

12-2012

# Development of Feature Recognition Algorithm for Automated Identification of Duplicate Geometries in CAD Models

Aravind Shanthakumar

Clemson University, [ashanth@g.clemson.edu](mailto:ashanth@g.clemson.edu)

Follow this and additional works at: [https://tigerprints.clemson.edu/all\\_theses](https://tigerprints.clemson.edu/all_theses)

 Part of the [Mechanical Engineering Commons](#)

---

## Recommended Citation

Shanthakumar, Aravind, "Development of Feature Recognition Algorithm for Automated Identification of Duplicate Geometries in CAD Models" (2012). *All Theses*. 1513.

[https://tigerprints.clemson.edu/all\\_theses/1513](https://tigerprints.clemson.edu/all_theses/1513)

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact [kokeefe@clemson.edu](mailto:kokeefe@clemson.edu).

DEVELOPMENT OF FEATURE RECOGNITION ALGORITHM FOR AUTOMATED  
IDENTIFICATION OF DUPLICATE GEOMETRIES IN CAD MODELS

---

A Thesis  
Presented to  
The Graduate School of  
Clemson University

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
Mechanical Engineering

---

by  
Aravind Shanthakumar  
December 2012

---

Accepted by:  
Dr. Joshua D. Summers, Committee Chair  
Dr. Georges M. Fadel  
Dr. Brian Malloy

## ABSTRACT

This research presents a feature recognition algorithm for the automated identification of duplicate geometries in the CAD assembly. The duplicate geometry is one of the seven indicators of the lazy parts mass reduction method. The lazy parts method is a light weight engineering method that is used for analyzing parts with the mass reduction potential. The duplicate geometry is defined as any geometries lying equal to or within the threshold distance with the user-defined orientation between them and have the percentage similarity that is equal to or greater than the threshold value. The feature recognition system developed in this research for the identification of duplicate geometries is also extended to retrieve the weighted bipartite graph of part connections for the assembly time estimation. The weighted bipartite graph is used as input for the part connectivity based assembly time estimation method.

The SolidWorks API software development kit is used in this research to develop a feature recognition system in SolidWorks CAD software package using C++ programming language. The feature recognition system built in the SolidWorks CAD software uses a combination of topology and geometric data for the evaluation of duplicate geometry. The measurement of distances between the sampling points strategy is used for the duplicate geometry feature recognition. The feature recognition algorithm has three phases of evaluation: first, is the evaluation for threshold distance condition of parts in the CAD assembly. Second, the part pairs that have satisfied the threshold distance condition are evaluated for the orientation condition. The threshold distance and

orientation are the necessary but not the sufficient conditions for duplicate geometries. In the third phase, the geometries that have satisfied orientation condition are evaluated for the percentage similarity condition. The geometries that satisfy the percentage similarity condition are highlighted in order to help designers review the results of the duplicate geometry analysis.

The test cases are used to validate the algorithm against the requirements list. The test cases are designed to check the performance of the algorithm for the evaluation of the threshold distance, orientation, and percentage similarity condition. The results indicate that the duplicate geometry algorithm is able to successfully conduct all the three phases of evaluation. The algorithm is independent of the geometric type and is able to analyze planar, cylindrical, conical, spherical, freeform, and toroidal shapes. The number of sampling points generated on the faces of parts for the orientation and percentage similarity evaluation has the significant effect on the analysis time. The worst case complexity of the algorithm is the big O ( ${}^nC_2 \times m_1^2 \times m_2^2 \times p^4$ ), where

$n$  = the number of parts in the assembly

$m_1$  = the number of faces in the parts that meet the threshold distance condition

$m_2$  = the number of faces that meet the orientation condition

$p$  = the number of sampling points on the face

The duplicate geometry feature recognition approach is used to demonstrate the applicability in the extraction of assembly relations for the part connectivity based assembly time estimation method. The algorithm is also able to extract part connectivity

information for the patterns. Further research is required to automate the identification of other laziness indicators in order to make the lazy parts method a completely automated tool. With regards to the complete automation of part connectivity based assembly time estimation method, the duplicate geometry feature recognition system needs integration with the algorithm for the computation of bipartite graph of part connections for the prediction of assembly time.

## DEDICATION

To my parents Shanthakumar and Leela Shanthakumar, and my hometown  
Bangalore.

## ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Joshua D. Summers for giving me an opportunity to work on this research. I also thank him for his support and patience that offered me confidence to do this research. He has provided me opportunities to work and collaborate on different projects and technical papers for which I will remain grateful to him.

I am equally thankful to Dr. Georges M. Fadel who had offered me financial assistance and the opportunity to work on a research project. I also thank him for his support and feedback on my research.

I am grateful to Dr. Brian Malloy who taught me C++ programming language which was essential for this research. The programming courses I took with him provided me the skill set to work on this research. I express my gratitude to Dr. Malloy for being supportive and agreeing to be a part of the committee.

I thank the members of the CEDAR lab for all the help. I would specially like to thank Eric Owensby, my research partner and a good friend. I would also like to thank Dr. Chiradeep Sen for his guidance during difficult times.

## TABLE OF CONTENTS

Abstract .....	ii
Dedication .....	v
Acknowledgments.....	vi
List of Tables .....	ix
List of Figures .....	x
Chapter One : motivation - needs for duplicate geometry feature recognition algorithm.....	1
1.1 Manual Identification of Lazy Parts Indicators Problem.....	1
1.2 Manual Retrieval of Physical Connections Problem for Assembly Time Estimation.....	7
1.3 Inference - Necessity for a Duplicate Geometry Algorithm.....	18
1.4 Overview of Thesis .....	21
Chapter Two : literature Review of feature recognition algorithms .....	23
2.1 Graph-Based Method.....	24
2.2 Hint-Based Method.....	30
2.3 Convex Hull Decomposition Method.....	32
2.4 Cell Based Volumetric Decomposition Method.....	35
2.5 Hybrid Method.....	37
2.6 Comparison of Techniques .....	39
Chapter Three : Research objective .....	43
3.1 Definition of Duplicate Geometry .....	43
3.2 Thesis Objective .....	49
3.3 Establishing Requirements .....	50
Chapter Four : Design and implementation .....	54
4.1 System Architecture.....	54
4.2 Duplicate Geometry Recognition Approach.....	57



4.3 Implementation .....	60
Chapter Five : Validation.....	87
5.1 Test-Cases to Check for Threshold Distance Condition.....	87
5.2 Test-Cases to Check for Orientation Condition.....	91
5.3 Test-Cases to Check for Percentage Similarity between Geometries .....	96
5.4 Highlight Duplicate Geometries .....	101
5.5 Effect of Geometric Types on the Evaluation .....	102
5.6 Effect of the Number of Parts on the Bounding-box Algorithm.....	107
5.7 Effect of the Number of Sampling Points on the Percentage Similarity Algorithm.....	108
5.8 Algorithm offers Extensibility to obtain Weighted Assembly Relations .....	110
5.9 External Validation .....	113
Chapter Six : Future work and conclusion.....	124
6.1 Research Contribution .....	124
6.2 Future Work .....	126
6.3 Conclusion .....	128
Chapter Seven : references.....	132
Appendix A: Test Cases to Check Threshold Distance Condition .....	139
Appendix B: Test Cases to Check Orientation Condition .....	141
Appendix C: Test-Cases to Check for Percentage Similarity between Geometries.....	145

## LIST OF TABLES

Table 1-2: All mate types offered by SolidWorks software [4].....	15
Table 2-1: Additional attributes of EAAG [31].....	38
Table2-2: Comparison of feature recognition techniques.....	41
Table 3-1: Subjectivity in the old definition addressed in revised definition .....	45
Table 4-1: Example list showing faces stored as pairs that have orientation within the user-defined angle and tolerance .....	78
Table 5-1: Test cases for threshold distance .....	89
Table 5-2: Test cases used check for orientation between face pairs .....	93
Table 5-3: Test cases to check for percentage similarity .....	98
Table 5-4: Duplicate geometry analysis results for different geometric types .....	103
Table 5-5: Input parameters for the algorithm.....	114
Table 5-6: Anticipated connections .....	114
Table 5-7: Input parameters for the caster assembly .....	118
Table 5-8: Anticipated part connections for caster assembly .....	118
Table 5-9: Input parameters for the punch assembly.....	122
Table 5-10: Anticipated part connections for punch assembly.....	122
Table 5-11: Assembly relations extracted for the punch assembly .....	122

## LIST OF FIGURES

Figure 1.1: Left – Section view of a fastener connection; Right – Bi-partite graph showing connectivity between three parts.....	9
Figure 1.2: Part connections and relationships developed for an automotive sub-assembly [18] .....	11
Figure 1.3: Hard drive packaging with foam [5] .....	16
Figure 1.4: Different 3D shapes in geometric modeling.....	20
Figure 2.1: Interacting features; square pocket is split into two halves by a slot.....	25
Figure 2.2: Cube with a pocket .....	26
Figure 2.3: AAG for the part in Figure 2.2 .....	27
Figure 2.4: Bi-connected and Tri-connected acyclic directed graph .....	28
Figure 2.5: Hints generated through ray-firing .....	32
Figure 2.6: Left: Part with a cylindrical protrusion; Right: Convex hull for the part.....	34
Figure 2.7: (a) Part; (b) cell decomposition of the delta volume .....	36
Figure 2.8: Two distinct maximal volume interpretation .....	37
Figure 3.1: Cable Guide Attached to the Underside of the Battery [2].....	44
Figure 3.2: Geometries that are lying within or equal to threshold distance are considered for duplicate geometry analysis .....	46
Figure 3.3: The angle between the outward normals from opposing geometries need to be within the threshold tolerance .....	47
Figure 3.4: The distance measurements between the sampling points.....	49
Figure 4.1: System Architecture .....	54
Figure 4.2: SolidWorks GUI showing Find Duplicate Geometries button built on the panel and drop down menu.....	55
Figure 4.3: High level description of the duplicate geometry algorithm .....	59
Figure 4.4: Flowchart representing duplicate geometry algorithm.....	60
Figure 4.5: SolidWorks GUI with the duplicate geometry button.....	61

Figure 4.6: List of visible parts in the SolidWorks feature manager tree for the motor assembly shown in the right.....	62
Figure 4.7: Example of an axis aligned bounding box .....	64
Figure 4.8: Bounding box types.....	65
Figure 4.9: Bounding box coordinates returned in SolidWorks .....	66
Figure 4.10: Bounding box expanded to check for threshold distance condition .....	67
Figure 4.11: Intersection calculation using bounding box.....	70
Figure 4.12: Parts meeting the threshold distance condition stored as pairs in multimap container .....	71
Figure 4.13: Topological data structure in SolidWorks.....	72
Figure 4.14: Faces extracted from the bodies in the part pair.....	73
Figure 4.15: Each face from one set is compared with all faces from the other set for orientation .....	74
Figure 4.16: Tessellating the face to generate sampling points .....	75
Figure 4.17: Surface outward normal for a face at different sampling points .....	76
Figure 4.18: Unit normals retrieved at sampling points for the two faces .....	77
Figure 4.19: Measurement of distance between sampling points .....	79
Figure 4.20: Face consisting of lesser number of sampling points is used to start the measurement .....	81
Figure 4.21: Distances measured from one sampling point on Face 1 to all sampling points on Face 2 .....	83
Figure 4.22: Calculating percentage similarity between two geometries .....	84
Figure 5.1: Instances of duplicate geometry highlighted in red by the algorithm .....	102
Figure 5.2: Time consumed for the evaluation of orientation condition for different geometric types.....	106
Figure 5.3: Effect of the number of parts on the bounding-box algorithm .....	108
Figure 5.4: Effect of the number of sampling points on the analysis time for the evaluation of percentage similarity .....	110

Figure 5.5: Weighted bipartite graph of part connectivity information extracted from the motor assembly .....	112
Figure 5.6: (a) The assembly of vice and the constituent parts; (b) Part connection faces are highlighted by the algorithm in red.....	115
Figure 5.7: Part connections extracted for the vice assembly.....	116
Figure 5.8: The caster assembly from SolidWorks library .....	117
Figure 5.9: Part connections retrieved for the caster assembly .....	120
Figure 5.10: The punch assembly .....	121

## CHAPTER ONE: MOTIVATION - NEEDS FOR DUPLICATE GEOMETRY FEATURE RECOGNITION ALGORITHM

Mechanical Computer Aided Design (CAD) software provides designers and engineers with various tools to create and work with the virtual representation of the physical artifact being designed. The CAD tools empower engineers to conduct design and analysis of the desired product with increased productivity and reduced errors. This research draws motivation from two distinct research works that compels developing a feature recognition system in CAD software to support design reasoning of duplicate geometry identification and analysis. The first application is for the automated identification of duplicate geometries in CAD assembly for the mass reduction analysis in lightweight engineering. The second application is to extract physical connections from the CAD assembly to develop the connectivity graph for assembly time estimation. Each of these applications will be discussed in greater detail below as system requirements are defined.

### 1.1 Manual Identification of Lazy Parts Indicators Problem

The Lazy Parts Indication Mass Reduction Method (LPIMRM) is a lightweight engineering tool that was developed at Clemson University to provide a systematic approach for engineers to select components for redesign [1–3].

#### 1.1.1 The Method and Benefits

This method was developed through collaboration between Clemson University and a major original equipment manufacturer (OEM) to develop lightweight engineering

tools [4–6]. The collaborative effort was focused on the application of lightweight engineering on five attributes of the design: requirements, concept development, optimization, assembly, and material replacement [7,8].

Originally, this method was envisioned to support lightweight engineering in automotive vehicles. However, the performance and scalability of this method to smaller mechanical systems was studied and assessed in [3]. The method provides a list of identifiers called laziness indicators to select components for mass reduction analysis. The method has five phases to estimate percentage of mass reduction of which reviewing the components against laziness indicators is one of the phase.

Formal definition of the lazy parts, description and examples for the laziness indicators, and the process for identifying lazy parts can be found in [1,2]. To help with understanding the motivation behind this research the definition of lazy parts and the laziness indicators are briefly discussed below.

#### 1.1.2 Lazy Parts Definition

The formal definition for lazy parts is *any part or assembly in an automobile that would include additional mass due to one of five reasons* [1]. First, the part's purpose may be only for the assembly process and therefore, after the assembly process, the presence of this part in the assembly is not necessary for full in-use performance. An example for this type of lazy part is a bracket used for connecting two spatially separated parts. Second, the part satisfies no functional requirement and the inclusion of this part may be due to the presence of certain specific features. The nuts are the example for this

type which is only used to fasten the bolts. If screws, rivets, or adhesive is used then, the use of nuts is not required. Third, the part or system could be redesigned and replaced by a lighter system. Fourth, two or more parts could be integrated into a single component and still maintain the same overall system function. The third and fourth type of lazy parts requires engineering knowledge for the manipulation of parts. Fifth, a part is considered lazy if there is a possibility for optimization of the part for mass reduction [1]. An example for this type of lazy parts is the structural parts that can be optimized for weight for the given mechanical stresses. Based on the five conditions, a list of indicators was developed to help in the identification of lazy parts. These indicators are pointers that would draw attention to the parts with mass reduction potential. The indicators are discussed in the next section.

### 1.1.3 Laziness Indicators

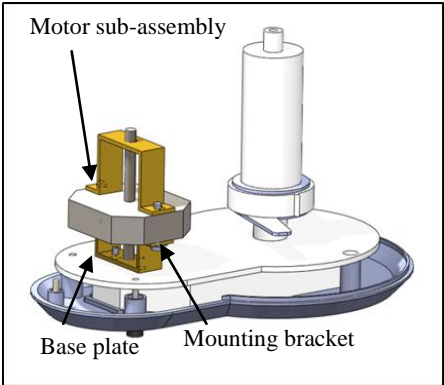
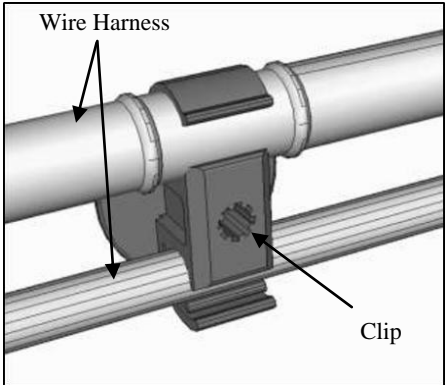
The laziness indicators represent a list of hints that could be referred to filter components for mass reduction analysis. The purpose of indicators are to draw the focus of a designer to components that has the potential for mass reduction [1]. Regardless of the expertise of the designer, the indicators help only in selecting the components for mass reduction. The seven indicators of LPIMRM are discussed in the following section (see Table 1-1 for examples).

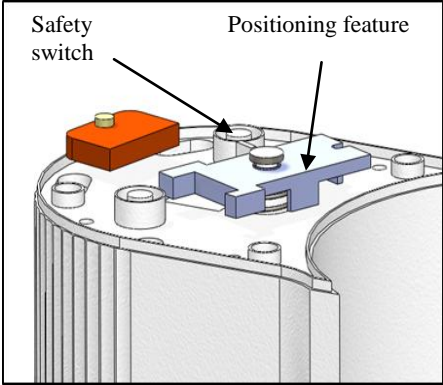
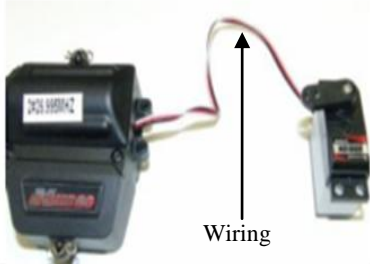

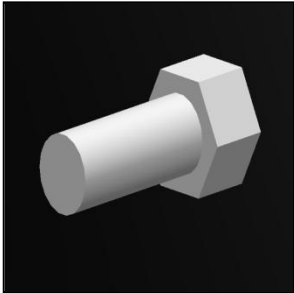
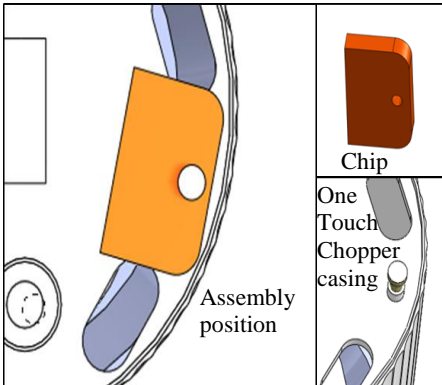
- Rigid-to-Rigid Connection – A component that connects one rigid component to another and prevents relative movements between them (Table 1-1 A).
- Support for a Flexible, Non-moving Part – A component that supports flexible parts and secures them from moving during vehicle operation (Table 1-1 B).



- Positioning Feature – A feature or a component that is useful only for positioning the component in the assembly (Table 1-1 C).
- Bridging System – A component that transfers material or energy between two systems that are separated (Table 1-1 D).
- Material Flow Restriction – A component whose purpose is to restrict the flow of material into or outside a system (Table 1-1 E).
- Fastener – A part that secures two or more components in place (Table 1-1 F).
- Duplicate Geometry – Two closely located geometries that are similar to each other. (Table 1-1 G). The research presented in this thesis addressed this identifier with an aim to automate the recognition of this identifier in CAD assemblies.

**Table 1-1: Examples of Laziness Indicators**

A. Rigid-to-Rigid Connection	B. Support for a Flexible, Non-moving Part
	
<p>CAD model of Black and Decker's One Touch Chopper showing an instance of rigid-to-rigid connection</p>	<p>Clip Securing Wire Harness [1]</p>

C. Positioning Feature		D. Bridging System	
			
Positioning feature on a safety switch		Electrical wire – bridging system between battery and servo from an RC car [3]	
E. Material Flow Restriction	F. Fastener	G. Duplicate Geometry	
			
Enclosure in headlight cluster	Hexagonal head bolt	Undersurface of the chip and top surface of One Touch Chopper casing are duplicate to each other [9]	

#### 1.1.4 Limitation and Motivation

For a large CAD assembly, supposing the assembly of an entire vehicle, manually parsing through the list of above discussed indicators against each component to identify lazy parts becomes tedious resulting in a large pre-analysis time and increased likelihood of human error. This limitation can be overcome by integrating the laziness indicators

into a CAD system that can use feature recognition technology to identify lazy parts indicators. Over one thousand components were manually evaluated for an automotive vehicle at a large OEM and recommended the development of a CAD system for the automation of laziness indicators [1].

#### 1.1.5 Research Challenges

Integration of all the seven laziness indicators into a CAD system necessitates separate research for each of the indicators. Feature Recognition (FR) of rigid-to-rigid connection and support-for-flexible part requires reasoning for differentiating a rigid component from a flexible component. One of the options could be to use material property information from the CAD software and use rule-based approach of FR to fulfill the task. Much of FR algorithms available in the literature could be explored and suitable ones adjusted to identify positioning features. Semantics or hint-based approach could be used to detect fasteners in the assembly. A FR algorithm for duplicate geometry needs to consider the degree of similarity and the proximity conditions. Certainly, all indicators require separate research to address and overcome the challenges.

The research of this thesis focuses on the development of a tool to automatically identify duplicate geometry as a laziness indicator. The definition of the duplicate geometry is broad and needs refinement for the purpose of automation [1]. To illustrate further, the definition “two closely located geometries that are similar to each other” presents three questions that needs to be answered. First, what distance between the geometries can be considered close? Second, how to determine if two geometries are similar and lastly, what is the amount of similarity that would make the two geometries

duplicate. While formalizing the definition for the duplicate geometry (discussed in Chapter Three), all the three questions are addressed. The answers for these questions may change based on the application and users, therefore these questions are treated as user-defined parameters in this research.

Although, duplicate geometry lazy part indicator is the primary motivation for this research, another research area where this FR system could be useful is for the automated assembly time estimation method that will be discussed in the next section.

## 1.2 Manual Retrieval of Physical Connections Problem for Assembly Time Estimation

Assembly Time Estimation (ATE) is a useful redesign tool that offers a quantitative scale to compare competitive designs. ATE is a part of Design for Assembly (DFA) method used for cost analysis, part count reduction, and comparison of different designs [10–12]. The research in the field of ATE has progressed from manual rule-based system [11,13,14] towards automation with integration into CAD system [9,15–19].

For this research, the motivation is the automation of ATE method that uses the information from a CAD system. The advantage of using a CAD system for DFA analysis is the ability to extract different types of data for automated reasoning; some examples for such type of data include geometry, assembly coordinates, volume, mass, part count, and assembly constraints. More recent works on ATE uses part connectivity information from the assembly [18,20] and the assembly mates [9,19] from a CAD file. This approach is aimed at reducing the number of user inputs and subjectivity elements

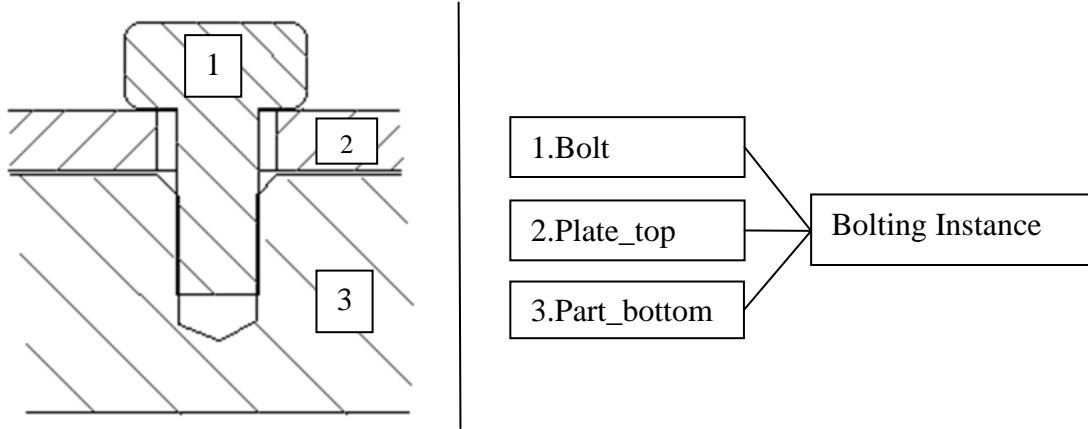
prevalent in previous DFA methods [21]. Both the approaches are semi-automated and offer scope for improvement that forms the motivation for this research. The discussion on these methods is presented in the following section. Presently, the part connectivity information is manually extracted from CAD data [6,22,23]. However, the automated retrieval of part connections from a CAD system could yield benefits such as reduced analysis time and reduced human inputs.

#### 1.2.1 Connectivity Based Assembly Time Estimation Method

The Assembly Time Estimation Method based on connective complexity metrics, developed at Clemson University, uses a mathematical model based on the part connections in the assembly to estimate the assembly time [20]. The assembly relations are manually retrieved from the CAD assembly file for input into the artificial neural-net. Based on the study in [21], this method is reported to be suitable for automation due to the use of objective information for inputs. The construction of assembly relations in this method is presently not automated and therefore, the method is time consuming and presents the possibility for human error in the construction of assembly relations [18]. Besides automation, another benefit of using objective information as input is the repeatability of the predicted assembly time for a given assembly.

Bi-partite graphs are used for the representation of the assembly relations from the CAD assembly file. The method lists four types of assembly relations that are based on the physical connections between parts in the assembly. A physical connection is the contact between parts in the assembly. The four assembly relations are: surface contact connection (two flat surfaces touching each other), fastener connection that includes all

types of clamping, snap, press, and interference fit connection, and other connections such as shaft and a hole instance and electrical types. Figure 1.1 shows an example of the bi-partite graph developed for a fastener assembly relationship [20]. In the example, the bolt fastens the Plate\_top having a clearance hole to the Part\_bottom having a tapped hole. The bi-partite graph used in this method (see Figure 1.1 right) only provides information about the assembly relationship between the three parts and not the assembly order.



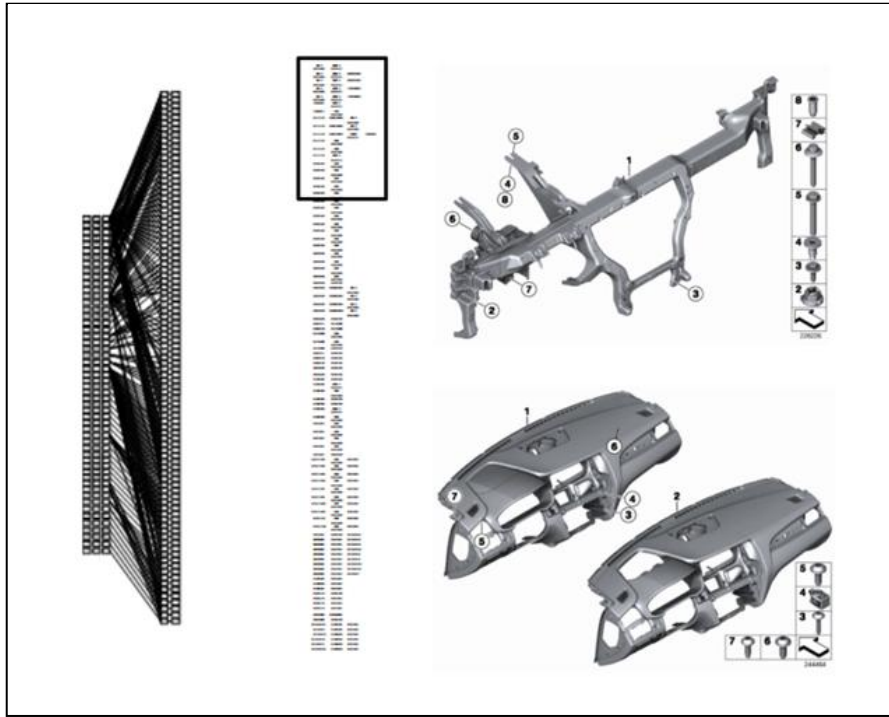
**Figure 1.1: Left – Section view of a fastener connection; Right – Bi-partite graph showing connectivity between three parts**

The part connectivity information and the metrics based on part connections are both fundamental to this method. Presently, the part connectivity information is captured in the form of a bipartite graph. The current research challenge is constructing the graph of part connections from the CAD assembly file. In the present state, the extraction of part connections and developing metrics are performed manually which is a tedious process. To illustrate further, the example shown in Figure 1.2 is from an automotive

sub-assembly [18] where the part connectivity graph is developed manually by examining the assembly relationship. The sub-assembly is manually analyzed and the connections between parts are recorded as a bipartite graph that leads to:

- Increased model set-up time: Depending on the size of the assembly, the time spent on the analysis and the verification of part connections varies and results in the time consumption for setting up the graph.
- Erroneous connections: The manually generated part connectivity graph requires quality check to ensure that the erroneous connections are not recorded or the connections are not missed.
- Integration of sub-assemblies and the main assembly: The presence of sub-assemblies requires the integration of the part connectivity graphs between sub-assemblies and the integration of part connectivity graphs of the main assembly and the sub-assemblies. This phase requires additional time and resources.

The presence of such issues in the development of complexity metrics can lead to erroneous assembly time estimation and in turn can lead to design reasoning on wrong data.



**Figure 1.2: Part connections and relationships developed for an automotive sub-assembly [18]**

Certainly, it is evident from the identified issues that there is a need for the automated generation of part connectivity graph from the CAD assembly models. To this end, the research in [9] demonstrates the use of assembly mates for the automated generation of part connectivity graph but is limited to the type of mates offered by the CAD software and the type and the number of mates used by the user. Also, the part connectivity graph used for input in this method does not consider the amount of overlap between the connected parts. The motivation of the research presented in this thesis is to extract the part connectivity graph and the amount of overlap between the connected parts.



### 1.2.2 Product Complexity Method Based on Neural Networks

The method is similar to the Connective Complexity method with regard to using part connections for the assembly time prediction but differs in the model development technique. In this method, the artificial neural network (ANN) approach is selected to develop the model in place of the previously used regression analysis [18]. The ANN was selected due to its capabilities of handling the non-linearity of the metrics [1]. The method is intended for the assembly time estimation of automotive systems and is derived from the original part connectivity based method [18].

The Product Complexity method demonstrates its applicability to the automotive industry with the assembly time estimates having a deviation of  $\pm 15\%$  from the target values. However, problems associated with manual construction of part connectivity graph are similar to the issues discussed in section 1.2.1 for the part connectivity method. Increased model set up time, erroneous connections, and integration of part connections between sub-assembly and main assembly offers a need for developing computer algorithm for automated generation of part connectivity graph.

The challenges that need to be overcome for the manual extraction of physical connections for automotive assemblies are further amplified due to the complexity of the system. Here complexity may be due to the large number of components in the system, difficulty in disassembling certain systems into smaller elements, identifying concealed connections such as adhesives and interference fits, and the size of the system to list a few.

A feature recognition algorithm to extract connectivity graph is not presently found in the literature that would help in the automated data collection process [18]. However, a more recent research looked at using the assembly mates from CAD system to build the connectivity graph, but that is dependent on the type of mate used, whether the assembly is fully constrained or partially constrained, and the user practices [19]. This approach leads to some amount of variation as the assembly mates selected depends on the user preference and practices. Therefore, a feature recognition algorithm to retrieve physical connections would be a useful tool repeatability of the results. The automation of the extraction of physical connectivity graph is common to both the product complexity method and the part connectivity based method. The algorithm can support both these methods and hence demonstrate the need in multiple DFA methods.

### 1.2.3 Assembly Mates Based Time Estimation Problem

Based on the study that evaluated Boothroyd and Dewhurst DFA method and the Connective Complexity DFA method for feasibility of automation, the Connective Complexity method was selected due to its objective inputs that could be retrieved from solid modeling software [9]. Solid modeling software is a popular tool used in the product development process [2]. The benefits offered by solid modeling software are improved product quality, reduced product development time, reduced product cost, and increased performance [3]. CAD software package is generally used across all product development companies for the representation and exchange of the part model data. The assembly mates based time estimation method makes use of the information contained in CAD models to build the complexity metrics.

The method uses the mates, which are used to constrain solid models in a CAD assembly file as a substitute for physical connections. The physical connections from Connective Complexity method represent the types of connections between components; for example, surface contacts, fasteners, fits (snap, press, and interference), and other connections (shafts, springs, and electrical). Extraction of such information from CAD software requires a feature recognition algorithm with the capability to identify physical connections. For this purpose, the feature recognition algorithm needs to evaluate all features in the solid model and perform comparisons with features from other models in the assembly to identify the physical connections. The computational effort of such an algorithm can get expensive depending on the size of the assembly and the number of features in the solid model. Therefore, as an alternate solution assembly mates were selected to represent the connections between the components in this method.

The mates are used between the assembly components to constrain their degrees of freedom at correct locations to simulate the real world assembly. Hence, the mates can offer information about the components' location and their connectivity relationship in the assembly. Adding mates is a necessary part of CAD modeling practice that is helpful in making assembly drawings and performing analyses (CAE, tolerance, motion, and packaging). In this method, the SolidWorks CAD software is used for the research and hence the mates offered by SolidWorks software were utilized to develop the complexity metrics. Table 1-2 shows the list of mates offered by SolidWorks software for the 2010 education edition.

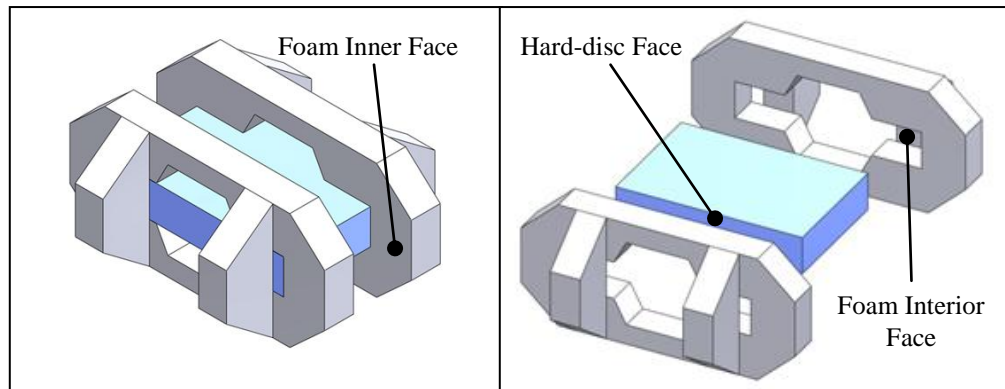
**Table 1-2: All mate types offered by SolidWorks software [4]**

	Mate Types	Geometric Entities
Standard	Coincident	Coincides faces, edges, planes, and vertices on the same plane
	Parallel	Makes selected geometric entities parallel
	Perpendicular	Makes selected geometric entities perpendicular to each other
	Tangent	Places a geometric entity tangential to a spherical or cylindrical entity
	Lock	Freezes the present position and orientation of the part
	Distance	Maintains specified distance between geometric entities
	Angle	Maintains specified angle between geometric entities (orientation)
Advanced	Symmetric	Makes similar entities symmetric about a plane
	Width	Centers to the width of the groove
	Path	Constrains a point to a path
	Linear	Establishes linear relationship between two components
	Limit	Limits movement of components to a specified tolerance
Mechanical	Cam	Makes a cylinder, plane, or point to be coincident or tangent to a series of tangent extruded faces
	Gear	Makes two components to rotate relative to one another about selected axes
	Hinge	Allows one rotational degree of freedom
	Rack and Pinion	Linear translation of a part causes rotation in the other
	Screw	concentric and pitch relationship between rotation of one and translation of the other
	Universal Joint	Rotation of one component about its axis is driven by rotation of the other about its axis

The mates based connectivity relationship established for all components in the assembly is a bi-partite graph of components' name that indicates if a mate was defined between the two components. Once the bi-partite graph of mate relationship is established, the process followed to develop the assembly time estimation model is similar to the process followed in the Product Complexity method with artificial neural-nets. The procedure for this method is, first, the SolidWorks add-in developed as part of this research gets the components name between which a mate is defined from the SolidWorks feature manager tree and forms a bi-partite graph [9]. Second, the graph is analyzed with a Matlab algorithm that generates twenty-nine different complexity

metrics. Third, these complexity metrics in conjunction with the respective MTM times for the assembly is used for neural-net training. Based on the neural-net training conducted for twenty-four products, a relationship is developed between the complexity metrics and MTM assembly times that is used for the assembly time estimation.

Although, the assembly mates based time estimation method demonstrates potential for complete automation of the DFA method it is shown that this method is sensitive to the number of mates defined in the assembly. The number of mates and the type of mates used are factors that depend upon the geometry, best practices, user preference, software, and the application the CAD assembly is intended for. A study was conducted to evaluate the variation in the predicted assembly time when different designers constrain the same assembly file and the general variation is observed to range from -7% to +27% [9]. The sensitivity of the assembly times with respect to the use of different mate types is acknowledged but not yet been explored. For instance, the assembly of hard-drive packaging with foam (see Figure 1.3) demonstrates a case where this assembly could be constrained alike with the use of different types of mates.



**Figure 1.3: Hard drive packaging with foam [5]**

The distance mate, lock, and coincident mate discussed in the Table 1-2: All mate types offered by SolidWorks software Table 1-2 can all be used to constrain the two foams in its proper location. The distance mate could establish a distance relationship between the two inner-faces of the foam; the lock mate can arrest the parts in their current location; and coincidence mate can mate interior faces of the foam with the respective hard disc faces. This type of variability can exist for all components in the assembly. Another type of variation discussed in the research is the variation in the number of mates used. Based on whether fully constrained assembly is used for neural-net training or the partially constrained assembly, the predicted time is shown to vary between -44.2% to +101.6% [9].

The issue of variability in the predicted assembly time due to the use of different number of mates and the different types of mates demonstrate the necessity for a feature recognition algorithm that could extract only the physical connections between the assembly components consistent with the original Connective Complexity method. Use of contact relationship between the components is both objective and independent of the mates' usage. The use of contact relationship also provides opportunity to develop weighted graph based on the area of contact for developing complexity metrics. The weighted graph could be used to explore the influence of additional metrics based on the minimum spanning tree, cycles, number of nodes and edges, traversability, graph connectivity, and isomorphism [6]. Previous work on the assembly time modeling has already investigated the performance of neural-nets with bipartite graphs, and hence there is an opportunity to explore the behavior of neural-nets with the weighted graphs. The

computer algorithm, therefore, exhibits a requirement for the automated retrieval of physical connections from the assembly which addresses the issue of subjectivity in the mates based method.

### 1.3 Inference - Necessity for a Duplicate Geometry Algorithm

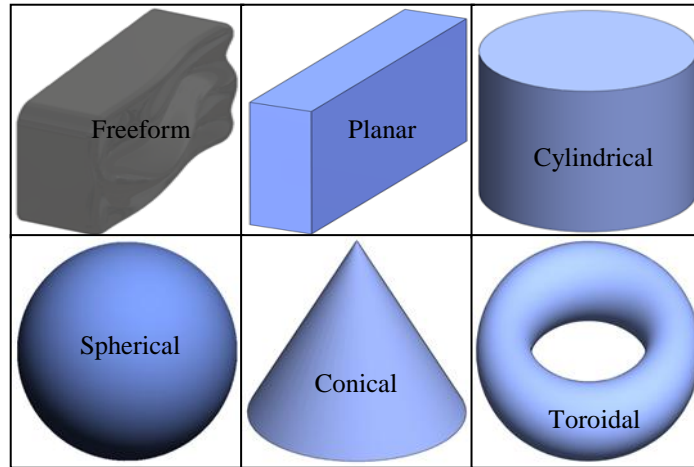
The discussion on lazy parts light weight engineering method and the assembly relations based DFA method both highlight the need for a feature recognition system that could support both applications towards automation. In the case of light weight engineering tool, the feature recognition (FR) algorithm needs to identify instances of duplicate geometries in the CAD assembly. Duplicate geometries are two geometries that possess certain user-defined amount of similarity lying within the threshold distance and threshold orientation (formal definition is provided in section 3.1). For the assembly relations based DFA method, the FR algorithm needs to identify and record the connectivity between components in the assembly. Thus, the focus of this research is developing a FR algorithm that consists of user controlled parameters that is useful for both lazy parts method and assembly relations based DFA method.

The current state of the art in the feature recognition technology focuses mainly on the integration of CAD and CAM, CNC visualization, process planning, and manufacturing [24–27]. A feature recognition algorithm to support the automation of duplicate geometry identification for the lazy parts method need to be developed with the focus on user-controllable parameters[2]. Also, the FR algorithm for the automated extraction of assembly relations from CAD data for assembly time estimation presents another opportunity for research[18]. That said, a tool to extract the CAD assembly

mates to represent assembly relations (also, design intent) is developed but that is user defined in nature and do not represent the actual physical contact based connectivity between parts [19].

Therefore, the intended requirement for the FR algorithm of this thesis is its extensibility to support both duplicate geometry identification and assembly relations extraction. The idea is to have single feature recognition system with user driven parameters that can provide the required extensibility. The value of the parameters could be controlled to have the FR algorithm to support either lazy parts method or connectivity based DFA method. Additionally, it is also desired to have the feature recognition system that is independent of the geometry type. The geometric shape of parts in the assembly can be formed of different types as shown in the Figure 1.4. Therefore, it is necessary for the feature recognition algorithm to be able to evaluate different geometric types. Such an algorithm would allow for the functioning with various types of geometry such as freeform, planar, cylindrical, spherical, conic, and toroidal to name a few (see Figure 1.4).





**Figure 1.4: Different 3D shapes in geometric modeling**

The benefit of the research presented in this thesis is the development of a feature recognition system that can support the automation of duplicate geometry identification of lazy parts method and assembly relations retrieval for connectivity based DFA method. The automation of both these methods will address the repeatability of the methods. Presently both methods are manual and therefore automation can prevent the potential errors arising from manual data collection. To illustrate further, the FR algorithm can help in the retrieval of the same instances of duplicate geometries for a given CAD assembly for lazy parts analysis. Similarly, for connectivity based DFA method the FR algorithm can ensure the extraction of same connectivity graph for a given CAD assembly. The potential errors associated with the manual construction of assembly relations are eliminated. Increased productivity is another benefit of the automated FR system [7]. This way, the FR system can allow designers more time to focus on the data rather than on the data collection processes.

In this chapter the motivation for the duplicate geometry FR algorithm is presented. Lazy parts light weight engineering method and physical connections based DFA method demonstrate a need for the duplicate geometry FR algorithm. The algorithm will consist of user-controllable parameters to modify the applicability of the system and would be independent of geometric types. The FR algorithm can help with reducing the inconsistencies associated with manual data collection and modeling technique. In the next chapter, the current state of the art in feature recognition technologies will be explored.

#### 1.4 Overview of Thesis

The motivation for the research presented in this thesis was discussed in this chapter. The rest of the thesis is organized in the following way:

The Chapter Two of this thesis presents the literature review of feature recognition algorithms that use b-rep data for the evaluation. Based on the motivation discussed in Chapter One and the existing feature recognition algorithms, the need is identified for the development of the duplicate geometry feature recognition algorithm to support lazy parts method and the part connectivity based assembly time estimation method.

The Chapter Three presents the research objective, definition of the duplicate geometry, and discussion on three conditions derived from the duplicate geometry definition. Furthermore, list of requirements is generated to meet the research objective and the definition of duplicate geometry.

The system architecture and the implementation details of the algorithm are presented in Chapter Four. The discussion on system architecture demonstrates the design that meets the usability requirements. The remainders of the system requirements are addressed in the implementation of the algorithm.

The Chapter Five presents the validation of the algorithm using the test cases. This chapter explains the design of test cases to check the algorithm against specific requirements and presents the results of the analyses.

The Chapter Six is the concluding chapter in this thesis that presents the research contribution and future work.

## CHAPTER TWO: LITERATURE REVIEW OF FEATURE RECOGNITION ALGORITHMS

Most Feature Recognition (FR) algorithms discussed in the literature are intended for extracting features for manufacturing [28–31] and Computer Aided Process Planning (CAPP) applications [28,32–34]. The FR algorithms intended for other domains such as structural design and analysis [35–38], sheet metal applications [25,26,28,29], and stress analysis [26,30,36,39] to name a few is less common. The extraction of manufacturing features from a solid model involves the conversion of low-level topological and geometric information contained in the CAD model to usually higher-level semantic information applicable to the Computer Aided Manufacturing (CAM) system usage [40]. To do this conversion, there are different types of feature recognition systems depending on the type of geometric engine used in the CAD software, underlying representation of the data, and the procedure used for reasoning in the algorithm. However, in this thesis the feature recognition algorithms discussed are based on the Boundary Representation (B-rep).

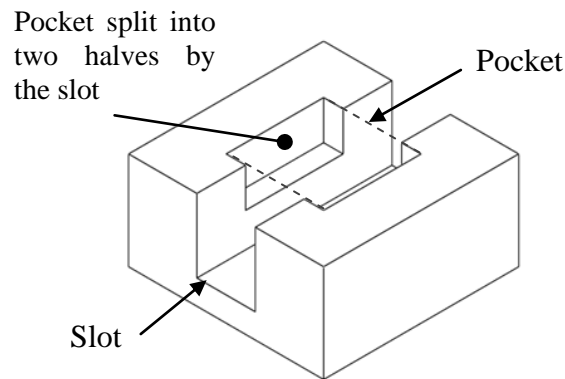
In the following sections, five popular methods for feature recognition are reviewed: graph-based method, hint-based method, convex hull decomposition method, cell based volumetric decomposition method, and the hybrid method. The discussion will focus on the feature representation used for recognition, types of features supported, adaptations to include additional features, reasoning procedure and strategies, merits and challenges, and the comparison of different approaches.

## 2.1 Graph-Based Method

In the graph based approach, the B-rep of the solid model is used to develop the attributed adjacency graph (AAG) for feature recognition [41]. B-rep is a graph representing the connectivity of topological elements (faces, edges, and vertices) in the solid model, each element having also associated geometric entities. Alongside B-rep, the adjacency information of faces, edges, and vertices are essential for feature recognition and may be represented through the AAG [41]. A node of the AAG is an identifier of the face and therefore, every face of the solid model consists of a unique node. Similarly, an arc is a unique identifier for every edge in the solid model. Attributes provide information regarding whether the two faces sharing an edge form a concave or convex angle. Other geometric information can also be attributed, but convexity is the most common attribute form.

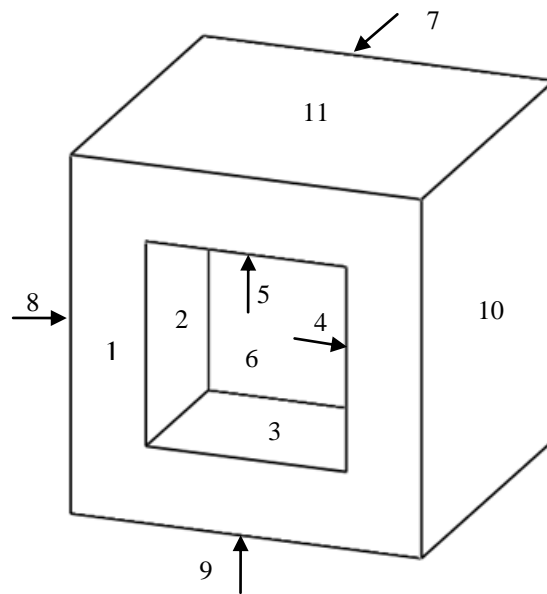
The example shown in Figure 2.3 is an AAG for the part with a pocket feature on its “face one” (see Figure 2.2). In this AAG, the numbers inside the circle nodes represent the unique identifiers for each of the eleven faces in the part. The connection between two nodes is an arc that is a unique to the corresponding edge. The number (0 and 1) linked to the arcs are attributes that inform if the two faces sharing an edge form concave or convex angle. Zero is used to represent concave angle and one is used to represent convex angle. The graph is then analyzed to delete nodes associated with the attribute one. The algorithm uses “if... else...” rules for the recognition of different types of features. The method is able to recognize wide range of polyhedral features and nested features [41]. The limitation of the method is recognizing all types of interacting

features. Interacting features are single or multiple features that are split by another feature. For example, see Figure 2.1 where a slot is machined over a square pocket thus splitting the pocket into two halves. Also, the method is only applicable to planar features while other features such as cylindrical, toroidal, spherical, conical, and freeform (see Figure 1.4) are not recognized.

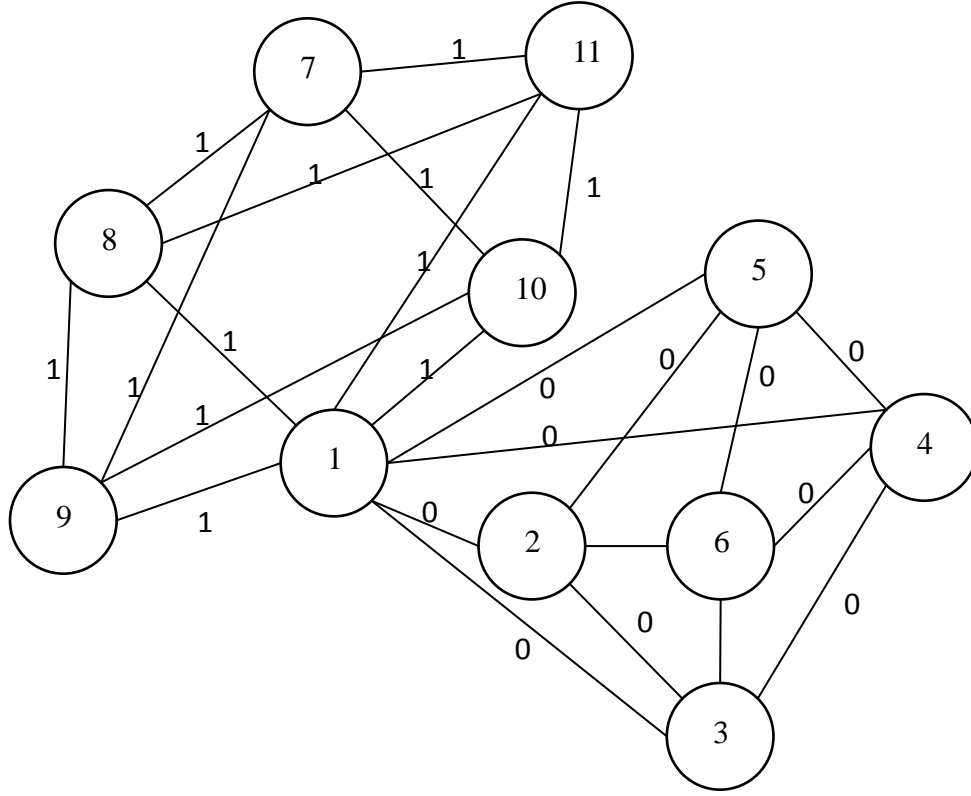


**Figure 2.1: Interacting features; square pocket is split into two halves by a slot**

Interacting features may be addressed using multi-attributed adjacency graphs (MAAG) [42]. The MAAG uses a modified winged edge data structure [43], called enhanced winged edge data structure (EWEDS) that has labeled faces containing pointers to boundary edges to construct the graph. Again, the algorithm for processing the graph is rule-based with graph matching conditions.



**Figure 2.2: Cube with a pocket**

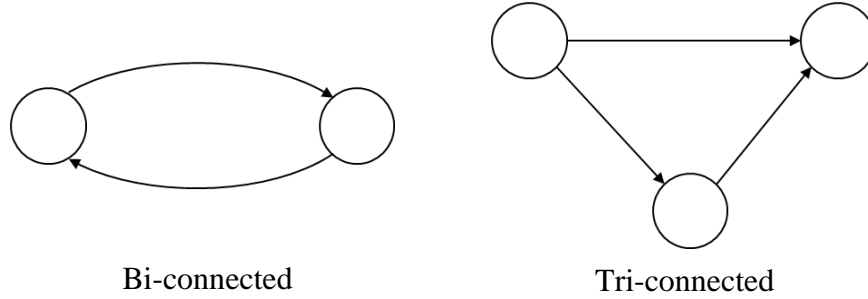


**Figure 2.3: AAG for the part in Figure 2.2**

An alternative method uses generalized edge-face graph (GEFG) to represent the solid object's boundary model [44]. GEFG provides the connectivity information about the topological entities in a solid model. In contrast with the AAG, the GEFG uses two additional topological entities, the shell and the loop for graph construction. The shell is the maximum number of connected faces and the loop is a closed loop of edges [44]. The method decomposes the GEFG into bi-connected and tri-connected sub-graphs for the recognition of depressions and protrusions on the face. This method also uses rules for feature recognition. A distinguishing aspect of GEGC is that the sub-graphs are directed and acyclic as shown in Figure 2.4, where each sub-graph represents a feature in the part.



This method can identify cylindrical features in addition to planar and features that lack axial symmetry [44].



**Figure 2.4: Bi-connected and Tri-connected acyclic directed graph**

The cavity graph algorithm is another graph based approach that uses convexity information for feature recognition [45]. The representation is modification of AAG, where the nodes also contain information pertaining to the orientation of the face. For example, a node with label {5: -Y} indicates that face five in the solid model has a topologically correct orientation (normal pointing away from material) in the negative Y direction. For this representation, a challenge in graph construction is the selection of the correct base face. Despite this, the representation has helped to overcome the problem of identifying interacting features. The algorithm uses the concept of virtual links to recognize interacting features. The virtual links are the edges that would be present in the absence of the interacting feature. The orientation labels used with the nodes are all aligned with orthogonal Cartesian directions. The method uses logic rules to evaluate the hypothesis.

Another type of graph used for feature recognition is the loop adjacency hyper graph (LAHG) for the boundary representation of a solid object [46]. LAHG is a modified form of face adjacency graph (FAG) that contains the additional hyper-arc showing the relationship between the inner and the outer loop. This approach further uses the matrix form of the LAHG called loop adjacency matrix (LAM) for computations. The method is intended for planar surfaces.

The multi-resolution reeb graph (MRG) is an extension of previous work [47] that is used for comparison of similar models [37]. The method generates a polyhedral approximation of the solid model through faceting and thereafter constructs the MRG. The MRG's of two geometries are used for graph based comparisons. The method is sensitive to topological relationship, but becomes less sensitive for complex geometries [37].

Reviewing the graph based approach for feature recognition indicates that the method works well for polyhedral features. Additional features, such as cylindrical, can be detected but requires geometric and adjacency information to be captured in the graphs. Preprocessing for the construction of solid model's representation is expensive [25]. The MRG approach has been shown to be useful for shape comparisons of diverse shapes, but method require further research dealing with missing faces and edges, and high sensitivity with the VRML format and topology.

## 2.2 Hint-Based Method

The hint-based approach uses logic rules based on the topology data of a solid model to generate hints for feature recognition. The faces in the solid model are the preferred topological entity used for hint generation and need to satisfy certain topological and geometric relationships. These hints only form a partial representation of the feature that still requires further analysis for full feature recognition [30]. Essentially, the hint-based approach incrementally examines possible instances of features, while the graph based approach defines the features all-at-once. For example, instances of cylindrical faces may serve as a hint for the presence of holes, while planar parallel faces with a floor may provide clues about the slots. This strategy was used to develop a feature recognizer for interacting features [25,30,48]. Hints may also be generated using other information such as semantics and geometric attributes from the part. As an example, a similar hint based feature recognition system uses geometric attributes from both part and stock for the construction of well-behaved feature instances [26,29,48]. The basic principle is that if a part can be produced by machining the stock, then the material removed from the stock represents features in the part. This approach helps in devising the strategies to machine a part from the stock.

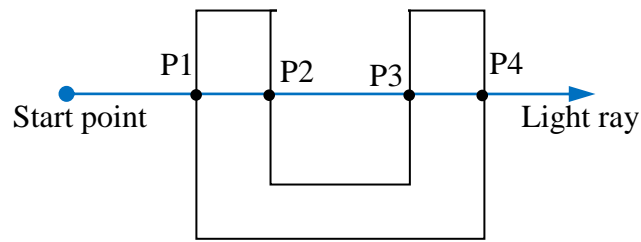
The objective of the hint based approach is to look for feature hints and then incrementally solve them to find full features [30]. As opposed to searching for full features, this approach helps in dealing with feature interactions. Rules are used to categorize hints and features into sub-classes, such as promising, unpromising, and rejected groups [30] and or to define accessibility of the features [48]. Feature hints are

then used to produce the largest nonintrusive feature volume by extending the feature along specific directions through feature completion. The extension could be both in one-dimension or two-dimension; linear extension is an example of one-dimensional feature extension and translational sweeping of the points on a feature's cross section is an example of two-dimensional sweep. The completed features are then verified using validity rules and invalid hints are dropped [30].

Besides using the topology relationship to generate hints for feature identification, a different approach is the ray-firing technique that has been used to generate hints based on the idea of human type analysis [49]. This method is illustrated with the example shown in Figure 2.5. The figure shows a slot machined into a rectangular part. The points P1 and P4 represent the points on the outer faces of the part and the points P2 and P3 represent the points on the inner faces of the part. When a ray is fired, the faces that are hit by the ray is flagged and checked whether they form alternate depressions and protrusions as shown in Figure 2.5. In this figure, the points pairs P1-P2 and P3-P4 represents a protrusion, and the points pair P2-P3 represents a depression which is used as a hint for the identification of features. The sequence of points is only a hint that needs to be solved for the full feature recognition.

To summarize, hint based approach is predominately adopted to address standard machining features formed from drilling, milling, chamfering, and filleting whose traces are stored in the pre-defined library. The hint based approach uses specific rules to generate, classify, and drop/select the hints. The information used to generate hints is the topological and geometric relationships in the part. Recent work [25] has extended this

method to also use tolerances and geometric attributes to generate hints. The algorithmic overhead for hint based approach is due to the storage of hints in the pre-defined library, the processing of these hints against rules to construct complete features, and the verification. However, the advantage of using hints is the reduction in the search space for features. As opposed to graph based approach that uses pattern matching and processing of all features, in the hint based approach only those features that are selected based on the hints are considered for further processing. The complexity of the algorithm for the hint based approach is polynomial in nature.



Hint generated: (P1-P2, P2-P3, P3-P4)

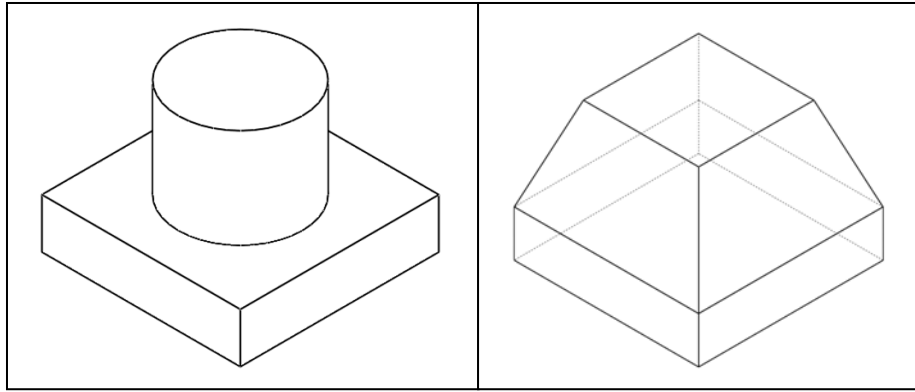
Alternate pair of points represents hints for protrusion and depression

**Figure 2.5: Hints generated through ray-firing**

### 2.3 Convex Hull Decomposition Method

A convex hull decomposition approach uses the constructive solid geometry (CSG) models of complex geometric shapes defined through a collection of regular primitives for feature recognition. The concept of representing a solid using primitive shapes are also observed in B-rep solid modeling, parametric solid modeling, and FEM

(finite element model) [50–52]. The method was originally conceptualized [25,28] and then extended into Alternating Sum of Volumes (ASV) decomposition [26,53]. The objective of the method is to create a convex hull around the boundary of the solid model. The convex hull represents the smallest non-concave envelop of the solid model consisting of planar faces as shown in Figure 2.6. The subtracted difference between the part and the convex hull represents delta features in the part. This approach is used for the recognition of depressions, such as slots, pockets, and holes, in the solid model and hence is suitable for non-convex parts [50]. The difference between the convex hull and the delta features provide the representation of the part. The creation of delta features from the convex hull is continued until all the features in the part are exhausted. In case of interacting features, the combination of multiple decomposed features may represent a single complex feature in the part. Otherwise, maximal features can be used to represent non-interacting features in the part [50]. As the process of decomposition is continued for all instances of delta features, the delta features of both convex and concave nature are obtained and hence the method is as the alternating sum of volumes [26,53].



**Figure 2.6: Left: Part with a cylindrical protrusion; Right: Convex hull for the part**

The lack of a termination criterion for the continued creation of the delta features result in the problem of non-convergence. However, the Alternating Sum of Volumes with Partitioning (ASVP) method addressed this problem by combining ASV decomposition and remedial partitioning [54,55]. The ASVP method was extended to extract Form Feature Decomposition (FFD) from each component in the assembly, which is a set of positive and negative form features [56]. Equivalent positive and negative form features from two distinct components provide the assembly mating relationship that is used for assembly planning. The conversion of positive form feature to negative form feature, called Negative Feature Decomposition (NFD), is used to obtain material removal volume from the components [57,58].

In summary, the convex hull decomposition approach uses the difference between convex hull and the part to represent features (delta volume). If the delta volume is empty then the algorithm terminates, otherwise the delta volume is recursively decomposed until termination. This method is suitable for polyhedral parts. The method was extended to

identify cylindrical features, but is not fully successful against cylindrical interacting features [28] because of approximation of all shapes into the polyhedral form. Due to this reason, the approach requires final reconversion of the cylindrical features from their polyhedral form [26]. This reconversion step and the decomposition of delta features make the algorithm expensive.

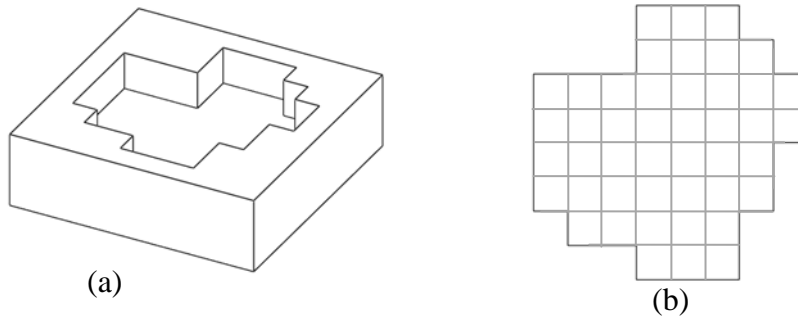
#### 2.4 Cell Based Volumetric Decomposition Method

The term ‘cell decomposition’ refers to representing a given shape in terms of constituent volumetric cells so that combining the cells back together gives the original shape [51]. In contrast to convex hull method, here the delta volume is decomposed into unit volumetric cells without the use of convex hull. Thereafter the unit volumetric cells are combined together to form maximal volumes that represent features. The voxel representation of a solid, that also uses unit cells, is different compared to the volumetric unit cells because the voxels may not always be able to combine to get the exact original geometry. Due to this reason, a voxel representation cannot be classified as cell decomposition [59,60]. The cell based volumetric decomposition method consists of three steps. First, the part is subtracted from the stock to obtain delta volume. The delta volume is decomposed into unit cells by using selected faces or half spaces. Second, the unit cells are combined to form maximal volumes based on the constraints related to manufacturing operations. Finally, the last step involves classifying the maximal volume as a specific type of machining feature.

However, the challenge associated with combining the unit cells back together results in the possibility of multiple feature interpretations. The condition that while



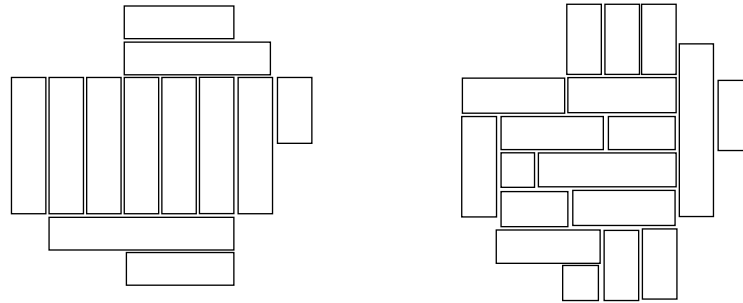
combining unit cells to maximal volume at least one face of the cell need to share a face with the part generates more than one possible combination. For instance, for the part shown in Figure 2.7, there are multiple ways of connecting the unit cells into maximal volumes as shown in Figure 2.8. Another problem, referred to as “the global effect of local geometry” where cell decomposition globally extends the surfaces or half spaces related to the faces of delta volume to regions where machining features would not extend. This results in the creation of cells that do not represent the machining feature that needs to be resolved to avoid multiple machining feature interpretation [25,28]. Also, in case of cylindrical and freeform surfaces some of the unit cells generated may represent voids or other unnecessary spaces that are discarded [26].



**Figure 2.7: (a) Part; (b) cell decomposition of the delta volume**

Two approaches are used to connect the unit cells into maximal volume. First, the connection is based on the adjacency relation between the unit cells which results in a non-convex volume [26,51,61,62]. The second approach uses a more selective strategy to combine cells based on adjacency rules [26,63]. Topology graph of the solid model and tool approach direction are other factors considered for the volume classification

[26,45,51,63]. Graph-pattern matching has been used in conjunction with heuristic rules to avoid unnecessary combination of the unit cells [51,61].



**Figure 2.8: Two distinct maximal volume interpretation**

In summary, the cell based volumetric decomposition method uses decomposition of the delta volume, re-composition of unit cells, and classification of maximal volumes as the three steps for the machining feature recognition. The algorithm for re-composition of the unit cells into maximal volumes is computationally expensive because of the reasoning required to interpret maximal volumes that do not match with pre-defined feature type. The approach is suitable for interacting features with planar surfaces, but problems persist with freeform and curved surfaces.

## 2.5 Hybrid Method

The hybrid approach uses a combination of previously discussed strategies to overcome the limitations that persist in the individual methods, mostly to deal with interacting features. It has been argued that three major feature recognition techniques, graph-based, hint-based, and volumetric decomposition (convex hull and cell decomposition both use volume decomposition) approaches, are unique and difficult to be

combined into a single algorithm [28]. However, the authors do recognize the benefits of combining such conventional feature recognition techniques referring to the work found in [31].

A combination of the graph-based and hint-based approaches is used to develop a general purpose algorithm to recognize interacting features and improve the computational efficiency [31]. This algorithm uses Extended Attributed Adjacency Graphs (EAAG) to represent features in the solid model. EAAG is an enhanced version of the attributed adjacency graph (AAG) [41], which includes additional arc and node attributes (see Table 2-1 for the additional attributes stored in EAAG).

**Table 2-1: Additional attributes of EAAG [31]**

Arc attributes	Node attributes
Concave edge or convex edge?	Stock face or part face?
Real edge or virtual edge?	Is face common to both the part and its convex hull?
Inner loop or outer loop?	Number of loops?
Curved edge or straight edge?	Is the split face unifiable or not?
Smooth blend or sharp edge?	Is the face planar or non-planar?

The EAAG is decomposed into manufacturing face adjacency graphs (MFAG's) obtained by deleting the stock faces and faces that are common to both the part and its convex hull. Each MFAG generated is compared with all the EAAG's corresponding to graphs of predefined features stored in the library. The feature recognition is accomplished by the graph matching between MFAG and the EAAG's in the library. For instance, if the MFAG of a particular feature in the part matches with the EAAG of a T-slot in the library, then the feature is declared as a T-slot. However, if no match is found

then the MFAG's are evaluated against the sequential list of heuristic rules in the library for the identification of other general features such as different kinds of pockets. If no match, either in the predefined feature library or the heuristic rule library, is found, then the feature is interacting and a minimal condition sub-graph (MCSG) is generated for feature recognition.

The MCSG is a sub-graph of EAAG generated through the decomposition of MFAG using the arc and node attributes shown in Table 2-1. MCSG's are used as hints for the identification of interacting features. The construction of MCSG is done in two steps: (1) virtual links between face pairs are generated based on conditions proposed in [64] and (2) features are constructed based on the virtual link classification. Once, the construction of MCSG's is completed the alternate feature interpretations are generated using heuristic rules from the library. Some of the advantages of this algorithm are the extensibility to include additional features in the library without modifications requiring to the code, reduction in the search space due to the use of virtual links and MFAG's, and the alternate interpretations of interacting features [31]. The limitation for this approach is with the identification of open pockets, but solution strategies are proposed to overcome the limitation.

## 2.6 Comparison of Techniques

Reviewing different feature recognition techniques, it is seen that the common challenges faced across all approaches are the recognition of interacting features, dealing with free-form surfaces, and having a general purpose algorithm for all feature types. The solid models' topological entity relationships with certain geometric attributes are

the preferred representation used in the graph-based, hint-based, and hybrid feature recognition approaches. Different kinds of representation used for the feature recognition purposes include the labeled graph, directed graph, bipartite graph, and undirected graph. The feature representation in convex hull decomposition and cell based decomposition techniques are volume based, and hence volumes of primitive shapes are used for feature representation. The comparison of previously discussed feature recognition techniques are shown in Table2-2.

Graph matching and logic rules are the commonly used reasoning procedure to identify features. In case of the graph-based, hint-based, and hybrid approaches, a pre-defined library of sub-graphs is used for the recognition of features. Due to the necessity for such a library, the types of features identified are limited depending on the library size and the code requires modification if new features are to be added into the library. However, one example demonstrates the potential to use pre-defined library while still allowing for the addition of new feature types without the need for changing the code [31]. Some of the other reasoning systems used for feature recognition are heuristic rules and artificial neural networks.

**Table2-2: Comparison of feature recognition techniques**

FR Technique	Feature Representation	Reasoning	Geometry	Independent of feature type?	Complexity
Graph-based	Topology, Geometry	Graph matching [31,47] Heuristic [34] neural nets [65] logic rules [41]	Planar, Cylindrical	No; includes pre-defined library of feature types	Exponential [28,66]
Hint-based	Topology, Geometry, Heuristics, Ray firing [49]	Graph matching, Rules	Planar, Cylindrical, Second-order curves	No; includes pre-defined library of feature types	Polynomial [28,48]
Convex hull	Delta volume of primitive shapes	Rules, Graph matching	Polyhedral, Cylindrical [58]	Independent of feature type	Exponential
Cell decomposition	Maximal volumes	Logic Rules, Heuristic [51], Graph matching [51]	Polyhedral	Independent of feature type	Exponential [28]
Hybrid	Topology, Geometry, Heuristics	Graph matching [31] Rules	Planar, Cylindrical, Second-order curves	No; includes library of feature hints	Polynomial [31]

The type of geometry supported by a feature recognition algorithm depends upon the underlying feature representation used and the reasoning structure. Most of the graph-based techniques are able to recognize planar and cylindrical features. In the hint based method, the use of partial features as traces and the subsequent reasoning on the incomplete feature hints has allowed for the identification of analytical surfaces. In the

case of the convex hull and volume decomposition methods, the feature types supported are limited to polyhedral and cylindrical volumes because of the approximations associated with the re-composition of the maximal volumes. The multiple-level reasoning in the hybrid approach has demonstrated much promise with the identification of analytical surfaces and interacting features.

The feature recognition techniques reviewed in this chapter were mostly intended for specific application domains. Most common application of the feature recognition algorithms are for the use in computer aided manufacturing (CAM) software for machining and computer aided process planning (CAPP). The types of features that need to be identified by the feature recognition system are governed by the definition of a feature for a particular application. Notably, a standard definition for features or feature classification is not found in the literature. The application domain for the feature recognition system developed in this research is for the design analysis of CAD assembly models. The specific requirements for the new system are found in Section 3.3 based, partially, on this review. For feature recognition, the definition of a feature for the scope of this research is discussed in Chapter Three.

## CHAPTER THREE: RESEARCH OBJECTIVE

### 3.1 Definition of Duplicate Geometry

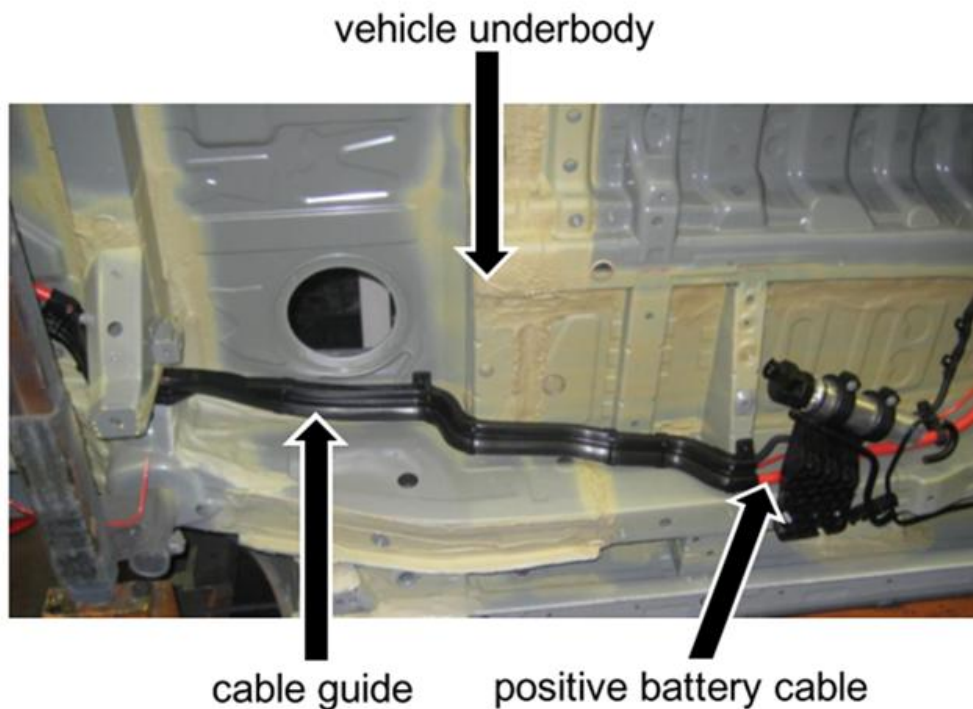
There is no standard definition for features and the current definitions found in the literature depend on the downstream application where the model will be used [41]. Features can hold different meanings based on use context. The definition of features vary depending on whether the FR algorithm is intended for identifying machining features, extruded features, polyhedral entities, or features for stress analysis. For example, extruded entities in the part are classified as a feature for the finite element modeling application for mesh generation. However, for machining purposes only concave features are classified as features to calculate the tool path and the amount of material that needs to be removed to produce that feature. Also, presently there is no standard definition for features and it is argued that it may not be possible to have a single definition covering all feature types [26,28].

For the research in this thesis, a feature recognition algorithm is needed to support the duplicate geometry identification and extraction of assembly relations from CAD assembly. Recalling the duplicate geometry identifier from lazy parts indicator mass reduction method in Chapter One, the definition is “two or more similar geometries that lie in close proximity to each other [2]”. An example for duplicate geometry is the vehicle underbody and cable guide as shown in Figure 3.1 where the profile of the cable guide follows the profile of the vehicle underbody and both geometries lie close to each other. However, as discussed earlier this definition is not comprehensive and therefore



identification depends on engineering judgment. To explain this further, some of the questions that need to be answered objectively for the identification of duplicate geometry are,

- Do the two geometries lie close to each other?
- What distance between the geometries can be regarded as close?
- Are the two geometries similar?
- If similar, what is the amount of similarity required?



**Figure 3.1: Cable Guide Attached to the Underside of the Battery [2]**

In order to remove the ambiguity involved with identifying duplicate geometry from the current definition and also to make the definition objective for the purposes of automation, the following definition is proposed:

*Geometries lying equal to or within a threshold distance (user defined) with the surface outward normals opposed to each other within a threshold tolerance (user defined) and the percentage of similarity between the two geometries is equal to or within a threshold value (user defined).*

In this definition, there are three user defined variables that determine if the geometries are duplicate. The ambiguity involved in the earlier definition is removed by the use of these user defined variables that are quantitative in nature. Table 3-1 provides a comparison of subjective questions in the earlier definitions to the user defined variables in the new definition. There is also a threshold tolerance for the surface outward normal that is not shown in Table 3-1. This parameter is used to ensure that the profiles of two geometries are opposed to each other, which is discussed with an example in the next section.

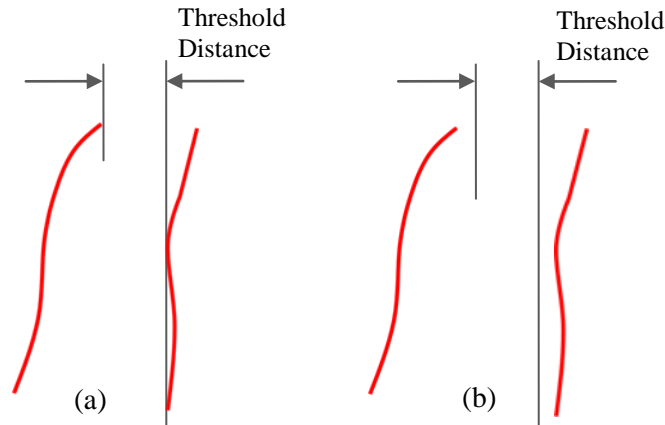
**Table 3-1: Subjectivity in the old definition addressed in revised definition**

<b>Questions in original definition</b>	<b>Addressed in revised definition</b>
Do the two geometries lie close to each other? What distance between the geometries can be regarded as close?	Threshold distance
Are the two geometries similar? If similar, what is the amount of similarity required?	Percentage value of similarity

The revised definition offers three conditions that need to be satisfied for the geometries to be evaluated as duplicate. The three conditions are the threshold distance condition, the orientation condition, and the percentage similarity condition. The next section presents the discussion on the three duplicate geometry conditions.

### 3.1.1 Threshold Distance

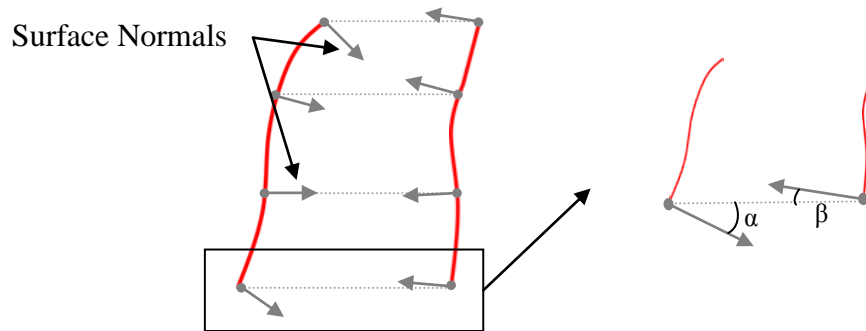
Threshold distance is the first condition in the definition of duplicate geometry. As per the definition, only those geometries that are lying within or equal to the threshold distance should be considered for duplicate geometry analysis. This condition is derived from the original definition of duplicate geometry from lazy parts mass reduction method that requires geometries to be in close proximity. By defining a threshold distance, the ambiguity involved with what distance can be considered close is removed. The example in Figure 3.2 shows two instances of same curve pairs but with different distances between them. In Figure 3.2 (a), the curves are considered for duplicate geometry analysis as the distance between them is equal to the threshold distance. However, in the Figure 3.2 (b) the same two curves cannot be considered for duplicate geometry analysis as the distance between them is greater than the threshold distance.



**Figure 3.2: Geometries that are lying within or equal to threshold distance are considered for duplicate geometry analysis**

### 3.1.2 Orientation Angle and Tolerance

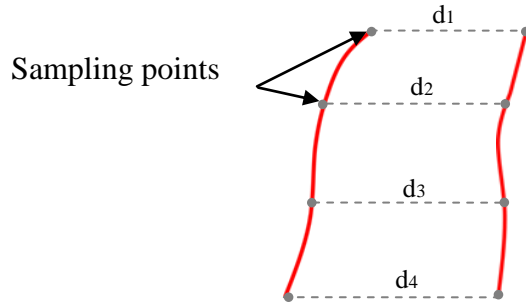
The orientation angle and tolerance is a user-defined input value for the algorithm that determines the angle between the duplicate geometries. From the definition of the duplicate geometries, it is required for the duplicate geometries to satisfy the angle condition. Typically in the assemblies the angle between the geometries is not always a single value, especially in the case of freeform and cylindrical surfaces. Moreover, the intent of identifying duplicate geometry is more of satisficing problem than an optimization problem [67]. Because of this reason a tolerance is used to compensate the variation of the angle along the surface. For the example shown in Figure 3.3, the angle between the two opposing topologically correct normal need to be within  $180^\circ \pm a$  tolerance band. Here the angle  $(\alpha - \beta)$  needs to be within the tolerance. If the angle  $\alpha$  is equal to the angle  $\beta$ , then the angle would be  $180^\circ$ . Therefore the difference between angle  $\alpha$  and  $\beta$  should be less than the orientation tolerance if the two geometries need to be considered for the duplicate geometry analysis.



**Figure 3.3: The angle between the outward normals from opposing geometries need to be within the threshold tolerance**

### 3.1.3 Percentage Similarity

Percentage similarity is the third, and final, condition in the revised definition of duplicate geometry. The percentage similarity is a user defined parameter used to address the ambiguity involved with the amount of similarity in the original definition. The original definition stated that the two geometries need to be similar in order to be considered duplicate, but did not mention the amount of similarity that was required. The revised definition provides control to the user to determine how much of similarity is required for the intended application. The similarity between the two geometries, upon satisfying the first two conditions, is calculated by measuring distance between sampling points on the two surfaces. The distances  $d_1$ ,  $d_2$ ,  $d_3$ , and  $d_4$  in the Figure 3.4 show the distance measurements between the corresponding sampling points. The two geometries are considered duplicate if the number of measurements between the sampling points from the two geometries meets the user defined percentage similarity value. In this example, if  $d_1$ ,  $d_2$ , and  $d_3$  were all equal to each other and the percentage of similarity defined was 75% or above, then the two geometries are duplicate. However, in the actual assembly it may not always be feasible to have all measurements equal to one another based on the number of sampling points used. For this reason, bounds are considered instead of a single value. That is if  $d_1$ ,  $d_2$ , and  $d_3$  are all equal to each other within a certain tolerance then the two geometries are duplicate. The bounds can be adjusted by the user based on the intended application.



**Figure 3.4: The distance measurements between the sampling points**

### 3.2 Thesis Objective

The identification of duplicate geometries in the lazy parts mass reduction method involves the tedious process of manually evaluating the CAD assembly for selecting duplicate geometries. This manual identification consumes considerable time and allows for the possibility for human subjectivity and error.

In addition to the above problem, the connectivity based assembly time estimation method [18,20] does not have an automated means for the extraction of the assembly relations from the CAD assembly absent of predefined assembly mates extraction [9]. The present manual construction of the assembly relations presents the same problem such as more time consumption and possibility for human error.

The objective of the research in this thesis, therefore, is to develop a feature recognition algorithm that can support both the automated identification of duplicate geometries for lazy parts mass reduction method and the automated extraction of assembly relations for the connectivity based assembly time estimation method from CAD assembly files.

### 3.3 Establishing Requirements

Establishing requirements is part of the software development process that helps in identifying the user, system, and functional requirements prior to software design and implementation [68,69]. In the software industry, there is no common definition for requirements. According to [69], requirement elicitation is a science of completely describing the behavior of software that aids in software development. Another definition for requirements according to [70] states that requirements are the condition needed by a user to achieve an objective. Although there is no common agreement on the definition for requirements, there is a common agreement about the need to document the requirements [68,69,71]. The system and user requirements for the research in this thesis are as follows:

- *The system allows users to define the threshold distance between duplicate geometries:* The user gets to decide the proximity between geometries based on the application and experience for the duplicate geometry analysis. The proximity between duplicate geometries could be different for different mechanical systems.
- *The system allows users to define the tolerance for surface outward normal:* The user can set the orientation that is required between the geometries with certain tolerance value for the duplicate geometry analysis. Depending on the geometric type of parts in the assembly, the user can choose the angle that is appropriate for the given assembly.

- *The system allows users to define the percentage of similarity:* The degree of similarity that is desired between geometries is decided by the user. This parameter indicates the extent of similarity between the geometries being compared. For example, the percentage of similarity value of 100% would mean identical geometries and the value zero would mean completely dissimilar.
- *The system allows users to adjust the bounds for the distance measurements between the sampling points:* The similarity between geometries is calculated by measuring the distance between sampling points on the two geometries. This list of distances between sampling points are analyzed to check if they are equal to each other within a certain tolerance. The tolerance value can be varied depending on the application and is decided by the user.
- *The system needs to highlight instances of duplicate geometry:* The system need to display the result of the duplicate geometry analysis to the user, so that the user is able to visualize the instances of duplicate geometries in the assembly. The highlighted geometries in the assembly would inform the user about regions where lazy parts mass reduction method could be applied.
- *The system needs to work with different geometric types:* The parts in the assembly may be composed of different geometric types. The examples of some of the geometric types are planar, cylindrical, spherical, conical,



freeform, and toroidal. The algorithm needs to function with such geometric types.

- *The system offers extensibility to extract assembly relations with weight:*

The algorithm need to extract the weighted bipartite graph of assembly relations to support the part connectivity based assembly time estimation method.

- *The system supports the assembly models from SolidWorks (licensed CAD software in the university) for design analysis:* The SolidWorks is the licensed CAD software in Clemson University that provides easy-to-use GUI for creating parts and assemblies.

- *The users are able to access the duplicate geometry program from within the SolidWorks software:* Presently, the duplicate geometries and part connectivity information are manually evaluated by loading the CAD assemblies in SolidWorks. For the automation of duplicate geometry identification and extraction of assembly relations, the system need to provide access to duplicate geometry program upon opening the CAD assembly file.

- *The users are able to start the duplicate geometry analysis by the click of a mouse button:* The system need to reduce the time required for the user to start the duplicate geometry analysis.

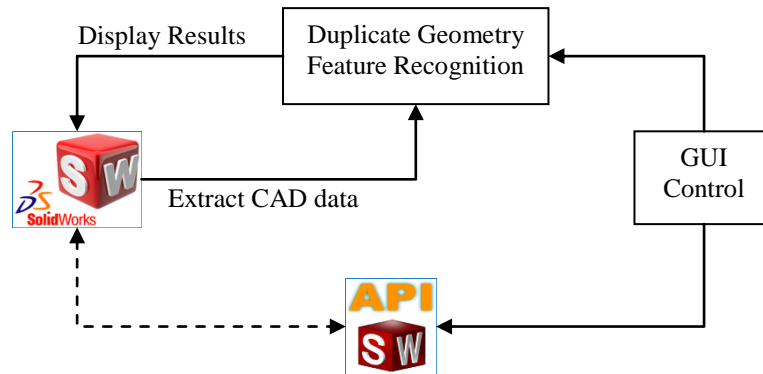
To summarize, there is no consensus on the definition of feature for feature recognition purposes and the definition of feature depends upon the application of feature recognition algorithm. The definition of feature discussed in this chapter is in the context

of duplicate geometry for lazy parts mass reduction method. The revised definition of duplicate geometry presented in this chapter removes the ambiguity involved with the original definition. Threshold distance, orientation tolerance, and the percentage similarity are the three parameters that determine the presence of duplicate geometry in a CAD assembly. Lastly, the research objective and algorithm requirements were discussed. In the next chapter software design and concepts will be reviewed.

## CHAPTER FOUR: DESIGN AND IMPLEMENTATION

### 4.1 System Architecture

The system architecture for the duplicate geometry feature recognition system is shown in Figure 4.1. This architecture supports the system requirements 5, 7, and 8 and the user requirements 9 and 10. The duplicate geometry feature recognition system is integrated into SolidWorks using 2010 SolidWorks API (Application Protocol Interface) Software Development Kit (SDK). Visual Studio C++ Professional 2008 was used for programming with the API (Application Programming Interface) and to register the DLL (Dynamic Link Library) as an add-in in the SolidWorks software. In this manner, the users will be able to access duplicate geometry algorithm from within SolidWorks software upon opening the assembly file. The users also have the advantage of reviewing the results of duplicate geometry analysis in SolidWorks GUI and focus on redesign efforts.



**Figure 4.1: System Architecture**

The feature recognition system developed inside SolidWorks GUI (Graphical User Interface) help users to start the analysis by the click of a mouse button (requirement 9 and 10, see Figure 4.2). The topology and geometric data of the CAD assembly in SolidWorks is accessed by the duplicate geometry algorithm through the API function calls. The duplicate geometry feature recognition analysis is performed in the background and the result of the analysis is displayed back in the SolidWorks software using function calls from the SolidWorks API (requirement 5, 7 and 8).



**Figure 4.2: SolidWorks GUI showing Find Duplicate Geometries button built on the panel and drop down menu**

#### 4.1.1 SolidWorks Software

SolidWorks is a commercial CAD software package used for creating parametric solid models and the production drawings. SolidWorks was selected as the CAD software for this research for the following reasons:

- Licensed CAD software at Clemson University
- Offers API SDK to build add-ins for customization
- Elaborate documentation on API functions with examples and help forum
- Supports multiple programming languages (VBA, VB.NET, C#, and C++)

- Offers easy-to-use GUI
- Provides option to build solid models of different geometric types

The version of SolidWorks used for this research was Education Edition 2010 x64. In addition to the above mentioned reasons, SolidWorks offers capability to model parts and assemblies, import assemblies from the library and online resources, conduct design analysis, and review the results.

#### 4.1.2 Application Programming Interface (API)

API is an interface that allows software developers to interact with the application software. APIs consist of function calls for the exchange of data between software. For this research, SolidWorks API is used to build a tool inside the SolidWorks CAD software for initiating the duplicate geometry analysis. The function calls from the API is used to access the data structures and its contents from SolidWorks. The version of API used in this research is SolidWorks 2010 API SDK (Software Development Kit) service pack 4.0 for Microsoft's Windows Vista 64-bit machine. The documentation for SolidWorks 2010 API SDK can be found at [72]. The API supports five languages: VBA, VB.NET, Visual C#, Visual C++ 6.0, and Visual C++/CLI [72].

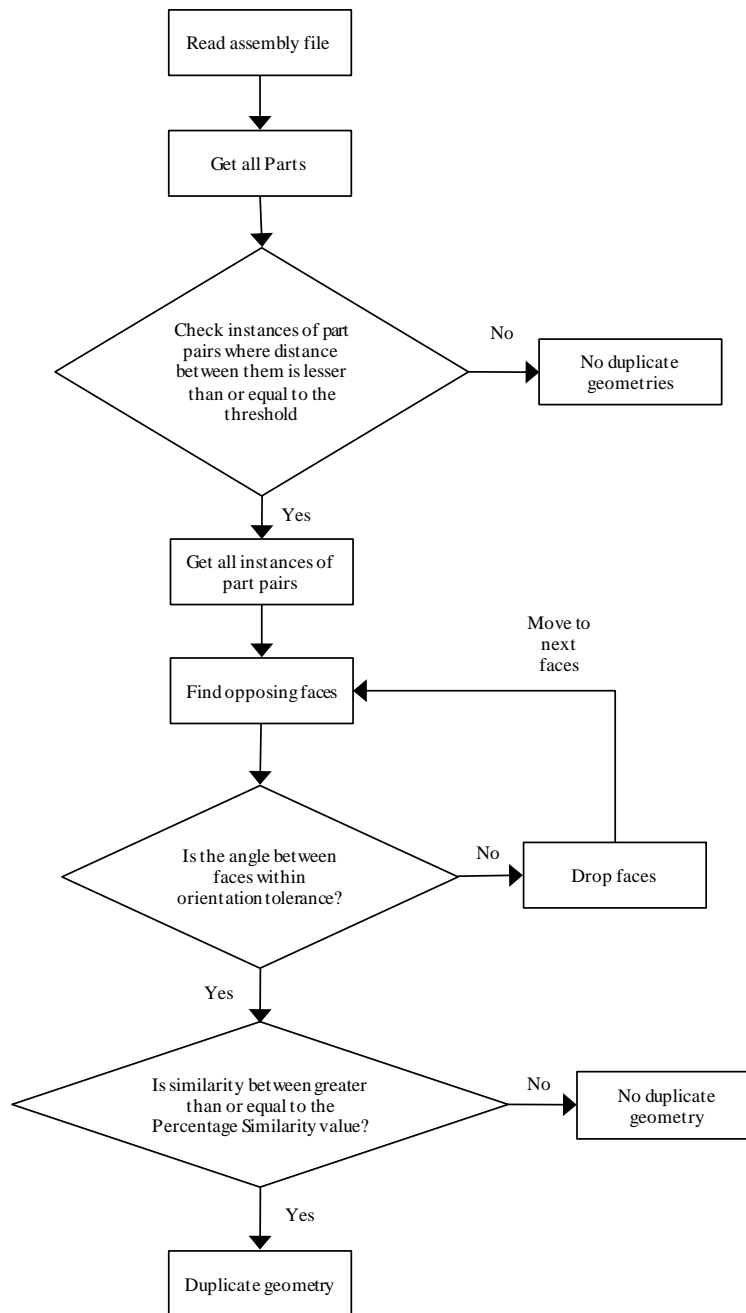
In this research, C++ was used as the preferred programming language to generate duplicate geometry feature recognition COM add-in (Component Object Model) in the SolidWorks software. COM add-in is a DLL that is registered in the SolidWorks software using the SolidWorks API. The C++ programming language provides for easy implementation of the COM objects and supports the Microsoft data structures that are

used as input and output variables in the SolidWorks API functions. Other benefits of C++ programming language are its extensibility, code reusability, and the modularity.

#### 4.2 Duplicate Geometry Recognition Approach

The general approach to find duplicate geometries is shown in Figure 4.3. This high level description of the duplicate geometry algorithm consists of nine steps. The first step is to read the CAD assembly file from SolidWorks software. The second step is to extract all visible parts from the assembly. The third step involves checking for threshold distance condition, where distance between parts is measured and compared with the user-defined threshold value. If the distance is less than or equal to the threshold value, then the instances of part pairs are stored in a list for further analysis. On the other hand, if the distance between parts is greater than the threshold value then such instances are dropped and the algorithm moves to the next parts. The first three steps are used to filter only those parts that lie in close proximity to each other within or equal to the threshold value that is defined in Requirement 1. In the fourth step, all the faces are extracted for parts that satisfy the threshold distance condition. The fifth step compares each face from one part with all the faces of the other part for orientation condition (Requirement 2) and storing the faces that satisfy this condition in a list. The orientation between the two faces is calculated by measuring the angle between the surface outward normals from the two faces as is explained in detail in the next section. The fifth step is a first pass check to ensure that only the required geometries are carried forward for further analysis. In the sixth step, sampling points are generated on the selected face pairs for the purpose of

distance measurements. Sampling points are necessary for the similarity analysis between the two faces. The seventh step is to measure the distance between sampling points from the two faces (Requirements 3 and 4). These measurements are used for the determination of percentage similarity between the two faces. In the eighth step, the total number of distance measurements that lie within a certain user-defined bounds, explained in detail in the next section, are compared with the user-defined Percentage Similarity value. The eighth step is used to differentiate duplicate geometries from the non-duplicate geometries for all faces that satisfy the distance and orientation condition. In the final step, the faces that satisfy the Percentage Similarity condition are highlighted. The description of the algorithm presented in this section is only a high level account of the general approach used for duplicate geometry feature recognition analysis. In the next section, a detailed description about the implementation for each of the above nine steps are provided.

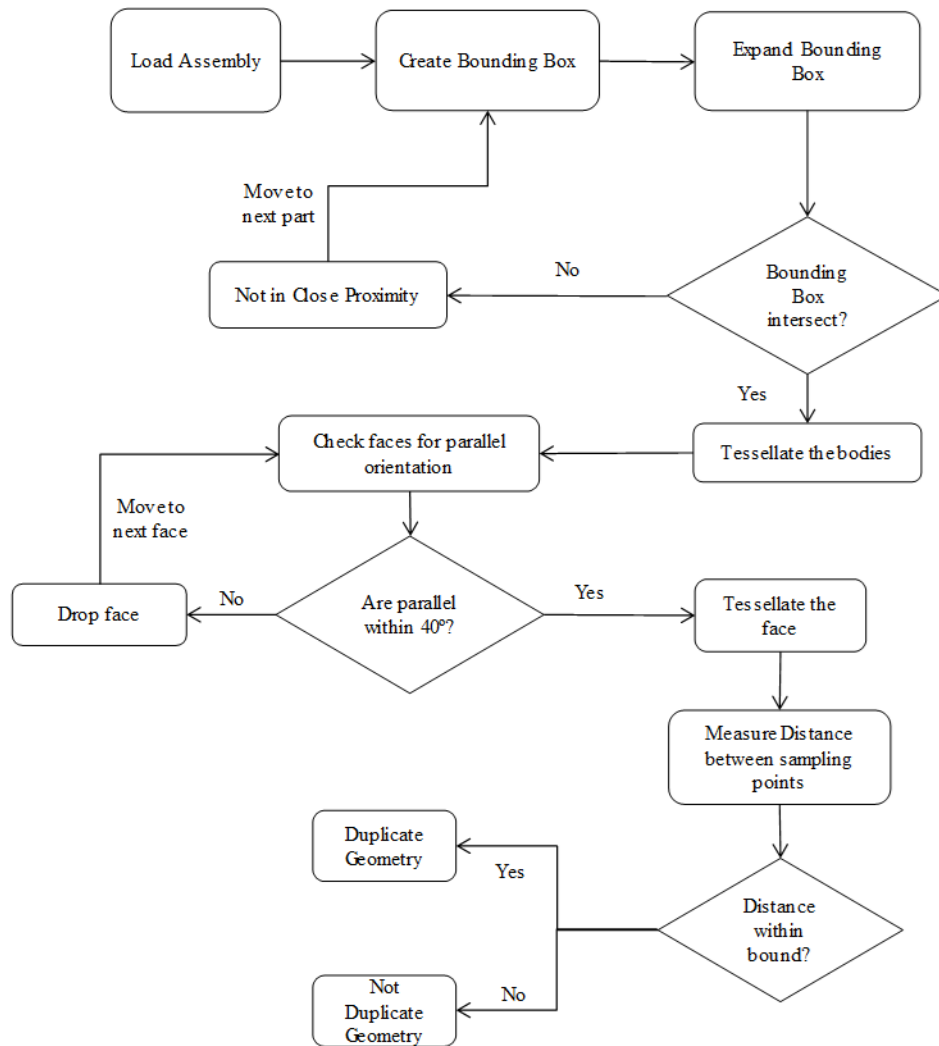


**Figure 4.3: High level description of the duplicate geometry algorithm**



### 4.3 Implementation

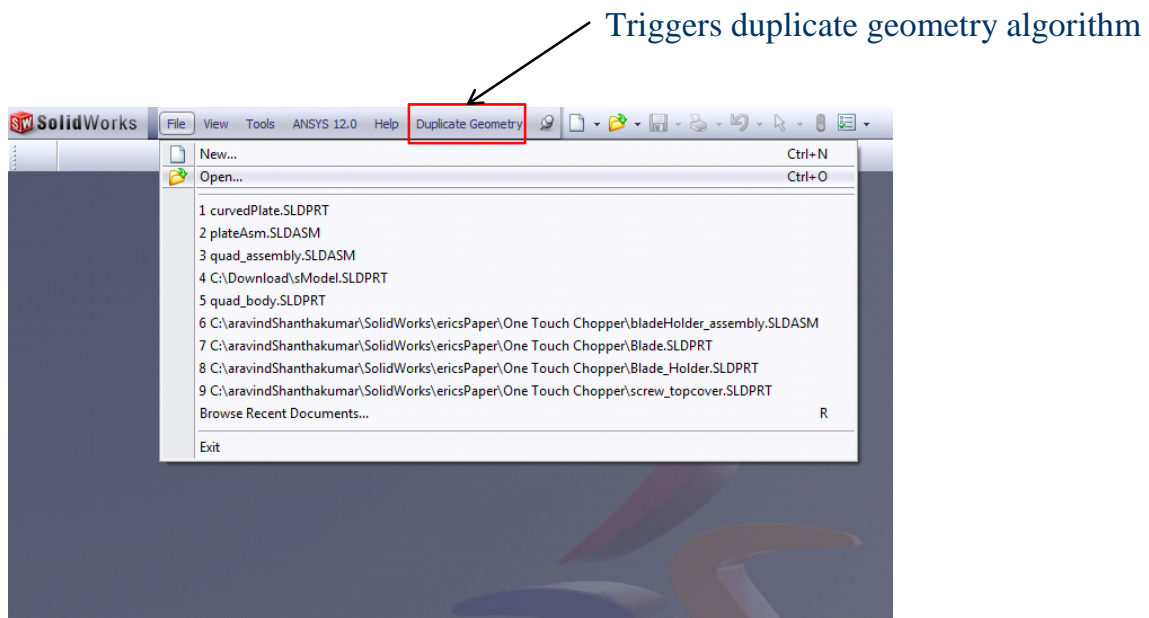
The implementation details of the general approach discussed above is presented in this section. The flowchart shown in Figure 4.4, illustrates the breakdown of the general approach to show details used for the identification of duplicate geometry. The detailed account of the stages of flowchart shown in Figure 4.4 is presented below.



**Figure 4.4: Flowchart representing duplicate geometry algorithm**

### *Step 1: Load Assembly*

The first step of the algorithm is to read the assembly file from SolidWorks. The Figure 4.5 shows the SolidWorks GUI used to load the assembly file. In SolidWorks, the extension \*.SLDASM in the file name represent the assembly file. Loading the assembly file in SolidWorks is the first step in the duplicate geometry analysis by the user. After opening the required assembly file, the duplicate geometry algorithm is activated by pressing the Duplicate Geometry button that is shown in Figure 4.5. Once the file is loaded, the algorithm reads the active assembly document.

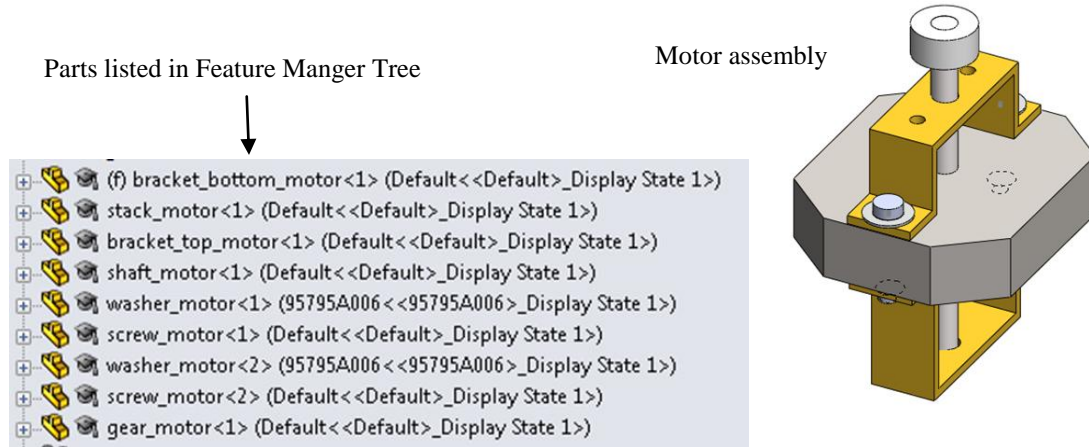


**Figure 4.5: SolidWorks GUI with the duplicate geometry button**

### *Step 2: Get Part Count*

The second step is to extract all the visible parts from the SolidWorks assembly. For example, the motor assembly shown in Figure 4.6 consists of nine visible parts that are also displayed in the SolidWorks feature manager tree. The feature manager tree is

the area in the SolidWorks GUI that shows the parametric CAD data of the active document. The data displayed includes information regarding parts name, construction history, assembly mates, and display properties to name a few.



**Figure 4.6: List of visible parts in the SolidWorks feature manager tree for the motor assembly shown in the right**

In the presence of sub-assemblies, the parts inside the sub-assemblies are considered towards the total part count. For example, if an assembly contains ‘n’ parts and a sub-assembly, then the total part count is equal to

$$PartCount = Part_1 + Part_2 + \dots + Part_n + Partsfromsubassembly_1$$

If the number of parts inside the sub-assembly is ‘m’, then

$$PartCount = n + m$$

In this research, sub-assemblies are treated as the assembly of parts and not as single units. However, there could be certain applications where sub-assemblies may be required to be treated as single units. An example for this case is the mechanical systems in the assembly from suppliers such as turbo charger assembly from a supplier.

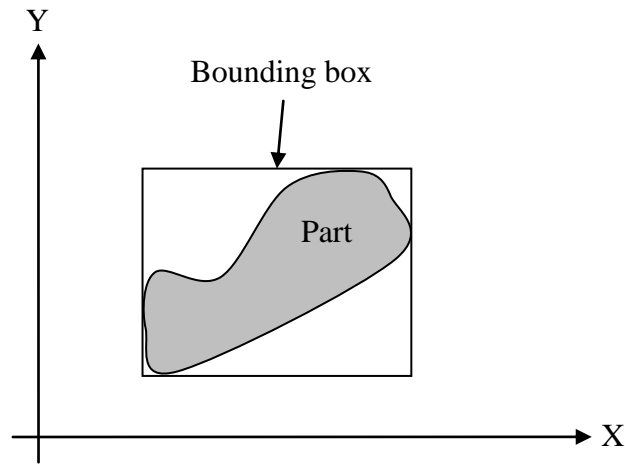
Therefore, if the requirement demands the sub-assemblies to be treated as parts, then the code used to extract parts from the assembly offers the option to treat sub-assemblies as single units.

*Step 3: Check for threshold distance condition*

The threshold distance between the two geometries is a user-defined value. This is the first necessary condition to be satisfied by the geometries before being analyzed for the orientation. Checking parts for the threshold distance condition is the third step of the algorithm. The details of this step are discussed in the following sub-section: extracting bounding box, expanding bounding box, and checking for intersection.

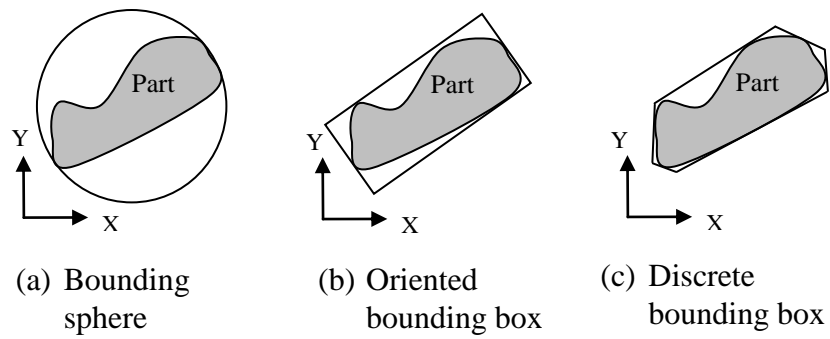
*Step 3.1: Get bounding box*

The first step in the process of determining the distance between two geometries is to retrieve the bounding box around each part in the assembly. The bounding box is a tight convex, prismatic, orthogonal, hexagon envelop around the boundary of the part (see Figure 4.7). The bounding box representation of the part is often used for intersection detection between parts due to its simple geometric representation [73]. Since, the bounding box represents the outer enclosure of the part, absence of intersection between bounding boxes ensures absence of intersection between the parts. This rule is used in this research to determine proximity between parts.



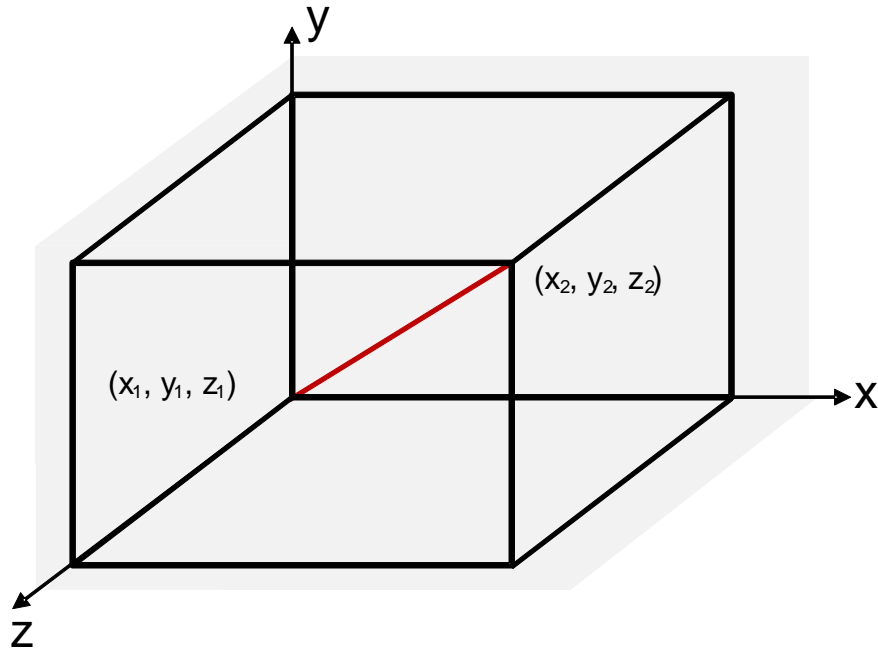
**Figure 4.7: Example of an axis aligned bounding box**

There are four different types of bounding boxes discussed in the literature. These are bounding sphere, axis aligned bounding box, oriented bounding box, and discrete bounding box [73]. The spherical bounding box forms a spherical envelope around the part (see Figure 4.8 (a)). The oriented bounding box is a rectangular bounding box whose orientation is along the axis of the part as shown in the Figure 4.8 (b). The discrete bounding box is a special envelope around the part that is non-orthogonal in nature, which is also referred to as fixed direction hull [73] as shown in Figure 4.8 (c). The bounding box used in this research is an axis aligned bounding box (see Figure 4.7). Axis aligned bounding box is rectangular in geometric shape similar to the oriented bounding box, but is aligned to the global Cartesian coordinates axes (see Figure 4.7).



**Figure 4.8: Bounding box types**

The SolidWorks API offers a function to extract axis aligned bounding box for visible parts in the assembly. The bounding box returned is the x, y, and z coordinates for the upper and lower diagonal points of the bounding box as shown in the Figure 4.9.



Bounding box co-ordinates = {x1, y1, z1, x2, y2, z2}

**Figure 4.9: Bounding box coordinates returned in SolidWorks**

*Step 3.2: Expand bounding box:*

After retrieving bounding box for all visible parts in the assembly, the second step is to expand the bounding box. The bounding box size is expanded in the three main Cartesian coordinate directions by the amount equal to half the user-defined threshold distance value. To explain this further, if the user-defined threshold distance value is 'x' unit then the bounding box around all parts are expanded by the amount equal to 'x/2' unit in the three main Cartesian coordinate directions. The pseudo-code used for expanding the bounding box is as follows:

Get the array of six bounding box coordinates (see Figure 4.9)

```
for inti = 0 to 5; i++
```

```
    if i < 3
```

```
        value[i] = value[i] - (half threshold distance value)
```

```
    end if
```

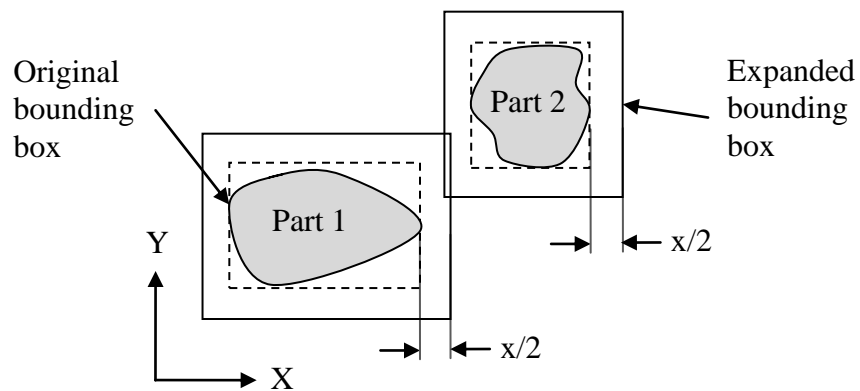
```
    else
```

```
        value[i] = value[i] + (half threshold distance value)
```

```
    end else
```

```
end for
```

As shown in the Figure 4.10, expanded bounding box are used to check for intersection between parts. Intersection between the expanded bounding boxes ensures that the distance between the original bounding boxes meets the threshold distance condition.



**Figure 4.10: Bounding box expanded to check for threshold distance condition**



It is to be noted here that, this strategy of using bounding box for measuring the distance between parts does not provide the accurate distance between the two geometries in consideration. However, it does provide an approximate and quick check to filter only those parts that satisfy the threshold distance value. The actual minimum distance between the two parts may be greater than the threshold distance value; however, no parts separated by the distance lesser than the threshold value will be missed.

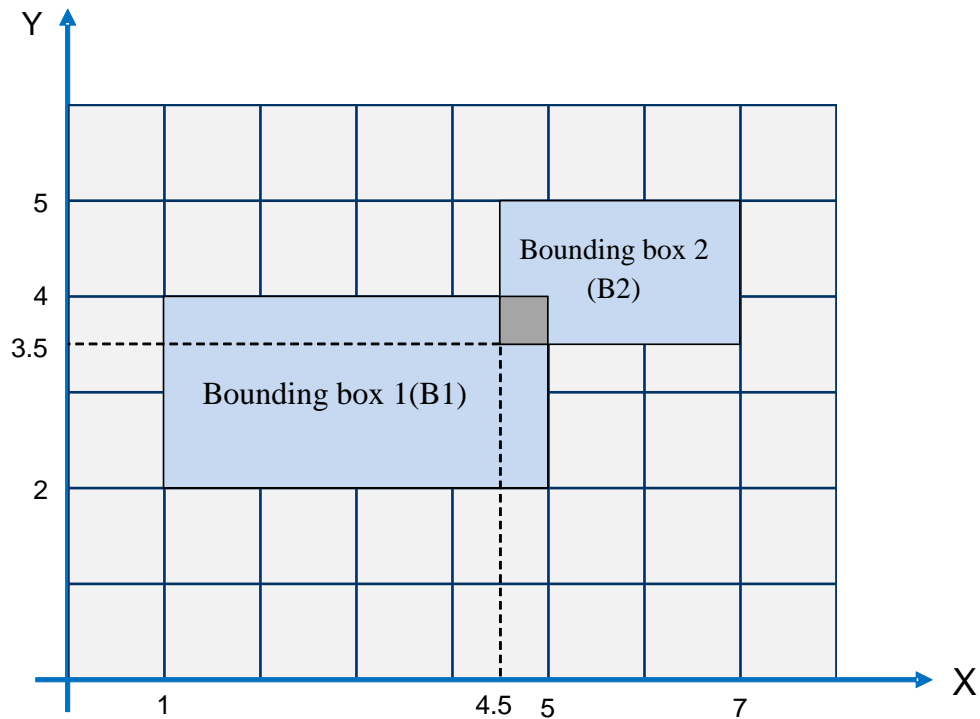
Because the bounding box used is axis aligned, the check for threshold distance condition made is in the principle axes direction in the Cartesian coordinate system. Therefore, if the distance between parts in all the x-, y-, and z-direction is less than or equal to the threshold value, then the parts will be considered as meeting the threshold distance condition. The next step explains the dynamics of intersection calculation.

*Step 3.3: Find intersection between bounding box:*

In the third step, expanded bounding box are used to check for intersection. The pseudo code used for intersection calculation is as follows:

```
for i = 0 to partCount-1; i++  
    itr_1 = part[i];  
    for j = 0 to part count; j++  
        itr_2 = part[j+1];  
        checkForIntersection (itr_1, itr_2)  
        if intersection == true  
            push (itr_1, itr_2) into a container as pair  
        end if  
    end for  
end for
```

If the assembly contains 'n' parts, then each part is checked for interference with (n-1) parts. The Big O complexity for this algorithm is  $O(N^2)$ , where N is the number of parts as there is a for-loop nested within another for-loop.



**Figure 4.11: Intersection calculation using bounding box**

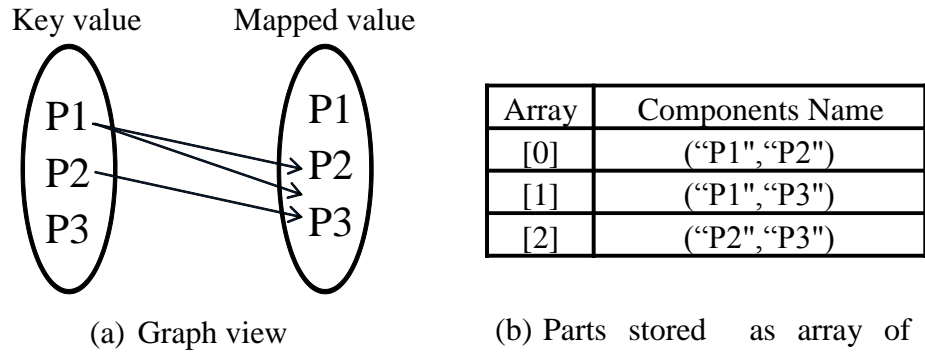
When the two parts are considered for the intersection calculation, the expanded bounding boxes for the two parts are retrieved first. The intersection between the two expanded bounding boxes is calculated as follows (see Figure 4.11 for reference):

```

if ( B1_right > B2_left && B1_top > B2_bottom )
    return true;
end if
else
    return false;
end else

```

Once the intersection between the two bounding boxes is determined, the parts associated with the two bounding boxes are stored as pair in a container. The standard template library's (STL) multimap (multiple-key map) data structure is used to store the parts as pairs. The multimap forms a link between key values and the mapped values (see Figure 4.12) allowing for multiple mapped values to have a single key value. To explain this in context, P1 can be stored as a pair with P2 and P3. In the Figure 4.12, P1 is paired with both P2 and P3. The pair P1-P2 represents part pairs that satisfy the threshold distance condition. The pair P1-P3 represents a different part pair indicating P1 and P3 satisfy threshold distance condition. In this example, both P2 and P3 have the single key value P1.



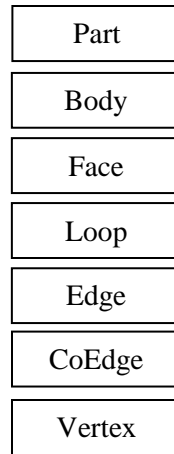
**Figure 4.12: Parts meeting the threshold distance condition stored as pairs in multimap container**

*Step 4: Iterate through part pairs and retrieve bodies and faces*

The fourth step in the algorithm is to access each pair of parts in the multimap list for the extraction of bodies and faces from the topology. Bodies and faces are the topological entities in a B-Rep data structure for a given part that are of interest in this

research. The list of topological entities for models in SolidWorks is shown in the Figure 4.13. The extraction of faces from the part pairs is necessary for the determination of orientation between faces.

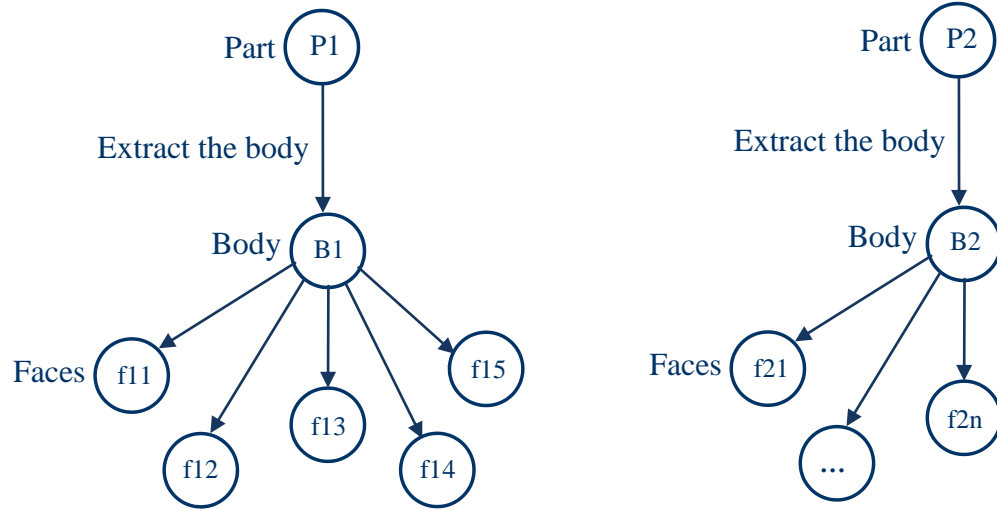
#### **Entities of Parasolid Topology**



**Figure 4.13: Topological data structure in SolidWorks**

Once the first pair of parts is accessed from the multimap list, as shown in Figure 4.12 (b), the bodies inside both parts are extracted first. In this research, only assemblies with single-bodied parts are considered for the duplicate geometry analysis and hence the bodies extracted from both parts are single objects instead of an array of bodies as observed in case of multi-bodied parts. However, the program offers extensibility to extract the array of bodies while dealing with assemblies that include multi-bodied parts. After the bodies are extracted from the two parts, all faces from the two bodies are extracted next. As shown in Figure 4.14, part P1 and P2 represent a part pair from the list in Figure 4.12 (b). B1 and B2 represent the bodies extracted from the parts P1 and P2

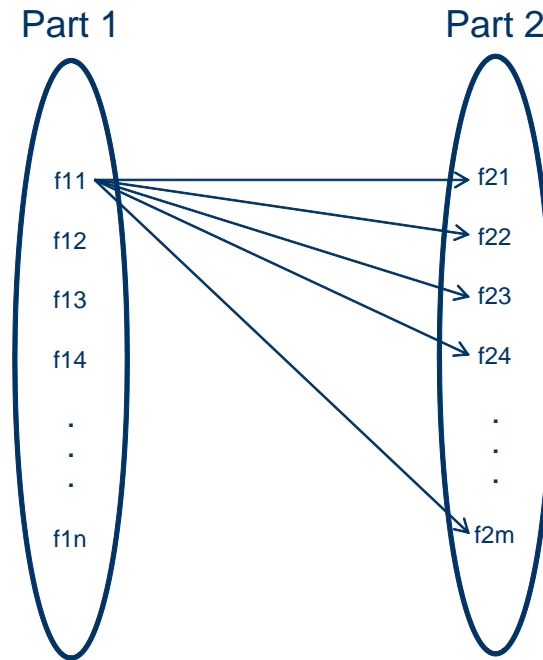
respectively. Faces f11 through f15 represent a total of five faces extracted from the body B1. Similarly, faces f21 through f2n represent a total of 'n' faces extracted from the body B2. Once all faces are extracted from both the bodies, each face from one set is compared with all faces in the other set for the orientation that is explained next.



**Figure 4.14: Faces extracted from the bodies in the part pair**

*Step 5: Check for orientation between faces*

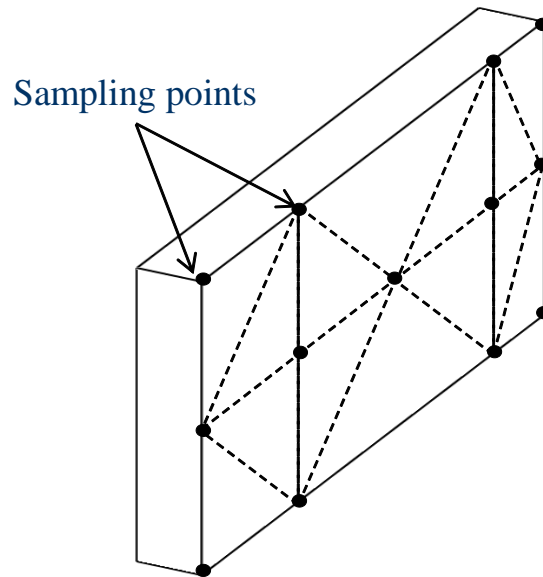
Determining the orientation between faces from the two bodies is the fifth step in the algorithm. As illustrated in the Figure 4.15, each face belonging to Part 1 is compared with all faces from Part 2 to ensure that all faces in Part 1 are compared with all faces from Part 2 for the orientation between them. Due to this strategy, the complexity of the algorithm is less than or equal to  $O(N^2)$ , where N represents the number of faces in Part 1 and Part 2. The details of the method used for the determination of orientation between the two faces are described below.



**Figure 4.15: Each face from one set is compared with all faces from the other set for orientation**

*Step 5.1: Tessellate the faces to generate sampling points*

All faces extracted from both the bodies are tessellated to generate sampling points. Tessellation is the process of representing the face in terms of triangles. For example, the planar face shown in Figure 4.16 is discretized into triangles to generate the sampling points for determining the orientation between faces.

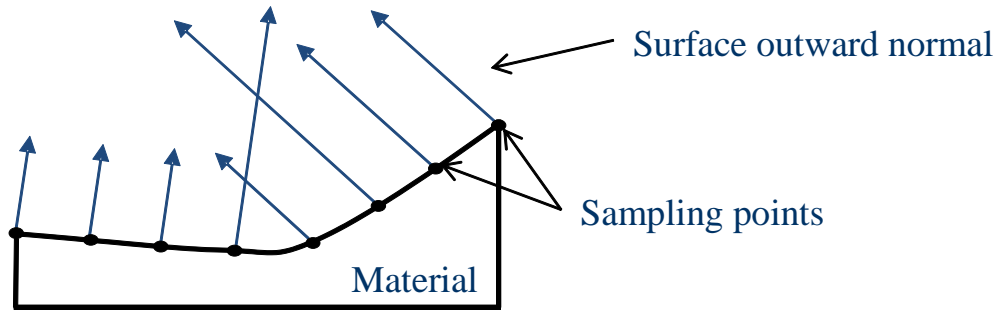


**Figure 4.16: Tessellating the face to generate sampling points**

*Step 5.2: Get surface outward normals at sampling points*

After generating sampling points on the faces, the surface outward normal for the face at each of the sampling points are retrieved. The direction of the surface outward normal is always away from the material and is orthogonal to the face at a given sampling point (see Figure 4.17). The surface outward normals retrieved are the unit normal vectors indicating the direction of the face at a given sampling point.





**Figure 4.17: Surface outward normal for a face at different sampling points**

*Step 5.3: Measure the angle between normal vectors*

In this step, all the unit normal vectors from the first face of Part 1 and the first face of Part 2 are sorted into two separate lists. First, the first unit normal vector is selected from list one and the dot product between this and all the unit normal vectors from the list two is calculated. For the example shown in Figure 4.18, 'n1' represent the set of unit normal vectors at all sampling points on the Face 1 of Part 1. Similarly, 'n2' represents the set of unit normal vectors at all sampling points on the Face 1 of Part 2. The dot product is calculated between the first unit normal in the list 'n1' and the all the unit normals in the list 'n2'. The formula used to calculate the dot product between two vectors is:

$$u \cdot v = |u||v| \cos \theta$$

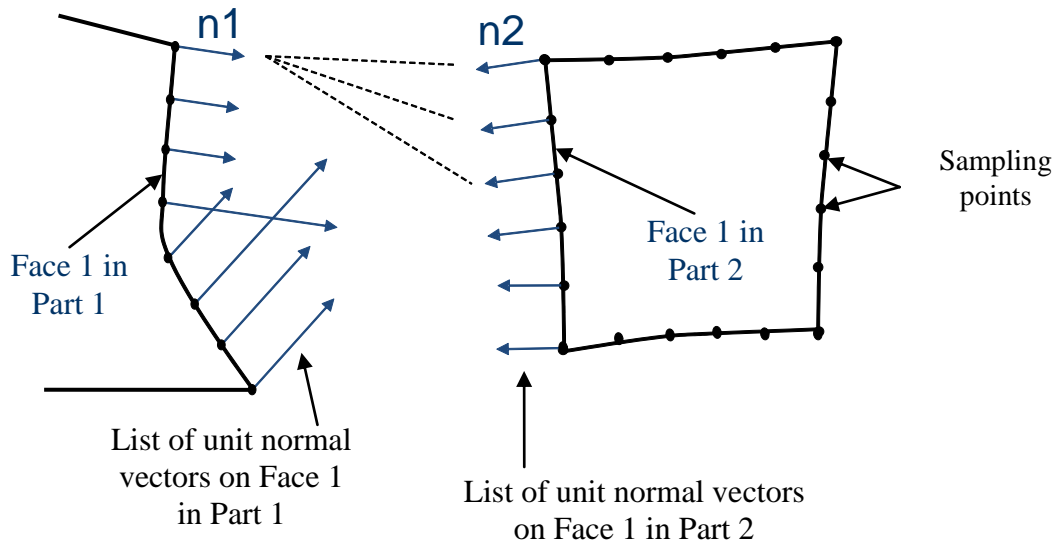
where,  $u = \text{unitnormalonFace 1}$

$v = \text{unitnormalonFace 2}$

Now, using the dot product and the magnitude of the two vectors the angle between the two vectors is determined using,

$$\theta = \text{acos}\left(\frac{u \cdot v}{|u||v|}\right) * \frac{180}{\pi}$$

If the angle  $\theta$  between the two vectors is within the user-defined angle and the tolerance then a counter is incremented by one unit. Next, the iteration is repeated with the second unit normal in the list 'n1' and all the unit normals in the list 'n2' until all the unit normal in the list 'n1' are exhausted. If at least three unit normals from 'n1' forms the angle with any unit normals from 'n2' that is within the user-defined bounds, then the two faces are considered to be meeting the orientation condition.



**Figure 4.18: Unit normals retrieved at sampling points for the two faces**

The reason for considering a minimum of three unit normals from ‘n1’ meeting the orientation condition is because; three is the minimum number of vectors required bound a facet. Therefore, the three vectors from the list ‘n1’ meeting the orientation condition provides an indication that at least one facet on the face from which ‘n1’ is derived forms a parallel orientation with the other face from which ‘n2’ is derived within the user-defined tolerance. Based on the angles calculated using the unit vectors, the pair of faces from the two parts that have parallel orientation within the user defined tolerance are stored in a container as pairs for further analysis. To explain further, let  $\{f_{11}, f_{12} \dots f_{1n}\}$  represent a list of faces in Part 1 and  $\{f_{21}, f_{22} \dots f_{2n}\}$  represent a list of faces in Part 2, then the pair of faces that have orientation within the user defined angle and tolerance is stored as pairs as shown in Table 4-1. This table indicates an example where the face pairs f11-f23, f15-f24, and f16-f25 have orientation within the user-defined value and can be considered for further analysis to determine percentage similarity.

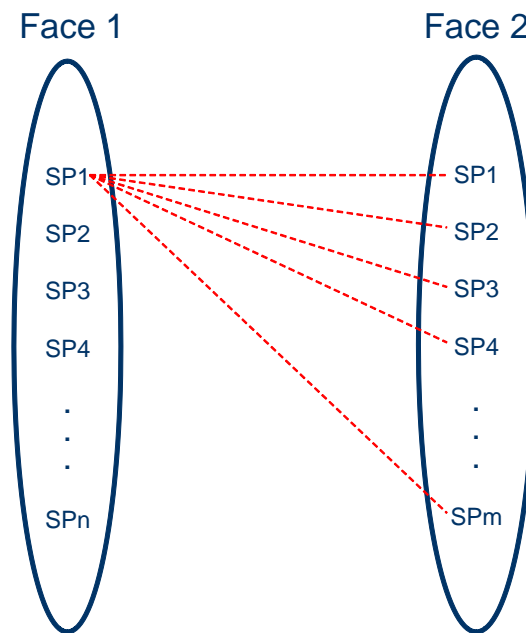
**Table 4-1: Example list showing faces stored as pairs that have orientation within the user-defined angle and tolerance**

Pair of faces that have parallel orientation
f11 – f23
f15 – f24
f16 – f25

*Step 6: Check for percentage similarity between the stored list of face pairs*

The percentage similarity between the two faces is evaluated by measuring the distance between sampling points from both faces. The face pairs are re-tessellated with

shorter edge length for the facet in order to generate more sampling points. The length of the facet edge is presently maintained at a length equal to the shortest edge in the assembly. The generated sampling points for both faces are stored in two separate lists. For example, if Face 1 has 'n' newly generated sampling points and Face 2 has 'm' sampling points as shown in Figure 4.19, then 'n' sampling points are stored in a list associated with Face 1 and 'm' sampling points are stored in a different list associated with the Face 2. After producing two lists of sampling points (SP), the steps described hereafter are used to determine the percentage similarity.



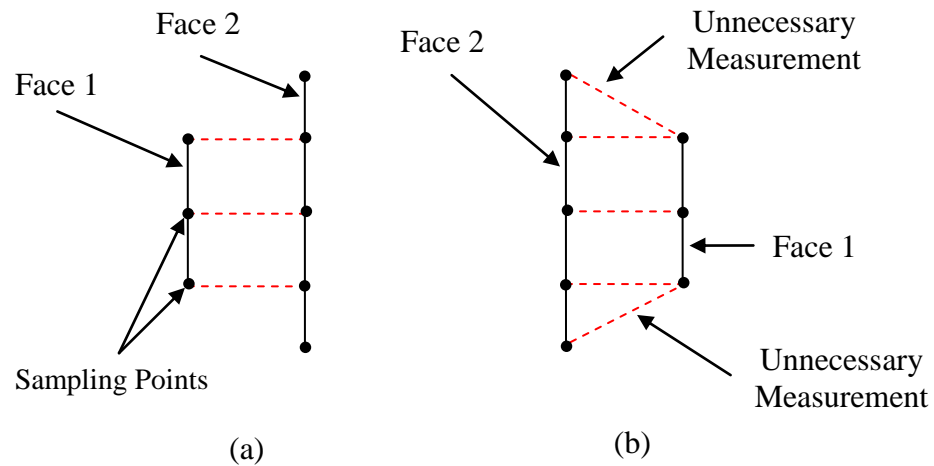
**Figure 4.19: Measurement of distance between sampling points**

*Step 6.1: Find the list with the lesser sampling points*

The two lists of sampling points are compared for the size. The smaller of the two lists (indicating fewer number of sampling points) is selected to be the first list of

sampling points that will be considered as start points for the distance calculation. The other list then forms the list of sampling points that will be considered as end points for the distance measurements. The reason for choosing the smaller list of sampling points as start points is because, in the case of two geometries of different sizes the larger geometry (with more sampling points) would allow unnecessary measurements between the sampling points of both faces.

To explain this further, consider the example shown in Figure 4.20 (a) where three sampling points on Face 1 are used as start points and the Face 2 with five sampling points are used as end points for the measurements. In this case, the number of measurements between Face 1 and Face 2 is only three and this number is used later to calculate the percentage similarity between the two faces. Now, for the case shown in Figure 4.20 (b) the sampling points on Face 2 becomes the start points and the three sampling points on Face 1 becomes the end points. For this case, the number of measurements between the two faces is five. Distances measured from the five start points to the smaller geometry with three end points would add two extra measurements that are unnecessary.



**Figure 4.20: Face consisting of lesser number of sampling points is used to start the measurement**

*Step 6.2: Find shortest distances between sampling points from both lists*

Before calculating the distance between sampling points from both lists, the X, Y, and Z coordinates of sampling points need to be transformed into the assembly coordinates. In SolidWorks, the coordinates for the sampling points generated on the part face returns the X, Y, and Z coordinates from the part file. Due to the fact that the part coordinate system is different from the assembly coordinate system, the sampling points generated on the part face need to be converted to assembly coordinate. For converting the X, Y, and Z coordinates of sampling points from the part into assembly coordinates, each sampling point in the list is multiplied by the assembly transformation matrix as shown below:

$$\text{Coordinates of a sampling point from part} = SP = \begin{bmatrix} x_{old} \\ y_{old} \\ z_{old} \\ 1 \end{bmatrix}$$

$$\text{Assembly transformation matrix} = M = \begin{bmatrix} 1 & 0 & 0 & x_a \\ 0 & 1 & 0 & y_a \\ 0 & 0 & 1 & z_a \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{New coordinates of the sampling point} = M \times SP$$

$$\begin{bmatrix} x_{new} \\ y_{new} \\ z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_a \\ 0 & 1 & 0 & y_a \\ 0 & 0 & 1 & z_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{old} \\ y_{old} \\ z_{old} \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_a + x_{old} \\ y_a + y_{old} \\ z_a + z_{old} \\ 1 \end{bmatrix}$$

After transforming all sampling points in both lists to the assembly coordinates, the distances between the first sampling point from Face 1 (see Figure 4.21) and all the other sampling points in Face 2 (see Figure 4.21) are calculated using the distance formula as shown below:

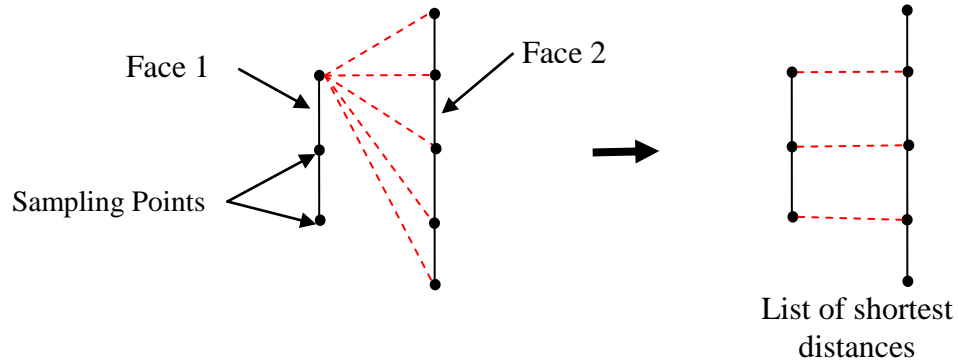
$$\text{Coordinates of first point} = (x_1, y_1, z_1)$$

$$\text{Coordinates of second point} = (x_2, y_2, z_2)$$

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Among distances calculated between the first sampling point from Face 1 and all the other sampling points in Face 2, the shortest distance is selected and stored in a list.

This process is repeated until all the sampling points from Face 1 are exhausted. The outcome of this step will be a list of shortest distances from the sampling points in Face 1 to the sampling points in Face 2 (see Figure 4.21).



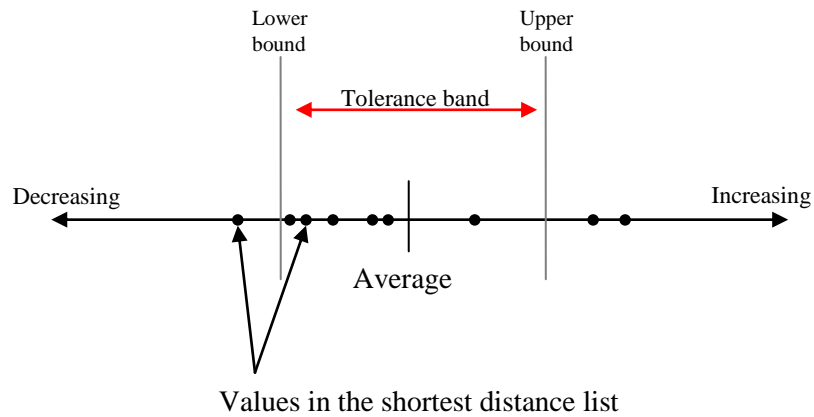
**Figure 4.21: Distances measured from one sampling point on Face 1 to all sampling points on Face 2**

*Step 6.3: Find the average of the entities in the shortest distance list and check for percentage similarity*

This step starts by calculating the average of all the distances in the shortest distance list. User defined upper and lower bounds (Requirement 4) are added to the newly calculated average. The upper and lower bounds represent a tolerance value for the average for comparing the number of distances in the shortest distance list that lie within this bound. The Figure 4.22 shows the spread of distances in the shortest distance list. The points lying inside tolerance band represent distances that are within the tolerance band. The point to the left of lower bound represents distances shorter than the lower tolerance limit. The points to the right of the upper bound represent distances that are



more than the upper tolerance limit. It is the number of distances that are within the tolerance band that are considered as meeting the tolerance bound condition. The distances (or points in the figure) inside the tolerance band are interpreted as being equal to the average value within a certain degree of tolerance that is defined by the user.



**Figure 4.22: Calculating percentage similarity between two geometries**

To determine the percentage similarity between the two faces, the number of distances in the shortest distance list that falls inside the tolerance band is calculated (see Figure 4.22). This number divided by the total number of points in the shortest distance list (size of the list) gives the actual similarity ratio. For the example shown in Figure 4.22, the actual similarity ratio is:

$$\text{Number of distances within the tolerance band} = 6$$

$$\text{Total number of distances in the list} = 9$$

$$\text{Actual similarity ratio} = \frac{6}{9} = 0.66$$

The two faces are evaluated to be duplicate geometries if the value of the actual similarity ratio is less than or equal to the user-defined percentage similarity value. The user-defined percentage similarity value is divided by 100, which is then compared with the actual similarity ratio. The need for geometries to satisfy the percentage similarity condition is the third condition of the duplicate geometry definition and the third requirement for the algorithm listed in Section 3.3.

*Step 7: Highlight duplicate geometries*

Highlighting duplicate geometry instances in the CAD assembly is the final step of the algorithm. All the face pairs that meet the percentage similarity condition are highlighted and displayed in the SolidWorks GUI. Highlighting the duplicate geometry instances helps the user to review the results of duplicate geometry analysis on the SolidWorks GUI. For the part connectivity based assembly time estimation method, in addition to highlighting part connections, the part connectivity graph with the degree of overlap is written to a \*.csv file which is discussed in section 5.9.

To review, in this chapter the system architecture for the algorithm, the high level description of the algorithm providing a concise overview of all steps, and the details of the implementation are discussed. The algorithm checks for the threshold distance condition and the orientation condition to filter only required geometries for duplicate geometry analysis. These two conditions are the necessary conditions derived from the definition of duplicate geometry. Each selected pair of geometries is then evaluated for percentage similarity in the final step and highlighted upon satisfying the percentage similarity condition that is displayed in the SolidWorks GUI. For part connectivity based

assembly time estimation method, in addition to highlighting instances of duplicate geometry the part connectivity graph and the amount of overlap are written to a \*.csv file. In the next chapter, validation of the algorithm using test cases will be discussed.

## CHAPTER FIVE: VALIDATION

The functioning of the algorithm is evaluated against the system requirements (discussed in Chapter Three) using test cases. Different test cases are designed to study the performance of the algorithm for each of the first seven requirements. The first seven requirements refer to the system requirements necessary for duplicate geometry analysis and displaying the results. The requirements 8 - 10 that are relevant to usability are already met as discussed in System Architecture in Chapter Four. The current system supports assembly models from SolidWorks CAD software for the analysis (requirement no. 8). Additionally, the users are able to access the duplicate geometry program from inside SolidWorks by using the duplicate geometry tool built in SolidWorks (requirement no. 9 and 10).

### 5.1 Test-Cases to Check for Threshold Distance Condition

The threshold distance condition is the first necessary condition for analyzing the parts for duplicate geometry and is also the first requirement for the algorithm. Presently the algorithm takes the input from the user for the threshold distance value (requirement no. 1). The parts that are lying closer than or equal to the threshold distance are filtered by the program to check for orientation and percentage similarity. The test cases presented in this section are designed to check for the performance of the algorithm in recognizing the geometries that satisfy the threshold distance condition.

The following test cases are designed to check the program's behavior for three different types of threshold distances between the geometries. The three distances

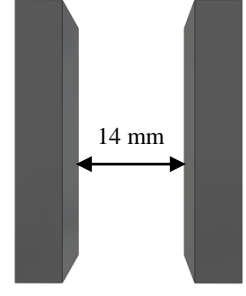
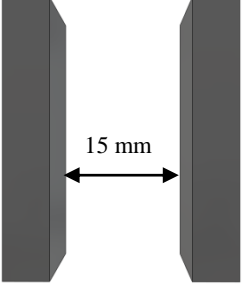
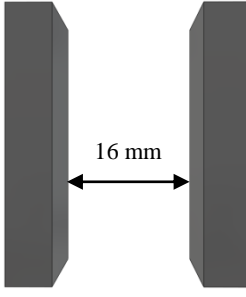
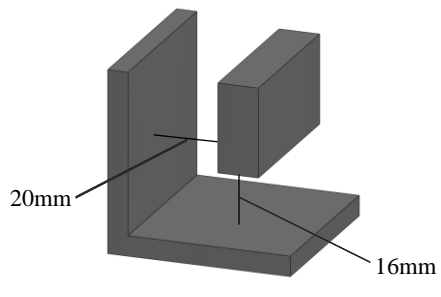
checked are the distance between parts less than the threshold distance, the distance between parts equal to the threshold distance, and the distance between parts greater than the threshold distance as these are the only three types of conditions that can be encountered by the algorithm. For this test, the user-defined threshold distance input is set at 15mm.

In the test case 1 shown in the Table 5-1, the two parts with planar faces are separated by a distance equal to 14mm that is less than the threshold distance. The bounding box algorithm that is used to test the distance between parts recognizes the pair of parts as meeting the threshold distance condition and stores the pair in a container for the orientation check.

In the test case 2 (see Table 5-1), the distance between the same part pairs are set at 15mm that is equal to the threshold distance value. The test case with the distance between parts equal to the threshold distance is used because the definition of duplicate geometry qualifies those geometries separated by the distance equal to threshold distance as meeting the threshold distance condition. For this test case, the bounding box algorithm calculates the distance between the part pairs to be equal to the threshold distance value and then stores them in a container for the orientation check.

In the test case 3 shown in Table 5-1, the distance between the part pair is increased to 16mm that is greater than the threshold distance value. The bounding box algorithm calculates the distance between the two parts to be more than the threshold distance value and discards the part pair from considering for further analysis as the part pair does not satisfy the first necessary condition.

**Table 5-1: Test cases for threshold distance**

Test Case No.	Test Case	Description
1		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be less than the threshold distance and stores the two parts in a container for orientation check.</p>
2		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be equal to the threshold distance and stores the two parts in a container for orientation check.</p>
3		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be more than the threshold distance and therefore discards this part pair from further analysis.</p>
4		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts as less than threshold distance. But the distance between the geometries from the two parts is greater than the threshold distance.</p>

Although the threshold distance condition need to be checked for the distance between geometries, the bounding box algorithm presently checks for the distance between the parts. The test case 4 shown in Table 5-1 has minimum distance between the parts set at 16mm that is greater than the threshold distance value. However, the bounding box algorithm uses the bounding envelopes of the two parts to check for the threshold distance condition. Due to the configuration, the algorithm detects the overlap between the two bounding box and treats the two parts as meeting the threshold distance condition.

The threshold distance check using bounding box is only a preliminary check intended purely to shortlist the component pairs for subsequent orientation and percentage similarity analysis. As demonstrated using test case 4, certain false positive (distance greater than threshold condition) part pairs are selected for further analysis by the algorithm. As bounding box check is only a preliminary filter, having false positives among the part pairs for orientation and percentage similarity is not going to affect the final results. The algorithm discards these false positive part pairs while performing the analysis for percentage similarity. The performance of the bounding box algorithm for threshold distance condition involving cylindrical and freeform surfaces is also tested (see Appendix A:). The observation is that the algorithm is consistent in detecting threshold distance across all geometric types and no part pairs that have the distance between them less than or equal to threshold distance is missed.

## 5.2 Test-Cases to Check for Orientation Condition

The orientation condition is the second necessary condition to be satisfied by the duplicate geometries and the second requirement in the requirement list presented in Chapter Three. The orientations between the geometries are determined by calculating the angle between outward unit vectors on the surface at different sampling points and comparing it with the user-defined angle value with a tolerance. Presently, the algorithm requires the user to input the value for the angle and tolerance.

The test cases showed in Table 5-2 for the validation of requirement no. 2 uses parts with planar faces to determine the performance of the algorithm for the angles that are within the user-defined orientation tolerance, equal to the upper or lower limit of orientation angle, and outside the user-defined orientation angle. The user-defined orientation angle and tolerance used for this test case is  $180^{\circ} \pm 10^{\circ}$ .

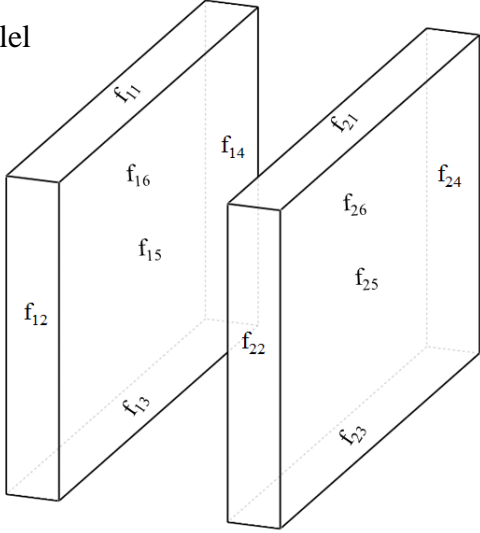
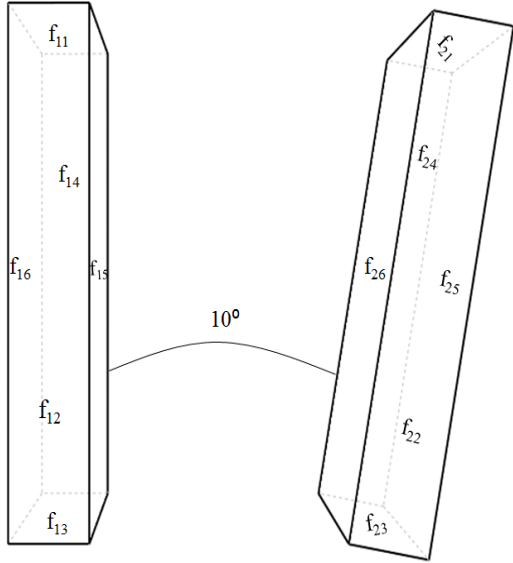
The test case 1 in Table 5-2 consists of two parts, each having planar faces that are aligned parallel to each other. The parallel orientations of the two parts suggest that the surface outward normal between the two opposing faces  $f_{15}$  and  $f_{26}$  (see test case 1 in Table 5-2) forms an angle of  $180^{\circ}$  that is between the user-defined angle tolerance of  $170^{\circ}$  and  $190^{\circ}$ . The algorithm evaluates the two parts for geometries that satisfy the orientation condition. The result from the analysis showed six pair of faces from the two parts as having satisfied the orientation condition. The six pair of faces that were identified are  $f_{11}$ - $f_{23}$ ,  $f_{12}$ - $f_{24}$ ,  $f_{13}$ - $f_{21}$ ,  $f_{14}$ - $f_{22}$ ,  $f_{16}$ - $f_{25}$ , and  $f_{15}$ - $f_{26}$ .

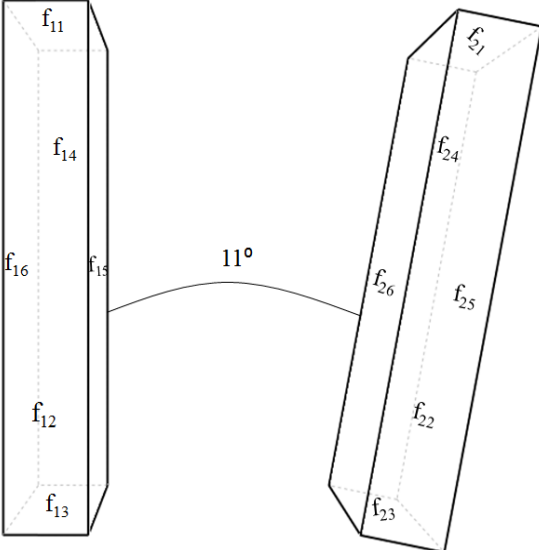
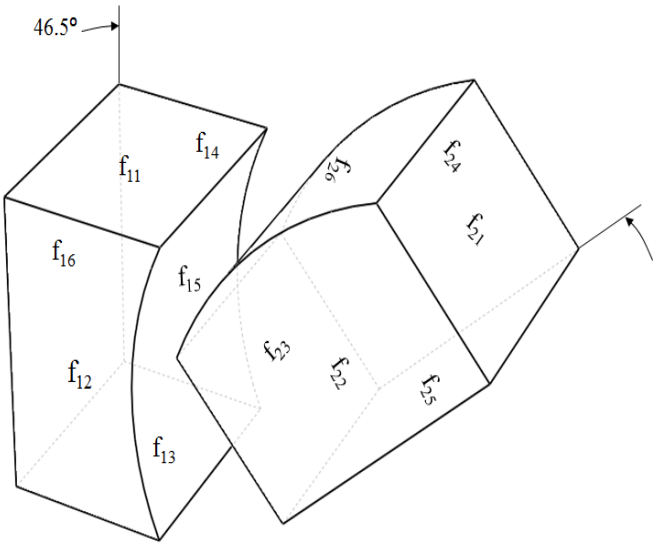


In the test case 2 shown in Table 5-2, the angle between the surface outward normal from the planar faces  $f_{15}$  and  $f_{26}$  is increased to  $190^\circ$ . The angle  $190^\circ$  represents the upper limit for the user-defined orientation angle tolerance. The algorithm evaluates the two parts and still identifies six pair of faces as having satisfied the orientation condition. The six pair of faces that meets the orientation condition are  $f_{11}$ - $f_{23}$ ,  $f_{12}$ - $f_{24}$ ,  $f_{13}$ - $f_{21}$ ,  $f_{14}$ - $f_{22}$ ,  $f_{16}$ - $f_{25}$ , and  $f_{15}$ - $f_{26}$ .

For the test case 3 shown in Table 5-2, the angle between the surface outward normal from the planar faces  $f_{15}$  and  $f_{26}$  is increased to  $191^\circ$  that is outside the user-defined angle tolerance. For this test case, the algorithm does not identify any pair of faces as meeting the orientation condition as none of faces are oriented within the orientation tolerance defined by the user.

**Table 5-2: Test cases used check for orientation between face pairs**

Test Case No.	Test Case	Description
1	<p>Parallel</p> 	<p><b>Input:</b> User-defined orientation = <math>180^\circ \pm 10^\circ</math></p> <p><b>Output:</b> The algorithm evaluated the following pair of faces as being opposed to each other within the user defined orientation angle,</p> <ol style="list-style-type: none"> <li>1. <math>f_{11}</math>-<math>f_{23}</math></li> <li>2. <math>f_{12}</math>-<math>f_{24}</math></li> <li>3. <math>f_{13}</math>-<math>f_{21}</math></li> <li>4. <math>f_{14}</math>-<math>f_{22}</math></li> <li>5. <math>f_{16}</math>-<math>f_{25}</math></li> <li>6. <math>f_{15}</math>-<math>f_{26}</math></li> </ol>
2		<p><b>Input:</b> User-defined orientation = <math>180^\circ \pm 10^\circ</math></p> <p><b>Output:</b> The algorithm evaluated the following pair of faces as being opposed to each other within the user defined orientation angle,</p> <ol style="list-style-type: none"> <li>1. <math>f_{11}</math>-<math>f_{23}</math></li> <li>2. <math>f_{12}</math>-<math>f_{24}</math></li> <li>3. <math>f_{13}</math>-<math>f_{21}</math></li> <li>4. <math>f_{14}</math>-<math>f_{22}</math></li> <li>5. <math>f_{16}</math>-<math>f_{25}</math></li> <li>6. <math>f_{15}</math>-<math>f_{26}</math></li> </ol>

3		<p><b>Input:</b> User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b> The algorithm evaluated the following pair of faces as being opposed to each other outside the user defined orientation angle. None of the faces meet the orientation condition.</p>
4		<p><b>Input:</b> User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b> The algorithm evaluated three pair of faces as being opposed to each other within the user defined orientation angle. The face pairs identified were,</p> <ol style="list-style-type: none"> <li>1. <math>f_{12}</math>-<math>f_{24}</math></li> <li>2. <math>f_{14}</math>-<math>f_{22}</math></li> <li>3. <math>f_{15}</math>-<math>f_{26}</math></li> </ol> <p>The angles between other face pairs are greater than the user-defined orientation angle.</p>

The test case 4 shown in Table 5-2 consists of two curved faces opposed to each other at an angle of  $46.5^0$ . This angle between the two faces is set by inclining the face  $f_{25}$  at an angle of  $46.5^0$  with respect to the face  $f_{16}$ . Although the angles between all the other

planar faces are greater than the threshold angle except for face pairs  $f_{12}$ - $f_{24}$  and  $f_{14}$ - $f_{22}$  that are parallel, the angle between surface outward normal at some sampling points on the curved faces  $f_{15}$  and  $f_{26}$  is within the user-defined orientation angle tolerance of  $170^\circ$  and  $190^\circ$ . This test case is designed to check the performance of the algorithm for instances when the face pair partially satisfies the orientation condition. As seen in the results for the test case 4 shown in Table 5-2, the algorithm identifies even the face pair that is only partially within the threshold angle tolerance. These faces ( $f_{12}$ - $f_{24}$ ,  $f_{14}$ - $f_{22}$ , and  $f_{15}$ - $f_{26}$ ) are recognized by the algorithm as satisfying the orientation condition and are stored in a container for percentage similarity analysis.

The algorithm was also tested for its performance against other geometric types such as cylindrical, freeform, and spherical surfaces that are shown in Appendix B: with results. The check for orientation between the faces is a pre-requisite step before evaluating the geometries for percentage similarity. The orientation is a necessary condition that needs to be satisfied by geometries to be considered for percentage similarity analysis. The geometries that satisfy the threshold distance condition (first necessary condition) but do not satisfy the orientation condition (second necessary condition) are dropped from further analysis, as geometries need to satisfy both the necessary conditions before being tested for percentage similarity. The results indicate that the algorithm is consistent with all geometric types that are tested in evaluating the faces for orientation condition.

### 5.3 Test-Cases to Check for Percentage Similarity between Geometries

The percentage similarity check is the final step in the duplicate geometry comparison analysis. The algorithm uses two user-defined values to check for duplicate geometry in this stage: the percentage similarity value (requirement no. 3) and the tolerance bound (requirement no. 4). The algorithm requires the user to input the values for the desired percentage similarity and the tolerance bound which are the requirements listed in chapter Three. The test cases designed to check the program's ability to recognize duplicate geometry uses different degree of similarity between the two faces, different amount of overlap between the faces, and varying geometric types. In order to discuss the performance of the algorithm for these different cases, the relevant test cases are selected from the complete list shown in Appendix C:.

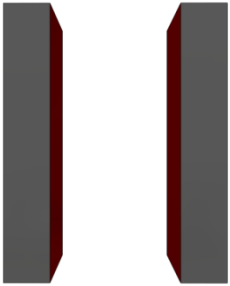
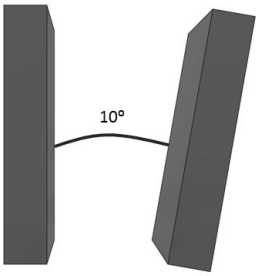
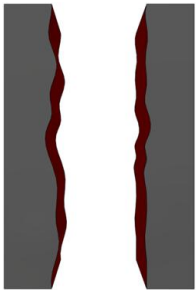
The test case 1 shown in Table 5-3 uses the assembly of two identical rectangular plates separated by a distance equal to 14mm. The user inputs for the algorithm are shown in the Table 5-3 under the column description. The algorithm successfully identifies the two planar faces as duplicate that is highlighted in red. The same test case is modified by changing the orientation of the rectangular plate to the right by  $10^{\circ}$  as shown in the test case 2. Although the orientation between the two opposing faces is within the user-defined orientation value, no instances of duplicate geometry are identified by the algorithm. From this observation, it is evident that the orientation and the threshold distance conditions are not sufficient (but necessary) for duplicate geometry analysis. The two opposed faces do not pass the percentage similarity evaluation for the given user inputs.

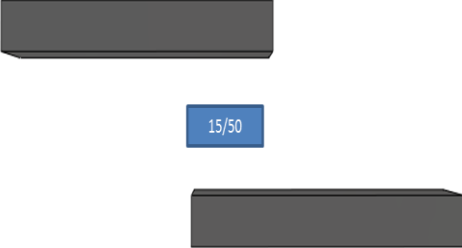
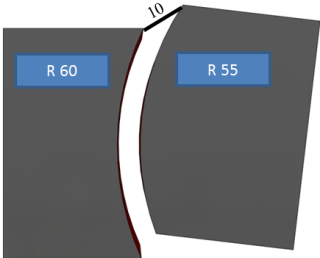
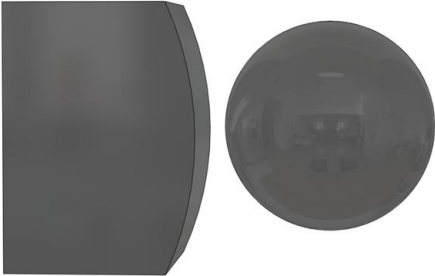
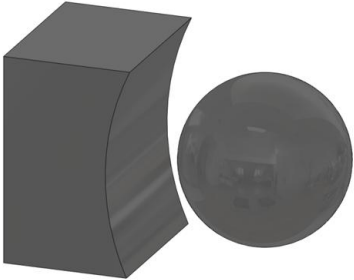
The two opposed planar faces from test case 1 are modified to have waviness as shown in test case 3 in Table 5-3. The waviness in this test case is only in 2-dimension and therefore is different from the freeform surface. This test case is similar to the test case 1 in terms of the assembly, but is used to check the performance of the algorithm for non-planar geometry. As observed in the results the algorithm identify the two wavy faces as duplicate that is highlighted in red. In another modification to test case 1, the amount of overlap between the two opposed planar faces is changed to be less than 50%. This test case demonstrates the ability of the algorithm to evaluate geometries having different amount of overlap. The results show that the algorithm evaluates the two parts to have no instances of duplicate geometry.

The test cases 5, 6, and 7 demonstrate the percentage similarity evaluation results for curved and spherical surfaces. The test case 5 in Table 5-3 has two curved surfaces opposed to each other at an angle that is within the user defined orientation angle. The radiuses of the two curved surfaces are different that represent certain amount of similarity but not identical surfaces. The algorithm evaluated these curved surfaces to be duplicate geometries and was highlighted in red. The results indicate that the two curved surfaces have percentage similarity greater than or equal to 80% and the surface variation within the 2mm tolerance bound. The test case 6 consists of two parts that has a convex curved surface opposed to a spherical surface. For this test case, the algorithm returned no instances of duplicate geometries, which is the indication of not satisfying that percentage similarity and tolerance bound condition. In the test case 7, convex curved surface from test case 6 was replaced with a concave curved surface where the profile of

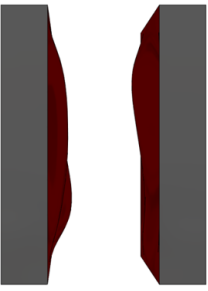
the concave curved surface followed the profile of the spherical surface. However, the algorithm did not recognize any instances of duplicate geometries between the parts.

**Table 5-3: Test cases to check for percentage similarity**

Test Case No.	Test Case	Description
1		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> Duplicate geometry shown in red
2		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> No duplicate geometries
3		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> Duplicate geometry shown in red

4		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
5		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>
6		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
7		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>



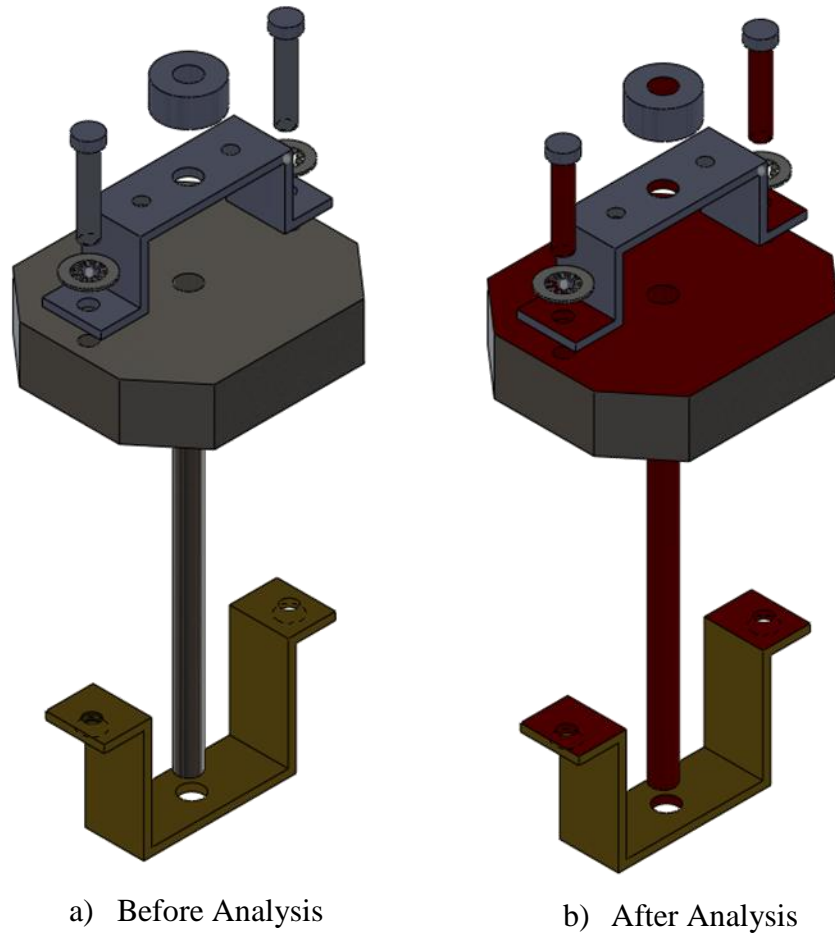
8		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>
---	---	--

The test case 8 in Table 5-3 shows two parts having freeform surfaces opposed to each other. This test case is used to check the performance of the algorithm in the evaluation of percentage similarity for freeform surfaces. The results indicate that the two freeform surfaces are duplicate to each other with similarity between the surfaces equal to or greater than the percentage similarity value and the surface variation between the two geometries within the user defined tolerance bound.

The test cases presented in this section covers different geometric types, different degrees of overlap between faces, and different amounts of similarity for the verification of the performance of the algorithm. The algorithm is also evaluated against other geometric types such as conical and cylindrical surfaces that are shown in Appendix C:. The percentage similarity evaluation is the final step in the algorithm for the identification of duplicate geometry instances. The use of test cases demonstrates that the algorithm is consistent in the evaluation of percentage similarity for the cases shown. In the next section, the effect of the geometric types on the analysis time is presented.

#### 5.4 Highlight Duplicate Geometries

The fifth requirement of the algorithm states that it is required to highlight the instances of duplicate geometries for the user to visualize the results on screen. This requirement is met by changing the color of duplicate geometries to red. To illustrate further, for a given CAD assembly all the instances of duplicate geometry pair are highlighted by changing the color of the face to red as shown in Figure 5.1. Due to this reason, it required not to have any parts in the assembly whose color is already set to red.




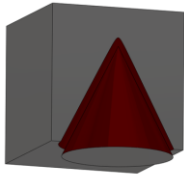
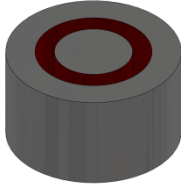

**Figure 5.1: Instances of duplicate geometry highlighted in red by the algorithm**

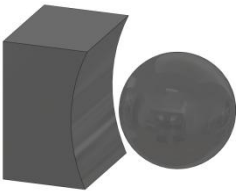
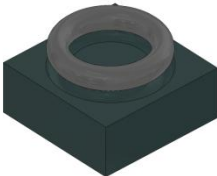
### 5.5 Effect of Geometric Types on the Evaluation

In this section, the ability of the algorithm to evaluate different geometric types for duplicate geometry analysis is presented. This is the sixth requirement of the algorithm listed in the requirement list in Chapter Three. The different geometric types tested were planar, cylindrical, spherical, freeform, conical, and toroidal shapes. Recalling from the requirement list presented in Chapter Three, the fifth requirement

necessitates the algorithm to work with different geometric types that may be encountered in the CAD assembly. The analysis time shown in Table 5-4 is the system time calculated using the number of ticks elapsed since the evaluation started for each of the three algorithms.

**Table 5-4: Duplicate geometry analysis results for different geometric types**

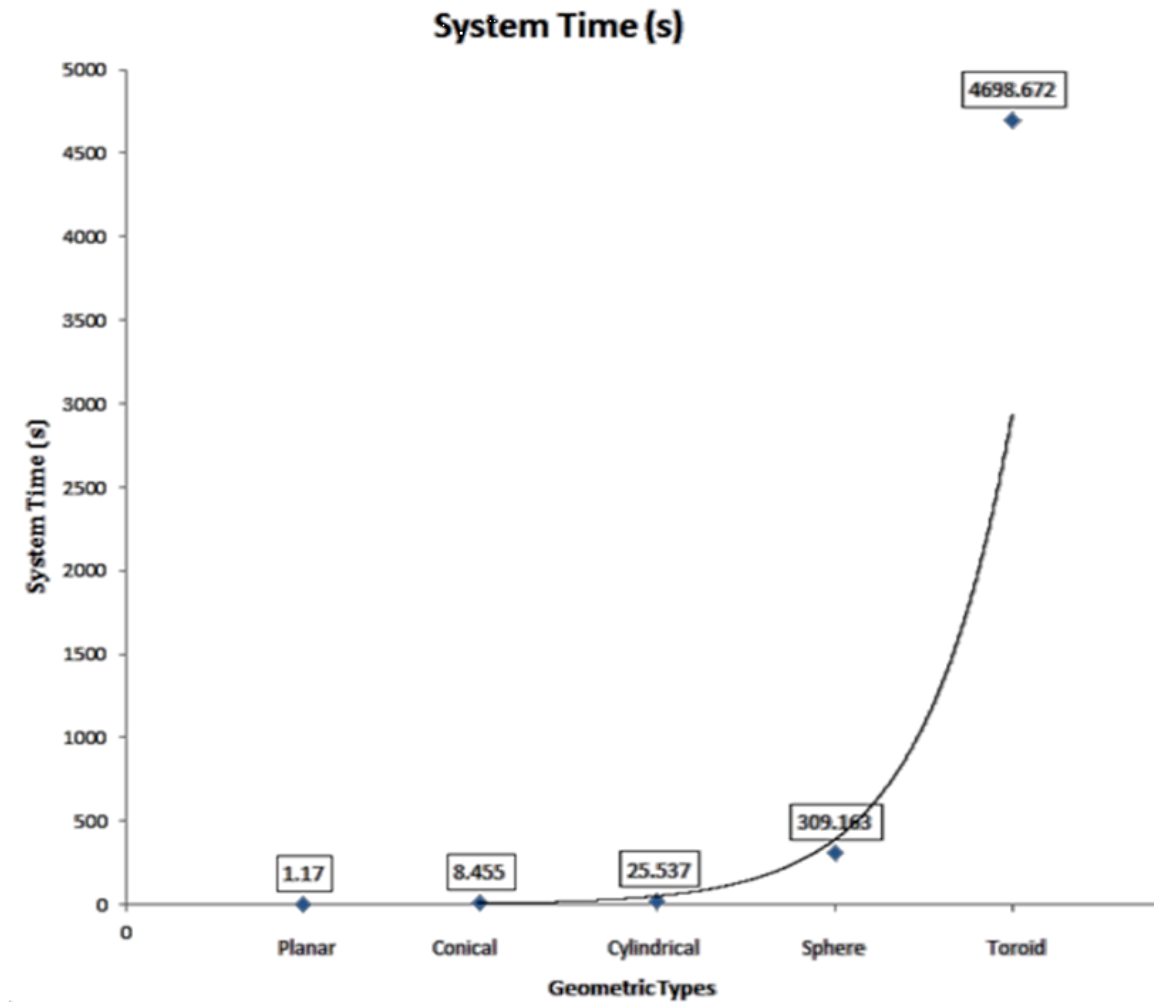
Geometric Type		Threshold Distance Evaluation Time	Orientation Condition Evaluation Time	Percentage Similarity Evaluation Time
Planar		0 ms	1.17 s	21.88 s
Conical		0ms	8.45 s	1.21 min
Cylindrical		0ms	25.53 s	4.09 min
Freeform		0ms	9.84 min	3.04 min

Spherical		0ms	5.15 min	11.01 min
Toroidal		0ms	78.31 min	11.38 min

The results indicate that the algorithm was successful in the evaluation of different geometric types. The three evaluation phases of the algorithm are: checking for threshold distance condition, checking for orientation condition, and checking for percentage similarity. It is observed that the threshold distance evaluation time is independent of the geometry. The bounding box algorithm that is used for calculating the distance between parts depends on the number of parts in the assembly (worst case complexity is  $O(N^2)$ , where  $N$  is the number of parts) and the geometric shape of the part has not effect on the bounding box calculations. All the test cases shown in Table 5-4 contain two parts and the threshold distance analysis time was zero milliseconds.

However, the geometric types affects the evaluation time for the orientation and percentage similarity analysis. The number of facets generated on the face depends upon the geometric shape of the face. It is observed that the planar face always generates fewer number of facets compared to a non-planar face for a given width of the facet edge. The worst case complexity for both the orientation and the percentage similarity algorithm is

$O(M^2 \times N^2)$ , where  $M$  is the number of faces in the part and  $N$  is the number of sampling points on the face. The analysis time presented in Table 5-4 for the orientation and percentage similarity evaluation indicate an increase in the analysis time from planar to non-planar geometric types. The analysis time consumed for the evaluation of different geometric types is presented in Figure 5.2. The graph shows exponential increase in the analysis time from the planar geometric type to the toroidal geometric type. This increase in the analysis time is because of the increase in the number of facets in the non-planar geometries. The number of facets required to represent a non-planar geometric type is more than the number of facets required to represent a planar geometry. Due to the increase in the number of facets, the number of sampling points on each face is also increased that affects the analysis time.



**Figure 5.2: Time consumed for the evaluation of orientation condition for different geometric types**

The analysis time for the evaluation of percentage similarity shown in the Table 5-4 varies for different geometric types. The number of sampling points generated on the face for percentage similarity evaluation is different for the number of sampling points used for the evaluation of orientation condition. The algorithm re-tessellates all the faces that have met the orientation condition to generate the facets. The re-tessellation of the

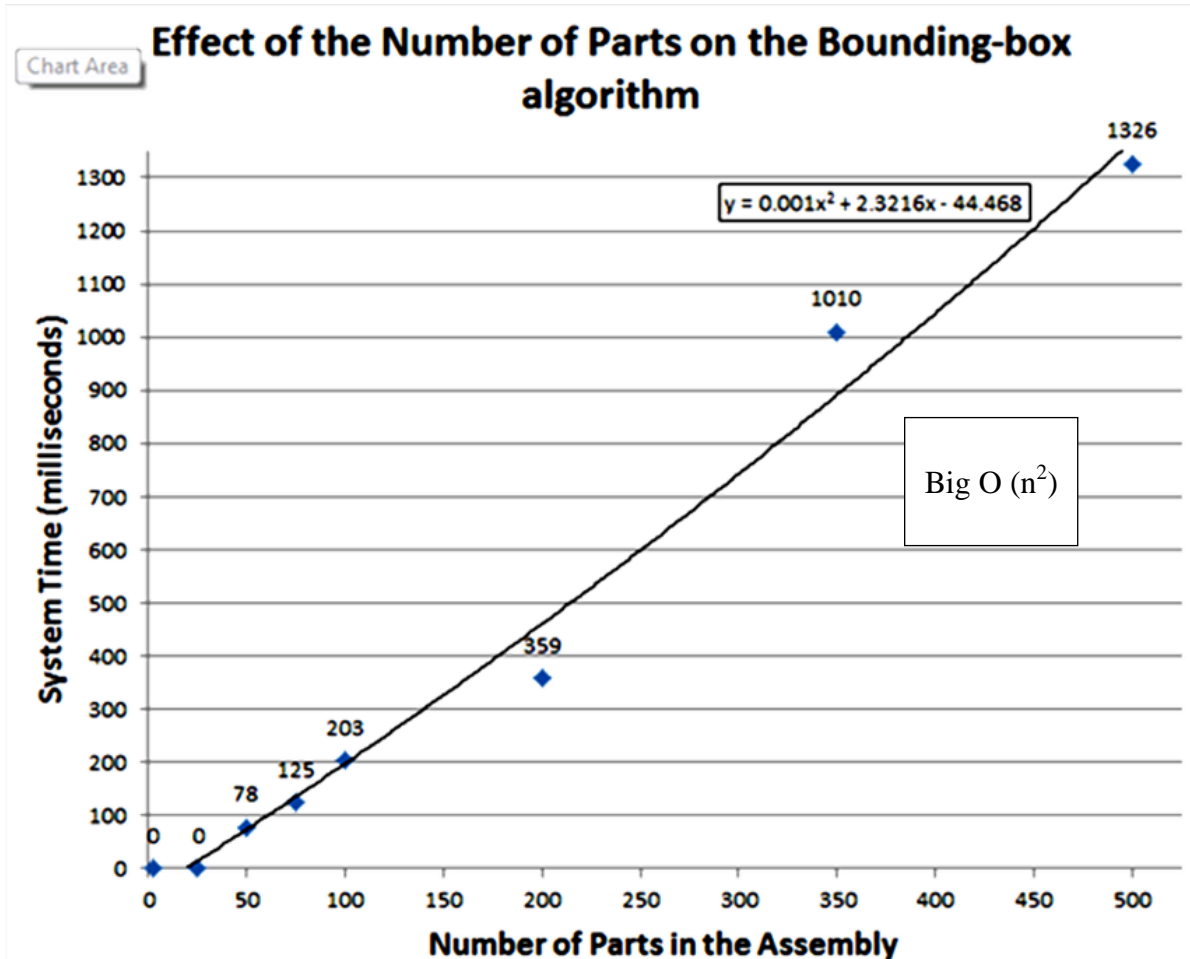
faces generates more number of sampling points by using shorter facet width. The combination of the number of faces and the coarse tessellation used for the orientation evaluation and the number of shortlisted faces and the fine tessellation used for the percentage similarity evaluation affects the analysis time that is presented in Table 5-4. The complexity of algorithm for percentage similarity evaluation is  $O(M^2 \times N^2)$ , where M is the number of faces and N is the number of sampling points. Although the complexity for the percentage similarity algorithm is same as the complexity of the orientation evaluation algorithm, the number of faces and the sampling points are different that varies the analysis time between the two algorithms. Because of this difference in the number of faces considered for the evaluation and the number of sampling points used, consistent trend between the analysis times for the orientation and percentage similarity evaluation is not observed.

#### 5.6 Effect of the Number of Parts on the Bounding-box Algorithm

The effect of the number of parts on the analysis time of the bounding-box algorithm for the evaluation of threshold distance condition is presented in this section. To study the effect of the number of parts, the pattern of cubes is used to generate more parts and the analysis time is recorded. The worst case complexity for this algorithm is  $O(n^2)$ , where n is the number of parts. The bounding-box algorithm compares each part in the assembly with all the other parts in the assembly until all the combinations of part comparisons are exhausted. This quadratic nature of the algorithm complexity is also observed in the analysis time consumed that is showed in the Figure 5.3. The graph shows the system time consumed in milliseconds for the evaluation of the threshold



distance condition for the assemblies with different number of parts. The results validate that bounding-box approach offers a faster first pass filtering of parts in close proximity. The algorithm consumed zero milliseconds for the assembly of up to 25 parts and 1.3 seconds for the assembly of 500 parts.

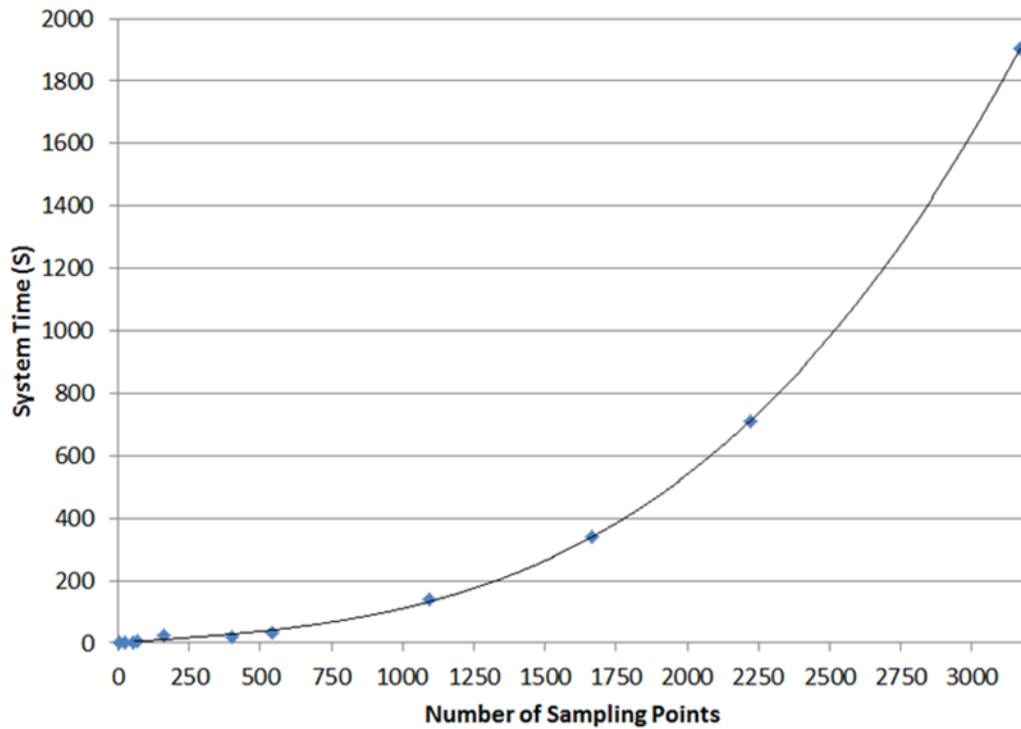


**Figure 5.3: Effect of the number of parts on the bounding-box algorithm**

### 5.7 Effect of the Number of Sampling Points on the Percentage Similarity Algorithm

The effect of the number of sampling points on the analysis time used for the measurement of percentage similarity is presented in this section. The results indicate a

polynomial increase in the analysis time with the increase in the number of sampling points as shown in Figure 5.4. The program consumed 31 milliseconds for the evaluation of percentage similarity when four sampling points were used on two planar faces. The analysis time increased to 0.2 seconds for 12 sampling points and 6.02 seconds for 70 sampling points. For two planar geometries of square cross sectional area, four sampling points were sufficient for the percentage similarity evaluation. However, for non-planar geometries more sampling points are required for the percentage similarity evaluation that would increase the analysis time. For a non-planar geometry, more sampling points are generated compared to a planar geometry of comparable area of cross section because the number of facets required to represent a non-planar geometry (curved, spherical, or freeform) is usually higher that would result in the increased number of sampling points. An example of a non-planar geometry with higher number of sampling points is a concave face that generated 3168 sampling points and consumed 31.75 minutes for the analysis.



**Figure 5.4: Effect of the number of sampling points on the analysis time for the evaluation of percentage similarity**

#### 5.8 Algorithm offers Extensibility to obtain Weighted Assembly Relations

This section presents the assembly test cases and their part connectivity information that is automatically retrieved by the algorithm. The requirement seven states that the algorithm needs to provide extensibility to extract the weighted part connectivity graph of the assembly file to support the part connectivity based assembly time estimation method [9,18,20]. The test cases used in this section demonstrates the ability of the algorithm to automatically extract the part connectivity information and thus meets requirement seven.

The duplicate geometry algorithm is modified to obtain weights for the amount of overlap between two connected faces. The amount of overlap is measured by calculating the ratio of the number of sampling points that meet the percentage similarity condition to the total number of sampling points. The threshold distance value for extracting the assembly relations is set at 1mm. The orientation angle and tolerance value used are  $180^{\circ} \pm 5^{\circ}$ . The percentage similarity value is set at 90% with a bound of -1mm to +1mm. The maximum facet size used to generate sampling points is set at 15mm. The algorithm is run on the motor assembly shown in Figure 5.1. The resulting weighted part connectivity bipartite graph is written to a “filename.csv” (comma-separated values) file.

The result of the analysis is presented in Figure 5.5. The result show the connections between one face to another from different parts with the amount of overlap. The first row of the bipartite graph in Figure 5.5 informs that a face from the part “stack\_motor-1” is connected to the face from another part “shaft\_motor-1” with 0.76 overlap between the two faces. The data in Figure 5.5 represents the list of physical connectivity between parts in the assembly. This part connectivity data is objective for a given assembly. The same connections between parts and the same weight are retrieved every time the analysis is run on the assembly that also validates the repeatability of the automated data collection method.

Bipartite graph of part connections		
Part Name	Part Name	Weight
stack_motor-1	shaft_motor-1	0.768421
stack_motor-1	bracket_top_motor-1	0.0666667
stack_motor-1	bracket_top_motor-1	0.0606061
stack_motor-1	bracket_top_motor-1	0.230769
stack_motor-1	bracket_top_motor-1	0.230769
stack_motor-1	screw_motor-1	0.794872
stack_motor-1	bracket_bottom_motor-1	0.060241
stack_motor-1	bracket_bottom_motor-1	0.0662651
stack_motor-1	bracket_bottom_motor-1	0.230769
stack_motor-1	bracket_bottom_motor-1	0.230769
stack_motor-1	screw_motor-2	1
shaft_motor-1	gear_motor-1	0.130435
shaft_motor-1	bracket_top_motor-1	0.0434783
shaft_motor-1	bracket_bottom_motor-1	0.0434783
bracket_top_motor-1	screw_motor-1	0.394737
bracket_top_motor-1	screw_motor-2	0.447368
screw_motor-1	bracket_bottom_motor-1	0.290323
screw_motor-1	bracket_bottom_motor-1	0.55
bracket_bottom_motor-1	washer_motor-2	0.0243902
bracket_bottom_motor-1	screw_motor-2	0.512195
bracket_bottom_motor-1	screw_motor-2	1

**Figure 5.5: Weighted bipartite graph of part connectivity information extracted from the motor assembly**

In this chapter, test cases are used to demonstrate that the algorithm meets the requirements presented in Chapter Three. The algorithm requires the user to input the values for threshold distance, orientation angle and tolerance, percentage similarity, and the bounds for the duplicate geometry evaluation. It is also shown that the algorithm works with different types of geometries. It is observed that the number of sampling points generated on the face for duplicate geometry comparison has the greatest effect on

the analysis time compared to the number of parts and the number of faces. The increase in the number of sampling points causes a quadratic increase in the analysis time. In order to help the designers review the result of the analysis, duplicate geometry instances are highlighted in red. The algorithm is also extendible to automatically extract the part connections from the assembly file. The motor assembly is used to demonstrate that the algorithm can extract a weighted bipartite graph of assembly relations. The usability requirements (8-10) are addressed in the system architecture presented in Chapter Four.

## 5.9 External Validation

The ability of the algorithm to use the duplicate geometry approach to extract the part connectivity graph for CAD assemblies is presented in this section. The test assemblies used in this section were externally developed and are only used in this research for validation purpose. The externally developed assemblies used for this section are of products encountered in the real world that would provide different connection types. The discussion of results from the three test cases used is presented in sections 5.9.1, 0, and 5.9.3.

### 5.9.1 Vise Assembly

The vise is a mechanical device used for clamping the work piece. The assembly of vise used in this analysis is selected from the library of assemblies in the SolidWorks folder. The assembly of vise and its parts are shown in Figure 5.6. The input parameters used for the algorithm is shown in Table 5-5.

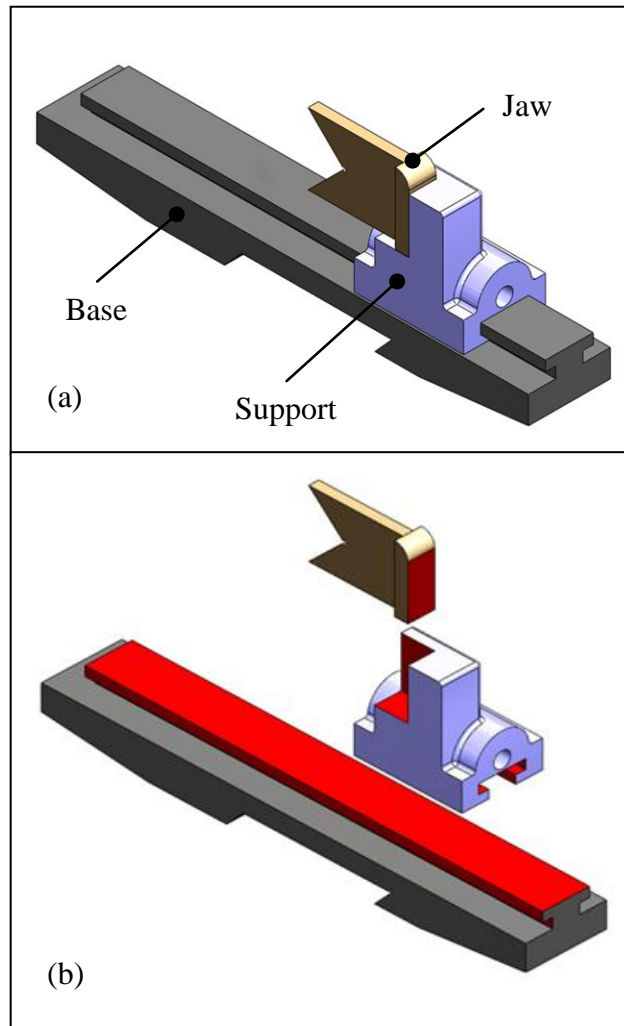
**Table 5-5: Input parameters for the algorithm**

<b>Name</b>	vise.sldasm
<b>Max Facet Size</b>	5 mm
<b>Threshold Distance</b>	1 mm
<b>Orientation</b>	$180^0 \pm 2^0$
<b>Percentage Similarity</b>	Not applicable for assembly relations extraction
<b>Bound</b>	2 mm

The assembly of vice consisted for four parts, but one of the part (clamp) was suppressed to check if the algorithm was able to detect and filter out the suppressed part. As a result, the number of active parts in the assembly was three. For this configuration the anticipated part connections are shown in Table 5-6.

**Table 5-6: Anticipated connections**

<b>Sl. No.</b>	<b>Part Name</b>	<b>Part Name</b>
1	Support	Base
2	Support	Base
3	Support	Base
4	Support	Base
5	Support	Base
6	Support	Base
7	Support	Base
8	Support	Jaw
9	Support	Jaw
10	Support	Jaw



**Figure 5.6: (a) The assembly of vice and the constituent parts; (b) Part connection faces are highlighted by the algorithm in red**

The algorithm was able to successfully identify all the ten instances of part connections and was able to filter out the suppressed clamp. The region of part connections are highlighted by the algorithm and shown in the Figure 5.7 (b). The weighted bipartite graph of part connections extracted by the duplicate geometry algorithm for the vice assembly is shown in Figure 5.7. The algorithm consumed 99.83 minutes for the complete analysis.



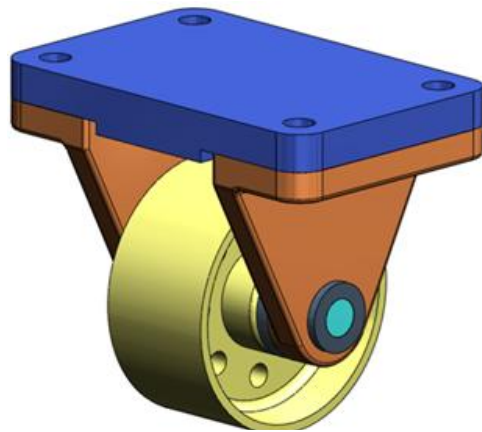
**Bipartite graph of part connections**

Part Name	Part Name	Weight
support-1	base-1	0.952381
support-1	base-1	0.483871
support-1	base-1	0.483871
support-1	base-1	0.952381
support-1	base-1	0.952381
support-1	base-1	0.941704
support-1	base-1	0.952381
support-1	jaw-1	0.42069
support-1	jaw-1	0.963636
support-1	jaw-1	0.456897

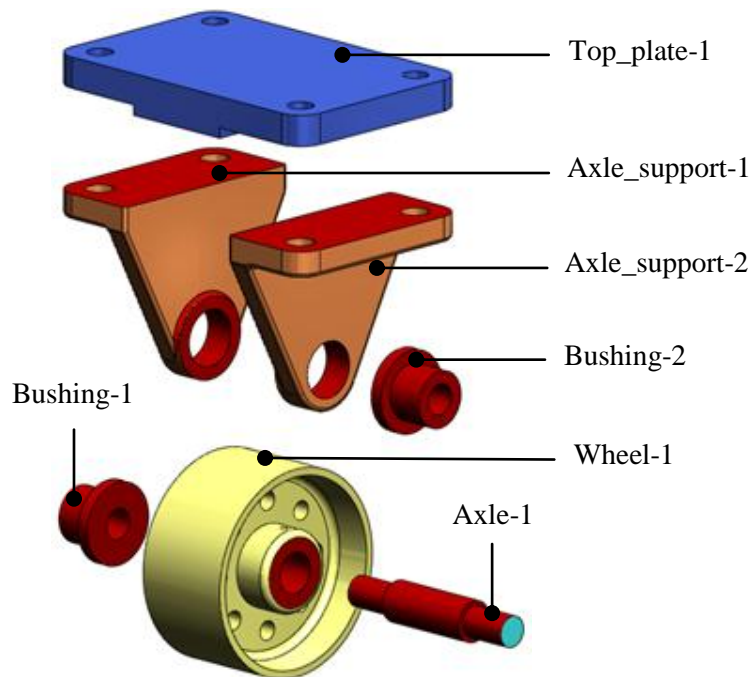
**Figure 5.7: Part connections extracted for the vice assembly**

#### 5.9.2 Caster Assembly

The second test case used is the caster assembly from SolidWorks library. The caster is an assembly of the wheel and supporting parts that is attached to the bottom of mechanical structure for the purpose of moving. The caster assembly consists of seven parts as shown in Figure 5.8.



(a) Caster assembly



(b) Caster assembly after the analysis

**Figure 5.8: The caster assembly from SolidWorks library**

The input parameter for the analysis of caster assembly is shown in Table 5-7. The maximum size for the facet edge was increased to 10mm from the initial 5mm that was used for the vice assembly. The other parameters were not changed.

**Table 5-7: Input parameters for the caster assembly**

<b>Name</b>	caster.sldasm
<b>Max Facet Size</b>	10 mm
<b>Threshold Distance</b>	1 mm
<b>Orientation</b>	$180^0 \pm 2^0$
<b>Percentage Similarity</b>	Not applicable for assembly relations extraction
<b>Bound</b>	2 mm

A total of eleven part connections are identified for the caster assembly that is shown in Table 5-8.

**Table 5-8: Anticipated part connections for caster assembly**

<b>Sl. No.</b>	<b>Part Name</b>	<b>Part Name</b>
1	Top_plate-1	Axle_Support-1
2	Top_plate-1	Axle_Support-2
3	Axle_Support-1	Bushing-1
4	Axle_Support-1	Bushing-1
5	Axle_Support-2	Bushing-2
6	Axle_Support-2	Bushing-2
7	Bushing-1	Wheel-1
8	Bushing-2	Wheel-1
9	Axle-1	Wheel-1
10	Axle-1	Bushing-1
11	Axle-1	Bushing-2

The algorithm was able to identify twenty five part connections in the assembly. This is fourteen part connections more than the anticipated part connections. The algorithm has identified other duplicate geometric pairs that have satisfied the 1mm threshold condition. The algorithm consumed 36.6 minutes to complete the analysis. The weighted bipartite graph of part connections for the caster assembly retrieved by the algorithm is shown in Figure 5.9.

**Bipartite graph of part connections**

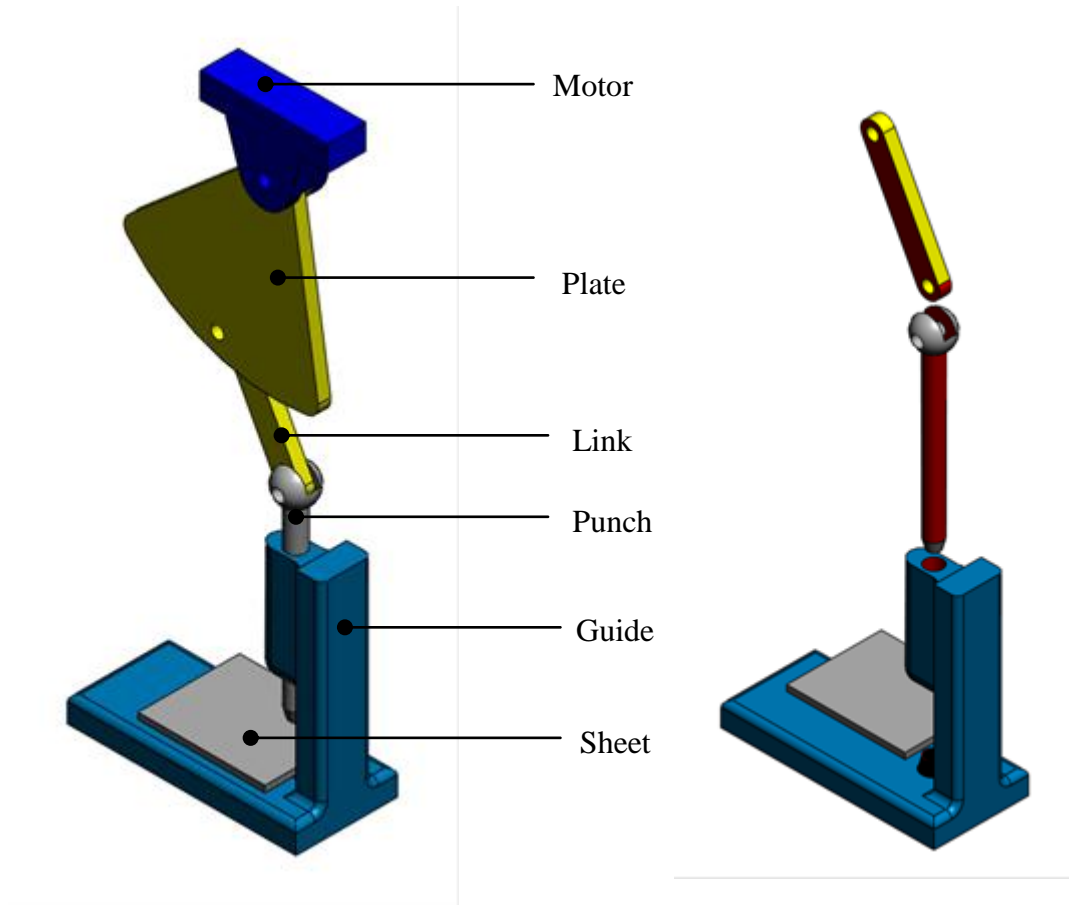
Part Name	Part Name	Weight
Axle-1	Wheel-1	0.865269
Axle-1	Wheel-1	0.357143
Axle-1	Wheel-1	0.357143
Axle-1	Bushing-2	0.46988
Axle-1	Bushing-2	1
Axle-1	Bushing-2	0.40678
Axle-1	Bushing-1	1
Axle-1	Bushing-1	0.40678
Axle-1	Bushing-1	0.46988
Wheel-1	Bushing-2	0.0714286
Wheel-1	Bushing-1	0.0357143
Bushing-2	Axle_support-2	0.513274
Bushing-2	Axle_support-2	0.0617284
Bushing-2	Axle_support-2	0.269767
Bushing-2	Axle_support-2	0.0617284
Bushing-2	Axle_support-2	0.974522
Bushing-2	Axle_support-2	0.451327
Bushing-2	Axle_support-2	0.383459
Axle_support-2	Top_plate-1	0.7343
Top_plate-1	Axle_support-1	0.7343
Axle_support-1	Bushing-1	0.898089
Axle_support-1	Bushing-1	0.452229
Axle_support-1	Bushing-1	0.383459
Axle_support-1	Bushing-1	0.0740741
Axle_support-1	Bushing-1	0.0864198

**Figure 5.9: Part connections retrieved for the caster assembly**

### 5.9.3 Punch Assembly

The punch is a mechanical system used for producing holes in sheet metals. The punch assembly test case used in this section is taken from the SolidWorks installation folder. The assembly consists of six parts in total as shown in Figure 5.10 (a). For this

analysis, the motor was hidden and the plate was suppressed to check the ability of the algorithm to analyze only the active parts. The assembly with only the active parts is shown in Figure 5.10 (b).



**Figure 5.10: The punch assembly**

The input parameters for the assembly with four active parts are as shown in Table 5-9.

**Table 5-9: Input parameters for the punch assembly**

<b>Name</b>	punch.sldasm
<b>Max Facet Size</b>	10 mm
<b>Threshold Distance</b>	1 mm
<b>Orientation</b>	$180^0 \pm 2^0$
<b>Percentage Similarity</b>	Not applicable for assembly relations extraction
<b>Bound</b>	2 mm

The anticipated part connections for the punch assembly are shown in Table 5-10.

**Table 5-10: Anticipated part connections for punch assembly**

<b>Sl. No.</b>	<b>Part Name</b>	<b>Part Name</b>
1	Link-1	Punch-1
2	Link-1	Punch-1
3	Link-1	Punch-1
4	Punch-1	Guide-1
5	Sheet-1	Guide-1

The algorithm consumed 2.23 minutes for the complete evaluation of the assembly with four active parts. The algorithm was able to identify six part connections in the assembly. This is one more than the anticipated part connections that were manually identified. The algorithm identified two extra connections between the Link-1 and the Punch-1, as opposed to only one connection that was identified manually, because of the split face in the punch. The part connectivity relation and the weights for the punch assembly are shown in Table 5-11. The part connections in the assembly are highlighted that is shown in Figure 5.10 (b).

**Table 5-11: Assembly relations extracted for the punch assembly**

<b>Part Name</b>	<b>Part Name</b>	<b>Weight</b>
Sheet-1	Guide-1	0.0652174
Guide-1	Punch-1	0.0192308
Link-1	Punch-1	0.428571
Link-1	Punch-1	0.428571
Link-1	Punch-1	0.411765
Link-1	Punch-1	0.842105

To summarize, this section presented the externally developed test cases to demonstrate the ability of the algorithm to retrieve part connectivity graph for the CAD assemblies. The algorithm was tested using the vice assembly, caster assembly, and the punch assembly. The results indicate that the algorithm is able to identify suppressed and hidden parts and consider only the active parts for the analysis. The assembly relations are exported to a \*.csv file with the part names and the corresponding weight. The research contribution and the future work are presented in the next section.



## CHAPTER SIX: FUTURE WORK AND CONCLUSION

The motivation for this research was to develop a feature recognition system that could automate the identification of duplicate geometries in CAD assemblies to support the lazy-parts lightweight engineering method. Also, a need was identified for the development of a feature recognition system for the automated extraction of assembly relations from CAD assembly file to support the part connectivity-based assembly time estimation method. Based on the identified needs, the objective of this research was to develop a feature recognition algorithm that could both identify duplicate geometries and retrieve assembly relations.

### 6.1 Research Contribution

The repeatability issue associated with the manual identification of duplicate geometry is addressed by this research. The original definition for duplicate geometry was subjective and therefore provided opportunity for subjectivity in the decision making. The formal definition of duplicate geometry proposed in this research removes the subjectivity in identifying duplicate geometries. In addition to addressing the issues of repeatability and subjectivity, the automated identification of duplicate geometry by the feature recognition algorithm removes the tediousness involved with the manual identification.

The part connectivity based assembly time estimation is a semi-automated method for the assembly time estimation that required manual construction of the assembly relationship for the input. The construction of the part connectivity graph manually was a

tedious process that required time and effort both for the construction and quality check for errors. The algorithm developed in this research allows for the automated extraction of part connectivity graph from an assembly file that reduces human effort required to study the assembly and prepare the graph. The algorithm eliminates the need for checking the graph for manual construction error and consistency. The automated retrieval of the assembly relations would allow designers more time on the data analysis by the reduction in time and effort required for data collection. The algorithm provides the way for complete automation of part connectivity-based assembly time estimation.

The research in [9] focused on the development of a tool for the complete automation of the assembly time estimation for CAD assemblies using the user-defined mates information. However, the limitation of this research was the inability to extract connections in the case of part patterns. The duplicate geometry algorithm presented in this thesis can extract connectivity information from the part patterns. The limitations of using user-defined mates for the assembly time prediction can be overcome by using the duplicate geometry algorithm that can extract the part connections which is objective.

The feature recognition algorithm developed in this research is independent of the geometric types. The test cases made of different geometric types demonstrate the ability of the algorithm to evaluate different kinds of geometries. It is observed that the analysis time was the only parameter affected by the different geometric types because of the change in the number of sampling points for orientation and percentage similarity calculation. The geometric type did not have any effect on the threshold distance calculation.

## 6.2 Future Work

The research presented in this thesis is the first attempt at the automation of the lazy parts mass reduction method. The lazy parts mass reduction method consists of seven identifiers for the identification of parts that have potential for mass reduction. The method requires manual effort to check the assemblies for parts that satisfy the definition of seven identifiers. The duplicate geometry identification was one of the indicators of the lazy parts method that has been automated through the algorithm presented in this research. However, for the complete automation of the lazy parts mass reduction method it is required to develop and integrate the algorithms for the identification of the other six indicators. The six other indicators that require further research for the automatic identification are: rigid-to-rigid connection, support for a flexible part, positioning feature, bridging systems, material flow restriction, and fasteners. The definitions for all of the indicators are presented in Table 1-1.

Some of the research challenges identified related to the automation of the other indicators using CAD data are:

- How to distinguish between rigid and flexible parts in the CAD assemblies for the automation of rigid-to-rigid connection and support for a flexible part indicator?
- Can the positioning feature be defined in the CAD terminology that would allow for the positioning feature identification using CAD data? Also, will the definition of the positioning feature be unique that would distinguish them from other parts?

- How to capture the engineering knowledge required for the decision making of identifiers such as material flow restriction and bridging systems?

The research challenges presented above is not a complete list but fundamental questions that need to be answered for the automation of other indicators of lazy parts method.

The algorithm can retrieve weighted bipartite graph of part connections that is used as input for the part connectivity based assembly time estimation. The method is semi-automated except for the process of data collection for the input. With this algorithm, automation of collecting part connectivity information is achieved. There is a need for the integration of the algorithm presented in this research with the semi-automated part connectivity based assembly time estimation method in order to make the assembly time estimation a completely automated tool. The current part connectivity method for the assembly time estimation uses a Matlab program for performing computations on the bipartite graph. It is required to integrate the SolidWorks add-in developed for this research with the Matlab code so that when the duplicate geometry algorithm is initiated from the SolidWorks the part connectivity graph is exported to the Matlab code for computations and the estimated assembly time is presented back in the SolidWorks software.

The limitation of the percentage similarity analysis that has been identified is with respect to the distance measurements between geometries of unequal sampling points. If

the distance measurements between the sampling points are performed from the geometry with fewer sampling points to the geometry with equal or more sampling points then no issue has been identified. However, if the distance measurements are calculated from the geometry with more sampling points to the geometry with fewer sampling points, then the algorithm interprets that the larger geometry as not being duplicate of the smaller geometry. Although, the interpretation is correct, the smaller geometry could be a duplicate of the larger geometry that is not considered in the program. In the case the algorithm identifies two geometries to be duplicate, then the two geometries are presented as duplicate to each other and not as one being the duplicate of another. Due to this reason, it is always required to measure distances from the geometry with fewer sampling points to the geometry with equal or greater sampling points. The future modification that could be implemented in the program to resolve the above mentioned limitation is to swap the two geometries if the second geometry to where the distance is measured to have fewer sampling points compared to the first geometry where the distance is measured from.

### 6.3 Conclusion

The research in this thesis is motivated from two distinct research topics that were developed at Clemson University. The first research topic is the lazy parts light weight engineering tool that has a time consuming process of identifying lazy parts through the use of indicators. The duplicate geometry is one of the seven indicators and this research focuses on the automation of duplicate geometry indicator. The second research topic from which this research was motivated is the part connectivity based assembly time

estimation method. The part connectivity based method required bipartite graph of part connections in the assembly as input for the assembly time prediction. The extraction of the part connectivity information from the assembly model is a manual process that requires time and effort. Hence, this research also focused on the automation of the extraction of part connectivity information using the duplicate geometry algorithm. The automation of duplicate geometry identification and the automation of the extraction of assembly relations using a feature recognition algorithm form the research objective for this thesis.

The feature recognition algorithms that use the B-rep data were reviewed as part of the background study. The algorithms discussed in the literature mainly focused on the manufacturing features that finds application in computer aided process planning (CAPP) and computer numerical controlled (CNC) machining. The performance of the algorithms discussed was dependent on the geometric type. The requirements derived from the motivation and the shortcomings of extending the existing feature recognition algorithms for duplicate geometry identification that required an algorithm independent of geometric type helped in the recognition of a need for this research. The need identified necessitated the development of a feature recognition algorithm that is independent of geometric type for the identification of duplicate geometries in CAD assemblies.

Based on the research objective, the definition for duplicate geometry and the requirements for the duplicate geometry feature recognition algorithm were developed. The requirements list consisted of system requirements and user requirements. The user requirements (8-10) were related to the usability parameters and were addressed while

developing the system architecture for this research. The system architecture adopted for this research was developing an add-in in the SolidWorks CAD software using SolidWorks API in C++ programming language. The add-in was developed to meet the system requirements (1-7) that were validated using test cases. The test cases demonstrated that the algorithm was successful in the evaluation of geometries for duplicate geometry identification of different geometric types.

The results indicated that the number of sampling points used for the percentage similarity evaluation in the duplicate geometry algorithm has the major effect on the analysis time. The worst case big O complexity of the algorithm is,

$$O (^nC_2 \times m_1^2 \times m_2^2 \times p^4)$$

Where,

$n$  = the number of parts in the assembly

$m_1$  = the number of faces in the parts that meet the threshold distance condition

$m_2$  = the number of faces that meet the orientation condition

$p$  = the number of sampling points on the face

The algorithm is independent of geometric type and consists of no predefined library for the recognition of duplicate geometries. The algorithm is capable of extracting part connections from the assembly in the form of bipartite graph. The bipartite graph of part connection extracted also contains weight that is an indication of area of overlap between the connected faces.

In addition to the duplicate geometry algorithm, further research is required for the development of feature recognition algorithms for the automated identification of

other six lazy parts indicators that is discussed in Chapter One. For the complete automation of lazy parts identification in the CAD assembly, development and integration of feature recognition algorithms of other indicators are necessary. The part connectivity based assembly time estimation method requires the integration of the algorithm presented in this research with the assembly time computation algorithm for complete automation.



## CHAPTER SEVEN: REFERENCES

- [1] Caldwell B. W., Namouz E. Z., Richardson J. L. I., Sen C., Rotenburg T., Mocko G. M., Summers J. D., and Obieglo A., 2010, "Automotive Lightweight Engineering: A Method for Identifying Lazy Parts," SAE World Congress, Detroit, pp. 1-21.
- [2] Namouz E. Z., 2010, "Mass and Assembly Time Reduction for Future Generation Automotive Vehicles Based on Existing Vehicle Model," Clemson University.
- [3] Griesse D., Namouz E., Shankar P., Summers J. D., and Mocko G., 2011, "Application of A Lightweight Engineering Tool: Lazy Parts Analysis and Redesign of a Remote Controlled Car," IDETC, Washington DC, pp. 1-10.
- [4] Maier J., Ezhilan T., Fadel G., Summers J., and Mocko G., 2007, "A Hierarchical Requirements Modeling Scheme to Support Engineering Innovation," Proceedings of the 16th International Conference on Engineering Design (IDED07).
- [5] Mocko G., Summers J., Teegavarapu S., Ezhilan T., Maier J., and Fadel G., 2007, "A Modeling Scheme for Capturing and Analyzing Conceptual Design Information: An Application to the Hair Dryer Example and Comparison to Existing Literature," International Conference for Engineering Design, Paris.
- [6] Snider M., Summers J., and Teegavarapu S., 2007, "Database Support for Reverse Engineering Product Teardown and Redesign as Integrated into a Mechanical Engineering Course," ASEE Computers in Education Journal.
- [7] Grujicic M., Arakere G., Pisu P., Ayalew B., Seyr N., and Erdmann M., 2009, "Application of Topology, Size, and Shape Optimization Methods in Polymer Metal Hybrid Structural Lightweight Engineering," Multidisciplinary Modeling in Materials and Structures, **4**(4).
- [8] Teegavarapu S., Snider M., Summers J., Thompson L., and Grujicic M., 2007, "A Driver for Selection of Functionally Inequivalent Concepts at Varying Levels of Abstraction," Journal of Design Research, **6**, pp. 239-259.
- [9] Owensby J. E., 2012, "Automated Assembly Time Prediction Tool Using Predefined Mates from CAD Assemblies," Clemson University.
- [10] Stone R. B., McAdams D. a., and Kayyalethekkel V. J., 2004, "A product architecture-based conceptual DFA technique," Design Studies, **25**(3), pp. 301-325.

- [11] Dewhurst B., and Knight W., 1993, "Design for Assembly," IEEE Spectrum, **30**(9), pp. 53-55.
- [12] Boothroyd G., 1982, "Design for Assembly Handbook."
- [13] Boothroyd G., and Alting L., 1992, "Design for Assembly and Disassembly," in CIRP Annals-Manufacturing Technology, **41**(2), pp. 625-636.
- [14] Boothroyd G., 1994, "Product Design for Manufacture and Assembly," Computer-Aided Design, **26**(7), pp. 505-520.
- [15] Rodriguez-Toro C., Jared G., and Swift K., 2004, "Product-development Complexity Metrics: A Framework for Proactive-DFA Implementation," International Design Conference-Design, Dubrovnik, pp. 483-490.
- [16] Sanders D. D., 2009, "An Expert System for Automatic Design for Assembly," Assembly Automation, **29**(4), pp. 378-388.
- [17] Barnes C. J., Dalglish G. F., Jared G. E. M., Mei H., and Swift K. G., 1999, "Assembly oriented design," Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99) (Cat. No.99TH8470), (45-50), pp. 45-50.
- [18] Miller M. G., 2011, "Product and Process Based Assembly Time Estimation in Engineering Design," Clemson University.
- [19] Owensby E., Namouz E. Z., Shanthakumar A., and Summers J. D., 2012, "EXTRACTING MATE COMPLEXITY FROM ASSEMBLY MODELS TO AUTOMATICALLY PREDICT ASSEMBLY TIMES," International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Chicago, pp. 1-9.
- [20] Mathieson J. L., Wallace B. a., and Summers J. D., 2010, "Assembly Time Modeling through Connective Complexity Metrics," 2010 International Conference on Manufacturing Automation, Ieee, pp. 16-23.
- [21] Owensby J. E., Shanthakumar A., Rayate V., and Summers J. D., 2011, "Evaluation and Comparison of Two Design for Assembly Methods: Subjectivity of Information Inputs," ASME Design Engineering Technical Conference, Washington DC.
- [22] Otto K. N., and Wood K. L., 1998, "Product Evolution: A Reverse Engineering and Redesign Methodology," Research in Engineering Design, pp. 226-243.

- [23] Snider M. R., 2006, "Extended Toolset for Reverse Engineering to Support Lightweight Engineering."
- [24] Kao C.-Y., Kumara S. R. T., and Kasturi R., 1995, "Extraction of 3D Object Features from CAD Boundary Representation Using the Super Relation Graph Method," **17**(12).
- [25] Babic B., Nesic N., and Miljkovic Z., 2008, "A review of automated feature recognition with rule-based pattern recognition," *Computers in Industry*, **59**(4), pp. 321-337.
- [26] Shah J. J., Anderson D., Kim Y. S., and Joshi S., 2001, "A Discourse on Geometric Feature Recognition From CAD Models," *Journal of Computing and Information Science in Engineering*, **1**(1), p. 41.
- [27] Wu M., and Lit C., 1996, "Analysis on machined feature recognition techniques based on B-rep," *Computer-Aided Design*, **28**(8), pp. 603-616.
- [28] Han J., Pratt M., and Regli W. C., 2000, "Manufacturing feature recognition from solid models: a status report," *Robotics and Automation*, **16**(6), pp. 782-796.
- [29] Regli W. C., *Geometric Algorithms for Recognition of Features from Solid Models*.
- [30] Vandenbrande J. H., and Requicha a. a. G., 1993, "Spatial reasoning for the automatic recognition of machinable features in solid models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(12), pp. 1269-1285.
- [31] Gao S., and Shah J. J., 1998, "Automatic recognition of interacting machining features based on minimal condition subgraph," *Computer-Aided Design*, **30**(9), pp. 727-739.
- [32] Kailash S. ., Zhang Y. ., and Fuh J. Y. ., 2001, "A volume decomposition approach to machining feature extraction of casting and forging components," *Computer-Aided Design*, **33**(8), pp. 605-617.
- [33] Jun Y., Raja V., and Park S., 2001, "Geometric Feature Recognition for Reverse Engineering using Neural Networks," *The International Journal of Advanced Manufacturing Technology*, **17**(6), pp. 462-470.
- [34] Y Z., Taib J. M., and Tap M. M., 2011, "Implementation of Heuristic Reasoning to Recognize Orthogonal and Non-Orthogonal Inner Loop Features From Boundary Representation (B-Reps) Parts," *Jurnal Mekanikal*, (33), pp. 1-14.

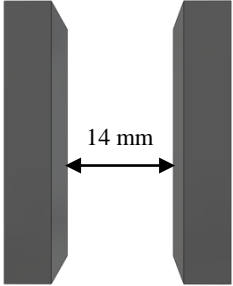
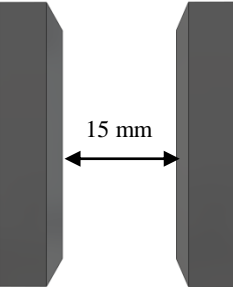
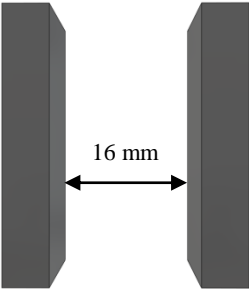
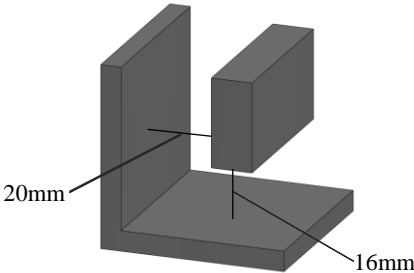
- [35] Anandan S., 2008, "Similarity Metrics Applied to Graph Based Design Model Authoring."
- [36] Zhang H. L., Van der Velden C., Yu X., Bil C., Jones T., and Fieldhouse I., 2009, "Developing a rule engine for Automated Feature Recognition from CAD models," 2009 35th Annual Conference of IEEE Industrial Electronics, pp. 3925-3930.
- [37] Bespalov D., Regli W. C., and Shokoufandeh A., 2003, "Reeb Graph Based Shape Retrieval for CAD."
- [38] Tate S. J., Jared G. E. M., Brown N. J., Swift K. G., and Cad A.-oriented, 2000, "An Introduction to the Designers' Sandpit," Proceedings of DFM 2000 Design for Manufacturing, Baltimore.
- [39] Summers J. D., Bettig B., and Shah J. J., 2004, "The Design Exemplar: A New Data Structure for Embodiment Design Automation," Journal of Mechanical Design, **126**(5), p. 775.
- [40] Shah J. J., and Mantyla M., 1995, Parametric and Feature Based CAD/CAM: Concepts, Techniques, and Applications, New York.
- [41] Joshi S., and Chang T. C., 1988, "Graph-based Heuristics for Recognition of Machined Features from a 3D Solid Model," Computer-Aided Design, **20**(2), pp. 58-66.
- [42] Venuvinod P. K., and Wong S. Y., 1995, "A graph-based expert system approach to geometric feature recognition," Journal of Intelligent manufacturing, **6**(3), pp. 155-162.
- [43] Baumgart B. G., 1972, Winged Edge Polyhedron Representation.
- [44] Floriani De L., 1987, "A Graph Based Approach To Object Feature Recognition," SCG '87 Proceedings of the third annual symposium on Computational geometry, pp. 100 - 109.
- [45] Marefat M., 1990, "Geometric reasoning for recognition of three-dimensional object features," IEEE Transactions on Pattern Analysis and Machine Intelligence, **12**(10), p. 949.
- [46] Qamhiyah A. Z., Venter R. D., and Benhabib B., 1996, "Geometric Reasoning for the Extraction of Form Features," Computer-Aided Design, **28**(11), pp. 887-903.

- [47] Hilaga M., "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," See note at the title.
- [48] Regli W. C., Gupta S. K., and Nau D. S., "Extracting alternative machining features: An algorithmic approach."
- [49] Sommerville M. G. L., Clark D. E. R., and Corney J. R., 2001, "Viewer-centered geometric feature recognition," *Journal of Intelligent Manufacturing*, **12**(4), pp. 359-375.
- [50] Woo Y., and Sakurai H., 2002, "Recognition of maximal features by volume decomposition," *Computer-Aided Design*, **34**(3), pp. 195-207.
- [51] Sakurai H., 1995, "Volume decomposition and feature recognition : Part 1 - polyhedral objects," *Computer-Aided Design*, **27**(11), pp. 833-843.
- [52] Lu Y., Gadh R., and Tautges T. J., 2001, "Feature Based Hex Meshing Methodology: Feature Recognition and Volume Decomposition," *Computer-Aided Design*, **33**(3), pp. 221-232.
- [53] Woo T., 1982, "Feature Extraction by Volume Decomposition," *Proc. Conf. on CAD/CAM Technology in Mechanical Engineering*, Cambridge, pp. 76-94.
- [54] Kim Y. S., 1992, "Recognition of form features using convex decomposition," *Computer-Aided Design*, **24**(9), pp. 461-476.
- [55] Kim Y. S., 1990, "A Convergent Convex Decomposition of Polyhedral Objects," *Journal of Mechanical Design*, **114**(3), p. 468.
- [56] Wang E., and Kim, 1999, "Feature-based assembly mating reasoning," *Journal of Manufacturing Systems*, **18**(3), pp. 187-202.
- [57] Waco D. L., and Kim Y. S., 1994, "Geometric reasoning for machining features using convex decomposition," *Computer-Aided Design*, **26**(6), pp. 477-489.
- [58] Wang E., and Kim Y. S., 1998, "Form feature recognition using convex decomposition: results presented at the 1997 ASME CIE Feature Panel Session," *Computer-Aided Design*, **30**(13), pp. 983-989.
- [59] Kriegel H.-peter, Kr P., and Seidl T., 2003, "Effective Similarity Search on Voxelized CAD Objects," *Database Systems for Advanced Applications*, pp. 27-36.

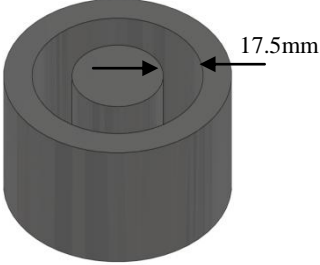
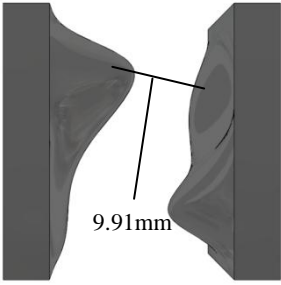
- [60] Tiwari S., 2009, "Development and Integration of Geometric and Optimization Algorithms for Packing and Layout Design."
- [61] Sakurai H., and Chin C., 1994, "Definition and Recognition of Volume Features for Process Planning," *Advances in Feature Based Manufacturing*, pp. 65-80.
- [62] Li W. D., Ong S. K., and Nee A. Y. C., 2002, "Recognizing manufacturing features from a design-by-feature model," *Computer-Aided Design*, **34**(11), pp. 849-868.
- [63] Shen Y.-te, and Shah J. J., 1994, "Feature recognition by volume decomposition using half-space partitioning," *ASME Computers in Engineering*.
- [64] Trika S. N., and Kashyap R. L., 1993, "Geometric Reasoning for Extraction of Manufacturing Features in Isooriented Polyhedrans," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16**(11), pp. 1087-1100.
- [65] Ding L., and Yue Y., 2004, "Novel ANN-based feature recognition incorporating design by features," *Computers in Industry*, **55**(2), pp. 197-222.
- [66] Garey M. R., and Johnson D. S., 1979, "Computers and Intractability: A Guide to the Theory of NP-Completeness."
- [67] Simon H. A., 1970, *The Sciences of the Artificial*, M.I.T Press, Massachusetts.
- [68] Wiegers K. E., 2003, *Software Requirements*, Microsoft Press.
- [69] Stellmann A., and Greene J., 2006, *Applied Software Project Management*, O'Reilly Media, Inc., North Sebastopol.
- [70] 1990, "IEEE Standard Glossary of Software Engineering Terminology."
- [71] Ulrich K. T., and Eppinger S. D., 1995, *Product Design and Development*, McGraw-Hill, Inc.
- [72] Help 2010 S. A., 2010, "<http://help.solidworks.com/2010/English/api/sldworksapiproguide/Welcome.htm>,"  
<http://help.solidworks.com/2010/English/api/sldworksapiproguide/Welcome.htm>.
- [73] Ding S., Mannan M. a., and Poo a. N., 2004, "Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces," *Computer-Aided Design*, **36**(13), pp. 1281-1294.

## Appendices

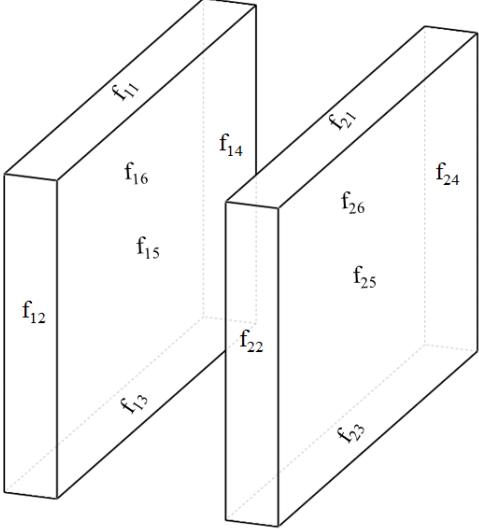
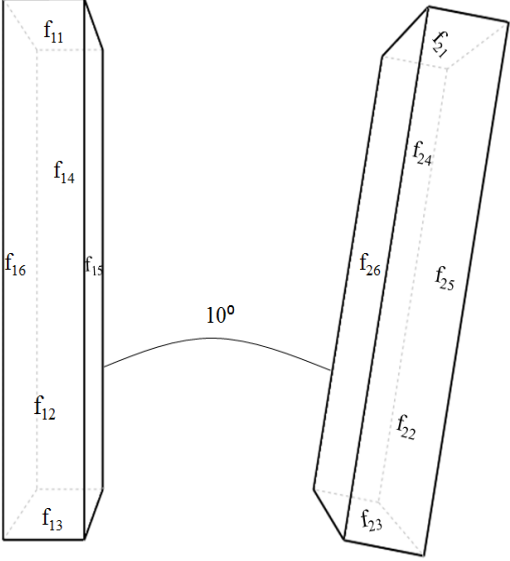
## Appendix A: Test Cases to Check Threshold Distance Condition

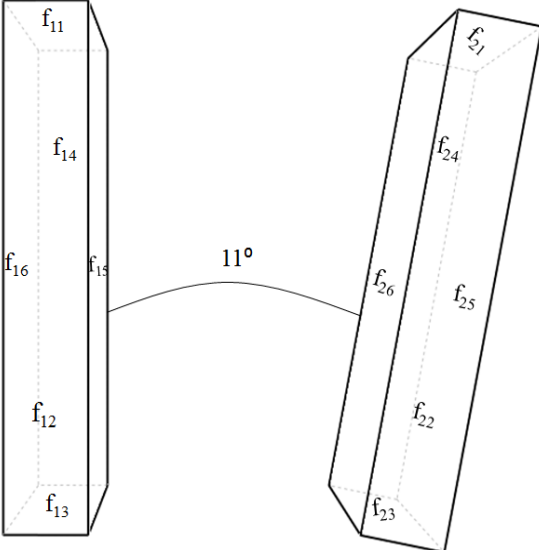
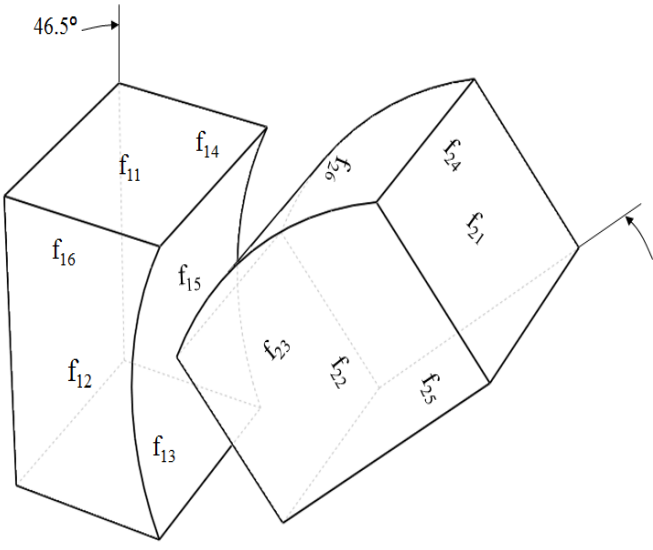
Test Case No.	Test Case	Description
1		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be less than the threshold distance and stores the two parts in a container for orientation check.</p>
2		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be equal to the threshold distance and stores the two parts in a container for orientation check.</p>
3		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts to be more than the threshold distance and therefore discards this part pair from further analysis.</p>
4		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two parts as less than threshold distance. But the distance between the geometries from the two parts is greater than the threshold distance.</p>

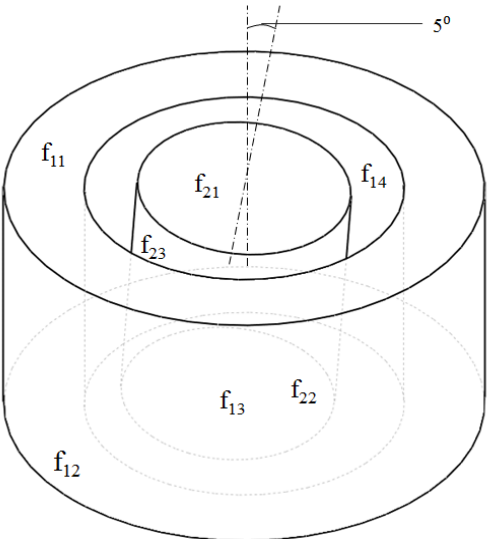
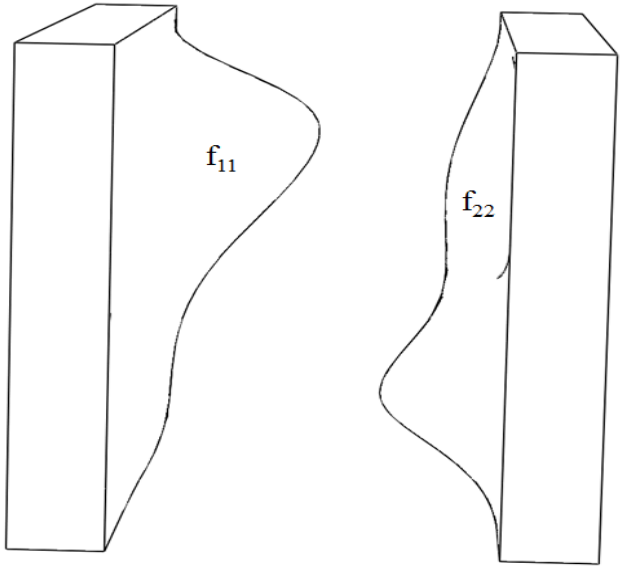


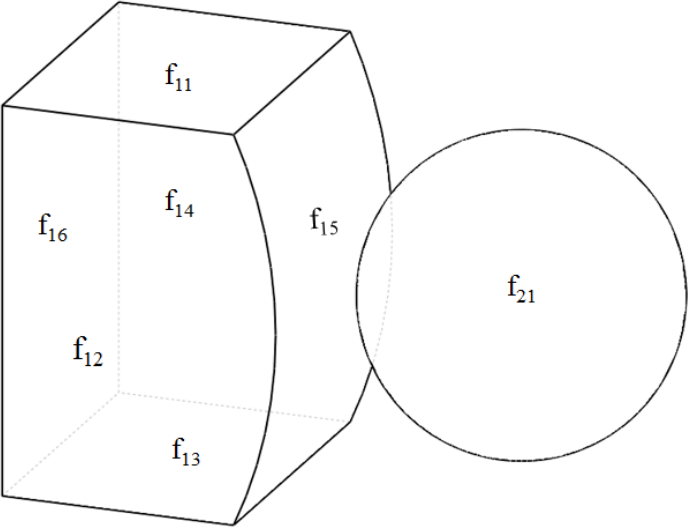
5		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two cylinders as less than threshold distance. But the distance between the inner cylinder and the inner diameter of the outer cylinder is greater than the threshold distance (17.5mm).</p>
6		<p><b>Input:</b> User-defined threshold distance = 15mm</p> <p><b>Output:</b> The algorithm calculates the distance between the two freeform geometries as less than the threshold distance. This test case is used to test the bounding box performance for freeform geometric type.</p>

## Appendix B: Test Cases to Check Orientation Condition

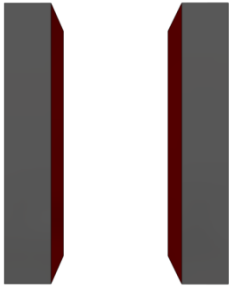
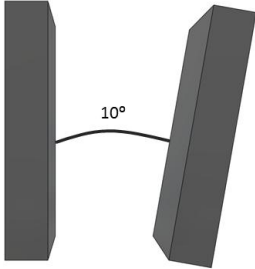
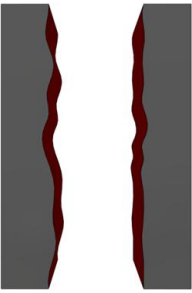
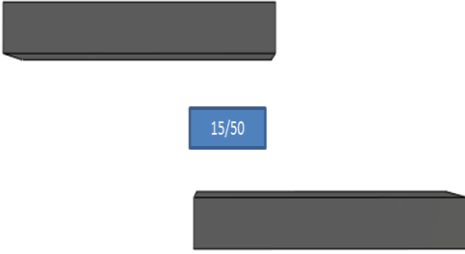
Test Case No.	Test Case	Description
1	<p>Parallel</p> 	<p><b>Input:</b>User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b>The algorithm evaluated the following pair of faces as being opposed to each other within the user defined orientation angle,</p> <ol style="list-style-type: none"> <li>7. <math>f_{11}</math>-<math>f_{23}</math></li> <li>8. <math>f_{12}</math>-<math>f_{24}</math></li> <li>9. <math>f_{13}</math>-<math>f_{21}</math></li> <li>10. <math>f_{14}</math>-<math>f_{22}</math></li> <li>11. <math>f_{16}</math>-<math>f_{25}</math></li> <li>12. <math>f_{15}</math>-<math>f_{26}</math></li> </ol>
2		<p><b>Input:</b>User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b>The algorithm evaluated the following pair of faces as being opposed to each other within the user defined orientation angle,</p> <ol style="list-style-type: none"> <li>7. <math>f_{11}</math>-<math>f_{23}</math></li> <li>8. <math>f_{12}</math>-<math>f_{24}</math></li> <li>9. <math>f_{13}</math>-<math>f_{21}</math></li> <li>10. <math>f_{14}</math>-<math>f_{22}</math></li> <li>11. <math>f_{16}</math>-<math>f_{25}</math></li> <li>12. <math>f_{15}</math>-<math>f_{26}</math></li> </ol>

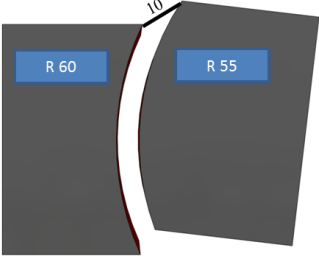
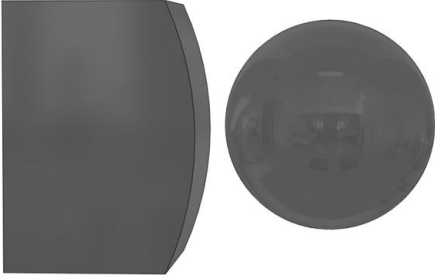
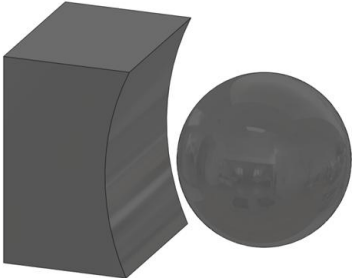
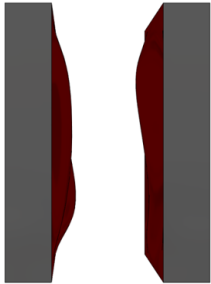
3		<p><b>Input:</b>User-defined orientation = <math>180^{\circ} \pm 10^{\circ}</math></p> <p><b>Output:</b>The algorithm evaluated the following pair of faces as being opposed to each other outside the user defined orientation angle. None of the faces meet the orientation condition.</p>
4		<p><b>Input:</b>User-defined orientation = <math>180^{\circ} \pm 10^{\circ}</math></p> <p><b>Output:</b>The algorithm evaluated three pair of faces as being opposed to each other within the user defined orientation angle. The face pairs identified were,</p> <ol style="list-style-type: none"> <li>4. <math>f_{12}</math>-<math>f_{24}</math></li> <li>5. <math>f_{14}</math>-<math>f_{22}</math></li> <li>6. <math>f_{15}</math>-<math>f_{26}</math></li> </ol> <p>The angles between other face pairs are greater than the user-defined orientation angle.</p>

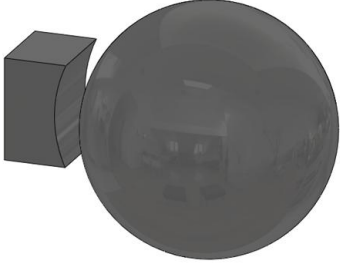
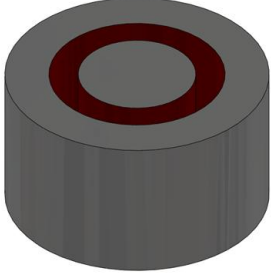
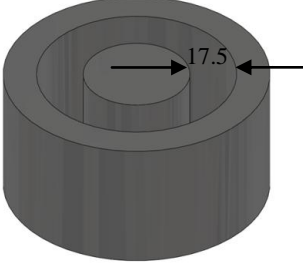
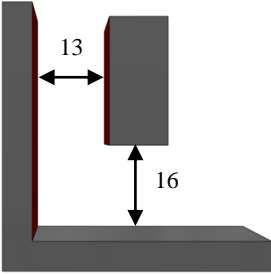
5		<p><b>Input:</b>User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b>The algorithm evaluated four pair of faces as being opposed to each other within the user defined orientation angle. The face pairs identified were,</p> <ol style="list-style-type: none"> <li>1. <math>f_{11}</math>-<math>f_{22}</math></li> <li>2. <math>f_{12}</math>-<math>f_{21}</math></li> <li>3. <math>f_{13}</math>-<math>f_{23}</math></li> <li>4. <math>f_{14}</math>-<math>f_{23}</math></li> </ol> <p>The angles between other combinations of face pairs are greater than the user-defined orientation angle.</p>
6		<p><b>Input:</b>User-defined orientation = <math>180^0 \pm 10^0</math></p> <p><b>Output:</b>This test case is used to check the algorithm's performance in checking orientation for freeform surfaces.</p> <p>Some portions of the faces <math>f_{11}</math>-<math>f_{22}</math> have orientation within the angle tolerance and the algorithm identifies that and stores the two faces for percentage similarity analysis.</p>

7	 <p>The diagram illustrates a 3D object with six faces labeled <math>f_{11}</math>, <math>f_{12}</math>, <math>f_{13}</math>, <math>f_{14}</math>, <math>f_{15}</math>, and <math>f_{16}</math>. The object is a rectangular prism with a curved side. A circular face labeled <math>f_{21}</math> is shown to the right of the object, representing a spherical body. The faces <math>f_{11}</math>, <math>f_{12}</math>, <math>f_{13}</math>, <math>f_{14}</math>, <math>f_{15}</math>, and <math>f_{16}</math> are the faces of the rectangular prism, and <math>f_{21}</math> is the face of the spherical body.</p>	<p><b>Input:</b>User-defined orientation = <math>180^\circ \pm 10^\circ</math></p> <p><b>Output:</b>The algorithm was checked using a spherical body and combination of planar and curved surfaces. The face pairs meeting the orientation condition that were identified are:</p> <ol style="list-style-type: none"> <li>1. <math>f_{11}</math>-<math>f_{21}</math></li> <li>2. <math>f_{12}</math>-<math>f_{21}</math></li> <li>3. <math>f_{13}</math>-<math>f_{21}</math></li> <li>4. <math>f_{14}</math>-<math>f_{21}</math></li> <li>5. <math>f_{16}</math>-<math>f_{21}</math></li> <li>6. <math>f_{15}</math>-<math>f_{21}</math></li> </ol>
---	--	---

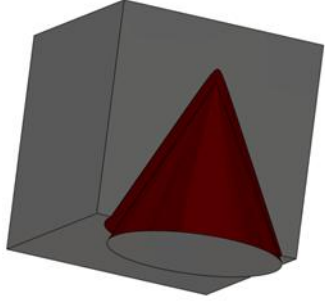
### Appendix C: Test-Cases to Check for Percentage Similarity between Geometries

Test Case No.	Test Case	Description
1		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> Duplicate geometry shown in red
2		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> No duplicate geometries
3		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> Duplicate geometry shown in red
4		<b>User input:</b> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <b>Result:</b> No duplicate geometries

5		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>
6		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
7		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
8		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>

9		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
10		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>
11		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> No duplicate geometries</p>
12		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Face pair that meets threshold distance and percentage similarity condition are highlighted in red as duplicate geometries.</p>



13		<p><b>User input:</b></p> <ol style="list-style-type: none"> <li>1. Threshold distance = 15mm</li> <li>2. Orientation = <math>180^0 \pm 10^0</math></li> <li>3. Percentage similarity = 80%</li> <li>4. Tolerance bound = 2mm</li> </ol> <p><b>Result:</b> Duplicate geometry shown in red</p>
----	---	--