess its performance.

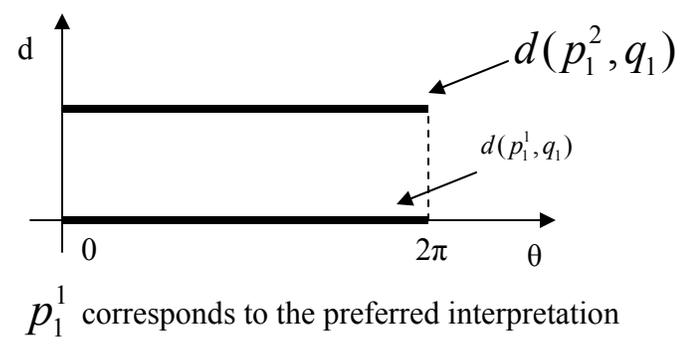
The feature-based algorithm described in this chapter is expected to perform better than the gross-shape based algorithms in similarity assessment from a machining cost point of view. This is because the machining cost mainly depends on the orientation, size, tolerance and group cardinality of the features and not on the gross shape of the part. The algorithm does not restrict the features to have a particular orientation as needed in some other techniques. It can handle features having any arbitrary orientation in space. It can also handle query parts that have features with multiple interpretations. The current algorithm can handle only parts for which individual feature interpretations are independent of each other. It accounts for the relative feature orientation that is not considered by other feature-based techniques. Also, the distance function includes feature attributes such as dimensional tolerances.



(a)



(b)



(c)

Figure 5.18: Dominance Analysis For The Two Interpretations Of Feature 1 of part M_P With Respect To Their Closest Neighbor Feature 1 of Part M_Q

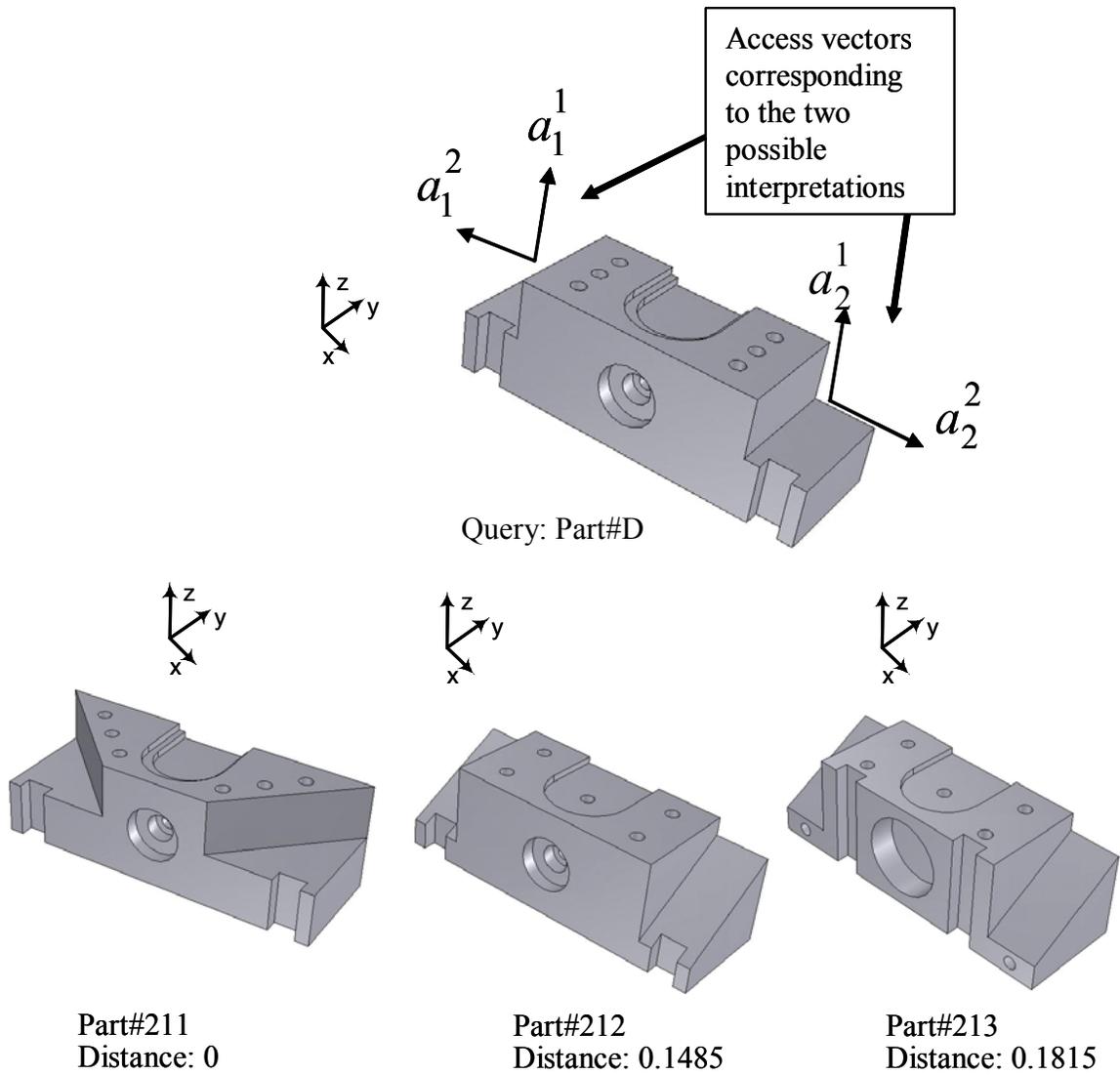


Figure 5.19: Results Obtained for Query Part#D Used As Input To The System

Chapter 6: Surface Feature-Based Shape Similarity Assessment Algorithms

This chapter is organized as follows. Section 6.1 gives the motivation behind the application addressed in this chapter. Section 6.2 provides the definitions that are needed and presents the problem formulation. Section 6.3 describes the algorithm for finding similar parts based on surface features. Section 6.4 describes the step of this algorithm that computes optimal alignment under one degree of freedom. Section 6.5 describes the iterative algorithm that computes optimal alignment under three degrees of freedom by utilizing the algorithm presented in Section 6.4. Section 6.6 provides the computational experiments that have been performed. Finally, Section 6.7 presents the summary of this chapter.

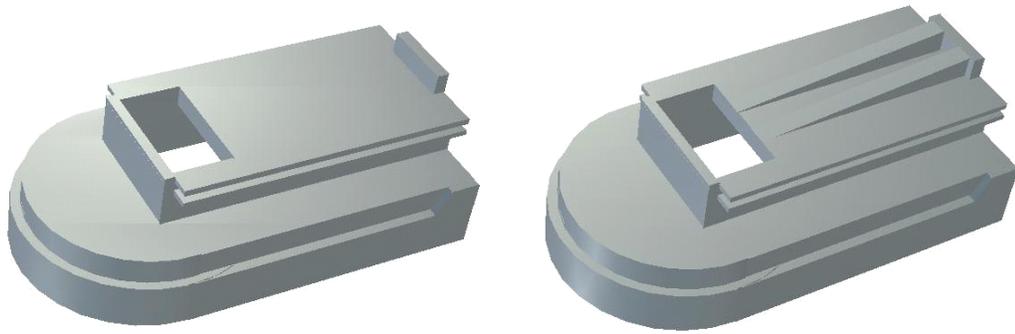
6.1 Motivation

Manufacturing of plastic parts is a two-step process. During the first step the tool is designed and constructed for making the parts. During the next step, parts are produced using the tool. Often tool makers and molders are two different organizations. Therefore, selecting a tool maker is an important step in molding of plastic parts. Many different kinds of tools exist that can be used to create plastic parts depending upon the shape of the part. Different tool makers specialize in different kinds of toolings. Therefore, one has to analyze the shape of the part to determine the most appropriate tool maker based on the type of tool needed for the part. Internet-based tool ordering systems give an organization an opportunity to contact a wide variety of tool makers (many of them located in different geographical locations) to solicit quotes from them in order to get the best deal. However, contacting a very large number of tool makers to get quotes is not practical due to the time needed to send the data and analyze the quotes. Therefore, designers and

manufacturers often rely on their prior experience to contact the tool makers that have capabilities to handle the new part. This model worked well when designers and manufacturers were dealing with a small number of local tool makers. In the era of global operations and access to a large number of tool makers, designers and manufactures can benefit from software support to help them in identifying potential tool makers.

Currently a fully generative method to determine the tool type based on the part shape does not exist. Therefore, another possible way to identify potential tool makers is to find similar parts to the given part and identify tool makers for the similar parts. This method is currently being practiced by experienced part designers. However, they currently rely on their memory to locate similar parts. We believe that a system that enables them to find similar parts based on surface features will be a useful system to them. Figure 6.1(a) shows a new part and Figure 6.1(b) shows a previously molded part whose tool maker can be approached to make the mold for the new part. The automatic part database search tool will clearly decrease the time needed to search for similar parts.

Similarity between two parts from the tool maker selection point of view needs to be assessed by referring to the surface features of the parts. In fact, the tooling for plastic parts depends mainly on their surface features. For example, surface parameters such as spatial location, type and curvature distribution determine the type and complexity of the tooling needed to manufacture the part. Similarly, the surface area determines the size of the tooling. Surface features do not always have explicitly defined parameters. Hence we need to identify components of surface feature vectors that are significant in determining the similarity between two parts from the tooling point of view.



(a): Newly designed part

(b): Previously molded part

Figure 6.1: The Tool Maker of Part (b) Can Be A Potential Tool Maker for the Newly Designed Part (a)

This chapter introduces reduced surface feature vector sets that are suitable for the tool maker selection application and provides alignment algorithms for the reduced surface vectors of two parts. Reduced surface feature vectors (RSFVs) for a part are defined for a specific coordinate system. In order to measure the distance between two sets of reduced surface feature vectors, one set is transformed with respect to the other by using rigid body transformations such that the minimum distance between two sets is obtained. The alignment algorithms rank order the parts in a database based on the degree of similarity with respect to the query part.

6.2 Background And Problem Formulation

6.2.1 Surface Features

A plastic part can be characterized from the tooling point of view by referring to its surface features. In this chapter surface patches represent the surface features of a part. A patch is defined as a surface region delimited by patch edges. A patch edge is a curve belonging to the surface of the part. The curve is either a segment corresponding to a

sharp corner or a set of points corresponding to locally maximum curvature values. Some definitions are given as follows.

Location of a surface patch is a point that gives the position of the patch in the space. *Orientation vector* of a surface patch is a unit vector that gives the orientation of a patch in the space. For some types of surface patches, such as spherical patches, this vector is not defined.

In order to give a formal definition of the surface patch location and orientation, assume without loss of generality that n points p_i are sampled from surface patch A along with the normal vector \mathbf{o}_i in correspondence of each point. Surface patch location l_A and orientation \mathbf{o}_A are computed as follows.

$$l_A = \frac{\sum_{i=1}^n p_i}{n} \quad \text{and} \quad \mathbf{o}_A = \frac{\sum_{i=1}^n \mathbf{o}_i}{\left| \sum_{i=1}^n \mathbf{o}_i \right|} \quad (6.1)$$

In case the surface patch is represented in the continuous domain (i.e. an equation representing the surface patch is defined), the sums in Equations (6.1) are replaced by the corresponding integrals. Observe that for some particular types of surface patches the second of Equations (6.1) cannot be used.

The following types of surface patches are considered in this chapter: cylindrical, planar, toroidal and spherical patches. All the rest are defined as general patches. They are shown in Figures 6.2 respectively with their access vectors and locations. The toroidal and cylindrical patches are examples of surface patches where the second of Equations (6.1) cannot be used. In these cases surface patch orientation is defined as the unit vector along the axis of the cylinder or torus. Orientation is not defined for spherical patches. Each of the previously listed surface patches can be completely characterized by

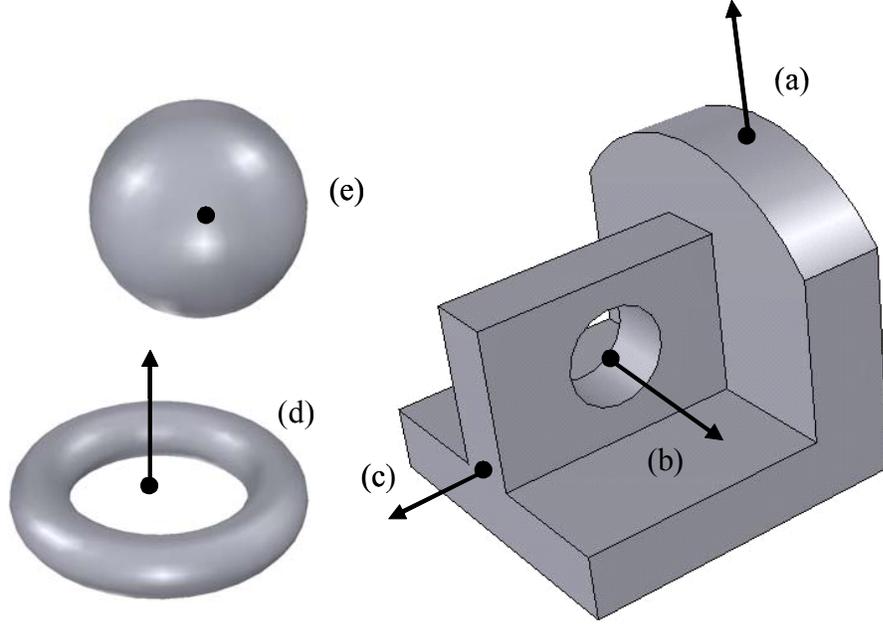


Figure 6.2: Types Of Patches That Are Considered With Corresponding Location Point and Orientation Vector: (a) General (b) Cylindrical (c) Planar (d) Toroidal (e) Spherical

providing the values of certain parameters such as area, curvature distribution and normal vector distribution. In particular, normal vector distribution is characterized by the orientation standard deviation. In order to formally define orientation standard deviation, consider surface patch orientation \mathbf{o}_A and the normal vectors \mathbf{o}_i sampled from patch A . Consider the discrete function $f_i = \mathbf{o}_A \cdot \mathbf{o}_i$. The orientation standard deviation is defined as the standard deviation of the discrete function f_i , which is defined as follows.

$$\sigma_f = \sqrt{\frac{\sum_{i=1}^n (f_i - \mu_f)^2}{n}}$$

where $\mu_f = \frac{\sum_{i=1}^n f_i}{n}$. A number of techniques can be used to compute the curvature in

correspondence of each sampled point p_i . We use the curvature computation technique suggested in [Hebe95]. As for the surface patches whose resolution is low (i.e. whose

number of sampled points p_i is low) the curvature is computed referring to the angles between the normal vectors \mathbf{o}_i in correspondence of each sampled point. Once the value of curvature c_i for each sampled point p_i of the surface patch is known, the mean curvature μ_c and the curvature standard deviation σ_c can be computed.

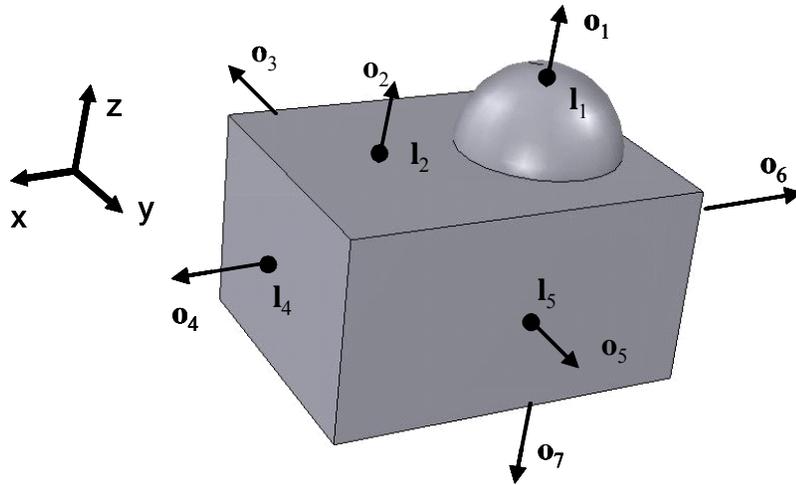
In order to formally define a RSFV, let us recall the definitions of free and applied vectors. A free vector is a vector whose orientation and magnitude are specified. An applied vector is a vector whose orientation, magnitude and point of application are defined. The point of application of a vector is the position of the vector in the space.

RSFVs of a surface patch consist of those surface patch components that are important from the tooling point of view. RSFVs are mathematically equivalent to attributed applied vectors in \mathbb{R}^3 , where the application points of the vectors correspond to the patch location and the vector orientations correspond to patch orientations. Figure 6.3 shows an example of RSFVs of a part and their equivalent attributed applied vectors in \mathbb{R}^3 . Therefore, the problem of aligning two sets of RSFVs is equivalent to the problem of aligning attributed applied vectors in \mathbb{R}^3 . Hence in this chapter we will use terms RSFVs and attributed applied vectors in \mathbb{R}^3 interchangeably.

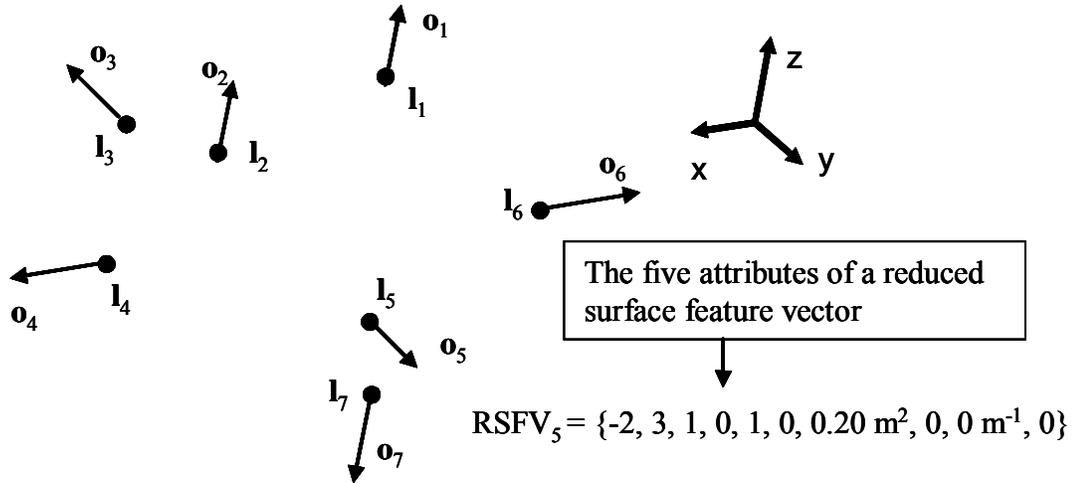
6.2.2 Distance Function For Similarity Assessment

Let $p \in P$ and $q \in Q$ be the two sets of RSFVs corresponding to parts M_P and M_Q . Then, P and Q are compared using the following distance function, that has the same expression as the one defined in Equation (3.1).

$$\bar{d}(P, Q) = \frac{\sum_{i=1}^n \min_{q \in Q} d(p_i, q)}{n} \quad (6.2)$$



(a): 3D Object With Surface Patch Locations and Orientations



(b): Equivalent Attributed Applied Vectors in \mathbb{R}^3

Figure 6.3: Equivalence Between Reduced Surface Feature Vectors and Set of Attributed Applied Vectors in \mathbb{R}^3

As observed in the previous section, the key drivers for tool maker selection of a plastic part are the surface patch relative locations and orientations, the surface patch curvature and the orientation distribution and surface patch type. The distance function between two RSFVs $p \in P$ and $q \in Q$ needs to account for them. Each RSFV is represented by using ten components. Specific components are $x^p, y^p, z^p, v_x^p, v_y^p, v_z^p, A(p),$

$\sigma_o(p)$, $\mu_c(p)$, $\sigma_c(p)$. The first three components x^p , y^p and z^p represent the location of the RSFV p , and are transformation-dependent. Similarly the second three components v_x^p , v_y^p and v_z^p represent the orientation of the RPV p and are also transformation-dependent. The other four components $A(p)$, $\sigma_o(p)$, $\mu_c(p)$ and $\sigma_c(p)$ are transformation-invariant. The seventh component $A(p)$ represents the normalized area of the RSFV. The areas are normalized using the maximum value of the area over all the surface patches of the parts being compared. The eighth component $\sigma_o(p)$ represents the normalized orientation standard deviation, which is not defined in the case where the surface patch is a sphere or a cylinder. For all the other types, the orientation standard deviation is normalized using the maximum value of the orientation standard deviation over all the surface patches of the parts being compared. The ninth component $\mu_c(p)$ represents the normalized average curvature, which is normalized using the maximum value of the average curvature over all the surface patches of the parts being compared. The tenth component $\sigma_c(p)$ represents the normalized curvature standard deviation, which is normalized using the maximum value of the curvature standard deviation over all the surface patches of the parts being compared.

To increase the efficiency of comparison and avoid the problem just described, parts that do not have a comparable value to the number of surface patches of the query part are discarded. This pruning step ensures that parts with a comparable number of surface patches are assessed for similarity, so that the retrieved parts have shapes comparable to the query part.

The distance function between RSFVs $p \in P$ and $q \in Q$ is defined as follows.

$$\begin{aligned}
d(p, q) = & w_L[(x^p - x^q)^2 + (y^p - y^q)^2 + (z^p - z^q)^2] + \\
& w_O[(v_x^p - v_x^q)^2 + (v_y^p - v_y^q)^2 + (v_z^p - v_z^q)^2] + (1 - \delta(p, q)) \\
& \left[w_A(A(p) - A(q))^2 + w_{\sigma_o}(\sigma_o(p) - \sigma_o(q))^2 \right] + \\
& \left[w_{\mu_c}(\mu(p) - \mu(q))^2 + w_{\sigma_c}(\sigma_c(p) - \sigma_c(q))^2 \right] + \\
& + w_T \delta(p, q)
\end{aligned} \tag{6.3}$$

The first three terms account for the difference in position between p and q and relate to surface patch interactions. The second three terms account for the difference in the orientation and relate to the surface patch interactions as well. The last five terms account for the difference in transformation-invariant attributes that are considered. Specifically, the seventh term accounts for the difference in area between the corresponding surface patches and relates to patch size. The eighth, ninth and tenth terms account for the difference in the orientation standard deviation, the average curvature and the curvature standard deviation between the corresponding surface patches and relate to patch complexity. The eleventh term accounts for the difference in type between the corresponding surface patches that has been defined in Subsection 6.2.1. It relates to patch complexity as well. The term δ has the following expression.

$$\begin{cases} \delta(p, q) = 0 & \text{if type of } p \text{ is equal to type of } q \\ \delta(p, q) = 1 & \text{if type of } p \text{ is different from type of } q \end{cases}$$

So all the key drivers for the surface feature-based shape recognition of parts are accounted for. The quantity $(1 - \delta(p, q))$ is defined so that when the types of surface patches p and q do not match most of the terms are not considered. The quantities w_O , w_L , w_A , w_{σ_o} , w_{μ_c} , w_{σ_c} , and w_T represent the weights given by the user to all the terms previously defined. The distance function can be customized by: (a) changing the weight

associated with each of the terms in the distance function, (b) considering additional transformation-invariant shape parameters as needed.

The distance function defined in Equation (6.2) is the measure of similarity between parts M_P and M_Q , represented by two sets of RSFVs; the smaller the value of the distance given by Equation (6.2), the more similar are the parts M_P and M_Q .

6.2.3 Problem Statement

The input to the system described in this chapter is a database of parts whose tool makers are known and a newly designed part for which a tool maker needs to be selected. The system outputs parts similar to the query part.

Each part has been modeled in its own coordinate system. Therefore, we need to align the parts using rigid body transformations before computing the distance. The parts are represented by using two sets of RSFVs. Hence, as stated previously, the problem of aligning two sets of RSFVs is equivalent to the problem of aligning attributed applied vectors in \mathbb{R}^3 . To align the two sets of attributed applied vectors in \mathbb{R}^3 , one set has to be moved with respect to the other set. Rigid body transformation of a set of attributed applied vectors in \mathbb{R}^3 involves six degrees of freedom. The distance function has to be minimized over all the possible configurations of the moving attributed applied vector set with respect to the stationary one. The transformation matrix for the six degrees of freedom transformation is given by $\mathbf{T} = \mathbf{T}(\Delta x, \Delta y, \Delta z, \theta, \phi, \psi)$ where Δx , Δy , Δz , θ , ϕ , and ψ are the six degrees of freedom considered. Assuming that P is the moving set, the transformed set P can be written as $\mathbf{T}P$. The distance function defined in Equation (6.2) can then be written as:

$$\bar{d}(\mathbf{TP}, Q) = \bar{d}(\mathbf{TP}, Q)(\Delta x, \Delta y, \Delta z, \theta, \varphi, \psi) \quad (6.4)$$

This chapter introduces an algorithm to find the best alignment between two sets of attributed applied vectors in \mathbb{R}^3 by transforming one attributed applied vector set such that the distance function is minimized.

In Sections 6.3, 6.4, and 6.5 the surface feature-based shape similarity assessment algorithm is described.

6.3 Computing Surface Feature-Based Similarity For Parts

As mentioned previously, aligning two sets of attributed applied vectors in \mathbb{R}^3 is a six degree of freedom problem. For selecting the tool maker of the new part based on the database of existing parts, the two parts should have at least one surface patch of the same type. If the two parts have no common surface patches, then one part cannot be used to select the tool maker for the other and hence the part needs to be pruned. Thus, three degrees of freedom in this problem can be constrained by considering combinations of surface patches. Each surface patch location of M_P is aligned with every surface patch location of M_Q having the same type. The total number of alignments that need to be performed is not large. This is because the number of combinations of surface patches of the two parts of the same type is not significantly large, as most of the reasonably complex plastic parts have fewer than 100 instances of surface patches.

Consider a pair of RSFVs $p_i \in P$ and $q_j \in Q$ of the same type equivalent to two attributed applied vectors in \mathbb{R}^3 . Initially, the translation represented by the matrix $\mathbf{T}_{i,j}$ is applied to the two sets P and Q such that the locations of $p_i \in P$ and $q_j \in Q$ are aligned. Then the set P is transformed with respect to Q using the three degrees of freedom left,

which are the rotations θ , φ and ψ around the three coordinate axis. The center of rotation is the location corresponding to the pair of RSFVs being aligned. An iterative scheme `THREEDOFITER` is used to solve the corresponding three degree of freedom alignment problem. The iterative scheme, defined in Section 6.5, will iterate through the algorithm `COMPUTEANGLE` applied to each of the three rotations θ , φ , and ψ around the three coordinate axis. The algorithm `COMPUTEANGLE`, described in Section 6.4, can solve separately the alignment problem for each degree of freedom. The value of the distance function corresponding to the outcome of the iterative scheme is the minimum value of the distance function for a particular RSFV pair alignment. Now, the next alignment is considered and the procedure is repeated. The output is the minimum value of the distance over all the RSFV pair alignments. The overall algorithm is given below.

Algorithm: `COMPUTESIMILARITYMEASURE_TWO`

Input:

- Parts M_P and M_Q .

Output:

- Degree of similarity between M_P and M_Q based on the distance function defined in Equation (6.2).

Steps:

1. Let P and Q be the RSFV sets corresponding to M_P and M_Q .
2. Initialize $d_{min} = \text{Infinity}$.
3. For each RSFV p_i of P , do the following.
 - a. Initialize $(d_{min})_i = \text{Infinity}$.
 - b. For each RSFV q_j of Q , do the following.

-
- i. If $p_i \in P$ and $q_j \in Q$ are of the same type, translate P using the transformation matrix \mathbf{T}_{ij} such that p_i aligns with q_j .
 - ii. Else go to next value of j in Step 3b.
 - iii. Compute the minimum distance value $(d_{min})_{i,j}$ using the algorithm **THREEDOFITER**.
 - iv. If $(d_{min})_i$ is greater than $(d_{min})_{i,j}$ then $(d_{min})_i = (d_{min})_{i,j}$.
- c. If d_{min} is greater than $(d_{min})_i$ then $d_{min} = (d_{min})_i$.
4. Return d_{min} .
-

6.4 Finding The Optimal Alignment Under One Degree Of Freedom Rotations

The algorithm **COMPUTEANGLE** finds the angle θ that minimizes the distance function given by Equation (6.2) between two sets of RSFVs in \mathbb{R}^3 . The angle θ represents a rotation around a fixed axis that can be any of the coordinate axes: the algorithm solves the one degree of freedom problem. The one independent variable of the problem is the rotation θ applied to one of the two sets. In order to describe the algorithm the rotation θ about Z axis will be considered. Clearly the algorithm can be applied to the rotations φ and ψ about Y axis and X axis as well. The overall algorithm is given below.

Algorithm: **COMPUTEANGLE**

Input:

- Sets P and Q of RSFVs.

Output:

- Angle θ_{min} that minimizes the distance function defined in Equation (6.2).

Steps:

-
- a. Partition the theta range $[0, 2\pi]$ into theta intervals such that the closest neighbor $q_j \in Q$ to each RSFV $p_i \in P$ is invariant within each interval by using the algorithm `FINDINVARIANTCLOSESTNEIGHBORS_TWO`
 - b. Within each theta interval c obtained from Step a compute the value of the rotation $\theta(c)$ that minimizes the distance function defined in Equation (6.2) for interval c .
 - c. Find interval c^* such that the distance function defined in Equation (6.2) reaches the minimum value over all the intervals obtained in Step a.
 - d. Return the corresponding value $\theta_{min} = \theta(c^*)$ of the rotation for the interval c^* found in Step c.

Note that many steps of algorithms `COMPUTEANGLE` and `FINDINVARIANTCLOSESTNEIGHBORS_TWO` defined in this chapter are coincident to the steps of algorithms `COMPUTETHETA` and `FINDINVARIANTCLOSESTNEIGHBORS` described in Chapter 5. However in this chapter RSFVs in \mathbb{R}^3 , while in Chapter 5 RFVs on the unit sphere were aligned. Hence there are some differences, as RSFVs are mapped to the attributed applied vectors in \mathbb{R}^3 while RFVs are mapped to the attributed points on the unit sphere.

In the following subsections the steps of algorithms `COMPUTEANGLE` and `FINDINVARIANTCLOSESTNEIGHBORS_TWO` will be described.

6.4.1 Step a: Building The Set Of Theta Intervals For The RSFVs Of Set P

To compute the distance value in Equation (6.2), the closest neighbor $q_j \in Q$ to each $p_i \in P$ needs to be determined. The closest neighbor $q_j \in Q$ to each $p_i \in P$ changes with the

rotation of set P with respect to set Q . Thus, the closest neighbors for each $p_i \in P$ need to be obtained by taking into account the rotation θ around the fixed axis as explained in the previous section. It is necessary to know, for each value of the rotation θ , the closest RSFV $q_j \in Q$ to each RSFV $p_i \in P$. The closest neighbor to each RSFV of P changes only at specific values of θ . Thus, the theta range $[0, 2\pi]$ can be partitioned into a set of theta intervals within which the closest neighbor to each RSFV of P is known and invariant. The following algorithm is used for this purpose.

Algorithm: FINDINVARIANTCLOSESTNEIGHBORS_TWO

Input:

- Sets P and Q of RSFVs.

Output:

- Set of theta intervals and for each interval the closest neighbor to every RSFV of P from set Q .

Steps:

1. For each RSFV p_i of P , do the following.
 - a. For each possible pair of distinct RSFVs q_k and q_l of Q , do the following. Partition the theta range $[0, 2\pi]$ into subintervals within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. The partitioning is performed by finding the values of θ such that $d(p_i, q_k) = d(p_i, q_l)$, where d is the distance function defined in Equation (6.3). This step can be carried out analytically and it will be described in more detail after the description of the overall algorithm.
 - b. Overlap the intersecting subintervals obtained in Step 1.a so that the range

$[0, 2\pi]$ is further partitioned into a set of intervals.

- c. For each interval being obtained in Step 1.b, do the following. Using the closest neighbors being obtained in Step 1.a, find the RSFV q_j of Q such that $d(p_i, q_j)$ is minimum over all the RSFVs of Q .
2. Overlap the set of intersecting intervals being obtained in Step 1 for each RSFV p_i of P . Within the set of intervals being obtained the closest neighbor to every RSFV of P from set Q is invariant and known.

The algorithm described previously yields the set of theta intervals for the RSFVs of P . In the next paragraphs Step 1.a and Step 2 will be explained in detail.

In Step 1.a, the closest neighbors for each RSFV $p_i \in P$ need to be obtained by using the distance function defined in Equation (6.2). The distance function accounts for the relevant surface patch attributes. The transformation-invariant attributes need to be considered in obtaining the closest neighbors. The task is carried out analytically as follows. In order to partition the theta range $[0, 2\pi]$ into subintervals within which either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$, it is necessary to find the values of θ such that $d(p_i, q_k) = d(p_i, q_l)$. If there are not such values, it is either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$ for all the values of θ . It is possible to verify that the values of θ are obtained solving the following equation:

$$A \cos \theta + B \sin \theta = C \tag{6.5}$$

The constant values A , B and C depend on the initial location and orientation of the RSFVs considered, on the center of rotation considered and on the transformation-invariant attributes of the RSFVs considered. In Appendix C more details on how to obtain Equation (6.5) will be given. The values of the angle θ that are obtained from

Equation (6.5) will partition the theta range $[0, 2\pi]$ into subintervals within which it is easy to verify whether $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$. In some cases Equation (6.5) might not have any real number solution for θ . In this case it is either $d(p_i, q_k) > d(p_i, q_l)$ or $d(p_i, q_k) < d(p_i, q_l)$, which can be verified by substituting in Equation (6.5) any real number value for θ .

Observe that Step 1 of the algorithm `FINDOPTIMALNEIGHBOR_2` yields the closest neighbors for each RSFV of P separately. A set of theta intervals is built for a particular RSFV $p_i \in P$ such that in each interval the closest RSFV of Q to p_i is known. In Figure 6.4 the set of theta intervals within the range $[0, 2\pi]$ for the RSFV $p_1 \in P$ is shown. Thus several sets of theta intervals are obtained, one for each RSFV of P . The overlapping of the sets of theta intervals being performed in Step 2 yields the set of theta intervals for the RSFVs of P . Within each of the intervals the distance given by Equation (6.2) can be minimized using closed form mathematical formulae. The only independent variable in the formulae is rotation θ . The single sets of theta intervals for each RSFV of P are combined into the set of theta intervals for the RSFVs of P by overlapping so that the resulting range $[0, 2\pi]$ is further partitioned into intervals. Each of the resulting intervals is obtained from the intersection of the intervals of the initial sets of intervals. Figure 6.5 shows two sets of intervals that are overlapped. One set of intervals is the set of theta intervals of RSFV p_1 of set P (see Figure 6.5(a)), the other one is the set of theta intervals of RSFV p_2 of set P (see Figure 6.5(b)). The interval c , indicated in Figure 6.5(c) by an arrow point, is clearly contained in one of the intervals of each of the two sets of theta intervals that have been overlapped. As shown in Figure 6.5(a) and Figure 6.5(b), the intervals c_1 and c_2 overlap to generate interval c . Thus, interval c represents a region in

the set of theta intervals for the RSFVs of P . Within c , q_1 is the closest neighbor to p_1 and q_2 is the closest neighbor to p_2 . Each point of c corresponds to a transformation applied to the set of RSFVs P while Q is fixed. Thus, within any interval of the set of theta intervals for the RSFVs of P , the closest RSFV of Q to each RSFV in P is known. The distance function defined in Equation (6.2) can now be computed for each interval. The distance function defined in Equation (6.2) for each interval can be expressed as a function of the location coordinates (x, y, z) and the orientation components (v_x, v_y, v_z) of the RSFVs of P and Q . The location coordinates and the orientation components of P and Q can be expressed as a function of θ , which is the angle of rotation. Thus the distance function defined in Equation (6.2) is expressed as a function of θ as explained in the next subsection.

6.4.2 Step b: Minimization Of The Distance Function Within A Given Theta Interval

Consider the location $(x^{p_i}, y^{p_i}, z^{p_i})$ and the orientation $(v_x^{p_i}, v_y^{p_i}, v_z^{p_i})$ of a RSFV p_i in \mathbb{R}^3 .

Let $v_{zo}^{p_i}$ be the initial Z component of the orientation for attributed point $p_i \in P$, while

$v_{xyo}^{p_i} = \sqrt{(v_{xo}^{p_i})^2 + (v_{yo}^{p_i})^2}$ is the initial component in the coordinate plane XY before

applying algorithm COMPUTEANGLE. Let (x_B, y_B) be the center of rotation. Define $\theta_o^{p_i}$ as

the known initial angle of each RSFV $p_i \in P$ with respect to the center of rotation before

applying algorithm COMPUTEANGLE. Similarly let d_{zi} be the Z component and d_{xyi} the XY

component of the Euclidean distance between each RSFV $p_i \in P$ and the center of rotation.

Let also $\theta_{vo}^{p_i}$ be the known initial angle of the XY component of the orientation of each

RSFV $p_i \in P$ with X axis before applying algorithm COMPUTEANGLE. The angle and

component values defined previously refer to the positions of RSFVs of P and Q after the initial alignment described in Section 6.3.

In the previous subsection the set of theta intervals for all the RSFVs of P was built by overlapping the single sets of theta intervals of each RSFV. The range $[0, 2\pi]$ is thus partitioned into a number of intervals. Within each interval the closest RSFV in Q to each of the RSFVs in P is known. The following definitions, valid within each single interval, will be used.

$$\left\{ \begin{array}{l} x^{q_j}(i) = x \text{ coordinate of the position of the closest RSFV } q_j(i) \in Q \text{ to RSFV } p_i \in P \\ y^{q_j}(i) = y \text{ coordinate of the position of the closest RSFV } q_j(i) \in Q \text{ to RSFV } p_i \in P \\ z^{q_j}(i) = z \text{ coordinate of the position of the closest RSFV } q_j(i) \in Q \text{ to RSFV } p_i \in P \\ v_x^{q_j}(i) = x \text{ component of the orientation } v^{q_j}(i) \text{ of the closest RSFV to RSFV } p_i \in P \\ v_y^{q_j}(i) = y \text{ component of the orientation } v^{q_j}(i) \text{ of the closest RSFV to RSFV } p_i \in P \\ v_z^{q_j}(i) = z \text{ component of the orientation } v^{q_j}(i) \text{ of the closest RSFV to RSFV } p_i \in P \end{array} \right. \quad (6.6)$$

Consider a single theta interval and a moving RSFV $p_i \in P$. Let θ be the rotation applied to the RSFVs of set P . Then,

$$\left\{ \begin{array}{l} x^{p_i}(\theta) = x_B + d_{xyi} \cos(\theta_o^{p_i} + \theta) \\ y^{p_i}(\theta) = y_B + d_{xyi} \sin(\theta_o^{p_i} + \theta) \\ z^{p_i} = z_B + d_{zi} \\ v_x^{p_i}(\theta) = v_{xyo}^{p_i} \cos(\theta_{vo}^{p_i} + \theta) \\ v_y^{p_i}(\theta) = v_{xyo}^{p_i} \sin(\theta_{vo}^{p_i} + \theta) \\ v_z^{p_i} = v_{zo}^{p_i} \end{array} \right. \quad \forall \text{ RSFV } p_i \in P \quad (6.7)$$

Within a single interval, it is necessary to compute $\bar{d}(P, Q)$ as a function of the transformation θ .

$$\bar{d}(\theta) = \frac{\sum_{i=1}^n \left\{ \begin{aligned} &w_L \left[(x^{p_i}(\theta) - x^{q_j(i)})^2 + (y^{p_i}(\theta) - y^{q_j(i)})^2 + (z^{p_i} - z^{q_j(i)})^2 \right] + \\ &w_O \left[(v_x^{p_i}(\theta) - v_x^{q_j(i)})^2 + (v_y^{p_i}(\theta) - v_y^{q_j(i)})^2 + (v_z^{p_i} - v_z^{q_j(i)})^2 \right] + \\ &(1 - \delta(p, q)) \left[w_A (A(p_i) - A(q_j(i)))^2 + w_{\sigma_o} (\sigma_o(p_i) - \sigma_o(q_j(i)))^2 + \right. \\ &\left. w_{\mu_c} (\mu(p_i) - \mu(q_j(i)))^2 + w_{\sigma_c} (\sigma_c(p_i) - \sigma_c(q_j(i)))^2 \right] + \\ &w_T \delta(p_i, q_j(i)) \end{aligned} \right\}}{n} \quad (6.8)$$

Using the notations introduced in Equations (6.6) and (6.7), Equation (6.8) can be simplified to,

$$\bar{d}(\theta) = \frac{\sum_{i=1}^n \left\{ \begin{aligned} &w_L \left[(x_B + d_{xyi} \cos(\theta_o^{p_i} + \theta) - x^{q_j(i)})^2 + \right. \\ &\left. (y_B + d_{xyi} \sin(\theta_o^{p_i} + \theta) - y^{q_j(i)})^2 + (z_B + d_{zi} - z^{q_j(i)})^2 \right] + \\ &w_O \left[(v_{xyo}^{p_i} \cos(\theta_o^{p_i} + \theta) - v_x^{q_j(i)})^2 + (v_{xyo}^{p_i} \sin(\theta_o^{p_i} + \theta) - v_y^{q_j(i)})^2 + (v_{zo}^{p_i} - v_z^{q_j(i)})^2 \right] + \\ &(1 - \delta(p, q)) \left[w_A (A(p_i) - A(q_j(i)))^2 + w_{\sigma_o} (\sigma_o(p_i) - \sigma_o(q_j(i)))^2 + \right. \\ &\left. w_{\mu_c} (\mu(p_i) - \mu(q_j(i)))^2 + w_{\sigma_c} (\sigma_c(p_i) - \sigma_c(q_j(i)))^2 \right] + \\ &w_T \delta(p_i, q_j(i)) \end{aligned} \right\}}{n} \quad (6.9)$$

In order to minimize $\bar{d}(\theta)$ its derivative with respect to θ must be set to zero. By doing this and simplifying, we get the following expression.

$$tg(\theta) = \frac{\sum_{i=1}^n \left(\begin{aligned} &w_L [y_B - y^{q_j(i)}] d_{xyi} \cos(\theta_o^{p_i}) - w_L [x_B - x^{q_j(i)}] d_{xyi} \sin(\theta_o^{p_i}) - \\ &-w_O v_y^{q_j(i)} v_{xyo}^{p_i} \cos(\theta_o^{p_i}) + w_O v_x^{q_j(i)} v_{xyo}^{p_i} \sin(\theta_o^{p_i}) \end{aligned} \right)}{\sum_{i=1}^n \left(\begin{aligned} &w_L [x_B - x^{q_j(i)}] d_{xyi} \cos(\theta_o^{p_i}) + w_L [y_B - y^{q_j(i)}] d_{xyi} \sin(\theta_o^{p_i}) - \\ &-w_O v_x^{q_j(i)} v_{xyo}^{p_i} \cos(\theta_o^{p_i}) - w_O v_y^{q_j(i)} v_{xyo}^{p_i} \sin(\theta_o^{p_i}) \end{aligned} \right)} \quad (6.10)$$

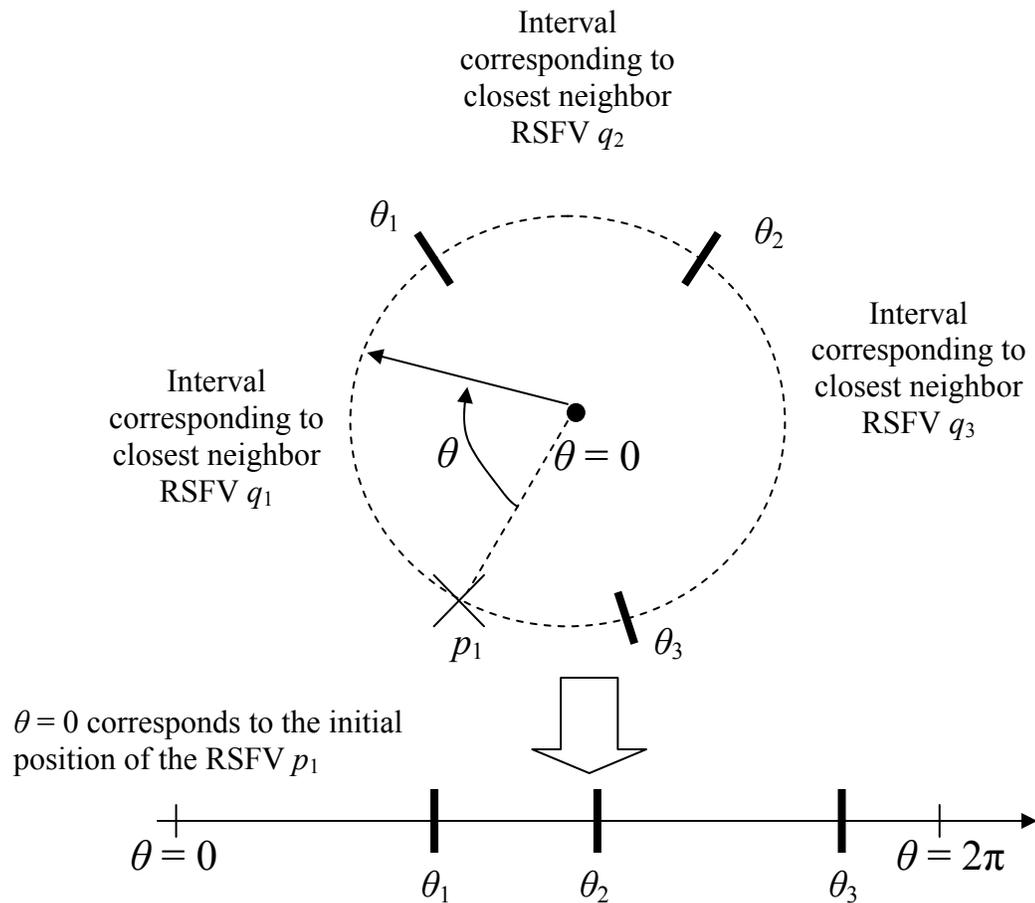
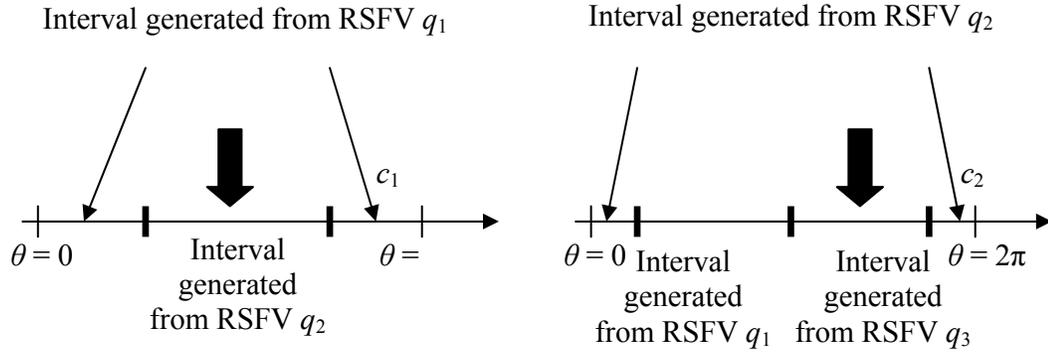
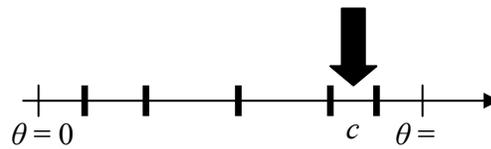


Figure 6.4: Set of Theta Intervals for Reduced Surface Feature Vector p_1 of P

Observe that the distance function defined in Equation (6.9) is a continuous function, and it is also bounded. The values of θ resulting from Equation (6.10) can identify local minima or local maxima of the distance function, depending on the sign of the second derivative. Hence it is necessary to check the sign of the second derivative by substituting the values of θ resulting from Equation (6.10) in the second derivative of the distance function defined in Equation (6.9). The values of θ that yield a positive value for the second derivative are local minima. Among them the θ value corresponding to the global minimum will be chosen.



(a): Set of Theta Intervals of RSFV p_1 of P (b): Set of Theta Intervals of RSFV p_2 of P



(c): Set of Theta Intervals Resulting From Overlapping of Sets of Theta Intervals (a) and (b)

Figure 6.5: Example of Set of Theta Intervals Resulting From Overlapping of Two Sets of Theta Intervals

Equation (6.10) yields the transformation θ , applied to the set of RSFVs P , which minimizes the distance between the sets of RSFVs P and Q . This value of the transformation is valid only within a single interval of the set of theta intervals for all the RSFVs of P . In general the value of θ that is found is not guaranteed to lie in the interval where the distance function is defined. Values of θ that lie outside the corresponding interval have no physical meaning and should be discarded. In fact Theorem 1 guarantees that none of them will be the θ value corresponding to the global minimum over all the intervals.

Equation (6.10) has been obtained by differentiating the distance function with respect to θ , which is a standard minimization technique in the continuous domain. Thus,

the transformation value obtained for an interval c of the set of theta intervals for all the RSFVs of P yields the best possible alignment between the two RSFV sets for all permissible transformations within the interval c .

6.4.3 Steps c and d: Computing The Value Of Theta That Minimizes The Distance Over All The Theta Intervals

The value of $\theta(c)$ obtained in the Equation (6.10) yields the rotation that minimizes the distance between the two RSFV sets P and Q within a single interval c of the set of theta intervals for all the RSFVs of P . To obtain the corresponding value of the distance $\bar{d}(c)$ it is sufficient to substitute the value of θ obtained from Equation (6.10) into Equation (6.9). Hence, for each interval, $\bar{d}(c)$ is the minimum distance. Finally Step d of the algorithm COMPUTETHETA involves finding the value of θ corresponding to the minimum distance over all the intervals. The minimum distance over all the intervals is obtained as in Subsection 3.5.3. The same formulae can be used and the same considerations are valid. They are reported for clarity as follows.

$$\bar{d}_{\min} = \min_{c \in C} \bar{d}(c) \quad (6.11)$$

where C is the set of all the intervals c of the partitioned theta range $[0, 2\pi]$. Equation (6.11) yields the minimum distance between sets P and Q . The corresponding rotation θ_{\min} is found as follows.

$$\theta_{\min} = \theta(c^*) \quad (6.12)$$

where c^* is the interval in which the minimum distance was found.

Equation (6.12) yields the rotation to apply to P in order to minimize the distance between P and Q and Equation (6.11) yields the minimum distance between two sets of RSFVs equivalent to applied vectors in \mathbb{R}^3 under one degree of freedom rotation.

6.5 Iterative Schemes To Find Optimal Alignment Under Three Rotational Degrees Of Freedom

In this section the algorithm `THREEDOFITER` used in Step 3.b.iii. of the algorithm `COMPUTESIMILARITYMEASURE_TWO` is described in detail. As explained in Section 6.3, a translation $\mathbf{T}_{i,j}$ has been applied to the two sets P and Q such that the locations of the pairs of two attributed applied vectors of the same type $p_i \in P$ and $q_j \in Q$ are aligned. Consequently the alignment problem to be solved involves only three rotational DOFs. The three rotational DOFs involved correspond to the three rotations θ , φ , and ψ around the three coordinate axis. Consider the corresponding transformation matrix $\mathbf{R} = (\theta, \varphi, \psi)$. The transformations $\mathbf{R}^1 \in \Gamma^1$, $\mathbf{R}^2 \in \Gamma^2$ and $\mathbf{R}^3 \in \Gamma^3$ are such that two of the components of \mathbf{R} are zero: $\mathbf{R}^1 = (\theta, 0, 0)$, $\mathbf{R}^2 = (0, \varphi, 0)$ and $\mathbf{R}^3 = (0, 0, \psi)$. The algorithm `COMPUTEANGLE` can perform alignment between P and Q in the lower dimension transformation spaces Γ^1 , Γ^2 and Γ^3 . Define the three optimal alignment algorithms based on partitioning of the transformation space `ALIGN- \mathbf{R}^i` with $i = 1, 2, 3$ as the optimal alignment algorithm `COMPUTEANGLE` under the transformation \mathbf{R}^i . The following notation introduced in Chapter 4 describes the effect of alignment.

$$P' = \text{ALIGN-}\mathbf{T}_i^s(P, Q)$$

where P' is rotated P after applying algorithm `COMPUTEANGLE`. Now consider the following sequence of algorithms.

$$(P_1 = \text{ALIGN-}\mathbf{R}^1(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^2(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^3(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^1(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^2(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^3(P_{k-1},Q)) \quad (6.13)$$

This sequence terminates when the following condition is met.

$$|\bar{d}(P_k, Q) - \bar{d}(P', Q)| < \varepsilon \quad (6.14)$$

The sequence defined in Equations (6.13) is an iterative strategy I_1 . There are $3!=6$ possible sequences of the three optimal alignment algorithms $\text{ALIGN-}\mathbf{R}^1$, $\text{ALIGN-}\mathbf{R}^2$ and $\text{ALIGN-}\mathbf{R}^3$, each corresponding to an iterative strategy I_i . The first iterative strategy I_1 has already been defined in Equations (6.13). The remaining five are listed as follows.

$$I_2 = (P_1 = \text{ALIGN-}\mathbf{R}^1(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^3(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^2(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^1(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^3(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^2(P_{k-1},Q))$$

$$I_3 = (P_1 = \text{ALIGN-}\mathbf{R}^2(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^1(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^3(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^2(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^1(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^3(P_{k-1},Q))$$

$$I_4 = (P_1 = \text{ALIGN-}\mathbf{R}^2(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^3(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^1(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^2(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^3(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^1(P_{k-1},Q))$$

$$I_5 = (P_1 = \text{ALIGN-}\mathbf{R}^3(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^1(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^2(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^3(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^1(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^2(P_{k-1},Q))$$

$$I_6 = (P_1 = \text{ALIGN-}\mathbf{R}^3(P,Q), P_2 = \text{ALIGN-}\mathbf{R}^2(P_1, Q), P_3 = \text{ALIGN-}\mathbf{R}^1(P_2, Q), P_4 = \text{ALIGN-}\mathbf{R}^3(P_3,Q), \dots, P_{k-1} = \text{ALIGN-}\mathbf{R}^2(P_{k-2},Q), P_k = \text{ALIGN-}\mathbf{R}^1(P_{k-1},Q)) \quad (6.15)$$

The algorithm `THREEDOFITER` aligns two sets of RSFVs under the three rotations θ , φ , and ψ for each pair of RSFVs $p_i \in P$ and $q_j \in Q$ of the same type being matched by

using the translation \mathbf{T}_{ij} . Algorithm THREEDOFITER uses the previously defined iterative schemes I_i .

Observe that the iterative schemes I_i are not guaranteed to optimize the distance function. They do not necessarily lead to a local minimum either. In order to reach the global minimum and so optimize the distance function, it is necessary to start from a number r of different initial conditions. The higher the number r of initial conditions being used, the higher the chances of optimizing the distance function by obtaining the global minimum. On the other hand, the complexity of the algorithm increases with the number of initial conditions used. Hence, in choosing the number of initial conditions r to use, it is necessary to tradeoff between the complexity of the corresponding algorithm and the accuracy of the outcome.

Using the iterative strategies defined in Equations (6.13) and (6.15) it is now possible to define the algorithm THREEDOFITER used in Step 3.b.iii of algorithm COMPUTESIMILARITYMEASURE_TWO. The algorithm is described in detail as follows.

Algorithm: THREEDOFITER

Input:

- Parts M_P and M_Q and number of different initial conditions r .

Output:

- Minimum distance value d_{min} between M_P and M_Q based on the distance function defined in Equation (6.2).

Steps:

1. Let P and Q be the RSFV sets corresponding to M_P and M_Q
 2. Initialize $d_{min} = \text{Infinity}$.
-

-
3. For $i = 1$ to 6, do the following.
 - a. Apply iterative scheme I_i to sets P and Q starting from r different initial conditions of set P and obtaining for the j -th each initial condition the distance d_j .
 - b. Among the obtained distances d_j with $j = 1, 2, 3, \dots, r$ find the minimum distance d .
 - c. If the minimum distance $d < d_{min}$ then $d_{min} = d$.
 - d. If $d_{min} = 0$ go to Step 4.
 4. Return d_{min} .
-

In the next section experimental results are presented.

6.6 Experimental Results

In this section it is verified experimentally whether iterative schemes I_i can be used to solve the attributed applied vectors alignment problems. Also, extensive experiments give an estimate of the number r of initial conditions needed to reach the global minimum. The experiments are similar to the ones carried out to verify the performance of iterative schemes I_{Ri}^3 on the attributed point alignment problems under three rotational DOFs described in Chapter 4. Furthermore experimental results to assess the performance of the algorithm COMPUTESIMILARITYMEASURE_TWO are presented.

6.6.1 Tests To Study The Performance Of Iterative Scheme

The first set of experiments was carried out to assess the performance of the iterative schemes defined in Section 6.5. A total of 1000 initial sets of 20 attributed applied vectors were randomly generated. In particular, the vector locations were randomly generated inside a sphere of a fixed size. The orientations and transformation-invariant

attributes were also generated randomly. Then a random transformation was applied to each of the 1000 sets, creating 1000 more sets of attributed applied vectors. Hence finally 1000 pairs of sets of applied vectors were obtained. Consider all the pairs of sets consisting of one initial set and one corresponding additional set created as explained previously. The iterative strategies I_i were applied to each pair of applied vector sets until convergence was reached following algorithm THREEDOFITER. A total of 1000 instances were evaluated. The expected minimum distance corresponding to the optimal alignment computed among the sets of each pair is 0. Cases in which the optimal alignment was not found were handled using the following procedure. A random transformation was applied to the initial set of the pair in order to create a different initial condition. Then the experiment was repeated with the different initial condition that had been obtained for those instances. This procedure was repeated until the optimal alignment was found or the limit of ten different initial conditions was reached. Out of the 1000 instances, the optimal alignment (i.e. distance = 0) was found in all of them. In Figure 6.6 a histogram representing the number of converging and non-converging instances versus the number of initial conditions used is shown.

The first set of experiments suggests that the number of initial conditions needed to obtain the optimal alignment is low. In fact, in 97.2 % of the cases only one initial condition was needed, and in the remaining 2.8 % of the cases two initial conditions were enough. This result is general as the applied vectors were randomly generated. In particular the applied vector locations were generated inside a sphere of fixed size in order not to have preferential directions or pattern in the applied vector sets. The obtained results suggest that the iterative scheme used in algorithm 3DOFITER has a good

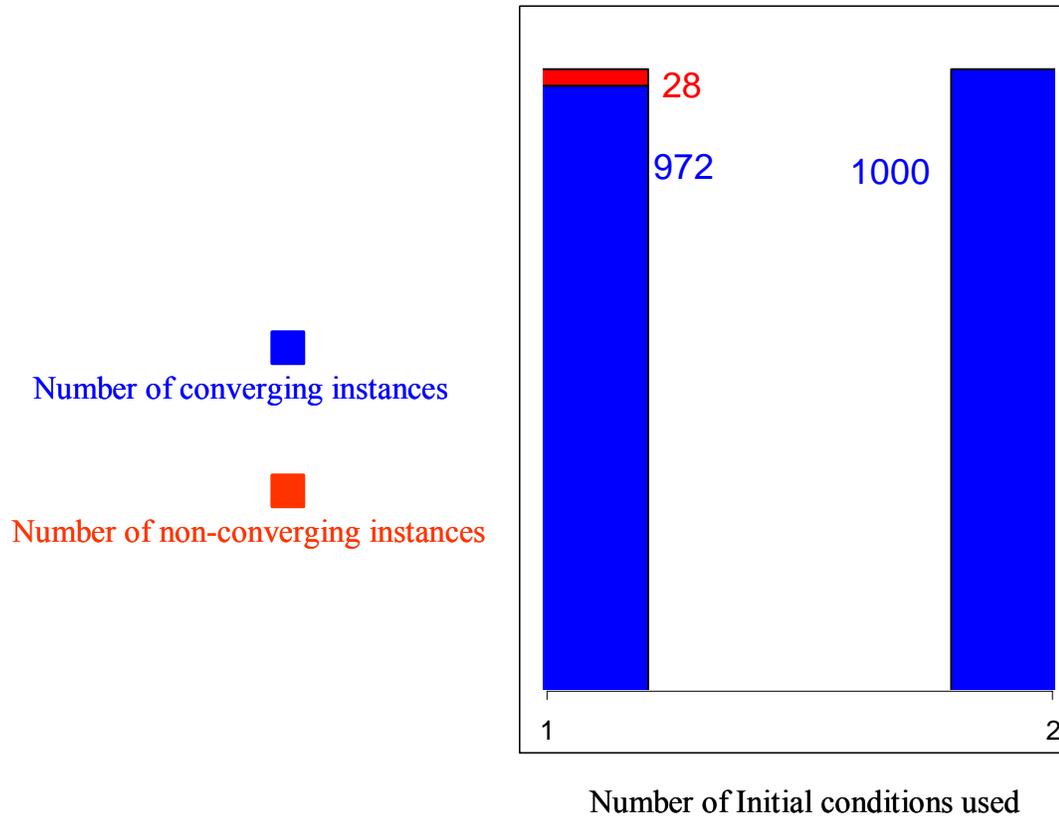


Figure 6.6: Histogram Representing Number of Converging and Non-converging Instances vs. Initial Conditions Used

performance on 3 rotational DOF attributed applied vector alignment problems under three rotations. Hence algorithm `THREEDOFITER` can be conveniently used within algorithm `COMPUTESIMILARITYMEASURE_TWO` to perform surface feature based shape similarity assessment of parts. Referring to the outcome of the first experiment we decided to use $r = 3$ initial conditions in the second experiment described in next subsection.

6.6.2 Tests On Mechanical Parts

A software system has been implemented based on the algorithms presented in this chapter in C++ programming language using Microsoft Foundation Classes (MFC) and

OpenGL on a Windows platform. The input to the system is a query part and the directory in which all the previously designed parts are stored. The system performs the alignment using the algorithms described previously and outputs those previously designed parts that are similar to the query part based on the distance function described in Section 6.2. The output models are rank ordered based on this distance function starting with the one having the smallest distance value. Surfaces feature parameters are computed from the boundary representation of the parts.

The procedure for aligning the two parts used as input to the system is illustrated using the example shown in Figure 6.7. Figure 6.7(a) shows the initial positions of two parts M_P and M_Q that are to be compared. Part M_P is obtained by randomly transforming part M_Q . The system, initially, translates part M_P such that one of its patches matches a patch of the same type of part M_P as shown in Figure 6.7(b). The system then computes the angles of rotation θ , φ and ψ such that the distance function is minimized. The final positions of the two parts are shown in Figure 6.7(c).

The database used for all the experiments consists of 150 parts and is different from the one used in Chapter 5. The weights w_L , w_O , w_A , $w_{\sigma\sigma}$, $w_{\mu c}$, $w_{\sigma c}$, and w_T are set to 1. The weights can be modified by the user to increase/decrease the influence of surface patch attributes on the distance function.

The first and second experiments test the algorithm performance by focusing on the surface patch area, the orientation, the average curvature, the standard deviations and the type. The parts being retrieved from the database will be the ones that are more similar to the query parts in these surface patch characteristics. Figures 6.8 and 6.9 show the two query parts and those parts from the database that are similar to the query parts. For each

experiment the top three matches will be shown. The value of the distance between the parts is also indicated. Let us consider Part#118 in Figure 6.9. The distance value between Part#118 and the query Part#B is $d = 0.0031$. The contribution of the location term to the distance defined in Equation (6.2) is 0.0013. There is no contribution of the orientation term to the distance. Among the transformation-invariant terms, the contribution of the area term is 0.0007 and the contribution of the average curvature term is 0.0011. Both the curvature standard deviation terms and the type term do not give contribution to the distance.

The performance of our surface feature-based shape similarity assessment algorithm was compared with a Fourier transformation based technique described in [Chak04, Chak05] that is the best-known technique for performing similarity analysis based on the boundary representation. Figures 6.8 and 6.9 show the results for our surface feature-based algorithm. Figures 6.10 and 6.11 show the results for the Fourier transformation based technique when applied to the same database. The three top matches retrieved by the Fourier transformation based technique are all similar in gross shape to the query parts. However, focusing on the shape details of the query part, it can be noticed that they are significantly different from the shape details of the retrieved parts. This did not happen using our surface feature-based shape similarity assessment algorithm, where the three top matches are very similar in surface features to the query parts as it can be noticed in Figures 6.8 and 6.9.

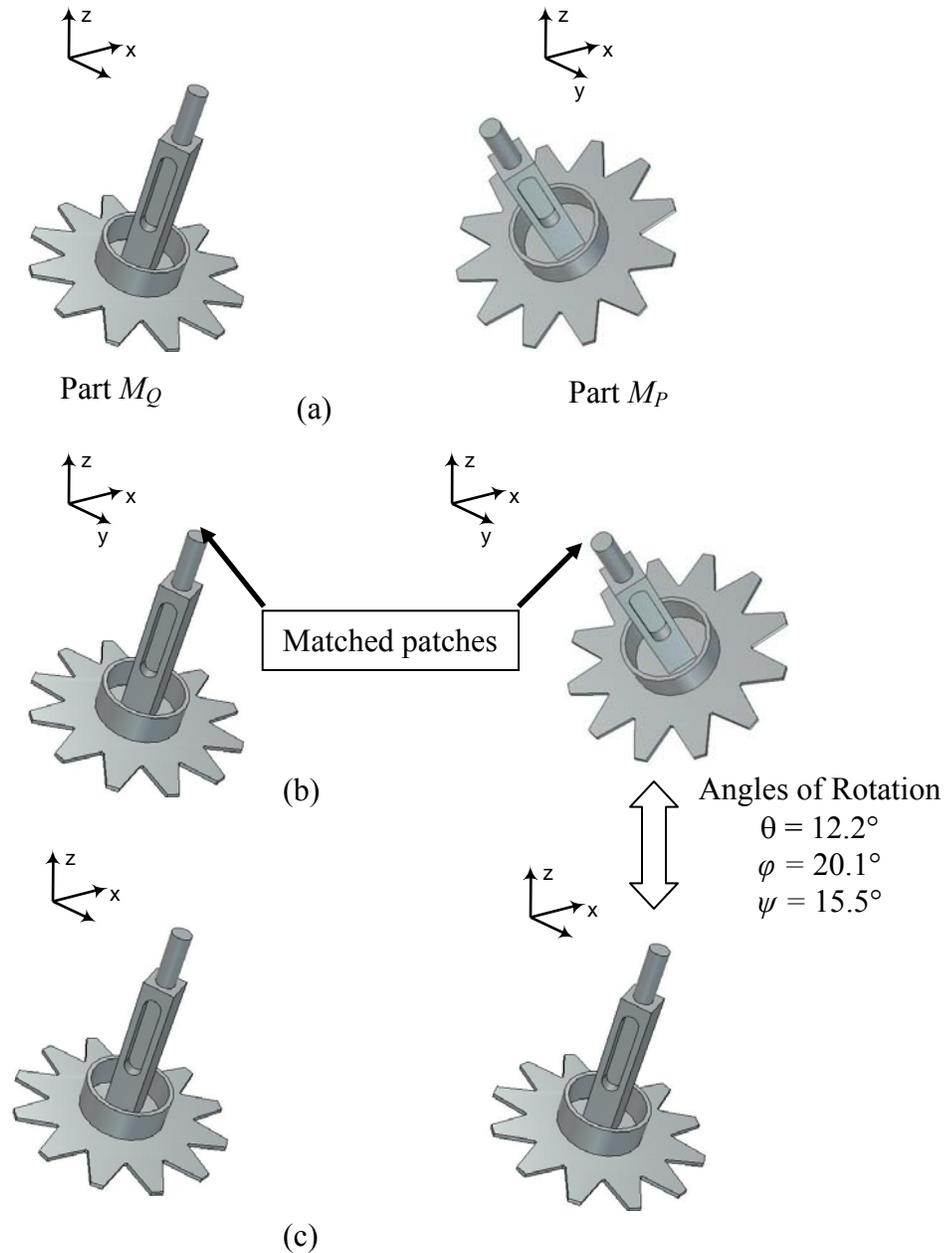


Figure 6.7: (a) Initial Position of Part M_Q and Its Randomly Transformed Version Part M_P ; (b) Position of Part M_P Before Step3b(iii) of the Algorithm COMPUTESIMILARITYMEASURE_TWO; (c) Final Position of Part M_P

As shown in Figure 6.8, Part#120 is almost identical to the query Part#A in surface features, and hence it is retrieved by the surface feature-based shape similarity assessment algorithm. However Part#120 has two cylindrical protrusions on the bottom

that make its gross shape different from the one of query Part#A. Hence Part#120 is not retrieved from the Fourier transformation based technique. Surface features of Part#120 and Part#A are similar in orientation, location, curvature, area and types, and hence they have similar molds. Hence, the tool maker of Part#120 can potentially be the tool maker of Part#A as the two parts have very similar surface feature characteristics. On the other hand Figure 6.10 shows that Part#122 has significantly different surface features from the query Part#A. Patch locations, orientation, area, curvature and type are different from query Part#A. So the mold for Part#122 is significantly different from the one used for query Part#A. Hence, our surface feature-based shape similarity assessment algorithm is more suitable for tool maker selection. Similar conclusions can be drawn from examples in Figures 6.9 and 6.11.

The third experiment assesses the performance of the algorithm by focusing on patch characteristics. The query part used in this case is Part#C. Our surface feature-based algorithm is applied to the same database using different patch characteristic weights. In the first case, location weight w_L and the area weight w_A are set to 10, while the type weight w_T is set to 0. All the other weights are set to 1. This way more importance is given to the surface patch area and the location than to the other surface patch characteristics, and the surface patch type is not taken into account. In the second case, the location weight w_L and the type weight w_T are set to 10, while the area weight w_A is set to 0. All the other weights are set to 1. This way more importance is given to the surface patch type and the location than to the other surface patch characteristics, and the surface patch area is not taken into account. Figure 6.12(a) shows the top two matches in the first case, while Figure 6.12(b) shows the top two matches in the second case. In Figure

6.12(a) the top match Part#323 is very similar to the query Part#C in surface patch area and location, but the surface patch types are significantly different. On the other hand Part#315, which is the second match, is very similar to the query Part#C in surface patch type and location but not in surface patch area. Hence Part#315 is ranked less similar to the query Part#C than Part#323 because a higher weight has been given to surface patch area than to surface patch type. The reverse reasoning can be made on the two top matches shown in Figure 6.12(b). In this case Part#315 becomes the top match, as it has a surface patch type and a location very similar to the query Part#C, and Part#323 becomes the second match as expected. This experiment shows that by modifying the weight values the user can determine the outcome of the database search. Hence, it is possible to modify the database search parameters depending on the particular part characteristics that are considered more important by the user. This gives flexibility to the surface feature-based shape similarity assessment algorithm presented.

6.7 Summary

This chapter provides algorithms for identifying those parts in a database that are similar to a given query part in surface features and hence can be potentially used as a basis for locating potential tool makers for the query part. We have developed a distance function to account for the key drivers for the surface features of a part. The selected distance function accounts not only for the explicit feature parameters such as the area, the location and the orientation, but also for features' implicit parameters such as the curvatures and the distribution of normal vectors. We have developed an algorithm that performs feature alignment to minimize this function. We have implemented the

algorithm to show the proof of the concept. We have tested the algorithm on some examples in order to assess its performance.

The surface feature-based shape similarity assessment algorithm described in this chapter can handle features having any arbitrary orientation in space. We have shown that the algorithm described in this chapter performs better than the best known technique for comparing parts based on the boundary representation.

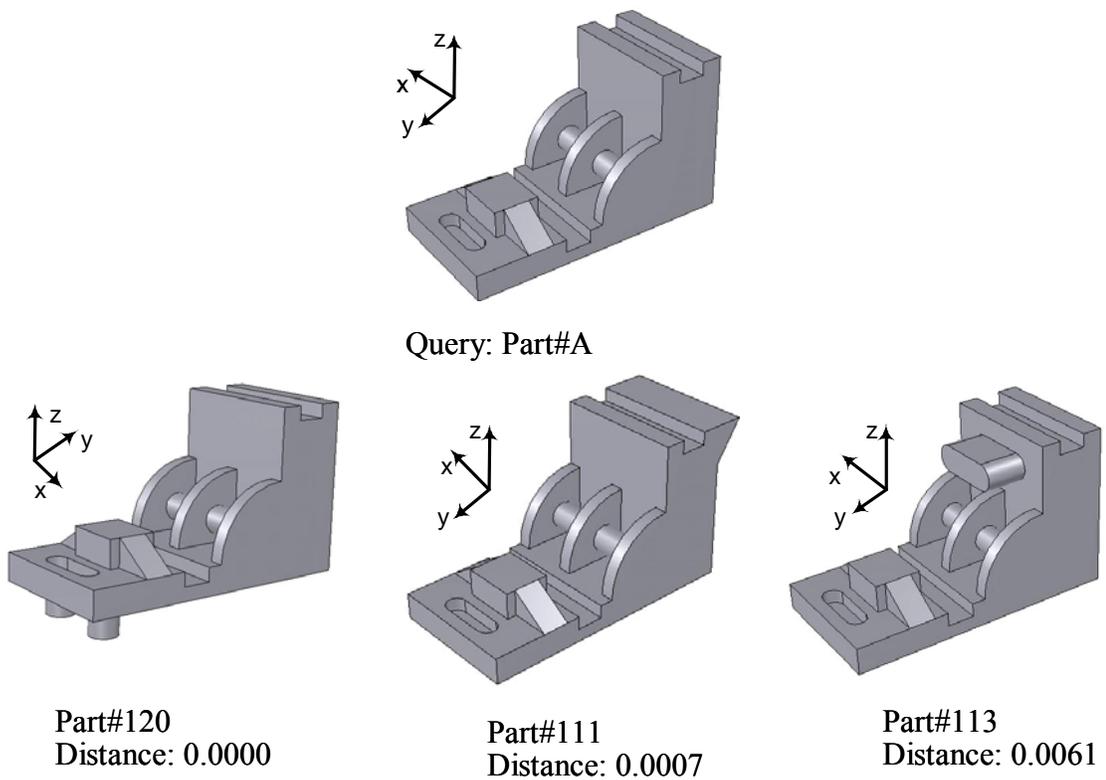


Figure 6.8: Results Obtained for Query Part#A Used As Input to the System

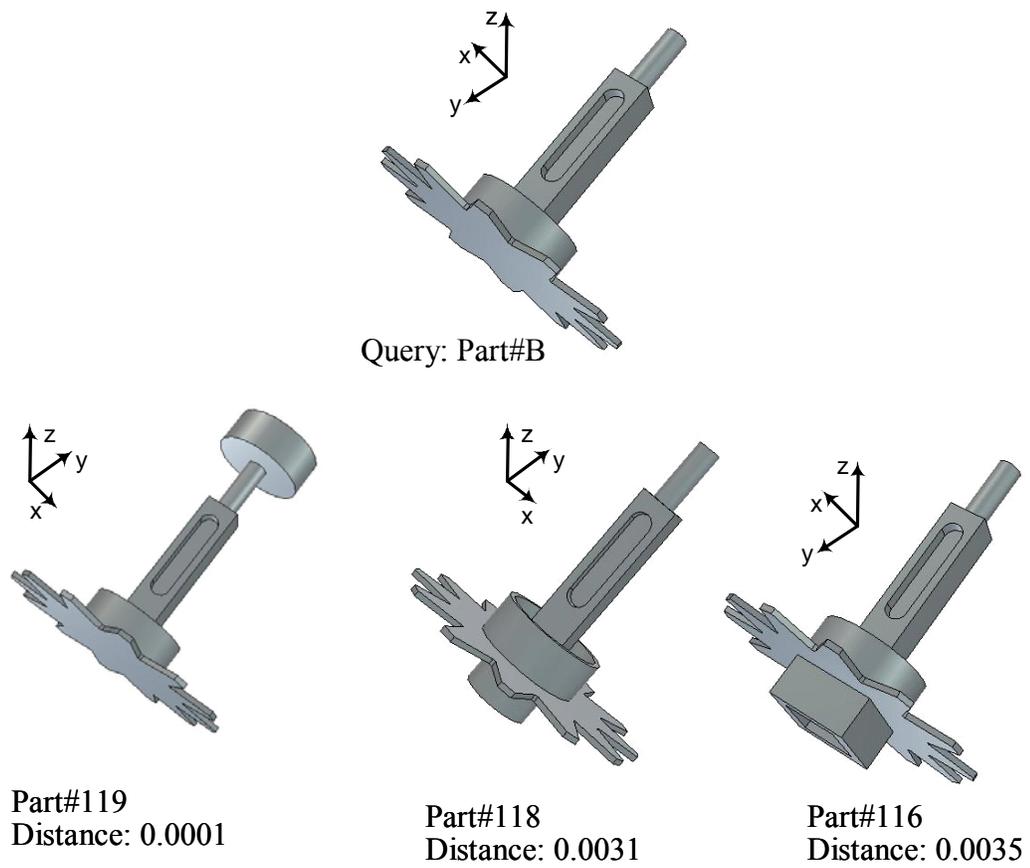
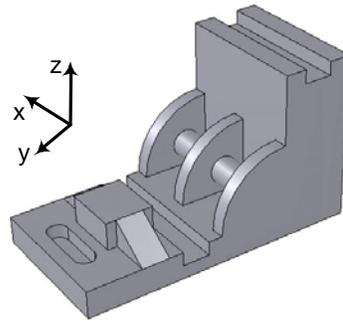
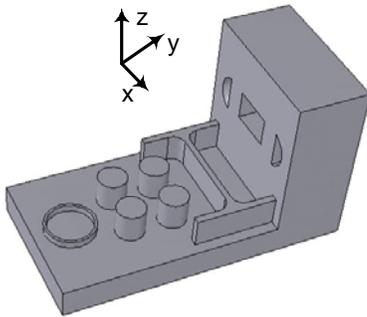


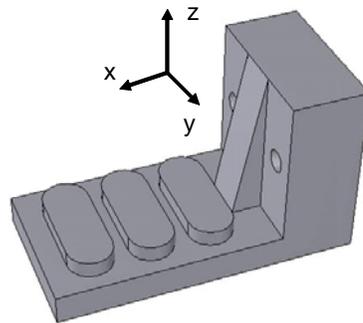
Figure 6.9: Results Obtained for Query Part#B Used As Input to the System



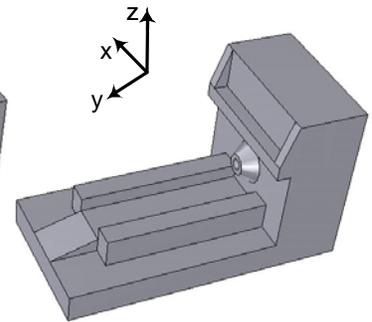
Query: Part#A



Part#122
Distance: 0.7857



Part#561
Distance: 1.0013



Part#117
Distance: 1.0609

Figure 6.10: Results Obtained for Query Part#A As Input Using Fourier Transformation Based Technique

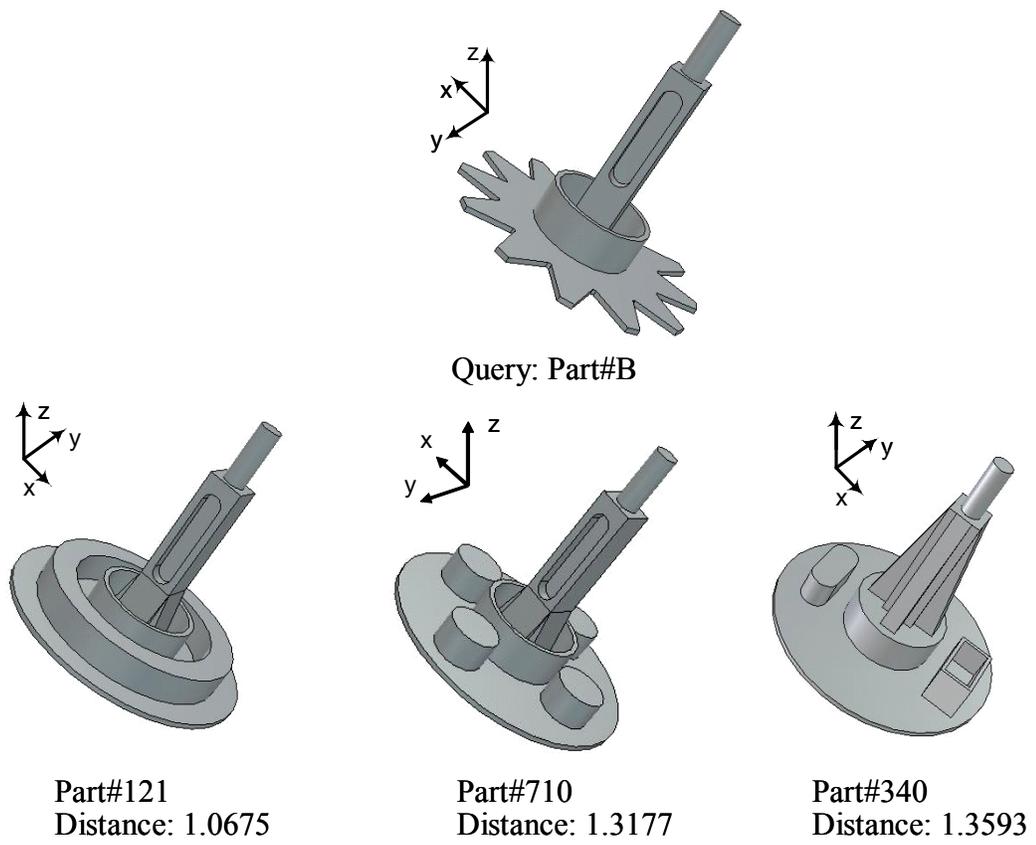
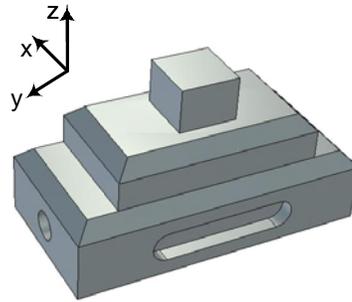


Figure 6.11: Results Obtained for Query Part#B As Input Using Fourier Transformation Based Technique



Query: Part#C

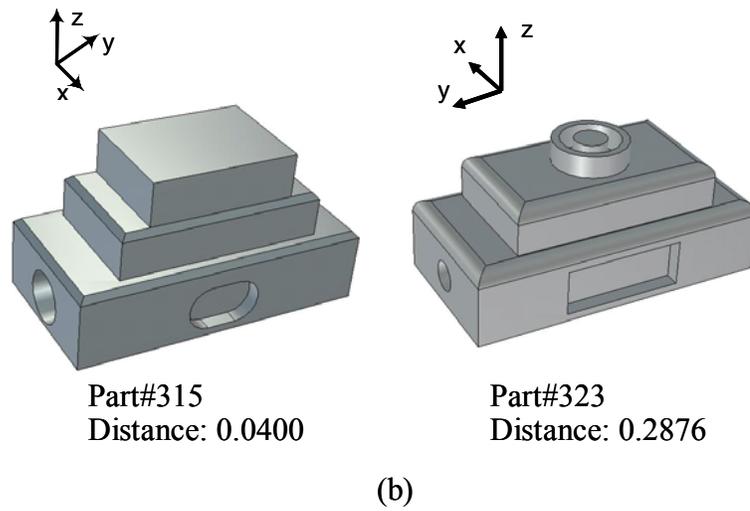
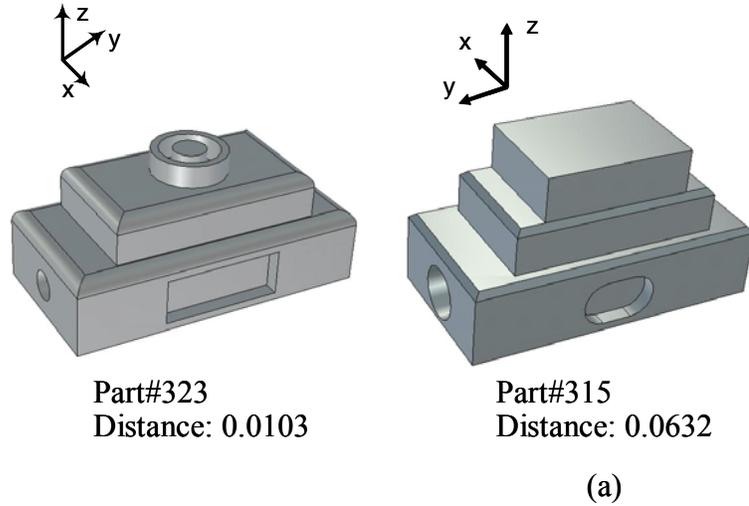


Figure 6.12: Results Obtained for Query Part#C As Input To The System; In Case (a) More Importance Is Given to Surface Patch Area and Location, in Case (b) More Importance Is Given to Surface Patch Type and Location

Chapter 7: Conclusions

This chapter is organized as follows. In Section 7.1 the intellectual contributions of this thesis are presented. In Section 7.2 the anticipated industrial benefits of the research work described in this thesis are identified. Finally, Section 7.3 suggests the future research directions resulting from this thesis.

7.1 Intellectual Contributions

This thesis makes the following intellectual contributions.

- **Optimal Feature Alignment Algorithms Based On Partitioning Of Transformation Spaces:** We have designed a new class of feature alignment algorithms based on partitioning of the transformation space. The transformation spaces corresponding to the following three transformations are used: two DOF translations in \mathbb{R}^2 , one DOF rotations in \mathbb{R}^2 and three DOF translations in \mathbb{R}^3 . The algorithms designed provide an optimal solution for the corresponding alignment problems. The optimal alignment is obtained by transforming one set of features such that a distance function between the two sets of features that are being compared is minimized. Any differentiable distance function whose form is consistent with Equation (3.1) can be used in our framework. In order to compute the distance function it is necessary to know for each feature the closest neighbor from the other set. The closest neighbor changes with the transformation applied to one of the two feature sets. As there are exponentially many closest neighbor combinations it would not be efficient to solve the problem by enumeration. Instead, in our framework, we partition the transformation space into regions such that the closest neighbors are known and invariant for each of them. Then the distance function is minimized within

each region by using standard analytical tools for optimization and finally the minimum value of the distance function over all the regions is found. The corresponding transformation is the optimal alignment. The complexity of the algorithms has been studied by assessing the complexity of spatial arrangements that are used to partition the transformation space. A low order polynomial upper bound for the spatial arrangement complexity with respect to the number of features has been found in the general case of \mathbb{R}^d spatial arrangements for well-behaved uniform feature distributions. The distance function used in our research accounts both for transformation-dependent attributes such as feature position and orientation and for transformation-invariant attributes such as feature size and type. The user can choose the number of feature parameters to take into account. The user can also assign a weight to each feature parameter. Hence the underlying distance functions are flexible and customizable.

- **Feature Alignment Algorithms Based On Iterative Strategies:** In theory, feature alignment algorithms based on partitioning of the transformation space can be used to find optimal solutions for alignment problems under transformations of any dimension. However in the case of higher dimension transformations, feature alignment algorithms based on partitioning of the transformation space involve very complex data structures. Therefore we have designed feature alignment algorithms based on iterative strategies. They solve higher dimension alignment problems by using optimal alignment algorithms based on partitioning of lower dimension transformation spaces. Optimal alignment algorithms based on partitioning of lower dimension transformation spaces represent partial optimal solutions for higher

dimension alignment problems. Therefore iterative strategies that use those partial solutions can find the optimal solution also for higher dimension alignment problems. The optimal solution corresponds to the global minimum of the selected distance function. The initial positions of the two sets being aligned are referred to as initial conditions. Initial conditions affect the performance of iterative strategies because depending on them an iterative strategy may reach a local minimum of the distance function rather than its global minimum. We have identified an iterative strategy in \mathbb{R}^2 that is guaranteed to lead to a local minimum of the distance function. We have provided empirical evidence that very few initial conditions are needed to reach the optimal alignment by performing extensive experiments in \mathbb{R}^2 and \mathbb{R}^3 . Hence alignment algorithms based on iterative strategies in \mathbb{R}^2 and \mathbb{R}^3 can be used to find the optimal solution for alignment problems under higher dimension transformations.

- **Surface Feature-Based Similarity Algorithms:** We have designed surface-feature based similarity assessment algorithms. These algorithms are capable of assessing similarity based not only on explicit feature parameters such as feature size, location or orientation, but also on implicit feature parameters such as surface curvature and distribution of normal vectors. Hence we have shown that the ideas presented in this thesis works both for explicit as well as implicit feature parameters.
- **Incorporation of Alternative Interpretations of Volumetric Features in Similarity Assessment Algorithms:** We have developed a mathematical framework that allows us to incorporate alternative interpretations of volumetric machining features in similarity assessment. This framework is applicable to parts for which

individual feature interpretations are independent of each other. It eliminates the need for considering the combinatorial enumeration of various alternative feature interpretations for parts.

7.2 Anticipated Benefits

This thesis provides a feature-based shape similarity assessment framework. This framework can be used to assess similarity between parts based on their feature characteristics. The feature characteristics can be chosen by the user depending on the application. The anticipated industrial benefits are as following.

- **Machining Feature-Based Shape Similarity Assessment:** The machining cost of parts depends on their machining feature characteristics. Cost estimators often estimate the cost of a new part by referring to similar parts whose cost has been already estimated. Searching large databases for similar parts can be time consuming. Our machining feature-based shape similarity assessment algorithm can help cost estimators in searching large databases of machined parts, automatically, in order to find the parts that are similar in machining features to the one whose cost has to be estimated. Then the cost estimators can analyze the cost of the retrieved parts in order to give an estimate of the cost of the newly designed part. This will reduce the time and effort required to locate parts in the database similar to the query part.
- **Surface Feature-Based Shape Similarity Assessment:** Nowadays many companies have global operations, and hence they use many tool makers, each specializing in different kinds of toolings. Designers rely on their own experience in choosing the most appropriate tool maker. An alternative way to identify the appropriate tool maker for a new part is to find similar parts to the new part. Tool makers used for the

similar parts can be approached by the designer to get quotes for the new part. Our surface feature-based shape similarity assessment algorithm can help designers in automatically searching large databases in order to find parts that are similar to the new one from the surface feature point of view. This will reduce the time and effort required to locate parts in the database similar to the query part. As the parts retrieved from the database are similar in surface feature characteristics to the new part, the type of tool used is potentially similar.

- **Feature-Based Shape Similarity Assessment To Search For Potentially Reusable Designs:** In many applications, designers need to use previously designed components in new designs. The use of archived design information improves the quality of the new designs by increasing their reliability and reducing part proliferation. It also decreases the cost of developing new designs. Nowadays companies are building large repositories of designs. Designers currently search these repositories manually. Searching large design repositories is time consuming. The reusable designs archived in the repository include the geometric model of the parts that have been designed. In manufacturing applications shape details of the parts to be designed are among the main elements that determine part design. For instance shape details determine the tooling needed to manufacture the part. Our feature-based shape similarity assessment algorithm can be used to automatically search repositories for the designs whose part feature characteristics are similar to the ones of the new part to be designed.
- **Feature-Based Shape Similarity Assessment To Search For Redesign Suggestion:** Archived redesign projects can provide meaningful suggestions on how to carry out

the redesign in a new project. This way redesign cost will be reduced by exploiting past redesign experiences. We expect repositories to include the models of both the initial and redesigned parts or assemblies. In manufacturing applications shape details of the parts or assemblies to be redesigned are among the main elements that determine the redesign process. For instance this applies to redesigning an assembly in order to make it manufacturable by multi-material molding. Our feature-based shape similarity assessment algorithms can be used to automatically search repositories for the previous redesign projects of parts or assemblies whose feature characteristics are similar to the part in a new redesign project. This will reduce the time and effort to locate redesign projects in large repositories.

7.3 Directions For Future Work

The following future work is suggested to overcome the limitations of the research work described in this thesis.

- **Enable Additional Optimal Alignment Algorithms Based On Partitioning Of Transformation Spaces:** Three optimal alignment algorithms based on partitioning of transformation spaces in \mathbb{R}^2 and \mathbb{R}^3 have been designed in this thesis. They partition the transformation space into regions within which the alignment problem can be solved directly. Spatial arrangements in \mathbb{R}^2 and \mathbb{R}^3 are used to partition the transformation space. The data structure for the spatial arrangements in higher dimensional transformations becomes more complex and theory for constructing them is not well studied. Therefore it is necessary to investigate if it is possible to define efficient data structures and procedures to build spatial arrangements to

- partition higher dimension transformation spaces. They will enable the optimal solution of alignment algorithms that involve higher dimension transformations.
- **Identify Characteristics Of Alignment Problems Such That The Optimal Solution Can Be Efficiently Found By Using Iterative Strategies:** The performance of iterative strategies depends on the number and distribution of local minima of the distance function. Hence it is necessary to investigate how exactly the number and distribution of local minima are affected by the characteristics of the alignment problems being addressed. In particular it is necessary to identify the characteristics of the alignment problems such that iterative strategies can provide the optimal solution by using very few initial conditions. This will enable a more efficient use of alignment algorithms based on iterative strategies to solve particular classes of alignment problems.
 - **Define Feature-Based Similarity Assessment Algorithms In Presence Of Multiple Feature Interpretations With Constraints On Feature Interpretation Combinations:** Machining feature-based shape similarity assessment algorithms have been extended to the case of multiple possible feature interpretations in this thesis. These algorithms are based on partitioning of the transformation spaces into regions such that the closest neighbor is invariant, like the algorithms designed for single feature interpretations. In this case the partitioning is built by assuming that selecting an interpretation for a feature does not depend on the interpretation chosen for some other feature. However, in some cases feature interpretations may not be independent. Therefore, algorithms developed in this thesis work need to be extended to cases in which feature interpretations need to meet certain constraints. In particular

the procedure and data structure used to define the partitioning of the transformation space need to be able to account for constraints on the possible closest neighbor combinations.

- **Develop Additional Pruning Criteria:** Pruning is a fundamental step towards efficient information retrieval from databases. The feature-based shape similarity assessment algorithms designed in this thesis are used to search part databases. They are based on optimal feature alignment algorithms. Finding the optimal alignment between two sets of features can be time-consuming if the number of features of the two sets that are being aligned is high. Therefore it is necessary to establish additional pruning criteria so that the part database search is performed in a more efficient way. The additional pruning criteria need to be consistent with the feature-based similarity assessment algorithms proposed in this thesis. In particular, they need to be customized for the distance function that is used to assess similarity between two parts.
- **Develop Clustering Techniques:** Clustering analysis is a fundamental data search technique to retrieve information from large databases. It consists of grouping database objects based on a similarity function. Then in searching databases it is possible to focus on the groups of interest. This results in a reduction of time and effort needed to search databases. Therefore it will be necessary to define clustering techniques customized for the feature-based shape similarity assessment algorithms that have been designed in this thesis. In particular the clustering criteria need to be consistent with the distance function that is used in the algorithms.

- **Perform More Extensive Tests To Assess The Performance Of The Feature-Based Shape Similarity Assessment Framework:** A feature-based shape similarity assessment framework has been developed in this thesis for two applications, and it has been tested with some examples to assess its performance. It will be useful to carry out more extensive tests in order to assess the performance of the feature-based shape similarity assessment framework developed in this thesis on different databases consisting of different types of parts that are obtained by using different manufacturing procedures. These tests will give an insight that can be used to improve the algorithms and decrease the number of false positives and false negatives.

References

- [Agar94] P. Agarwal, M. Sharir, and S. Toledo. Applications Of Parametric Searching In Geometric Optimization. *Journal Of Algorithms*, 17(3): 292-318, 1994.
- [Agar03a] P. Agarwal, S. Krishnan, N. H. Mustafa, and S. Venkatasubramanian. Streaming Geometric Optimization Using Graphics Hardware. In *Proceedings Of 11th Annual European Symposium On Algorithms*, 2003.
- [Agar03b] P. Agarwal, S. Har-Peled, M. Sharir, and Y. Wang. Hausdorff Distance Under Translations Of Points, Disks, And Balls. In *Proceeding Of 19th Annual ACM Symposium On Computational Geometry*, 282-291, 2003.
- [Alt88] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, Similarity And Symmetries Of Geometric Objects. *Discrete And Computational Geometry*, 3: 237-256, 1988.
- [Alt96] H. Alt and L.J. Guibas. Discrete Geometric Shapes: Matching, Interpolation, And Approximation: A Survey. *Technical Report B96-11, EVL-1996-142*, Institute of Computer Science, Freie Universität Berlin, December 1996.
- [Ande88] J. A. Anderson and E. Rosenfeld. Neurocomputing – Foundations Of Research. *MIT Press*, Cambridge, MA, 1988.
- [Anke99] M. Ankerst, G. Kastenmuller, H.-P. Kriegel, and T. Seidl. 3D Shape Histograms For Similarity Search And Classification In Spatial Databases. In *Proceedings Of 6th International Symposium On Large Spatial Databases*, 207-226, 1999.
- [Arbt90] K. Arbter, W.E. Snyder, H. Burkhardt, and G. Hirzinger. Application Of Affine-Invariant Fourier Descriptors To Recognition Of 3D Objects. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 12(7): 640-647, July 1990.
- [Arki91] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S. Mitchell. An Efficiently Computable Metric For Comparing Polygonal Shapes. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 13(3): 209-216, March 1991.
- [Arma93] F. Arman and J. Aggrawal. Model-Based Object Recognition In Dense-Range Images - A Review. *ACM Computing Surveys*, 25(1): 5-43, March 1993.
- [Atki87] M. D. Atkinson. An Optimal Algorithm For Geometrical Congruence. *Journal Of Algorithms*, 8(2): 159-172, 1987.
- [Aure91] F. Aurenhammer. Voronoi Diagrams - A Survey Of A Fundamental Geometric Data Structure. *ACM Computing Survey*, 23(3): 345-405, 1991.

- [Belo01] S. Belongie, J. Malik, and J. Puzicha. Matching Shapes. In Proceedings Of *8th IEEE International Conference On Computer Vision*, 454-461, Vancouver, Canada, July 2001.
- [Besp03a] D. Bespalov, W. C. Regli, and A. Shokoufandeh. Reeb Graph Based Shape Retrieval For CAD. In Proceedings Of *23rd ASME DETC Computers And Information In Engineering (CIE) Conference*, Chicago, IL, 2003.
- [Besp03b] D. Bespalov, A. Shokoufandeh, W. C. Regli, W. Sun. Scale-Space Representation Of 3D Models And Topological Matching. In Proceedings Of *Symposium On Solid Modeling And Applications*, 208-215, 2003.
- [Bias05] S. Biasotti and S. Marini. 3D Object Comparison Based On Shape Descriptors. *International Journal Of Computer Applications In Technology*, 23(2-4): 57-69, 2005.
- [Bien05] M. Bienkowski, V. Damerow, F. Meyer auf der Heide, and C. Sohler. Average Case Complexity Of Voronoi Diagrams Of N Sites From The Unit Cube. In Proceedings Of *The 21st European Workshop on Computational Geometry (EWCG)*, 167-170, Eindhoven, Holland, March 2005.
- [Burb75] J.L. Burbidge. The Introduction Of Group Technology. *Heinemann*, London, UK, 1975.
- [Camp01] R. J. Campbell and P. J. Flynn. A Survey On Free-Form Object Representation And Recognition Techniques. *Computer Vision And Image Understanding*, 81(2): 166-210, February 2001.
- [Card03] A. Cardone, S. K. Gupta, and M. Karnik. A Survey Of Shape Similarity Assessment Algorithms For Product Design And Manufacturing Applications. *Journal Of Computing And Information Science In Engineering*, 3(2):109-118, June 2003.
- [Chak04] T. Chakraborty, S. Venkataraman, and M. Sohoni. A fast 3D Shape Search Technique For 3D Cax/PDM Repositories. *Technical Paper, Society Of Manufacturing Engineers*, August 16th 2005.
- [Chak05] T. Chakraborty. Shape-Based Clustering Of Enterprise CAD Databases. *Computer Aided Design and Applications*, 2(1-4): 145-154, 2005.
- [Chen03] D. Y. Chen, X. P. Tian, Y. T. Shen, and M. Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, 22(3): 223-232, September 2003.
- [Chew97] L.P. Chew, M. T. Goodrich, D.P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric Pattern Matching Under Euclidean Motion. *Computational Geometry: Theory And Application*, 7(1-2): 113-124, January 1997.

- [Chew99] L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric Pattern Matching In D Dimensional Space. *Discrete And Computational Geometry*, 21: 257-274, 1999.
- [Chun94] Y. Chung and A. Kusiak. Grouping Parts With A Neural Network. *Journal Of Manufacturing System*, 13(4): 262 – 275, 1994.
- [Chun97] F.R.K. Chung. Spectral Graph Theory. In Proceedings Of *American Mathematical Society's Regional Conference Series In Mathematics No. 92*, Providence, RI, 1997.
- [Cici00] V. Cicirello and W.C. Regli. Managing Digital Libraries For Computer-Aided Design. *Computer Aided Design*, 32(2): 119-132, February 2000.
- [Cici01] V.A. Cicirello and W.C Regli. Machining Feature-Based Comparisons Of Mechanical Parts. In Proceedings Of *International Conference On Shape Modeling And Applications*, Genova, Italy, May 2001.
- [Cici02] V. Cicirello and W.C. Regli. An Approach To Feature-Based Comparison Of Solid Models Of Machined Parts. *Journal of Artificial Intelligence For Engineering Design Analysis And Manufacturing (AI EDAM)*, 16(5): 385-399, November 2002.
- [Corm01] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. Introduction To Algorithms. *MIT Press*, Cambridge, MA, 2001.
- [Corn03] J. Corney, H. Rea, D. Clark, J. Pritchard, R. MacLeod, and M. Breaks. Coarse Filters for Shape Matching. *IEEE Computer Graphics and Applications*, 22(3): 65-74, May/June 2003.
- [deBe97] M. deBerg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational Geometry. *Springer-Verlag*, New York, 1997.
- [Dey03] T. K. Dey, J. Giesen, and S. Goswami. Shape Segmentation And Matching With Flow Discretization. In Proceedings Of *Workshop On Algorithms And Data Structures*, 25-36, 2003.
- [Dwye91] R. A. Dwyer. Higher-Dimensional Voronoi Diagrams In Linear Expected Time. *Discrete and Computational Geometry*, 6(4): 343-367, May 1991.
- [Elad01] M. Elad, A. Tal, and S. Ar. Content Based Retrieval Of VRML Objects - An Iterative And Interactive Approach. In Proceedings Of *6th Eurographics workshop in Multimedia*, 107-111, Manchester, UK, September 2001.
- [Elin96] A. Elinson, D. Nau, and W. C. Regli. Solid Similarity Measurements. *Technical Report ISR-TR96-63*, Institute for Systems Research, University of Maryland, 1996.

- [Elin97] A. Elinson, D.S Nau, and W.C. Regli. Feature-Based Similarity Assessment Of Solid Models. In Proceedings Of *ACM Symposium On Solid Modeling And Applications*, 297-310, Atlanta, GA, 1997.
- [ElMe03a] M. El-Mehalawi and R. A. Miller. A Database System Of Mechanical Components Based On Geometric And Topological Similarity. Part I: Representation. *Computer-Aided Design*, 35(1): 83-94, 2003.
- [ElMe03b] M. El-Mehalawi and R. A. Miller. A Database System Of Mechanical Components Based On Geometric And Topological Similarity. Part II: Indexing, Retrieval, Matching, And Similarity Assessment. *Computer-Aided Design*, 35(1): 95-105, 2003.
- [From04] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing Objects in Range Data Using Regional Point Descriptors. In Proceedings Of *European Conference on Computer Vision*, 224-237, Prague, Czech Republic, May 2004.
- [Funk03] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A Search Engine For 3D Models. *ACM Transactions on Graphics*, 22(1): 83-105, January 2003.
- [Gavr99] M. Gavrilov, P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric Pattern Matching: A Performance Study. In Proceedings Of *15th ACM Symposium On Computational Geometry*, 79-85, 1999.
- [Ghos93] P. K. Ghosh. A Unified Computational Framework For Minkowski Operations. *Computers And graphics*, 17(4): 357-378, 1993.
- [Ghos96] P. K. Ghosh and R. M Haralick. Mathematical Morphological Operations Of Boundary-Represented Geometric Objects. *Journal Of Mathematical Imaging And Vision*, 6(2-3): 199-222, June 1996.
- [Good94] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky. Practical Methods For Approximate Geometric Pattern Matching Under Rigid Motion. In Proceedings Of *10th Annual ACM Symposium On Computational Geometry*, 103-112, 1994.
- [Grim85] G. R. Grimmett and D. R. Stirzaker. Probability And Random Processes. *Clarendon Press*, Oxford, UK, 1985.
- [Grun67] B. Grunbaum. Convex Polytopes. *Interscience Publishers*, 1967.
- [Gupt95] S.K. Gupta and D. Nau. A Systematic Approach For Analyzing The Manufacturability Of Machined Parts. *Computer Aided Design*, 27(5): 343 – 352, 1995.

- [Gupt99] S.K. Gupta and D.A. Bourne. Sheet Metal Bending: Generating Shared Setups. *Journal Of Manufacturing Science And Engineering*, 121(4): 689-694, November 1999.
- [Hebe95] M. Hebert, K. Ikeuchi, and H. Delingette. A Spherical Representation For Recognition Of Free-Form Surfaces. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 17(7): 681-690, July 1995.
- [Hech89] R. Hecht-Nielsen. Theory Of Backpropagation Neural Networks. In Proceedings Of *IEEE International Joint Conference On Neural Networks*, Washington D.C., June 1989.
- [Heff94] P. J. Heffernan and S. Schirra. Approximate Decision Algorithms For Point Set Congruence. *Computational Geometry: Theory And Applications*, 4(3): 137-156, 1994.
- [Herr00] J. W. Herrmann, S. Balasubramanian, and G. Singh. Defining Specialized Design Similarity Measures. *International Journal of Production Research*, 38(15): 3603-3621, 2000.
- [Hila01] M. Hilaga, Y. Shinagawa, T. Kohmura, and T.L. Kunii. Topology Matching For Fully Automatic Similarity Estimation Of 3D Shapes. In Proceedings Of *SIGGRAPH*, 203-212, Los Angeles, CA, August 2001.
- [Hutt90a] D. Huttenlocher and S. Ullman. Recognizing Solid Objects By Alignment With An Image. *International Journal Of Computer Vision*, 5(2): 195-212, 1990.
- [Hutt90b] D. Huttenlocher and K. Kedem. On Computing The Minimum Hausdorff Distance For Point Sets Under Translation. In Proceedings Of *6th ACM Symposium On Computational Geometry*, Berkeley, CA, June 1990.
- [Hutt92] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On Dynamic Voronoi Diagrams And The Minimum Hausdorff Distance For Point Sets Under Euclidean Motion In The Plane. In Proceedings Of *8th annual ACM Symposium On Computational Geometry*, 110-120, 1992.
- [Hutt93a] D. P. Huttenlocher and W. J. Rucklidge. A Multi-Resolution Technique For Comparing Images Using The Hausdorff Distance. In Proceedings Of *IEEE Conference In Computer Vision And Pattern Recognition*, 705-706, New York, NY, 1993.
- [Hutt93b] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing Images Using The Hausdorff Distance. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 15: 850-863, 1993.

- [Hutt93c] D. P. Huttenlocher, K. Kedem, and M. Sharir. The Upper Envelope Of Voronoi Surfaces And Its Applications. *Discrete And Computational Geometry*, 9: 267-291, 1993.
- [Indy99] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric Matching Under Noise: Combinatorial Bounds And Algorithms. In Proceedings Of *10th ACM-SIAM Symposium On Discrete Algorithms*, 457-465, 1999.
- [Ip02] C. Y. Ip, D. Lapadat, L. Sieger, and W.C. Regli. Using Shape Distributions To Compare Solid Models. In Proceedings Of *7th Symposium On Solid Modeling And Applications*, Saarbrücken, Germany, June 2002.
- [Ip03] C. Y. Ip, W. C. Regli, L. Sieger, and A. Shokoufandeh. Automated Learning Of Model Classifications. In Proceedings Of *8th ACM Symposium On Solid Modeling And Applications*, 322-327, Seattle, WA, June 2003.
- [Iran96] S. Irani and P. Raghavan. Combinatorial And Experimental Results For Randomized Point Matching Algorithms. In Proceedings Of *12th Annual ACM Symposium On Computational Geometry*, 68-77, 1996.
- [Iyer03] N. Iyer, Y. Kalyanaraman, K. Lou, S. Jayanti, and K. Ramani. A Reconfigurable, Intelligent 3D Engineering Shape Search System Part I: Shape Representation. In Proceedings Of *23rd ASME DETC Computers And Information In Engineering (CIE) Conference*, Chicago, IL, 2003.
- [Josh88] S. Joshi and T. C. Chang. Graph-Based Heuristics For Recognition Of Machined Features From A 3D Solid Model. *Computer Aided Design*, 20: 58-66, 1988.
- [Kapa91] S. Kaparthi and N.C. Suresh. A Neural Network System For Shape Based Classification And Coding Of Rotational Part. *International Journal Of Production Research*, 29(9): 1771-1784, 1991.
- [Karn05] M.V. Karnik, S.K. Gupta, and E.B. Magrab. Geometric Algorithms for Containment Analysis of Rotational Parts. *Computer Aided Design*, 37(2): 213-230, February 2005.
- [Keim99] D.A. Keim. Efficient Geometry-Based Similarity Search Of 3D Spatial Databases. In Proceedings Of *ACM SIGMOD International Conference On Management Of Data*, Philadelphia, PA, June 1999.
- [Khan90] T. Khanna. Foundations Of Neural Networks. *Addison Wesley*, 1990.
- [Kim03] Y. S. Kim, Y. H. Jung, B. G. Kang, and H. M. Rho. Feature-Based Part Similarity Assessment Method Using Convex Decomposition. In Proceedings Of *23rd ASME DETC Computers And Information In Engineering (CIE) Conference*, Chicago, IL, 2003.

- [Ko03a] K. H. Ko, T. Maekawa, and N. M. Patrikalakis. An Algorithm For Optimal Free-Form Object Matching. *Computer Aided Design*, 35(10): 913-923, 2003.
- [Ko03b] K. H. Ko, T. Maekawa, N. M. Patrikalakis, H. Masuda, and F.-E. Wolter. Shape Intrinsic Fingerprints For Free-Form Object Matching. In Proceedings Of *8th ACM Symposium On Solid Modeling And Applications*, 196-207, Seattle, WA, June 2003.
- [Lamd88a] Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General And Efficient Model-Based Recognition Scheme. In Proceedings Of *2nd International Conference On Computer Vision*, 238-249, 1988.
- [Lamd88b] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. Object Recognition By Affine Invariant Matching. In Proceedings Of *Computer Vision And Pattern Recognition*, 335-344, 1988.
- [Lamd88c] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson. On Recognition Of 3D Objects From 2D Images. In Proceedings Of *IEEE International Conference On Robotics And Automation*, 3: 1407-1413, 1988.
- [Laza99] F. Lazarus and A. Verroust. Level Set Diagrams Of Polyhedral Objects. In Proceedings Of *ACM Solid Modeling*, Ann Arbor, MI, June 1999.
- [Lin92] Y. Lin, J. Dou, and H. Wang. Contour Shape Description Based On Arch Height Function. *Pattern Recognition*, 25(1): 17-23, 1992.
- [Lipp89] R.P. Lippmann. Pattern Classification Using Neural Networks. *IEEE Communications Magazine*, 27(11): 47-64, November 1989.
- [Lonc98] S. Loncaric. A Survey Of Shape Analysis Techniques. *Pattern Recognition*, 31(8): 983-1001, August 1998.
- [Lou04] K. Lou, S. Prabhakar, and S. Ramani. Content Based Three Dimensional Engineering Shape Search. In Proceedings Of *20th International Conference On Data Engineering*, 754-765, Boston, MA, 2004.
- [McWh01a] D. McWherter, M. Peabody, A. Shokoufandeh, and W.C. Regli. Database Techniques For Archival Of Solid Models. In Proceedings Of *6th ACM/SIGGRAPH Symposium on Solid Modeling and Applications*, Ann Arbor, MI, June 2001.
- [McWh01b] D. McWherter, M. Peabody, W.C. Regli, and A. Shokoufandeh. Transformation Invariant Shape Similarity Comparison Of Solid Models. In Proceedings Of *ASME 6th Design For Manufacturing Conference*, Pittsburgh, PA, Sept 2001.

- [McWh01c] D. McWherter, M. Peabody, A. Shokoufandeh, and W.C. Regli. Solid Model Databases: Techniques And Empirical Results. *Journal Of Computer And Information Science In Engineering*, 1(4): 300-310, December 2001.
- [Mori01] G. Mori, S. Belongie, and J. Malik. Shape Contexts Enable Efficient Retrieval Of Similar Shapes. In Proceedings Of *IEEE Computer Society Conference On Computer Vision And Pattern Recognition*, Kauai Marriott, Hawaii, December 2001.
- [Moun99] D. M. Mount, N. S. Netanyahu, and J. Le Moigne. Efficient Algorithms For Robust Point Pattern Matching. *Pattern Recognition*, 32: 17-38, 1999.
- [Novo03] M. Novotni and R. Klein. 3D Zernike Descriptors For Content Based Shape Retrieval. In Proceedings Of *8th ACM Symposium On Solid Modeling And Applications*, 216-225, Seattle, WA, USA
- [Ohbu02] R. Ohbuchi, T. Minamitani, and T. Takei. Shape Similarity Search of 3D Models By Using Enhanced Shape Functions. In Proceedings Of *10th Pacific Conference On Computer Graphics And Applications*, 97-104, Beijing, China, October 2002.
- [Ohbu03a] R. Ohbuchi, T. Minamitani, and T. Takei. Shape-Similarity Search Of 3D Models By Using Enhanced Shape Functions. *International Journal Of Computer Applications In Technology*, 23(3/4/5): 70-85, 2005.
- [Ohbu03b] R. Ohbuchi and T. Takei. Shape-Similarity Comparison Of 3D Shapes Using Alpha Shapes. In Proceedings Of *11th Pacific Conference On Computer Graphics And Applications*, 293-302, Canmore, Canada, October 2003.
- [Osad01] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3D Models With Shape Distributions. In Proceedings Of *International Conference On Shape Modeling And Applications*, Genova, Italy, May 2001.
- [Osad02] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape Distributions. *ACM Transactions On Graphics*, 21(4): 807-832, 2002.
- [Rame01] M. Ramesh, D. Yip-Hoi, and D. Dutta. Feature-Based Shape Similarity Measurement For Retrieval Of Mechanical Parts. *Journal Of Computing And Information Science In Engineering*, 1(3): 245-256, September 2001.
- [Rea01] H. J. Rea, J. R. Corney, D. E. R. Clark, J. Pritchard, M. L. Breaks, and R. A. MacLeod. Part-Sourcing In A Global Market. In Proceedings Of *2001 International Conference On Ecommerce Engineering, Icece 2001*. Xi'an, P.R. China, September 2001.

- [Reeb46] G. Reeb. On The Singular Points Of A Completely Integrable Pfaff Form Or Of A Numerical Function. *Comptes Rendus De l'Academie De Sciences, Paris, France*, 222: 847-849, 1946.
- [Rubn98] Y. Rubner, C. Tomasi, and L. Guibas. A Metric For Distributions With Applications To Image Databases. In Proceedings Of *6th International Conference On Computer Vision*, Bombay, India, January 1998.
- [Sant95] S. Santini and R. Jain. Similarity Matching. In Proceedings Of *2nd Asian Conference On Computer Vision*, Singapore, December 1995.
- [Schw87] J. Schwarz and M. Scharir. Identification Of Partially Obscured Objects In Two And Three Dimensions By Matching Noisy Characteristic Curves. *International Journal Robotics Research*, 6(2): 29-44, Summer 1987.
- [Shen83] H. C. Shen and A.K.C. Wong. Generalized Texture Representation And Metric. *Computer Vision, Graphics And Image Processing*, 23(2): 187-206, August 1983.
- [Shum96] H. Shum, M. Hebert, and K. Ikeuchi. On 3D Shape Similarity. In Proceedings Of *IEEE Conference On Computer Vision And Pattern Recognition*, San Francisco, CA, June 1996.
- [Sidd99] K. Siddiqi, A. Shokoufandeh, S. J. Dickenson, and S.W. Zucker. Shock Graphs And Shape Matching. *International Journal Of Computer Vision*, 35(1):13-32, 1999.
- [Sprin94] J. Sprinzak and M. Werman. Affine Point Matching. *Pattern Recognition Letters*, 15(4): 337-339, 1994.
- [Srin98] G. Srinivas, D.E. Fasse, and M.M. Marefat. Retrieval Of Similarly Shaped Parts From A CAD Database. In Proceedings Of *IEEE International Conference On Systems, Man, And Cybernetics*, San Diego, CA, October, 1998.
- [Sun95] T.L. Sun, C.J. Su, R.J. Mayer, and R.A. Wysk. Shape Similarity Assessment Of Mechanical Parts Based On Solid Models. In Proceedings Of *Design For Manufacturing Symposium, ASME Design Technical Conference*, Boston, MA, September 1995.
- [Sund03] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton Based Shape Matching And Retrieval. In Proceedings Of *Shape Modeling And Applications Conference SMI*, 130-142, Seoul, Korea, May 2003.
- [Sung02] R. Sung, H. J. Rea, J. R. Corney, D. E. R. Clark, J. Pritchard, M. L. Breaks, and R. A. MacLeod. Assessing The Effectiveness Of Filters For Shape Matching. In Proceedings Of *2002 Asme International Mechanical*

Engineering Congress & Exposition, IMECE '02. New Orleans, LA, November 2002.

- [Taka97] S. Takahashi, Y. Shinagawa, and T. L. Kunii. A Feature-Based Approach For Smooth Surfaces. In Proceedings Of *4th ACM Symposium On Solid Modeling And Applications*, Atlanta, GA, May 1997.
- [Tang04] J. Tangelder and R. Veltkamp. A Survey Of Content-Based 3D Shape Retrieval Methods. In Proceedings Of *Shape Modeling International*, 145-156, Genoa, Italy, June 2004.
- [Thac95] N. Thacker, P. Riocreux, and R. Yates. Assessing The Completeness Properties Of Pairwise Geometric Histograms. *Image And Vision Computing*, 13(5): 423-429, June 1995.
- [Tsai85] W.H. Tsai and S.S. Yu. Attributed String Matching With Merging For Shape Recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 7(4): 453-462, July 1985.
- [Tuzi00] A. V. Tuzikov, J.B.T.M. Roerdink, and H.J.A.M. Heijmans. Similarity Measures For Convex Polyhedra Based On Minkowski Addition. *Pattern Recognition*, 33(6): 979-995, June 2000.
- [Velt01] R.C. Veltkmap. Shape Matching: Similarity Measures And Algorithms. In Proceedings Of *International Conference On Shape Modeling And Applications*, Genova, Italy, May 2001.
- [Vran01] D. V. Vranic, D. Saupe, and J. Richter. Tools For 3D-Object Retrieval: Karhunen-Loeve Transform And Spherical Harmonics. In Proceedings Of *IEEE 2001 Workshop Multimedia Signal Processing*, 293-298, Cannes, France, October 2001.
- [Werm85] M. Werman, S. Peleg, and A. Rosenfeld. A Distance Metric For Multidimensional Histograms. *Computer Vision, Graphics And Image Processing*, 32(3): 328-336, December 1985.
- [Wolf97] H.J. Wolfson and I. Rigoutsos. Geometric Hashing: An Overview. *IEEE Computational Science And Engineering*, 4:10-21, 1997.
- [Youn74] I. Young, J. Walker, and J. Bowie. An Analysis Technique For Biological Shape. *Information And Control*, 25(4): 357-370, August 1974.
- [Yu03] M. Yu, I. Atmosukarto, W. K. Leow, Z. Huang, and R. Xu. 3D Model Retrieval With Morphing-Based Geometric And Topological Feature Maps. In Proceedings Of *IEEE Computer Society Conference On Computer Vision And Pattern Recognition*, 656-661, Madison, WI, June 2003.

- [Zhan99] D. Zhang and M. Hebert. Harmonic Maps And Their Applications In Surface Matching. In Proceedings Of *IEEE Conference On Computer Vision And Pattern Recognition*, Fort Collins, CO, June 1999.

Appendix

A. Calculation Of Partitioning Lines And Planes For Attributed Points In \mathbb{R}^2 And

\mathbb{R}^3

Consider two attributed points $b_1 = (x_1, y_1, z_1, w_1)$ and $b_2 = (x_2, y_2, z_2, w_2)$ in \mathbb{R}^3 , where w_1 and w_2 are the transformation-invariant attributes of the two points. Consider also point $p = (x, y, z, w)$ in \mathbb{R}^3 , where w is the transformation-invariant attribute. Suppose the distance between p and b_1 is computed using the following equation.

$$d(p, b_1) = (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 + \Delta w_1^2 \quad (\text{A.1})$$

Distance between p and b_2 is computed using a similar equation.

The differences $\Delta w_1 = w - w_1$ and $\Delta w_2 = w - w_2$ correspond to the transformation-invariant terms of respectively $d(p, b_1)$ and $d(p, b_2)$. We are interested in finding the partitioning plane for points b_1 and b_2 , that is locus of attributed points $p = (x, y, z, w)$ in \mathbb{R}^3 such that $d(p, b_1) = d(p, b_2)$.

First of all observe that it is always possible to apply a rigid body transformation to the attributed points b_1 and b_2 in \mathbb{R}^3 such that the following constraints are valid.

$$\begin{cases} x_1 = -x_2 \\ z_1 = z_2 \\ y_1 = y_2 = 0 \end{cases} \quad (\text{A.2})$$

In order to solve the previously formulated problem, it is also necessary to comply with the following constraint.

$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 + \Delta w_1^2 = (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 + \Delta w_2^2 \quad (\text{A.3})$$

Equation (A.3) constrains point $p = (x, y, z, w)$ to be at the same distance from points b_1 and b_2 . Using the Equations (A.2), Equation (A.3) simplifies in the following manner.

$$(x+x_2)^2 + y^2 + (z-z_2)^2 + \Delta w_1^2 = (x-x_2)^2 + y^2 + (z-z_2)^2 + \Delta w_2^2 \quad (\text{A.4})$$

It is possible to assume without loss of generality that $\Delta w^2 = \Delta w_2^2 - \Delta w_1^2 > 0$. Using this assumption and the notations introduced previously, the second Equation (A.4) can be simplified to

$$x = \frac{\Delta w^2}{4x_2} \quad (\text{A.5})$$

Equation (A.5) represents a plane π parallel to plane YZ. If the value of Δw is set to 0, then the plane π corresponds to plane YZ itself.

So the locus of attributed points in \mathbb{R}^3 such that their distance from $b_1 = (x_1, y_1, z_1, w_1)$ and $b_2 = (x_2, y_2, z_2, w_2)$ is equal corresponds to plane π parallel to coordinate plane YZ. Plane π is the plane perpendicular to the line segment joining attributed points b_1 and b_2 . The intersection between plane π and the line segment joining attributed points b_1 and b_2 has a distance from the midpoint of the line segment that is equal to the offset value defined by Equation (A.5).

The reasoning and equations described in this appendix can be easily modified to address the same problem in \mathbb{R}^2 by not considering coordinate Z. Therefore it is easy to verify that the locus of attributed points in \mathbb{R}^2 such that their distance from $b_1 = (x_1, y_1, w_1)$ and $b_2 = (x_2, y_2, w_2)$ is equal corresponds to line L perpendicular to the line segment joining b_1 and b_2 and whose distance from the midpoint of the line segment is equal to the offset value defined by Equation (A.5).

B. Calculation Of Partitioning Curves For Attributed Points On The Unit Sphere

Consider two attributed points $b_1 = (x_1, y_1, z_1, w_1)$ and $b_2 = (x_2, y_2, z_2, w_2)$ that lie on the unit sphere, where w_1 and w_2 are the transformation-invariant attributes of the two points. Consider also the attributed point $p = (x, y, z, w)$ of the unit sphere, where w is the transformation-invariant attribute. Suppose the distance between p and b_1 is computed using the following equation.

$$d(p, b_1) = (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 + \Delta w_1^2 \quad (\text{B.1})$$

Distance between p and b_2 is computed using a similar equation.

The differences $\Delta w_1^2 = w^2 - w_1^2$ and $\Delta w_2^2 = w^2 - w_2^2$ correspond to the transformation-invariant terms of respectively $d(p, b_1)$ and $d(p, b_2)$. We are interested in finding the partitioning curve for points b_1 and b_2 , that is locus of attributed points $p = (x, y, z, w)$ on the unit sphere such that $d(p, b_1) = d(p, b_2)$.

First of all observe that it is always possible to apply a rigid body transformation to the attributed points b_1 and b_2 that lie on the unit sphere such that the following constraints are valid.

$$\begin{cases} x_1 = -x_2 \\ z_1 = z_2 \\ y_1 = y_2 = 0 \end{cases} \quad (\text{B.2})$$

In order to solve the previously formulated problem, it is also necessary to comply with the following constraints.

$$\begin{cases} x^2 + y^2 + z^2 = 1 \\ (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 + \Delta w_1^2 = (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 + \Delta w_2^2 \end{cases} \quad (\text{B.3})$$

The first equation constrains the attributed point $p = (x, y, z, w)$ to lie on the unit sphere. The second equation constrains point $p = (x, y, z, w)$ to be at the same distance

from points b_1 and b_2 . Using Equations (B.2), Equations (B.3) simplify in the following manner.

$$\begin{cases} x^2 + y^2 + z^2 = 1 \\ (x+x_2)^2 + y^2 + (z-z_2)^2 + \Delta w_1^2 = (x-x_2)^2 + y^2 + (z-z_2)^2 + \Delta w_2^2 \end{cases} \quad (\text{B.4})$$

It is possible to assume without loss of generality that $\Delta w^2 = \Delta w_2^2 - \Delta w_1^2 > 0$. Using this assumption and the notations introduced previously, the second Equation (B.4) can be simplified to

$$x = \frac{\Delta w^2}{4x_2} \quad (\text{B.5})$$

Equation (B.5) represents a plane π parallel to plane YZ. If the value of Δw^2 is set to 0, then the plane π corresponds to plane YZ itself. If the expression for x being obtained in Equation (B.2) is substituted into the first of Equations (B.3), a circle is obtained with having following equation.

$$y^2 + z^2 = 1 - \frac{\Delta w^4}{16x_2^2} \quad (\text{B.6})$$

Equations (B.5) and (B.6) represent a circle for values of $\Delta w^4 < 16 x_2^2$. Otherwise Equation (B.6) is not defined.

For values of $\Delta w^4 \leq 16 x_2^2$ the partitioning curve defined as the locus of attributed points p on the unit sphere such that $d(p, b_1) = d(p, b_2)$ corresponds to a circle C that lies on the unit sphere. The circle C is obtained from the intersection between the unit sphere and the plane π parallel to plane YZ. The distance of the plane π from plane YZ is given by Equation (B.5).

For values of $\Delta w^4 > 16 x_2^2$ Equation (B.6) becomes unsolvable. It means that the offset value of the plane parallel to plane YZ given by Equation (B.5) is such that the

plane does not intersect the unit sphere. In this case the partitioning curve is not defined. This means that for all the attributed points p on the unit sphere it is always $d(p, b_2) > d(p, b_1)$ as we defined $\Delta w^2 = \Delta w_2^2 - \Delta w_1^2$.

C. Calculation Of Partitioning Theta Values For Attributed Applied Vectors Under 1 DOF Rotations In \mathbb{R}^3

Consider two attributed applied vectors a and b whose locations are (x^a, y^a, z^a) and (x^b, y^b, z^b) and whose orientations are (v_x^a, v_y^a, v_z^a) and (v_x^b, v_y^b, v_z^b) that lie in \mathbb{R}^3 . Define w^a and w^b as the transformation-invariant attributes of the two applied vectors. Consider also the attributed applied vector p whose location is (x^p, y^p, z^p) and whose orientation is (v_x^p, v_y^p, v_z^p) that lie in \mathbb{R}^3 as well, where w^p is the transformation-invariant attribute.

Suppose the distance between p and a is computed using the following equation.

$$d(p, a) = (x^p - x^a)^2 + (y^p - y^a)^2 + (z^p - z^a)^2 + (v_x^p - v_x^a)^2 + (v_y^p - v_y^a)^2 + (v_z^p - v_z^a)^2 + (w^p - w^a)^2 \quad (\text{C.1})$$

Distance between p and b is computed using a similar equation.

Imagine that the attributed applied vector p is rotated about Z axis of θ . Let v_{zo}^p be the initial Z component of the orientation for attributed point p , while $v_{xyo}^p = \sqrt{(v_{xo}^p)^2 + (v_{yo}^p)^2}$ is the initial component in the coordinate plane XY before applying rotation θ . Let (x_B, y_B) be the center of rotation. Define θ_o^p as the known initial angle of p with respect to the center of rotation before applying rotation θ . Similarly let d_z be the Z component and d_{xy} the XY component of the Euclidean distance between p and the center of rotation. Let

also θ_{vo}^p be the known initial angle of the XY component of the orientation of p with X axis before applying rotation θ .

Given the previous definitions, the following equations hold.

$$\begin{cases} x^p(\theta) = x_B + d_{xy} \cos(\theta_o^p + \theta) \\ y^p(\theta) = y_B + d_{xy} \sin(\theta_o^p + \theta) \\ z^p = z_B + d_{zi} \\ v_x^p(\theta) = v_{xyo}^p \cos(\theta_{vo}^p + \theta) \\ v_y^p(\theta) = v_{xyo}^p \sin(\theta_{vo}^p + \theta) \\ v_z^p = v_{zo}^p \end{cases} \quad (C.2)$$

Hence the distance function defined in Equation (C.1) can be written as follows.

$$\begin{aligned} d(p(\theta), a) = & (x^p(\theta) - x^a)^2 + (y^p(\theta) - y^a)^2 + (z^p - z^a)^2 + \\ & (v_x^p(\theta) - v_x^a)^2 + (v_y^p(\theta) - v_y^a)^2 + (v_z^p - v_z^a)^2 + (w^p - w^a)^2 \end{aligned} \quad (C.3)$$

We are interested in finding the values of the rotation θ such that $d(p(\theta), a) = d(p(\theta), b)$, which are the partitioning theta values. By using the definitions given previously and Equation (C.3) the following equation must hold.

$$\begin{aligned} & (x^p(\theta) - x^a)^2 + (y^p(\theta) - y^a)^2 + (z^p - z^a)^2 + \\ & (v_x^p(\theta) - v_x^a)^2 + (v_y^p(\theta) - v_y^a)^2 + (v_z^p - v_z^a)^2 + (w^p - w^a)^2 = \\ & (x^p(\theta) - x^b)^2 + (y^p(\theta) - y^b)^2 + (z^p - z^b)^2 + \\ & (v_x^p(\theta) - v_x^b)^2 + (v_y^p(\theta) - v_y^b)^2 + (v_z^p - v_z^b)^2 + (w^p - w^b)^2 \end{aligned} \quad (C.4)$$

Equation (C.4) can be simplified to

$$\begin{aligned} & 2x^p(\theta)(x^a - x^b) + 2y^p(\theta)(y^a - y^b) + 2v_x^p(\theta)[v_x^a - v_x^b] + \\ & 2v_y^p(\theta)[v_y^a - v_y^b] = (x^a)^2 + (y^a)^2 + (v_x^a)^2 + (v_y^a)^2 - (x^b)^2 - \\ & (y^b)^2 - (v_x^b)^2 - (v_y^b)^2 + \Delta w^2 \end{aligned} \quad (C.5)$$

where

$$\Delta w^2 = (z^p - z^a)^2 + (v_z^p - v_z^a)^2 + (w^p - w^a)^2 - (z^p - z^b)^2 - (v_z^p - v_z^b)^2 - (w^p - w^b)^2.$$

From Equations (C.2) and (C.5) it can be inferred that the values of the rotation angle θ such that $d(p(\theta), a) = d(p(\theta), b)$ can be found solving an equation of the following type:

$$A \cos\theta + B \sin\theta = C \tag{C.6}$$

where the terms A , B and C are function of locations (x^a, y^a, z^a) and (x^b, y^b, z^b) , orientations (v_x^a, v_y^a, v_z^a) and (v_x^b, v_y^b, v_z^b) , rotation-invariant attributes w^a and w^b and the constant terms in Equations (C.2) defined previously. Equation (C.6) might have one, more or no solution depending on the value of the terms A , B and C . There are well-known mathematical procedures to solve it.