12-2007

A CASE STUDY ON THE USE OF DESIGN EXEMPLAR AS A SEARCH AND RETRIEVAL TOOL

Narasimha murty Srirangam
Clemson University, nsriran@clemson.edu

A CASE STUDY ON THE USE OF THE DESIGN EXEMPLAR AS A SEARCH AND RETRIEVAL TOOL

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
Murty Srirangam
December 2007

Accepted by:
Dr. Joshua Summers, Committee Chair
Dr. Gregory Mocko
Dr. Gang Li

ABSTRACT

This thesis presents a case study examining the suitability of design exemplar technology as a CAD query tool for an industrial scenario. The search and retrieval of geometrically similar mold inserts to save tooling cost for a tire manufacturing company is taken as the case study. During the implementation of the design exemplar as a CAD search and retrieval tool, several limitations of it are identified, such as the difficulty and tediousness in authoring exemplars for real world problems. To overcome these limitations, a new mathematical-based exemplar approach (mathematical model) is developed for tire mold insert retrieval. This approach calculates a set of maxima and minima based on the specifications of the target mold insert and identifies similar molds that fall within these specifications. Though the mathematical model requires less effort to author the exemplar queries than the initial boundary envelope approach, it has an unreasonably high time complexity when implemented using design exemplar. A partially developed mathematical-based exemplar takes 30 seconds to run through a sample database of 10 mold inserts. Assuming that a fully built exemplar would take more time when run on a database of 55000 mold inserts as it would have more number of constraints added to it, the approach was not implemented through design exemplar tool. However, as this exemplar approach gives a satisfactory theoretical solution to the problem, the exemplar is hard-coded in an independent C++ program to suit the requirements at hand. The mathematical-model which is implemented in an independent C++ program successfully searches through the complete database of 5500 tire mold inserts, retrieving similar mold inserts, in a span of one second, a huge efficiency gain when compared with the traditional exemplar system. Apart from the time reduction for the search and retrieval, the exemplar inspired program has an additional advantage in that no geometrical entities must be handled to build exemplars. However, the mathematical model may result in several false positives that must be manually eliminated. In an experiment, 15 mold inserts were randomly selected from the database and were searched for similar mold inserts using the program developed. In the experiment, it was observed that there were no false negatives and the number of false positive ranged from 0 – 2. Considering the size of the database which is roughly 5500 mold inserts, the mathematical still provides advantages in search and retrieval. It is still good because 5500 have been

reduced on an average to 9. During the experiment, it as been found that 80 percent of the mold inserts which were tested have similar mold inserts. The design exemplar tool could have helped company to save the tooling cost of $184,000 per annum by providing them with a search and retrieval tool. However, it was not implemented because of some of the limitations mentioned above. Still the strategies of design exemplar are valid for querying CAD models. This is proved by the fact that the mathematical model which is an essentially an exemplar can successfully finds and retrieves similar mold inserts.

DEDICATION

I dedicate this work to my parents whom I love the most.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Joshua D Summers, for his constant support and guidance. Without him as a driving force, the work wouldn't have taken the present form. I appreciate his patience in handling students like me. I would also like to express my gratitude to the members of my committee, Dr. Gregory Mocko and Dr. Gang Li for their support and advice, particularly in the situations when I am in a state of unrest and confusion which most of the graduate students face. I would also like to thank Dr. Farhad Ameri who helped at various stages of this project and thesis.

I am thankful to my friends and colleagues in AID and CREDO lab, especially Srinivasan Anandan, Shashidhar Putti, Sudhakar Teegavarapu, Ajay Nowlay, Raji Raxevier, Min Wang, Madhusudhan Kayyar and Siva Chavali for providing a stimulating and fun environment in which I could learn, work and grow. I would also like to thank, my room mates Pavan Krishna Seemakurthy, Suman Velvaluri, and Atul Kale who have made my stay at Clemson memorable.

I specially thank my family members Kiran Mayi Srirangam, Ravi Shankar Bulusu and Jyothirmayi Srirangam for their continuous support.

Above all, I would like to thank my parents who made all this possible.

TABLE OF CONTENTS

(Table of Contents continued)

Page

LIST OF TABLES

LIST OF FIGURES

(List of Figures continued)

Figure Page

(List of Figures continued)

Figure Page

CHAPTER 1

INTRODUCTION AND MOTIVATION

Introduction

In engineering design, it is a common practice to develop a solution set to the problems at hand by comparing them with known 'similar' design problems and modifying them to satisfy the new design requirements [1]. As this saves the designer a considerable amount of time and effort when compared to working from scratch, the concept of similarity is applied in several product design and manufacturing scenarios. This is evident from the fact that a designer spends about 60% of the time searching for information that is relevant to the given design problem [2]. Also, it has been found that about 75% of the design activity consists of reusing existing data to address the design problem [3]. Some of the different areas in which application of similarity can improve product design and manufacturing are cost estimation, product platform development, and part reuse [4].

Cost Estimation of Machined Part:

In the modern computer era where the design data is maintained digitally, many companies allow clients to submit the 3D models of the part they wish to produce over the web to get a manufacturing cost quotation. Some of the companies, which fall into this category, are Mfg Quote[1], Job Shop[2], and Global Spec[3]. The time to estimate the production cost of the submitted product can be greatly reduced as compared to the manual estimation [5], if a similar product that has been previously manufactured can be retrieved from the database and its cost is recalculated according to new specifications.

---

[1] http://www.mfgquote.com/http://www.mfgquote.com/

[2] http://www.jobshop.com/

[3] http://www.globalspec.com/

Part Family Formation:

In manufacturing, the setup time and cost while switching between different products is considerable. This could be reduced significantly, if parts of similar shape are grouped together to share common tools and setups[6].

Reuse of Previously Designed Parts:

In a company that has a database of previously designed parts, a designer may find it convenient and time saving to reuse designs from the database to create new product. For example, a designer in a brake manufacturing company may find that the brake shoes of previously designed brakes can be used for the present design with small or even no modifications.

In all the cases mentioned above, finding similar parts is motivated by the objective of saving time and/or money. Especially with huge data generated due to wide spread use of CAD systems in engineering design, designers prefer to reuse the CAD data as this would help them to channel their time and efforts to create better designs rather than spending their time in generating CAD models. Reusing previously designed components while designing a new part can save the company considerable time and money [4]. This is the general motivation of this case study.

Motivation

To create treads on a tire, a manufacturer uses mold inserts which it in turn must produce. A typical mold insert is shown in the Figure 1.1. It is a metallic piece is stamped and bent to obtain the required shape. The mold insert is then inserted into a mold during the manufacturing process of a tire so that it leaves its impression and thus creating tread on the tire.

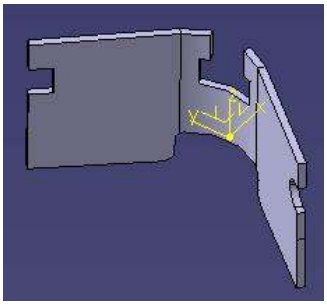


Figure 1.1 Typical mold insert manufactured by the company

When a mold insert is inserted into a tire to create tread on it, the tread on the tire takes the shape of the top-view of the mold insert. The top view of the mold insert for the present case is a line-arc-line profile as it can be seen from the

Figure 1.2. The line-arc-line profile of the mold insert is explained as a line followed by an arc which is tangential to the line. The arc in turn is followed by a line which is tangential to the arc itself. Mold inserts manufactured to create treading on a tire may consist up to 18 such line arc lines as their top profiles. An example of treading on the tire can be seen in Figure 1.3. In the figure, a mold insert is inserted into a tire to create treading on it. The treading takes profile of the top view of the mold insert which is of line-arc-line shape.

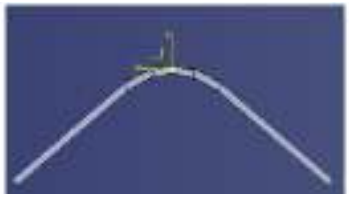


Figure 1.2 Top view of the Mold insert which is a line-arc-line profile

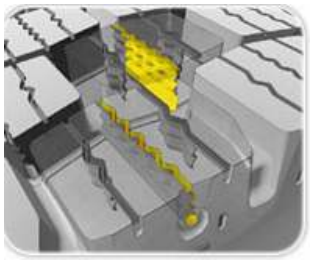


Figure 1.3 Treading made of mold inserts with multiple line-arc-line profile

The tooling cost for each type of mold insert is approximately $2,000 and each tire would need three to five different kinds of such mold inserts. Hence the tooling cost of the mold inserts for a newly designed tire would be around $6,000 to $10,000. When considering the different varieties of tires the company manufactures, this amount is huge and so the company wants to reduce cost. As the company has been manufacturing tires for a few years and has a database of mold insert designs, (CAD models and the specifications their line-arc-line profiles such as the length of the line segment, angle subtended by the line

segment, radius of the arc that lies between the two line segments, length of the other line segment in the line-arc-line profile and the angle subtended by it, that have been used previously, the tooling cost can be reduced by finding and replacing an adequately "*similar mold insert*" from their database in place of the mold insert designed for their newly designed tire. The manufacturer defines an adequately "similar mold insert" based on the tolerance envelope. The concept of similarity based on tolerance envelope is explained with an example. Consider a typical mold insert (green color) as shown in

Figure1.4 (a). This mold insert is initially designed to create treading on a newly designed tire and is called target mold insert. To find a "similar" mold insert for this target mold insert, a tolerance envelope is drawn around the target envelope which is shown in blue color in the figure, the tolerance being defined by the designer. A mold insert that falls within this tolerance envelope is considered to be a similar mold insert.

Figure1.4 (b) shows a similar mold insert that fits in the tolerance envelope.

Figure1.4 (c) shows a mismatch of the mold inserts for the tolerance envelope retrieved from the database. The mold insert is a mismatch since it does not fit within the target envelope. Once a similar mold insert is retrieved from the database, it is used instead of the target mold insert, thus saving the tooling cost.

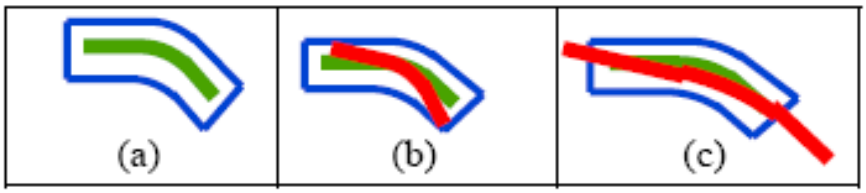


Figure1.4 (a) Target Mold Insert (b) Match from the database (c) Mismatch

Figure1.5 shows the ideal procedure which the designers would like to follow after they have designed a new mold insert for a newly designed tire. The first step while designing a new mold insert is to prepare the sketch of the new mold insert. Once the sketch is prepared, the database is searched for the existence of a similar mold insert. If the database consists of the similar mold insert, then the designed mold insert is replaced with the existing similar mold insert. If no similar mold inserts exist in the database, then the newly designed mold insert is manufactured.

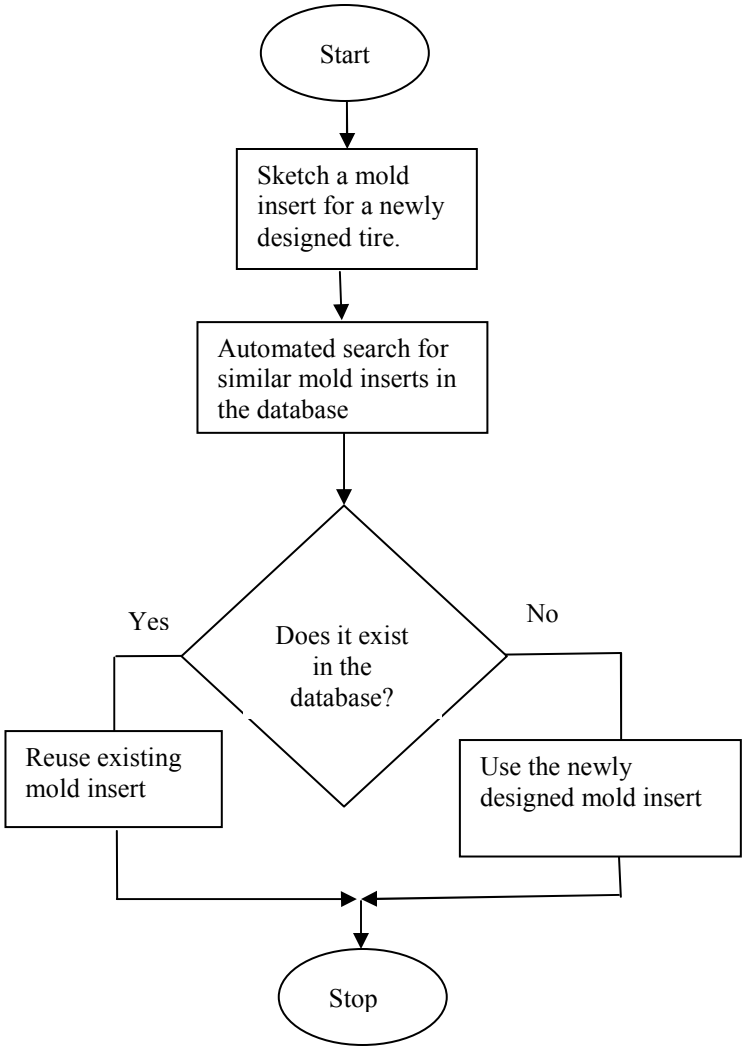


Figure1.5 Flow chart showing the process of using a newly designed mold insert

Without an automated search and retrieval procedure, the database must be navigated manually to find similar mold inserts. The manual search and retrieval process has two primary disadvantages. The database consists of around 5500 mold inserts. Hence, it is a tedious and time consuming job to search the entire database for similar mold inserts manually. As well, due to human error, there are many chances that the designer may miss a potential candidate during the search process. This error may cost several thousands of dollars to the manufacturer, as they may have to manufacture a new mold insert. Due to these limitations, the company prefers automating the search and retrieval process. A potential challenge to automate the search and retrieval process of mold inserts is absence of algorithms that deal with this type of

geometric similarity. In the present case, all the mold inserts are of line-arc-line models or extended line arc line models which are almost of the same shape and differ only in dimensions. In order to search and retrieve similar mold inserts to cut down the expenses on tooling cost, it is necessary provide a search and retrieve algorithm that can deal with geometric similarity of this kind. The design exemplar is proposed as a solution to the problem.

The design exemplar is a CAD query language that can be used to search and retrieve geometric models. In the literature, the design exemplar has been presented an effective concept that has the qualifications of a general query tool. While there has been much research and development to extend the exemplar capabilities, there is insufficient evidence regarding its applicability to real world problems. This research aims at investigating the possibility of using the design exemplar system for industrial applications such as search and retrieval of mold inserts.. This thesis discusses the limitations that the design exemplar, which prevented its implementation in the traditional format as a tool to search and retrieve mold, inserts while presenting an algorithm based on the design exemplar that can be used for the retrieval of similar mold inserts efficiently.

Problem Statement

This thesis presents a case study in building a mold insert retrieval system based on exemplar technology to support mold insert design. In this case study, the applicability of the design exemplar as a CAD query tool to find and retrieve potential candidates for a given target mold insert is studied. The limitations of the exemplar are identified and alternative methods are explored. Briefly, this research seeks to answer the question:

Q1: Can design exemplar be implemented to find potential candidates of mold inserts for a given target mold insert?

Hypothesis: Yes, design exemplar can be used for search and retrieval of mold inserts as there is sufficient evidence from the literature that it is a concept upon which a commercial CAD query tool can be built. Also, it has been used for purposes such as feature recognition in the field of academia. Due to the

reasons mentioned, it has been assumed that the design exemplar tool can be used for the search and retrieval process of similar mold inserts.

Answering this research question is the goal of this thesis. The remaining part of the thesis is organized in the following way. In chapter two, a brief literature review on the design exemplar has been presented. In chapter three, the proposed approach of using the design exemplar as a search and retrieval tool is presented. Limitations in using this approach are discussed. In chapter four, an alternative approach that is adapted is explained in detail underlining its advantages and limitations. Chapter five discusses other algorithms developed for the same application which is followed by a section on conclusions and future work.

CHAPTER 2

DESIGN EXEMPLAR

Introduction to Design Exemplar

The design exemplar is a data structure used to represent design data, with an integral generic constraint solving algorithm that facilitates querying, solving, and modification of design data represented in a geometric model [7]. It provides a standard representation of the mechanical engineering design knowledge for representing topological and geometric design problems based on a canonically derived set of entities and relationships [8]. The design exemplar uses a bipartite graph representation to model design data. As the name bipartite representation implies, an exemplar consists of two groups of graph nodes. The entities that build the model form one group while the relationships between these entities form the other group. A node from one group is connected to the node of the other group through edges. If the edges are represented in dotted lines, it indicates that this relation is implicitly stated in the model. If the relations are explicitly stated in the model, such as boundary constraints, nodes are connected through a solid line. Two entities can be connected only through a relational node as two nodes of same group cannot be connected. For example, the exemplar representation of a model consisting of a pair of parallel lines shown in Figure 2.1 is presented in Figure 2.2. In the exemplar shown in Figure 2.2, the pair of lines, Line1 and Line2, are represented with two nodes.

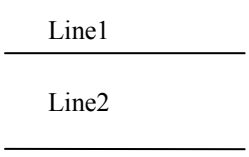


Figure 2.1 Model

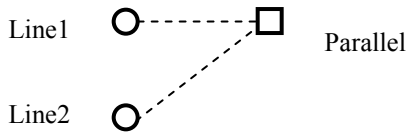
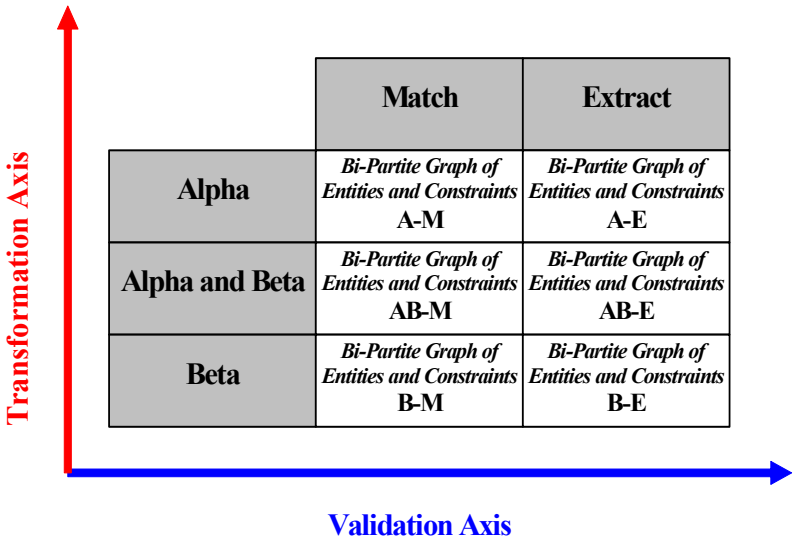


Figure 2.2 Exemplar Representation

These nodes are grouped together and are connected to the parallel relation implying that the two lines are parallel. The edges are represented with dotted lines implying that the parallel relation is implicit.To facilitate querying of the design data in a model, exemplar supports features called *match* and *extract* [9]. The match portion supports the data that is explicit and is being interrogated in the model by the user where as the extract portion supports and evaluates the implicit data represented in the model [9]. Thus, the extract part is the conditional part that holds the relations which should be satisfied by the match

part. In this way it facilitates reasoning about the match part of the exemplar. The modification capability of design data of the design exemplar is due to its ability to support alpha and beta states [10]. The alpha state represents the constraints and entities that exist in the model before modification, while the beta state represents entities and relations that exist in the model after modification. Thus, modification means transformation of entities and constraints from alpha state to beta state[9]. If there are entities that exist in a model both before and after modification, then they are said to be in alpha-beta state. The alpha and beta states lie on the transformation axis of the design exemplar. These fundamental definitions are better explained using an illustration.



	Match	Extract
Alpha	*Bi-Partite Graph of Entities and Constraints* **A-M**	*Bi-Partite Graph of Entities and Constraints* **A-E**
Alpha and Beta	*Bi-Partite Graph of Entities and Constraints* **AB-M**	*Bi-Partite Graph of Entities and Constraints* **AB-E**
Beta	*Bi-Partite Graph of Entities and Constraints* **B-M**	*Bi-Partite Graph of Entities and Constraints* **B-E**

Figure 2.3 Components of the design exemplar [7]

An example has been explained to illustrate the working of the exemplar algorithm and the match/extract features of the exemplar. Consider a model which consists of three lines as shown in the Figure 2.4.

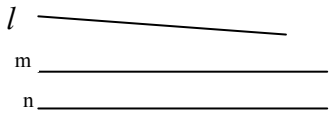


Figure 2.4 Model representing 3 lines

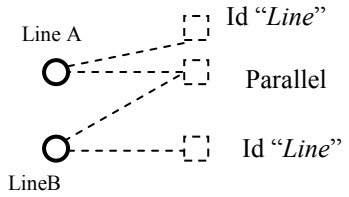
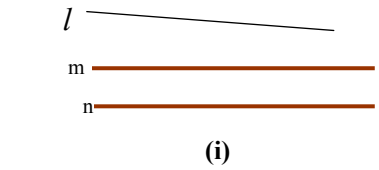
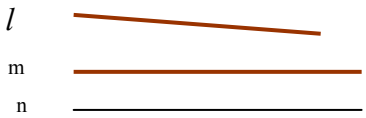


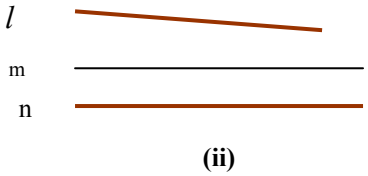
Figure 2.5 Exemplar to find parallel lines



(i)



(iii)



(ii)

(iv) Final result displayed to the user

Figure 2.6 Steps (i), (ii), (iii) and (iv) illustrate the working of design exemplar algorithm [7]

An exemplar authored to find if a pair of parallel lines exist in the model, is shown in Figure 2.5. In the exemplar the explicit design data that the user is investigating in the model is a pair of lines which is represented by two circles Line A and Line B. The circles are in solid lines indicating that the data is explicit. The implicit data of the model that the lines must be parallel is shown in dotted lines. The id tags (*id "line"*) are useful in highlighting the identified pair of parallel lines in the model. When the model shown in Figure 2.4 is queried with the exemplar shown in Figure 2.5, the design exemplar algorithm first identifies the match part of the model. In the current exemplar, a pair of lines form the match part. Since there are six pairs of lines present in the model (though there are three pairs of lines, the design exemplar algorithm identifies a pair of lines l, n and n, l as two different sets) these are recognized in the model as the first step. Steps (i), (ii), and (iii) of Figure 2.6 show possible matches in thick lines when the model has been queried with the exemplar to find a pair of parallel lines. Secondly, each of these pair of lines is checked for the parallelism as a constraint. Since only two sets of parallel lines exist which are (m, n) and (n, m), these sets will be returned and are highlighted at the end. This example is further

extended to demonstrate modification of the design data. The capability of the design exemplar to modify the design data is by the virtue of its ability to support alpha and beta states [7]. Consider the model shown in the Figure 2.4. It is required modify the lengths of the line segments to a specific value of 35mm if they are parallel. Modification requires identification of the relations which exist before and after modification. In the above example, as the line segments and the angles between them remains same before and after modification, they exist in both alpha and beta states. To modify the model, a condition on the length parameter is introduced. Since the condition of equal lengths, exists only after the modification, it exists only beta states. The modified model is presented in Figure 2.7 and the exemplar used for this purpose is shown in the Figure 2.8 .

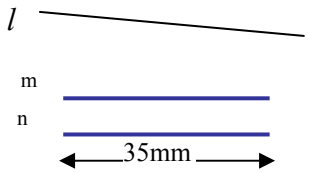


Figure 2.7 Modified model

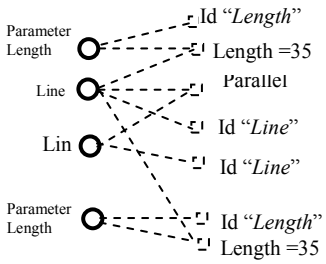


Figure 2.8 Exemplar to find a pair of parallel lines and modify their length

In Figure 2.7 the model obtained after modification process is presented. For better understanding, the features that are present in both alpha and beta states are shown in black color and the features that are present only after modification are present in blue color. This modifying capability of the design exemplar by using alpha and beta representations is used to add or delete features to a design model. The algorithms for transformation and validation of the design models are explained in [7]. These algorithms support the alpha, beta, alpha_beta changes with respect to the transformation axis as shown in the Figure 2.3.

To improve the querying capabilities of the design exemplar, it has been provided with tools such as exemplar networks and logical connectives. Logical connectives [11] provide a vocabulary which can be used to author complex exemplar queries and static networks [12] help in authoring exemplars by the reusing existing exemplars. These tools are discussed briefly in the following sections.

Exemplar Networks:

Authoring design exemplars to represent generalized characteristics of design models is a tedious task [7]. To reduce the effort needed to author complex exemplars and as a step towards providing a new level of abstraction to reuse existing exemplars [11], Summers has developed exemplar networks. Exemplar Networks provide a way to reuse the existing exemplars to reduce the effort required to develop new exemplars. Three types of reuse were identified which are: inheritance, composition, and *ad-hoc* modification. In inheritance reuse, existing exemplars are used as a base and are extended to serve additional functionalities by adding new entities and relationships. Linking some of the existing exemplars to form a new exemplar is called composition. *Ad-hoc* modification is the reuse of existing exemplars by changing their existing values. Thus, development of static exemplar networks helps in overcoming a limitation of reusability of existing exemplars by integrating them into other exemplars to accommodate the inclusion of new characteristics.

Logical Connectives:

Divekar has proposed the design exemplar as a concept upon which a CAD query language could be built [10]. As a part of his research, the components and tasks of a standard de-facto query language were studied. A list of the qualifications of a textual query language is found in Table 2-1. These qualifications include its ability to perform tasks such as data retrieval, data addition, data update and data deletion, and data types such as logical connectives and predicates as its components to form queries is made which are presented.

Table 2-1 Qualification of a Query Language[12]

Components	Data Types
	Predicates
	Logical Connectives
Tasks	Retrieval
	Addition
	Update
	Delete

In addition to these qualifications, a CAD query language should also satisfy additional requirements that allow it to handle spatial data efficiently. Graphical display of the information sought and its context, dynamic interaction of the results with the previous queries, selection of references for upcoming queries, labels are some of the requirements. A list of requirements that should be satisfied by a CAD query language are presented in Table 2-2.

Table 2-2 Requirements of a Spatial Query Language[12]

Spatial Data Requirements	Graphical Display
	Independence of Spatial data from coding
	Dynamic interaction of the queries and results.
	Labels
	Selection of results as references for future, through pointing or direct selection.
	Display Context

The capabilities of the design exemplar are compared with the de-facto query language and the requirements that a CAD query language in Table 2-3. In Table 2-3 the various tasks performed by the design exemplar are listed and a comparison is drawn relating it to the SQL. It was found that the design exemplar has most of the qualifications that SQL has. Also, when analyzed, the design exemplar complies with the most of the requirements that a CAD query language have to handle spatial data. The analysis done is presented in

Table 2-4. Hence, it was concluded that the design exemplar is concept upon which a commercial CAD query language can be built.

Table 2-3 Query Language Qualifications Vs Design Exemplar[12]

	Qualifications of a Query Language	Design exemplar
Components	Data-Types	Real parameter, Integer parameter, Vector, Rotation Matrix (Algebraic), Point, Direction, Plane, Line, Circle Ellipse, Cylinder, Sphere (Geometric), Solid Volume (Topological), Form Features, Part, Assembly (Semantic)
	Predicates	Scalar equations, Scalar inequalities, Cross Product(Algebraic), Fixed Tables, Distance Angle, Radius, Focal Distance, Distance to resolve geometry, Control points, Knot values, Continuity conditions, In_set, Mao Coincident, Incident parallel, Right angle (Geometric) Inciden, Length, Area, Volume, Directed left of, Curve direction, Curve direction TC, Surface Normal, Surface Normal TC, Same direction (Topological)
	Logical Connectives	AND, OR, NOT, MINUS (to be implemented)
Tasks	Retrieval	Pattern Matching (Alpha/Match) Query Extraction (Alpha/Match and Alpha/Extract) Design Validation
	Modification, Addition, Deletion	Model Modification (Alpha/Match, Alpha/ Extract, Beta Match/Beta Extract)

Table 2-4 Requirements of a Spatial Query language Vs Design Exemplar[12]

Requirements of Spatial query language	**Does Exemplar Comply?**
Ability to treat spatial data at a level Independent from internal coding such as x-y Coordinates	Yes
Display results in graphical form	Yes
Combine one query results with one Or more previous queries.	Yes
Display of context in addition to the information Sought	Yes
Extended dialog allowing selection by pointing and direct selection of a result as a Reference to the upcoming query	No
Labels to aid understanding of models so that users are able to select specific instance of objects	Limited

As a step towards improving the capabilities of the design exemplar as a CAD query language, logical connectors AND, OR, NOT were implemented in the design exemplar system[9]. With the help of

these logical connectors, a designer can reuse the existing exemplars to author complex exemplars. While the design exemplar system inherently supports the AND logical connector, OR and NOT connectors are implemented in [11]. For example, in the exemplar shown in the Figure 2.9, OR logical connective is used to find a pair of planes in a model that are either parallel or perpendicular. When a model is queried using this exemplar, the pairs of planes that are either parallel or perpendicular to each other are found. As an illustration for the NOT block, Figure 2.10 shows a design exemplar that is used to find a pair of planes that are perpendicular to each other but do not intersect at a line. Since, it is required that the planes do not intersect, the coincidence constraint is included into the NOT block of the exemplar.

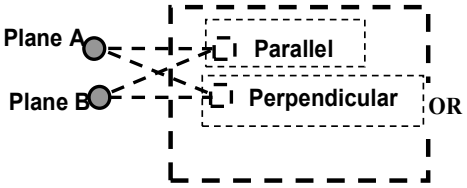


Figure 2.9 Exemplar to find a pair of planes that are either parallel or perpendicular

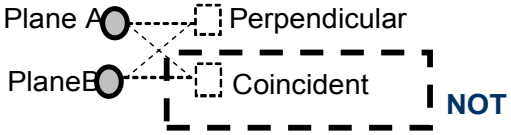


Figure 2.10 Exemplar to find a pair of pines perpendicular to each other but does not intersect.

Design exemplar as Visual Programming Language (VPL):

While the design exemplar technology was initially developed as a CAD query tool and implements it in a declarative fashion [11], a new dimension was added in [8] by implementing the procedural use of design exemplar. Further, the development of the design exemplar based visual programming language (VPL) was proposed by Putti [7]. At this point, it is identified that the design exemplar supports two of the three important components of an iconic visual programming language, namely icons, iconic system and the compiler. While, the proposed design exemplar based VPL uses visual objects (icons) to represent geometric, parametric and topologic entities and their relations, the inclusion of programming constructs for looping, conditional branching were found to important. This work introduces the "*dynamic exemplar node*" and "*dynamic network"*, two data structures to achieve procedural programming with the existing design exemplar system. However, this work does not address the development of the third important component of a visual programming language-- the compiler.

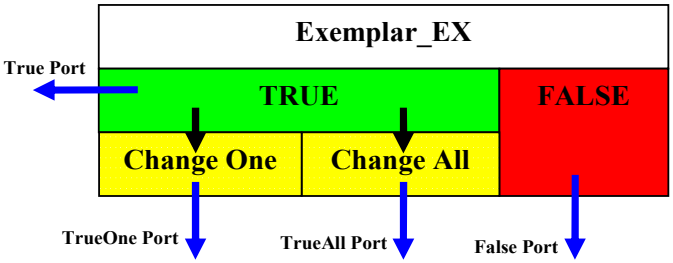


Figure 2.11 Complete exemplar node for dynamic networking[7]

Research and development is still being conducted in order to improve the capabilities of the design exemplar. However, a case study of the applicability of the design exemplar to support industrial scenarios has been notably absent. To this end, this research presents a case study on using the design exemplar as a search and retrieval tool for retrieving mold inserts in order to verify its usability and applicability.

CHAPTER 3

SEARCH AND RETRIEVAL OF MOLD INSERTS USING DESIGN EXEMPLAR

In this chapter the approaches proposed for the implementation of the design exemplar as a search and retrieval tool of mold inserts are presented and their limitations are discussed.

The Exemplar Approach

In the exemplar approach, a set of conditions are imposed on the salient vertices or points of the geometry of the mold insert to constrain it fully outside or inside a bounding geometry so that the "*similarity*" condition is satisfied. The most common condition is checking if a given point lies within a particular area of the tolerance envelope or bounding geometry. Hence this is called boundary constraint approach in this thesis. The geometry on which the conditions are imposed is called the exemplar and the geometry on which the exemplar is constrained is called the model. The model and exemplar concepts are demonstrated using the following example. To check if a line segment can fit within a tolerance envelope of rectangular shape, conditions can be imposed on the entities of the line segment with respect to the entities of the rectangle to form fully constraint problem. In this example, the line segment forms the exemplar of the query as conditions are imposed on it and envelope forms the model of the query as line segment is constrained about it. If all the salient points or vertices of geometry satisfy the conditions imposed, then the query is said to be similar to the target; otherwise it is not similar. Before continuing with the discussion on the exemplar approach, a standard understanding of the terminology used is required. Following the definitions, two different exemplar approaches are discussed and illustrated with simple examples.

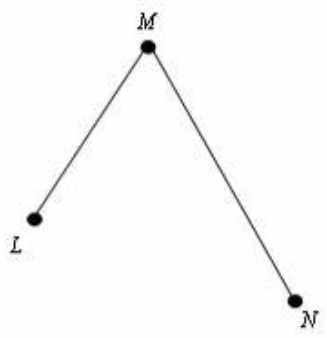
Nomenclature:

Some of the key terms used in this research are defined below using Figure3.1 and Figure 3.2.

End Tolerance Boxes: The tolerance envelope drawn around a target mold insert will have two tolerance boxes, one at the beginning and one at the end. For a mold insert to be considered similar, its starting and ending points should lie within these end tolerance boxes of the envelope. For example, from the Figure 3.2, P1 P2 P7 P6 and P9 P4 P10 P5 form the end tolerance boxes. For the mold insert to be

considered similar, it should not only fit within the envelope shown in Figure 3.2, but its endpoints *L* and *N* should lie within these tolerance boxes

Primary and Secondary Tolerance: The tolerance envelope drawn around the target envelope (shown in dotted lines) consists of two different types of tolerances. The width of the tolerance envelope, the distance between P1 and P6, which determines the tolerance of the envelope is called primary tolerance. This is indicated by Tol1. The tolerance provided at the legs that form the tolerance boxes is called secondary tolerance. This is indicated by Tol2.



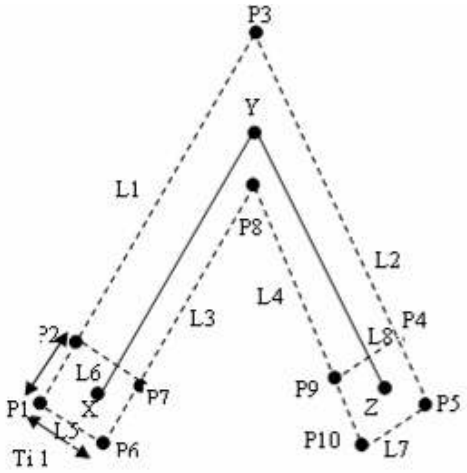


Figure 3.1 Mold insert LMN **Figure 3.2 Target mold insert XYZ and its tolerance envelope**

Other Terminology Used: Since in Figure 3.1, line segments LM and MN represent the top view of the legs of the mold insert, the line segments are often referred to as the legs rather than line segments to fit the context. The end points P1 and P6 are referred to as corners of the tolerance envelope. P1 P2 P3 P4 P5 is called outer boundary of the tolerance envelope and P6 P7 P8 P9 P10 is called the inner boundary of the tolerance envelope. Therefore, when a line is drawn from P1, it would be referred to as a line drawn from the corner of the outer boundary of the envelope. To avoid confusion between the vertices of the envelope and mold insert when they are superimposed, the vertices of the mold insert are italicized to make them look different from the vertices of an envelope.

Using this nomenclature, the approach is demonstrated for a line-line mold insert before applying to a line-arc-line mold insert.

Boundary Constraint Approach Applied to Line-Line Mold Insert:

For a mold insert in the database to be considered as a similar mold insert to the target, it should satisfy the following conditions:

- The starting and the ending points should lie within the end tolerance boxes.
- The mold insert should lie completely within or on the tolerance envelope.

The process of verifying if a mold insert can fit within a tolerance envelope can be done in two different methods. In the first method, the tolerance is adjusted around the mold insert to find if there is at least one configuration that exists such that the necessary conditions are met. Whereas, in the second method, the mold inset is adjusted such that it fits within the envelope satisfying the necessary conditions. Though these methods may sound alike, there is a significant difference based on the number of constraints applied which may effect the time to find a mold inset in the database. This difference is explained later after explaining the exemplars needed for the both procedures.

Boundary Constraint Approach 1

In this approach, the tolerance envelope is adjusted around the mold insert retrieved from the database to find if there exists a configuration such that the two conditions necessary to qualify as a similar mold insert are met. Since conditions are imposed on the tolerance envelope, it forms the extract part of the query and the mold insert form the alpha match part of the query.

Conditions that should be satisfied by the tolerance envelope so that the mold insert LMN is similar to XYZ are (referring to Figure 3.1 and Figure 3.2):

- The point *L* should lie within the end tolerance box either P1 P2 P7 P6 or P4 P5 P10 P9.
- The point *N* should lie within the box P1 P2 P7 P6 or P4 P5 P10 P9.
- *M* should lie within the envelope.

Figure 3.3 presents an enlarged view of the tolerance boxes of the tolerance envelope shown in Figure 3.2. For point *L* to lie within the box P1 P2 P7 P6, which is a necessary condition for the mold insert *LMN* to be similar, it should satisfy the following equations that are derived below:

From Figure 3.3 (a) and Figure 3.3 (b), if the distances of the point *L* from the lines L1, L3, L5, L6 are d1, d2, d3, d6 respectively and the length and breadth of the rectangle box P1 P2 P7 P6 be Ti1 and Ti2, the conditions that should be satisfied so that the point L lies within the rectangle box are:

Ti1=d1+d2. (3.1)

and

Ti 2 =d3+d4. (3.2)

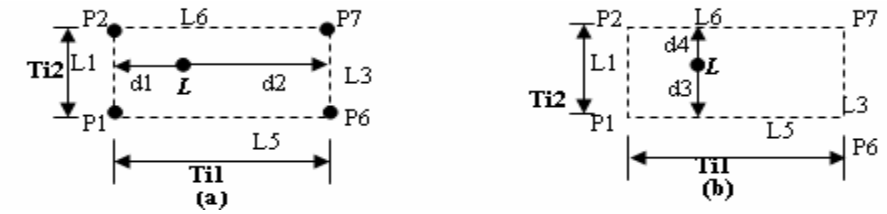


Figure 3.3 Conditions need to be satisfied by point L

Also, there is a possibility that the point *L* lies in the tolerance box lying on the other side of the tolerance envelope, which is P4 P5 P10 P9. Therefore, similar equations are formulated using Figure 3.4. In Figure 3.4, the enlarged view of the tolerance box P4 P5 P10 P9 is shown. Therefore, to constrain *L* within this tolerance box, it should satisfy the equations listed below:

Ti1=d5+d6. (3.3)

and

Ti 2 =d7+d8. (3.4)

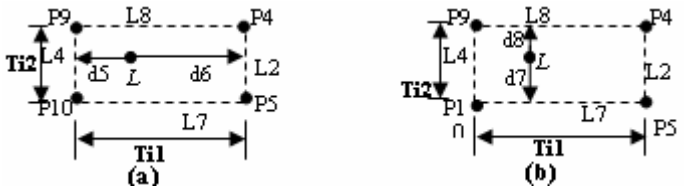


Figure 3.4 Conditions need to be satisfied by point L for it to lie within the box

A similar set of equations are used to constrain point *N* such that it lies within the tolerance box. The equations needed to constrain point *M* within the tolerance envelope are derived using Figure 3.5. In Figure 3.5, the tolerance envelope shown in Figure 3.2 is split into two halves. For point *M* to lie within the envelope, it should lie in either of the halves. Referring to Figure 3.5, point *M* should lie between L1 and

L3 or L2 and L4. If d9 and d10 are the distances of *M* from L1 and L3, and d11 and d12 are the distances of *M* from L4 and L2, then for *M* to lie within the envelope, it should satisfy at least one of the two conditions stated below:

$$d9+d10=Ti1 \text{ AND } d11 \leq L1+Ti1 \quad \text{OR} \quad d12+d13=Ti1 \text{ AND } d11 \leq L2+Ti1. \qquad (3.5)$$

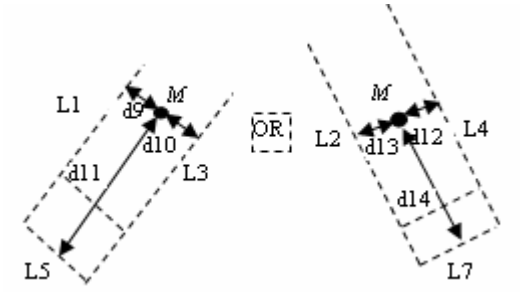


Figure 3.5 Conditions need to be satisfied by point M to lie within the tolerance envelope

If the equation (3.5) is divided into condition 1 and condition 2, the condition before the OR being condition 1 and the latter being condition 2, then the sufficiency of these conditions is checked using Figure 3.6. Figure 3.6 shows the various possible positions of point *M* inside and outside of the envelope.

Table 3-1Truth table to verify sufficiency of the conditions

Vertex name	Is condition 1 of the equation 3.2 is satisfied	Is condition 2 of the equation 3.2 is satisfied	If "No", reason why it is failed	Result: Does the point lie with in the envelope
M1	No	---	d9+d10>Ti1	No
M2	Yes	Yes		Yes
M3	_ _ _	No	d12+d13=Ti1	No
M4	Yes	Yes	d11 ≤ L1+Ti1 and	Yes
M5	No	No	d12+d13=Ti1	No

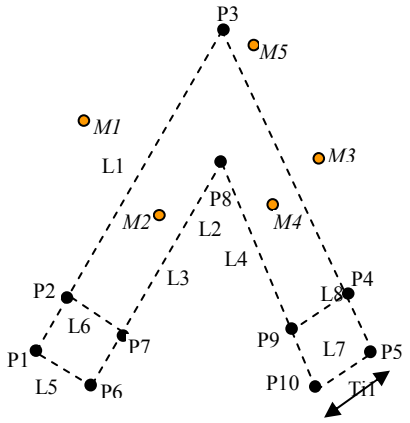


Figure 3.6 Verifying the sufficiency of the conditions developed

The exemplar authored for the retrieval of line-line mold shown using Figure 3.7. In the Figure 3.7, the tolerance envelope is represented in the dotted line indicating that it forms the extract part of the exemplar and mold insert LMN which is being checked for similarity forms the match part of the exemplar. The specifications of the tolerance envelope such the lengths of the legs, angle between the legs and tolerances are indicated in the figure.

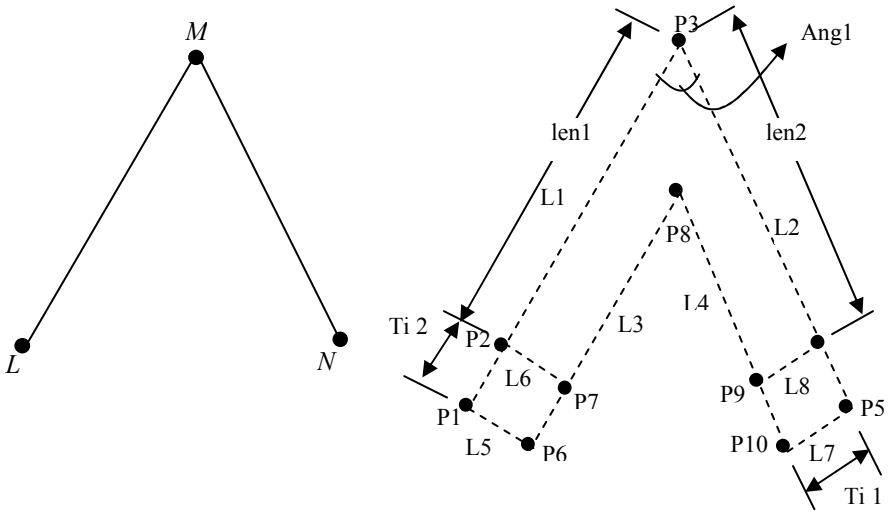


Figure 3.7 Exemplar with mold insert as match and envelope as extract

In Figure 3.8, lines one to seven describe the entities that form the mold insert and lines eight to15 describe the incident, tangency and distance constraints such that the entities form they form the mold insert. These lines fall under Alpha match category indicating that in this approach the mold insert forms the match part of the exemplar. Lines 16 to 78 describe the entities of mold insert in the database which

form the extract part of the exemplar. From line 79, a set of distances are calculated to form a set of equations. These equations are formed into set of blocks such that when a mold insert satisfies these set of blocks of equations, it is said to be a match to the query else not.

Alpha Match:

1. Parameter Len1;
2. Parameter Len2;
3. Line L_A;
4. Line L_B;
5. Point L;
6. Point M;
7. Point N;
8. *Incident (L_A, L);*
9. *Incident (L_ A, M);*
10. *Incident (L_B, M);*
12. *Incident (L_B, N);*
13. *Angle "Ang1" (L_A, L_B);*
14. *Distance "Len1" (L, M);*
15. *Distance "Len2" (M, N);*

Alpha Extract:

16. Parameter len1;
17. Parameter len2;
18. Parameter Ang1;
19. Parameter Ti 1;
20. Parameter Ti 2;
21. Line " L1";
22. Line "L2";
23. Line "L3";
24. Line "L4";
25. Line "L5";
26. Line "L6";
27. Line "L7";
28. Line "L8";
29. Point "P1";
30. Point "P2";
31. Point "P3";
32. Point"P4";
33. Point "P5";
34. Point "P6";
35. Point "P7";
36. Point "P8";
37. Point "P9";
38. Point "P10;
40. *Incident* (L1, P3);
41. *Incident* (L1, P2);
42. *Incident (L1, P1);*
43. *Distance "len1+Ti 1" (P1,P 3)*
44. *Distance "len1" (P1, P 2);*
45. Angle (Ang1, L1);
46. *Incident (L2, P3);*
47. *Incident (L2, P4);*
48. *Incident (L2, P5);*
49. *Distance "len2+Ti 2" (P3, P5);*
50. *Distance "len2" (P3, P4);*
51. *Angle (Ang2, L1);*
52. *Incident (L3, P6);*
53. *Incident (L3, P7);*
54. *Incident (L3, P8)*
55. *Distance "len11" (P6, P8);*
56. *Distance "len11-Tol2" (P7, P8);*
57. *Incident (L4, P8);*
58. *Incident (L4, P9);*
59. *Incident (L4, P10);*
60. *Distance "len2" (P8, P10);*
61. *Distance "len2" (P9, P10);*
62. *Parallel (L4, L2);*
63. *Distance "Ti1" (L1, L3);*
64. *Distance "Ti1" (L2, L4);*
65. *Parallel (L3, L1);*
66. *Incident (L5, P1);*
67. *Incident (L5, P6);*
68. *Perpendicular (L1, L5);*
69. *Incident (L6, P2);*
70. *Incident (L6, P7);*
71. *Perpendicular (L1, L6)*
72. *Distance "Tol2" (L6, L5);*
73. *Incident (L8, P4);*
74. *Incident (L8, P9);*
75. *Perpendicular (L8, L2);*
76. *Incident (L7, 10);*
77. *Incident (L7, P5);*
78. *Perpendicular (L7, L2);*
79. *Distance "Tol2" (L7, L8);*
80. *Distance "d1" (L, L1);*
81. *Distance "d2" (L, L3);*
82. *Distance "d3" (L, L5);*
83. *Distance "d4" (L, L6);*
84. *Distance "d5" (L, L4);*
85. *Distance "d6" (L, L2);*
86. *Distance "d7" (L, L7);*
87. *Distance "d8" (L, L8);*
88. *Distance "d9" (M, L1);*
89. *Distance "d10" (M, L3);*
90. *Distance "d11" (M, L5);*
91. *Distance "d12" (M, L2);*
92. *Distance "d13" (M, L4);*
93. *Distance "d14" (M, L7);*
94. *Distance "d15" (N, L4);*
95. *Distance "d16" (N, L2);*
96. *Distance "d17" (N, L7);*
97. *Distance "d18" (N, L8);*
98. *Distance "d19" (N, L1);*
99. *Distance "d20" (N, L3);*
100. *Distance "d21" (N, L5);*
101. *Distance "d22" (N, L6);*
102. *Equation "EQ1" (Ti 1, d1, d2);*
103. *Equation "EQ2" (Ti 2, d3, d4);*
104. *Equation "EQ3" (Ti 1, d9, d10);*
105. *Equation "EQ4" (d11, Len1, Tol2);*
106. *Equation "EQ5" (Ti 1, d13, d12);*
107. *Equation "EQ6 (d14, Len2, Tol2);*
108. *Equation "EQ7" (Ti 1, d15, d16);*
109. *Equation "EQ8" (Ti 2, d17, d18);*
110. *Equation "EQ9" (Ti 1, d5, d6);*
111. *Equation "EQ10" (Ti 2, d7, d8);*
112. *Equation "EQ11" (Ti 1, d19, d20);*
113. *Equation "EQ12" (Ti 2, d21, d422);*

Blocks

Sub Block "Block1" (EQ1, EQ2, EQ7, EQ8);
Sub Block "Block 2" (EQ9, EQ10, EQ11, EQ12);
OR Block "Block 3" ("Block1", "Block 2");
Sub Block "Block 4" (EQ3, EQ4);
Sub Block "Block 5" (EQ5, EQ6);
OR Block "Block 6" ("Block 4", "Block 5");

Equations:

Equation "EQ1" (Ti 1= d1 + d2);
Equation "EQ2" (Ti 2= d3 + d4);
Equation "EQ7" (Ti 1= d15+ d16);
Equation "EQ8" (Ti 2= d17 + d18);
Equation "EQ9" (Ti 1= d5+ d6);
Equation "EQ10" (Ti 2= d7 + d8);
Equation "EQ11" (Ti 1= d19+ d20);
Equation "EQ12" (Ti 2= d21 + d22);
Equation "EQ3" (Ti 1= d9 + d10);
Equation "EQ4" (d11 $\leq$ Len1+Ti 2);
Equation "EQ5" (Ti 1= d12 + d13);
Equation "EQ6 (d14 $\leq$ Len2+Ti 2);

Figure 3.8 Exemplar authored for mold insert retrieval with envelope as extract

Boundary Constraint Approach 2

In this approach, the mold insert form the database is adjusted within the tolerance envelope to find if at least one configuration exists such that it meets the conditions necessary for the similarity. Figure

3.9 schematically represents this approach. In Figure 3.9, the tolerance envelope is grounded and is drawn in solid lines indicating that it forms the match part of the exemplar. The mold insert LMN is drawn in dotted lines indicating that it forms the extract part of the exemplar and various degrees of freedom the mold insert may have inside the envelope are schematically shown in the figure.

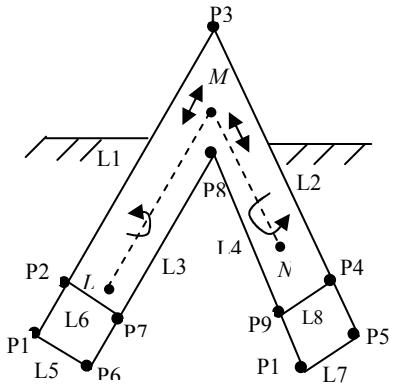


Figure 3.9 The match and extract parts of the exemplars are interchanged in the approach 2

The conditions that has to be satisfied by the mold insert remains the same, that is the points *L, M, N* should satisfy the same conditions mentioned in the exemplar approach 1 but the difference is that now these conditions are applied on the mold insert rather than the tolerance envelope. The exemplar authored is explained using Figure 3.10. In Figure 3.10, the tolerance envelope which forms the match part of the query is indicated with solid lines whereas the extract part is formed by the mold insert from the database which s indicated in dotted lines. (Alpha extract). The exemplar authored is presented in Figure 3.11.

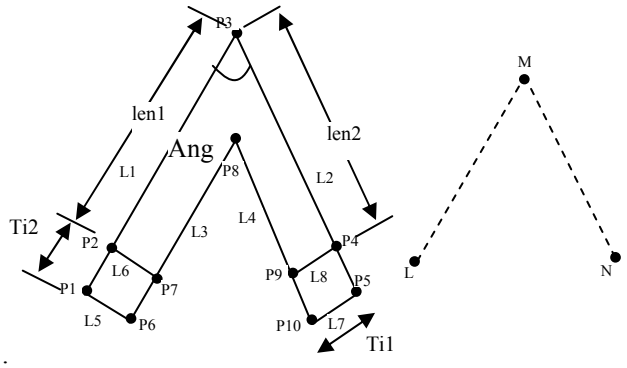


Figure 3.10 Exemplar query with envelope as match and mold insert as query

In Figure 3.11, lines to 23 describe the entities that form the envelope and lines 24 to 78 describe the incident, tangency and distance constraints such that the entities form they form the tolerance envelope. These lines fall under Alpha match category indicating that in this approach, the tolerance envelope form the match part of the exemplar. Lines 79 to 94 describe the entities of the mold insert in the database which form the extract part of the exemplar. From line 95, a set of distances are calculated to form a set of equations. These equations are formed into set of blocks such that when a mold insert satisfies these set of blocks of equations, it is said to be a match to the query else not.

Alpha Match:

1. Parameter len1;
2. Parameter len2;
3. Parameter Ang1;
4. Parameter Tol1;
5. Parameter Tol2;
6. Line " L1";
7. Line "L2";
8. Line "L3";
9. Line "L4";
10. Line "L5";
11. Line "L6";
12. Line "L7";
13. Line "L8";
14. Point "P1";
15. Point "P2";
16. Point "P3";
17. Point "P4";
18. Point "P5";
19. Point "P6";
20. Point "P7"
21. Point "P8";
22. Point "P9";
23. Point "P10;

24. Incident (L1, P3);
25. Incident (L1, P2);
26. Incident (L1, P1)
. Distance "len1+Tol1" (P1,P 3)
28. Distance "len1" (P1, P 2);
29. Angle (Ang1, L1);
30. Incident (L2, P3);
31. Incident (L2, P4);
32. Incident (L2, P5);
Distance "len2+Tol2" (P3, P5);
34. Distance "len2" (P3, P4);
35. Angle (Ang2, L1);
36. Incident (L3, P6);
37. Incident (L3, P7);
38. Incident (L3, P8)
39. Distance "len11" (P6, P8);
Distance "len11-Tol2"(P7,P8);
41. Incident (L4, P8);
42. Incident (L4, P9);
43. Incident (L4, P10);
44. Distance "len2" (P8, P10);
45. Distance "len2" (P9, P10);
46. Angle (Ang1, L1);
47. Incident (L2, P3);
48. Incident (L2, P4);
49. Incident (L2, P5);
Distance "len2+Tol2" (P3, P5);
51. Distance "len2" (P3, P4);
52. Angle (Ang2, L1);
53. Incident (L3, P6);
54. Incident (L3, P7);
55. Incident (L3, P8)
56. Distance "len11" (P6, P8);.
Distance "len11-Tol2" (P7, P8);
57. Incident (L4, P8);
58. Incident (L4, P9);
59. Incident (L4, P10);
60. Distance "len2" (P8, P10);
61. Distance "len2" (P9, P10);
62. Parallel (L4, L2);
63. Distance "Tol1" (L1, L3);
64. Distance "Tol1" (L2, L4);
65. Parallel (L3, L1);
66. Incident (L5, P1);
67. Incident (L5, P6);
68. Perpendicular (L1, L5);
69. Incident (L6, P2);
70. Incident (L6, P7);
71. Perpendicular (L1, L6);
72. Distance "Tol2" (L6, L5);
73. Incident (L8, P4);
74. Incident (L8, P9);
75. Perpendicular (L8, L2);
76. Incident (L7, 10);
77. Incident (L7, P5);
78. Perpendicular (L7, L2);

Alpha Extract:

79. Parameter Len1;
80. Parameter Len2;
81. Line L_A;
82. Line L_B;
83. Point L;
84. Point M;
85. Point N;

86. Incident (L_A, L);
87. Incident (L_ A, M);
88. Incident (L_B, M);
89. Incident (L_B, N);
90. Distance "Len1" (L, M);
91. Distance "Len2" (M, N);
92.Angle "Ang" (L_A, L_B);
93. Distance "Len1" (L, M);
94. Distance "Len2" (M, N);
95. Distance "d1" (L, L1);
94. Distance "d2" (L, L3);
95. Distance "d3" (L, L5);
96. Distance "d4" (L, L6);
97. Distance "d5" (L, L4);
98. Distance "d6" (L, L2);
99. Distance "d7" (L, L7);
100. Distance "d8" (L, L8);
101. Distance "d9" (M, L1);
102. Distance "d10" (M, L3);
103. Distance "d11" (M, L5);
104. Distance "d12" (M, L2);
105. Distance "d13" (M, L4);
106. Distance "d14" (M, L7);
107. Distance "d15" (N, L4);
108. Distance "d16" (N, L2);
109. Distance "d17" (N, L7);
110. Distance "d18" (N, L8);
111. Distance "d19" (N, L1);
112. Distance "d20" (N, L3);
113. Distance "d21" (N, L5);
114. Distance "d22" (N, L6);
115. Equation "EQ1" (Tol1, d1 , d2);
116. Equation "EQ2" (Tol2, d3, d4);
117. Equation "EQ3" (Tol1, d9, d10);
118. Equation "EQ4" (d11, Len1,Tol2);
119. Equation "EQ5" (Tol1, d13, d12);
120. Equation "EQ6 (d14, Len2, Tol2);
121. Equation "EQ7" (Tol1, d15, d16);
122. Equation "EQ8" (Tol2, d17, d18);
123. Equation "EQ9" (Tol1, d5, d6);
124. Equation "EQ10" (Tol2, d7, d8);
125. Equation "EQ11" (Tol1, d19, d20);
126. Equation "EQ12" (Tol2, d21, d422);

Blocks

Sub Block "Block1" (EQ1, EQ2, EQ7, EQ8);
Sub Block "Block 2" (EQ9, EQ10, EQ11, EQ12);
OR Block "Block 3" ("Block1", "Block 2");
Sub Block "Block 4" (EQ3, EQ4);
Sub Block "Block 5" (EQ5, EQ6);
OR Block "Block 6"("Block 4", "Block 5");

Equations:

Equation "EQ1" (Tol1= d1 + d2);
Equation "EQ2" (Tol2= d3 + d4);
Equation "EQ7" (Tol1= d15+ d16);
Equation "EQ8" (Tol2= d17 + d18);
Equation "EQ9" (Tol1= d5+ d6);
Equation "EQ10" (Tol2= d7 + d8);
Equation "EQ11" (Tol1= d19+ d20);
Equation "EQ12" (Tol2= d21 + d22);
Equation "EQ3" (Tol1= d9 + d10);
Equation "EQ4" (d11 $\leq$ Len1+Tol2);
Equation "EQ5" (Tol1= d12 + d13);
Equation "EQ6 (d14 $\leq$ Len2+Tol2);

Figure 3.11 Exemplar authored for mold insert retrieval with envelope as match

Observation

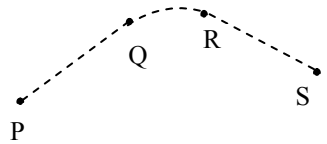
When a model is queried with an exemplar, the constraint solver of the design exemplar technology tries to constraint the entities of the geometry such that the conditions contained in the extract

part of the exemplar are satisfied. The greater the number of conditions present in the extract part of the exemplar, the more is time taken by the constraint solver to check if a configuration exist such that these conditions are satisfied. By checking the exemplars authored for the two approaches, approach 2 has fewer constraints in the extract part as compared to the approach 1. This means that the number of constraints that the constraint solver of the design exemplar algorithm has to solve is considerably less in the approach 1 when compared to approach 2. In approach 1 where the tolerance envelope forms the extract part of the exemplar, eight lines and ten points of the tolerance envelope are constrained around three points and two lines of the mold insert which is retrieved from the database. However, in approach 2, where the mold insert forms the extract part of the exemplar query, three points and two lines of the mold insert retrieved from the database are constrained within eight lines and ten points of the tolerance envelope. Hence, the constraint solver takes a considerable amount of time to check for the similarity condition in the approach 1 when compared to the approach 2.

Exemplar Approach Applied to Line-arc-Line Mold Inserts:

The approach discussed for line-line mold inserts is applied for the retrieval of line-arc-line mold inserts. Since apart for the curve which has been introduced in between the two lines, the example discussed has similar geometrical characteristics to those of the line-arc-line profile. Hence most of the conditions that have to be applied to constraint a line-arc-line mold insert within a line-arc-line tolerance envelope remain the same. Figure 3.12 shows a mold insert PQRS of line-arc-line profile that has to be verifed if it can fit within the tolerance envelope shown in Figure 3.13. For PQRS to be considered as a similar mold insert with reference to the envelope shown in the Figure 3.13, it should satisfy the following conditions:

- Point *P* should lie within the end tolerance box P1 P2 P11 P12.
- Point *S* should lie within the end tolerance box P5 P6 P7 P8.
- *Q* and *R* should lie within the tolerance envelope.



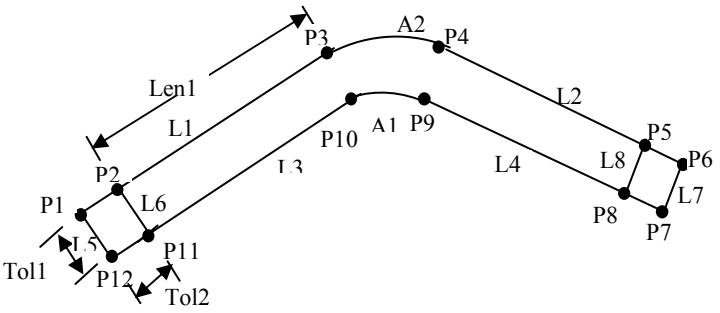


Figure 3.12 Mold insert PQRS **Figure 3.13 Tolerance envelope from target mold insert**

From Figure 3.14, if the perspendicular distances between point *Q* and lines L1, L3 and L5 are d9, d10, d11 respectively and the perpendicular distances between point *Q* and lines L2, L4 and L7 are d12, d13, d14, then for point *Q* to lie within the tolerance envelope, it should satisfy the following equations:

$d9+d10=Ti1$ AND $d11 \leq L1+Ti1$ (3.6) OR

$d12+d13=Ti1$ AND $d11 \leq L2+Ti1$ (3.7) OR

$Ri \leq d15 \leq Ro$ AND $\theta 1 + \theta 2 = \alpha$, Where α is angle subtended by the arc A1 at its center (3.8)

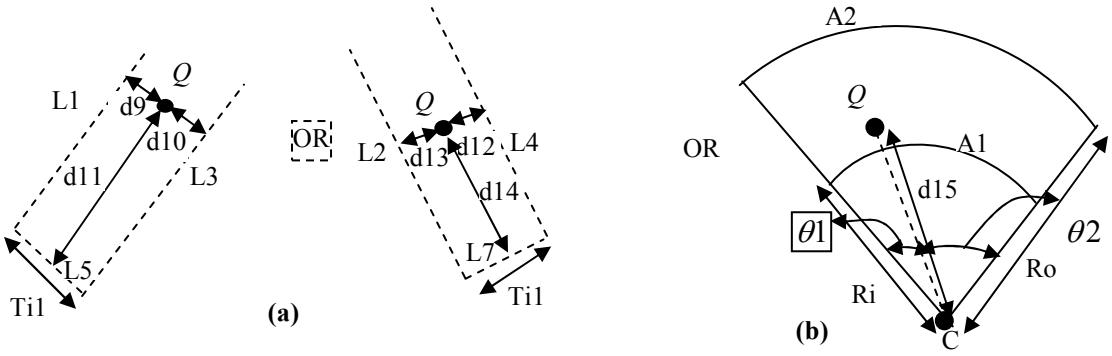


Figure 3.14 Conditions needed to be satisfied by the point M to lie within the tolerance envelope

Similar conditions are also applied on point R to check if it lies within the tolerance envelope. Figure 3.15 presents an enlarged view of the tolerance boxes. To constraint point *P* within the tolerance envelope, a set of conditions are developed. Let the distance of point *P* be d1, d2, d3, d4 from the lines L1, L3, L5, L6 respectively, the point *P* should satisfy following equations for it to be contained in the end tolerance box P1 P2 P11 P12 :

$d1+d2=Ti\ 1$ (3.9)

d3+d4=Ti 2 (3.10)

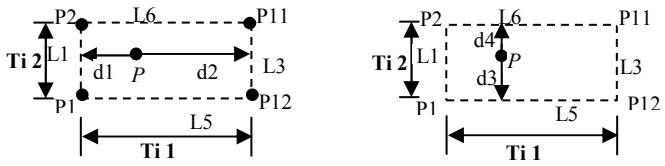


Figure 3.15 Conditions needed to be satisfied by the point P to lie within the box P1 P2 P7 P6

Also, there is a possibility that point P can stay in the other end tolerance box P7 P8 P5 P6. Therefore, the conditions it needs to satisfy in order it be contained in the end tolerance box are:

d5+d6=Ti 1 (3.11)

d7+d8=Ti2 (3.12)

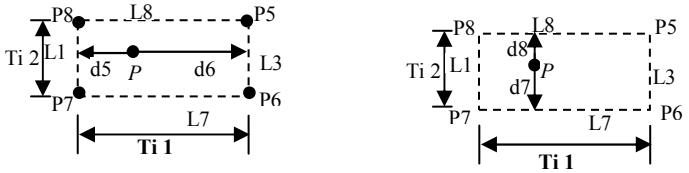


Figure 3.16Conditions needed to be satisfied by point P to lie within the tolerance box P7 P8 P5 P6

The conditions developed along with the geometric entities that form the match and extract parts form the exemplars. The exemplar authored is explained using Figure 3.17 where the envelope is represented in solid lines indicating that it forms the match part of the exemplar and the mold insert PQRS in dotted lines indicating it forms the extract part of the exemplar. .

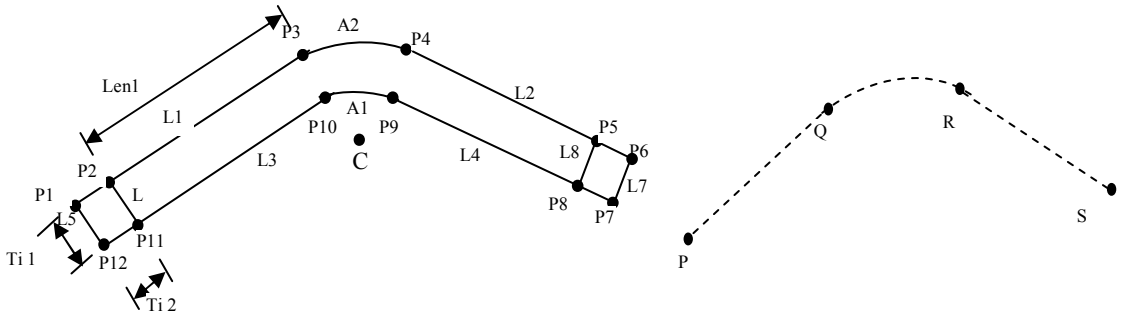


Figure 3.17 Exemplar Query with envelope as match and mold insert as exemplar

In Figure 3.18, lines one to 34 describe the entities that form the envelope and lines 35 to 77 describe the incident, tangency and distance constraints such that the entities form they form the tolerance

envelope. These lines fall under Alpha match category indicating that in this approach, the tolerance envelope forms the match part of the exemplar. Lines 78 to 94 describe the entities and conditions that constitute the mold insert in the database, which form the extract part of the exemplar. From line 95, a set of distances are calculated to form a set of equations. These equations are formed into set of blocks such that when a mold insert satisfies these set of blocks of equations, it is said to be a match to the query else not.

Alpha Match:

1. Parameter Len1;
2. Parameter Len2;
3. Parameter Ang;
4. Parameter Tol1;
5. Parameter Tol2;
6. Parameter rad;
7. Line “L1”;
8. Line “L2”;
9. Line “L3”;
10. Line “L4”;
11. Line “L5”;
12. Line “L6”;
13. Line “L7”;
14. Line “L8”;
15. Line CP3;
16. Line CP4;
17. Point “P1”;
18. Point “P2”;
19. Point “P3”;
20. Point “P4”;
21. Point “P5”;
22. Point “P6”
23. Point “P8”;
24. Point “P9”;
25. Point “P10;
26. Point “P11”;
27. Point “P12”;
28. Point P;
29. Point Q;
30. Point R;
31. Point S;
32. Point C;
33. Arc “A1”;
34. Arc “A2”;

35. Tangency (A1, L1);
36. Tangency (A1, L2);
37. Tangency (A2, L4);
38. Tangency (A2, L3);
39. Incident (L1, P3);
40. Incident (L1, P2);
41. Incident (L1, P1);
Distance “len1+Tol1” (P1, P 3);
42. Distance “Tol1” (P1, P 2);

43. A1 (Rad1+Tol/2, Ang1);

44. Incident (A1, P3);
45. Incident (A1, P4);

46. A2 (Rad1+Tol/2, Ang1);

47. Incident (L2, P6);
48. Incident (L2, P4);
49. Incident (L2, P5);
. Distance “len2+Tol2” (P4,P 6);
51. Distance “len2” (P4, P 5);
52. Incident (L4, P9);
53. Incident (L4, P7);
54. Incident (L4, P8);
Distance “len2+Tol2” (P7, P 9);
55. Distance “len2” (P8, P 9);
56. Parallel (L4, L2);
57. Incident (L7, P6);
58. Incident (L7, P7);
59. Perpendicular (L7, L2);
60. Perpendicular (L7, L4);
61. Incident (L8, P8);
62. Incident (L8, P5);
63. Parallel (L8, L7);
64. Concentric (A1, A2);
65. Incident (L3, P 10);
66. Incident (L3, P11);
67. Incident (L3, P12);
68. Distance “len2+Tol2” (P10, P 12);
69. Distance “len2” (P11, P 10);
70. Parallel (L3, L1);
71. Incident (L5, P1);
72. Incident (L5, P12);
73. Perpendicular (L7, L1);
74. Perpendicular (L7, L3);
75. Incident (L6, 2);
76. Incident (L8, 11);
77. Parallel (L8, L7);

Alpha Extract

78. Point P;
79. Point Q;
80. Point R;
81. Point S;
82. Line Li1;
83. Line Li2;

84. Incident (P, Li1);
85. Incident (Q, Li1);
86. Incident (Q, Ar2);
87. Incident (R, Ar2);
88. Incident (R, Li2);
89.Equation “EQ1” (Tol1= d1 + d2);
90.Equation “EQ2” (Tol2=d3+d4);
91. Equation “EQ3” (Tol1= d5 + d6);
92. Equation “EQ4” (Tol2=d7+d8);
93. Equation “EQ5” (Tol1= d22+ d23);
94. Equation “EQ6” (Tol2, d2, d25);
95. Equation “EQ7” (Tol1, d26, d27);
96.Equation “EQ8” (Tol2 d28, d29);
97. Equation “EQ9” (Tol1, d9, d10);
98. Distance “d1” (P, L1);
99. Distance “d2” (P, L2);
100. Distance “d3” (P, L5);
101. Distance “d4” (P, L6);
102. Distance “d5” (S, L4);
103. Distance “d6” (S, L2);
104. Distance “d7” (S, L8);
105 Distance “d8” (S, L7);
106. Distance “d9” (Q, L1);
107. Distance “d10” (Q, L3);
108. Distance “d11” (Q, L5);
109. Distance “d12” (Q, L2);
110. Distance “d13” (Q, L4);
111. Distance “d14” (Q, L7);
112. Distance “d15” (Q, C);
113. Distance “d16” (R, L1);
114. Distance “d17” (R, L3);
115. Distance “d18” (R, L5);
116. Distance “d19” (R, L2);
117. Distance “d20” (R, L4);
119. Distance “d21” (Q, L7);
120. Distance “d22” (Q, C);
121. Equation “EQ10” (d1, L1, Ti1);
122. Equation “EQ11” (Tol1, d12, d13);
123. Equation “EQ12” (d14, L1, Ti1);
124 .Equation “EQ13” (Ang, Ang1, Ang2)
125. Equation “EQ14” (rad, Ti1, d15);
126. Equation “EQ15” (Tol1, d16, d17);
127. Equation “EQ16” (d18, L1, Ti1);
128. Equation “EQ17” (Tol1, d19, d20);
129 Equation “EQ18” (d21, L1, Ti1);
130 Equation “EQ19” (Ang, Ang3, Ang4);
131 Equation “EQ20” ((rad, T, d30);
132. Equation “EQ6” (Tol2, d2, d25);
133. Equation “EQ7” (Tol1, d26, d27);
134. Equation “EQ8” (Tol2 d28, d29);
135. Equation “EQ9” (Tol1, d9, d10);
136 Equation “EQ10” (d1, L1, Ti1);
137. Equation “EQ11” (Tol1, d12, d13);
138. Equation “EQ12” (d14, L1, Ti1);
139. Equation “EQ13” (Ang, Ang1, Ang2);
140. Equation “EQ14” (rad, Ti1, d15);
141. Equation “EQ15” (Tol1, d16, d17);
142 Equation “EQ16” (d18, L1, Ti1);
143. Equation “EQ17” (Tol1, d19, d20);
144. Equation “EQ18” (d21, L1, Ti1);
145. Equation “EQ19” (Ang, Ang3, Ang4);
146. Equation “EQ20” ((rad, T, d30);

Blocks:

Sub Block “Block1” (*EQ1, EQ2, EQ3, EQ4*);
Sub Block “Block2” (*EQ5, EQ6, EQ7, EQ8*);
OR Block (“Block1”, “Block2”);
Sub Block “Block3” (*EQ9, EQ10*);
Sub Block “Block4” (*EQ11, EQ12*);
Sub Block “Block5” (EQ13, EQ14);
OR Block (“Block3”, “Block4”, “Block5”);
Sub Block “Block6” (*EQ15, EQ16*);
Sub Block “Block7” (*EQ17, EQ18);*
Sub Block “Block8” (*EQ19, EQ20*);
OR Block (“Block6”, “Block7”, “Block8”);

Equations:

Equation “EQ1” (Tol1= d1 + d2);
Equation “EQ2” (Tol2=d3+d4);
Equation “EQ3” (Tol1= d5 + d6);
Equation “EQ4” (Tol2=d7+d8);
Equation “EQ5” (Tol1= d22+ d23);
Equation “EQ6” (Tol2= d2+, d25);
Equation “EQ7” (Tol1= d26+ d27);
Equation “EQ8” (Tol2= d28+ d29);
Equation “EQ9” (Tol1= d9+ d10);
Equation “EQ10” (d11 ≤ = L1+Ti1);
Equation “EQ11” (Tol1= d12+ d13);
Equation “EQ12” (d14 ≤ = L1+Ti1);
Equation “EQ11” (Ang= Ang1+Ang2);
Equation “EQ13” ((rad +Ti) ≤ d15 ≤ (rad-Ti1));
Equation “EQ14” (Tol1= d16+ d17);
Equation “EQ15” (d18 ≤ = L1+Ti1);
Equation “EQ16” (Tol1= d19+ d20);
Equation “EQ17” (d21 ≤ = L1+Ti1);
Equation “EQ18” (Ang= Ang3+Ang4);
Equation “EQ19” ((rad +Ti) ≤ d30 ≤ (rad-Ti1));

Figure 3.18 Exemplar authored for line arc line mold insert retrieval

Limitations of the Exemplar Approach:

The following are the limitations observed with these approaches:

- From the anecdotal experience of developing exemplars for the retrieval of mold inserts for profiles line-line and line-arc-line, it can be inferred that authoring exemplars is a tedious job The exemplar built for the retrieval of a simple line-arc-line profile consists of 40 entities and 80 geometric constraints and to author an exemplar to obtain a generic solution for mold insert retrieval problem, 18 different exemplars have to be authored, which means the number of geometric entities and constraints that should be handled will be of the order of thousands.
- A typical mold insert can have up to 18 line-arc-line profiles. The exemplars authored works only for a line-arc-line profile but not for mold inserts with a different profile. Therefore, to author an exemplar for the retrieval of similar mold inserts of all profiles, different exemplars have to be written for each increment of line-arc-line. This introduces a number of complex issues which are discussed below:
    - With each increment of arc and line in the profile of the mold insert, the number of entities that have to be handled increases gradually. Authoring exemplars for the mold inserts may become quite difficult due to the high number of entities and constraints that have to be handled.
    - All the exemplars written for different profiles of mold inserts have to be networked to obtain a general solution for the similar mold an exemplar, in worst case, the mold inserts is checked 18 times with 18 different exemplars in the first step, and is then checked for the constraints applied. The procedure may become quite time consuming.

Since the design exemplar has the limitations mentioned above, a mathematical-based exemplar has been developed to address the problem of search and retrieval of similar mold inserts. The mathematical-based exemplar developed has been explained in the following chapter.

CHAPTER 4

MATHEMATICAL MODELING FOR MOLD INSERTS RETRIEVAL

In order to address the limitation of tediousness of authoring exemplars faced in the Exemplar Approach1 and Exemplar Approach2, a Max-min exemplar approach is developed. The principle behind the approach is illustrated with an example. Consider a line segment AB of certain length and a tolerance envelope P Q S R of rectangular shape as shown in the Figure 4.1(a) and Figure 4.1 (b) respectively. It is desired to determine if the line can fit within the tolerance envelope. The minimum length of the line that can fit within the envelope is zero and the maximum length of the line that can fit within the envelope is the diagonal of the rectangle QS (or PR) which is given by the formula $\sqrt{l^2 + b^2}$. To frame the formula $\sqrt{l^2 + b^2}$, as the first step, a configuration of the line which has the longest length and can fit within the tolerance envelope is found. Then a mathematical expression is determined for the length of this line in terms of known parameters, which are the length and the breadth of the rectangle. If the length of the line AB falls between the maxima and minima ($\sqrt{l^2 + b^2}$ and 0), then it can be inferred that the line *can* fit within the envelope. Otherwise, it can not. Since maxima and minima of the known parameters are calculated in this approach, it is also referred to as Max- min exemplar in this thesis.

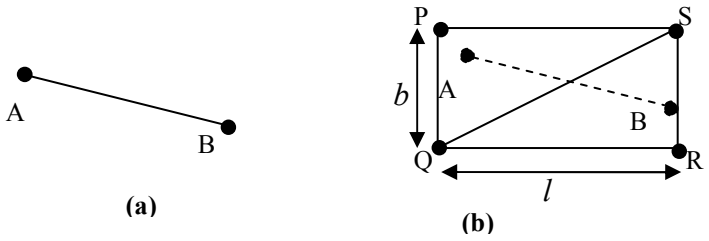


Figure 4.1 Concept of mathematical model

The Min-max exemplar has been developed for the implementation of design exemplar as search and retrieval tool. As the design exemplar can support equality and inequality relations, when a set of equality and inequality relations that can completely classify a mold insert to be a similar or non-similar can be developed, these conditions could be incorporated into design exemplar to use it as a search and retrieval tool. However, when a partially developed exemplar is run to find the mold inserts which have

radii less than a given value on a database of ten mold inserts, it took 30 seconds to obtain an output. Assuming that a fully developed exemplar would take a longer to run on this subset and much longer when run on a database of 5500 mold inserts, the idea of using the existing design exemplar technology is aborted. However, as this exemplar approach gives a satisfactory solution to the problem, albeit in a long time, it is decided to develop the exemplar as a standalone program. In writing an exemplar inspired program, in Visual Studio C++, the need to handle geometric entities, and their corresponding degrees of freedom, is eliminated. The database of mold inserts that is available is written such that the critical parameters, such as number of legs, leg lengths, arc radii, and angles between legs, are explicitly provided. The reduction in information and the ability to code explicitly is believed to yield a much more efficient search algorithm. Thus, the effectiveness of the design exemplar is combined with the efficiency of direct coding. The three main steps of the algorithm of the mathematical-based exemplar are presented below:

- Divide the target mold insert and the tolerance envelope around it into line-arc-line entities.
- Find maximum and minima of the parameters that can fit within individual tolerance envelopes obtained by division of the whole tolerance envelope.

To check if a mold insert is similar, verify if all the specifications of the mold insert fall with in the maxima and minima calculated.

These three steps are detailed in the following sections.

Step1: Division of Mold inserts into Line-Arc-Line Profiles

Before expressions for maxima and minima are developed for mold inserts, it is necessary to ensure that the expressions are general and not specific to a certain configuration. If different configurations of mold inserts formed by each increment of geometric entities arc and line, such as line-arc-line and line-arc-line-arc-line, need different formulae, then the development of mathematical model would not only become tedious and difficult but also moves from the intent of replacing the design exemplar tool. Hence, a degree of freedom analysis is done on the geometry of mold inserts to verify if general expressions can be developed that can be extended to all the mold inserts.

Consider a simple mold insert of line-arc-line profile as shown in the Figure 4.2. Here, various geometric entities, such as points, lines, and arcs, forming a mold insert are presented. Also, a tolerance envelope is drawn around it. The mold inserts that fall within this tolerance envelope are considered to be similar to this mold insert.

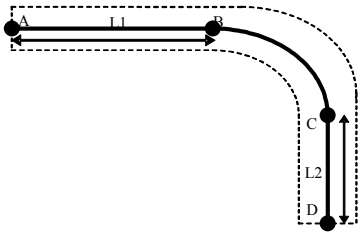


Figure 4.2Line-Arc-Line model of a mold insert

Table 4-1 shows the total degrees of freedom associated with the entities of the line-arc-line profile. The mold insert consists of several entities such as Point A, Point B, Point C, and Point D. Each of these points has two degrees of freedom (in 2D); translation along the x-axis and translation along the y-axis. Similarly each line has three degrees of freedom; translation along the axes and the angle. Each arc has seven degrees of freedom; x and y coordinates of the center, start, and end points and the radius. The degrees of freedom of all the entities contained in the model are summed to 21.

Table 4-1 Degrees of Freedom associated with a line-arc-line profile

Entities	**DOF**	**Description**
Point A	2	X and Y coordinates
Point B	2	X and Y coordinates
Point C	2	X and Y coordinates
Point D	2	X and Y coordinates
Line 1 (L1)	3	X and Y coordinates, angle of line
Line 2 (L2)	3	X and Y coordinates, angle of line
Arc	7	X and Y coordinates of center, radius and end points
Total	21	

These degrees of freedom are constrained through the set of relations.

Table 4-2 shows the tangency and incident constraints that have to be satisfied by these entities to form a line-arc-line profile. These constraints arrest 18 of the 21 degrees of freedom of the entities. Therefore, a mold insert of a line-arc-line profile, has three degrees of freedom in the space. Thus, it can be treated as a rigid body.

Table 4-2 Degrees of Freedom that can arrested

Constraints	DOF	Description
Incident (A, L1)	2	Constraints the x and y coordinates
Incident (D, L2)	2	Constraints the x and y coordinates
Distance (A, D)	2	Constraints the distance between A and D in a range (lower limit < L(A, D) < upper limit)
Distance (AB + BC + CD)	2	Constraints the total length of the profile in a range (lower limit < L(AB+BC+CD) < upper limit)
Incident (Arc, B, L1)	2	Constraints the x and y coordinates
Incident (Arc, C, L2)	2	Constraints the x and y coordinates
Angle (L1, L2)	2	Constraints the angle between L1 and L2 in a range (lower limit < angle(A, D) < upper limit)
Tangent (arc and L1, at B)	1	
Tangent (arc and L2, at C)	1	
Radius of Arc	2	Constraints the radius of the arc in a range (lower limit < rad(arc) < upper limit)
Total	**18**	

The three degrees of freedom associated with the mold insert in a tolerance envelope are the rotation of the mold insert in the two dimensional space within the envelope and the translation of the mold insert inside the envelope in horizontal and vertical directions. Also, a similar degree of freedom analysis is done on a line-arc-line-arc-line profile shown in Table 4-3 consisting of a line-arc-line-arc-line mold insert profile and a target envelope drawn around it.

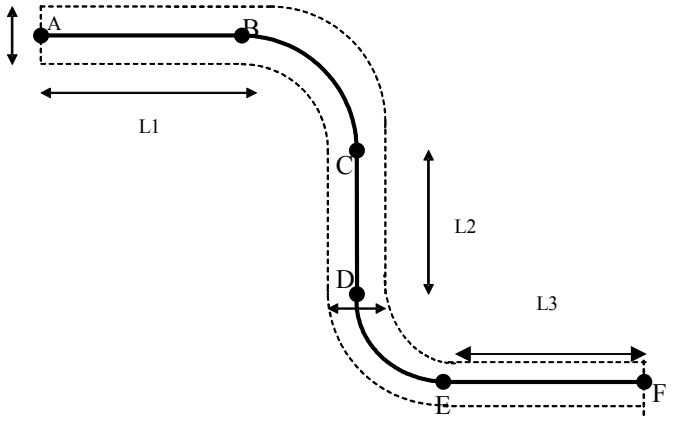


Figure 4.3 Line -Arc-Line-Arc-Line model of a mold insert

The total degrees of freedom that the points, lines, and arcs that form the model have are found in Table 4.3. The degree of freedom for the line-arc-line-arc-line mold insert is 35.

Table 4-3 Degrees of freedom associated with a line-arc-line-arc-line profile of a mold insert

Entity	**DOF**	**Description**
Point A	2	X and Y coordinates
Point B	2	X and Y coordinates
Point C	2	X and Y coordinates
Point D	2	X and Y coordinates
Point E	2	X and Y coordinates
Point F	2	X and Y coordinates
Line AB	3	X and Y coordinates, angle of line
Line CD	3	X and Y coordinates, angle of line
Line EF	3	X and Y coordinates, angle of line
Arc	7	X and Y coordinates of center, radius and end points
Arc	7	X and Y coordinates of center, radius and end points
Total	**35**	

These degrees of freedom are controlled through a series of constraints as found in Table 4-4. The number of degrees of freedom controlled is 32, a difference of three. Thus, the line-arc-line-arc-line insert may also be considered a rigid body with translation in the x and y directions and rotation about z.

Table 4-4 Degrees of freedom that can be arrested for a line-arc-line-arc-line profile.

Constraints	DOF	Description
Incident (A, L1)	2	Constraints the x and y coordinates
Incident (F, L2)	2	Constraints the x and y coordinates
Distance (A, D)	2	Constraints the distance between A and D in a range (lower limit < L(A, D) < upper limit)
Distance (AB + BC + CD+DE+EF)	4	Constraints the total length of the profile in a range (lower limit < L(AB+BC+CD) < upper limit) Constraints the total length of the profile in a range (lower limit < L(AB+BC+CD) < upper limit
Incident (Arc, B, L1)	2	Constraints the x and y coordinates
Incident (Arc, C, L2)	2	Constraints the x and y coordinates
Angle (L1, L2)	2	Constraints the angle between L1 and L2 in a range (lower limit < angle(A, D) < upper limit)
Tangent (arc and L1, at B)	1	
Tangent (arc and L2, at C)	1	
Radius of Arc BC	2	Constraints the radius of the arc in a range (lower limit < rad (arc) < upper limit)
Boundary(Arc, E, L2)	2	Constraints the x and y coordinates
Boundary(Arc, F, L3)	2	Constraints the x and y coordinates
Angle (L2, L3)	2	Constraints the angle between L1 and L2 in a range (lower limit < angle(E, F) < upper limit)
Tangent (arc and L2, at C)	1	
Tangent (arc and L3, at D)	1	
Radius of Arc DE	2	Constraints the radius of the arc in a range (lower limit < rad (arc) < upper limit)
Distance (C,F)	2	Constraints the distance between A and D in a range (lower limit < L(A, D) < upper limit)
Distance	2	
Total	**32**	

Therefore, in general, for any mold insert, the number of degrees of freedom associated with the mold insert within a tolerance envelope is three. As no extra degrees of freedom are present for the mold

insert presented in a two dimensional space within the tolerance envelope, the formulae developed for a simple mold insert with a line-arc-line will be extended to all the mold inserts.Following the principle of the mathematical model, these three degrees of freedom can be arrested by calculating the maxima and minima of the known specifications or known parameters of the mold insert. The known specifications of the mold insert are the:

- Radius of the mold inserts.
- Angle between the legs of the mold insert.
- Lengths of the legs of the mold insert.

Therefore, in order to assess if a line-arc-line mold insert from the database can fit within the tolerance envelope, the maximum and minimum values of the radius of the arc, angle between the legs, and the length of the legs are calculated. Also the additional condition that the end points of the legs should lie within end tolerance boxes gives rise to constraint on the maximum and minimum distance between the end points of the legs that can fit within the tolerance envelope. Then the specifications of the mold insert which must be verified for similarity are checked to determine if they fall within the maxima and minima. If all the specifications fall within calculated range, the mold insert is considered to be similar. Otherwise, it is deemed dissimilar. As stated before, since the mold insert of any general line-arc-line profile has three degrees of freedom, the formulae developed for a line-arc-line profile are easily extendable to the remaining profiles.

To explain this, a simple observation suggests that all the mold inserts consists of line-arc-line profile as a fundamental element. Therefore, if a mold insert is divided into line-arc-line elements, the formulae developed for a line-arc-line profile can be applied to these line-arc-line elements obtained after division. To apply the approach to a target mold insert which has the profile of multiple line-arc-lines, a tolerance envelope is drawn around the target mold insert. The tolerance envelope is then divided into line-arc-line profiles. The maxima and minima of the parameters that can fit into each of these tolerance envelopes with line-arc-line profiles are calculated. To check if a mold insert can fit within the whole tolerance envelope, the specifications of the mold insert are checked if they fall within the maxima and minima calculated for each individual tolerance envelopes. Figure 4.4 shows a mold insert with line-arc-

line-arc-line profile. To apply to it the formulae developed for a line-arc-line profile to it, the mold insert is divided into two line-arc-line profiles. Then the formula is applied to each line-arc-line profile separately to obtain their maxima and minima.

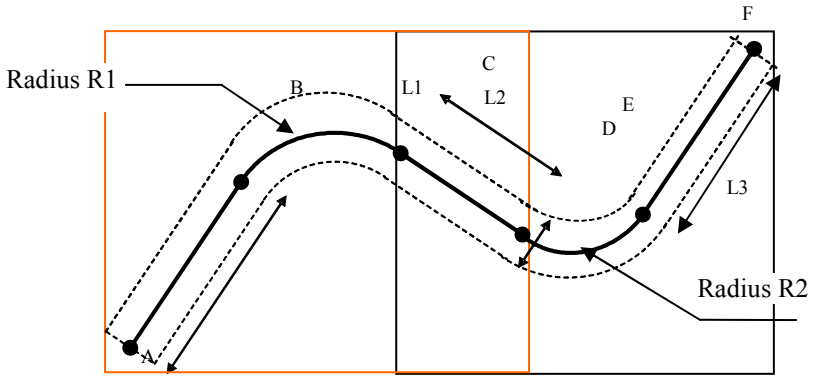


Figure 4.4 A complex mold insert divided into line-arc-line profiles.

Thus the first step in assessing the similarity of the mold inserts is to divide the tolerance envelope of the target mold insert into line-arc-line profiles.

Step 2: Calculation of the Parameters:

In this step, formulae to estimate maximum and minimum radius of the arc ,the maximum and minimum distance between the end points of the leg, ,and maximum and minimum lengths of the leg that can fit with a given tolerance envelope are framed in terms of known parameters of the mold insert which are tolerance values, the length of the leg, and the radius of the arc of the target mold insert. This section elaborates on the equations needed. However the order in which these parameters is dealt is different to make explanation easier.

Expression for maximum radius of the circle that can fit in the tolerance envelope:

To find the formula for the maximum radius of the arc that can fit within a given tolerance envelope, it is necessary to determine the configuration that represents the maximum radius of the arc. Consider a tolerance envelope ABCDEF as shown in the Figure 4.5. At a given point on the axis O1-Omax of the tolerance envelope, the arc with the greatest radius is the one which passes through that point and is

tangential to the lines of the outer boundary of the envelope B O1 and D O, as the arc can not be extended any further. Now, it is required to find out where exactly on the axis, the radius has the highest value.

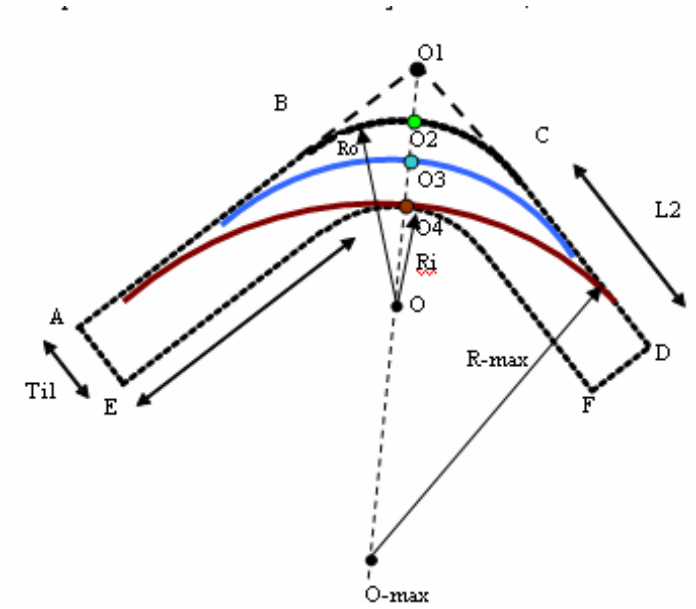


Figure 4.5 Configuration of the arc of the maximum radius that can fit within a tolerance envelope

Consider the outer boundary of the mold insert as shown in Figure 4.6(a). The tangent BO1, radius BG and the axis GO1 form a right angled triangle G O1 B. Also the tangent CO1, radius CG, and the radius GO1form a right angled triangle. The enlarged view of these triangles is shown in the Figure 4.6 (b). Let the distance between the points of intersection of the tangents O1 and point of intersection of the arc and axis O2 be l. Now from Figure 4.6,(b), in triangle GO1B.

$$\sin\alpha = \frac{r}{r+l}$$

$$l = r(1-\sin\alpha)$$

$$l \propto r \qquad \text{---------- (4.1)}$$

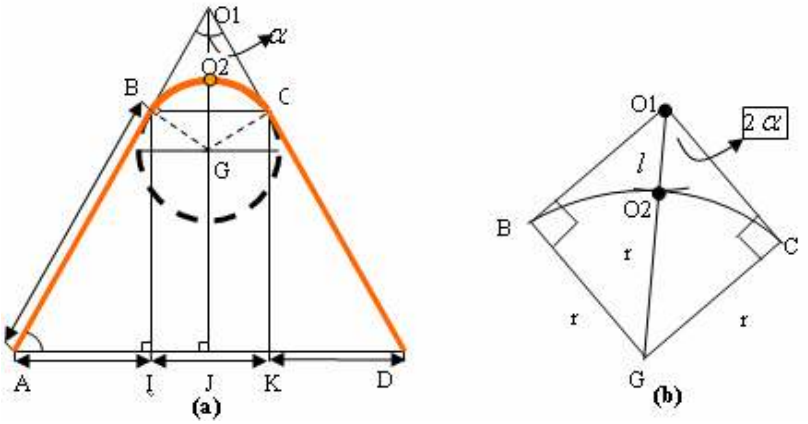


Figure 4.6 Mathematical derivation of the configuration for maximum radius of the circle

So, the length l which is the length between O1 and O2 is directly proportional to the radius of the arc. As the length increases, that is O2 moves away from O1, the radius of the arc that is tangential to both the legs increases. This can be clearly seen in the Figure 4.5. As the position of the point O1 shifts from O2 to O3 and O4, or as the distance between the points O1 and O2 increases, the radius of the arc increases. Since the maximum distance at which the point O2 can be position is at O4, the arc through O4 and tangential to the legs of the outer envelope determines the maximum radius. In general, the arc with the maximum radius is the one that is tangent to the either of the legs of the outer boundary of the envelope and is also tangent to the arc of the inner boundary of the tolerance envelope. The derivation of the expression for maximum radius of the arc is presented below.In Figure 4.7, let Ro and Ri be the radii of the outer and inner arcs, or circles in this case, of the tolerance envelope represented in red and purple colors respectively. The legs of the tolerance envelope are represented by blue dashed lines. The green circle with the radius R which is tangential to the legs of the outer envelope and the arc of the inner envelope represents the circle with the largest radius that can fit within a given tolerance envelope

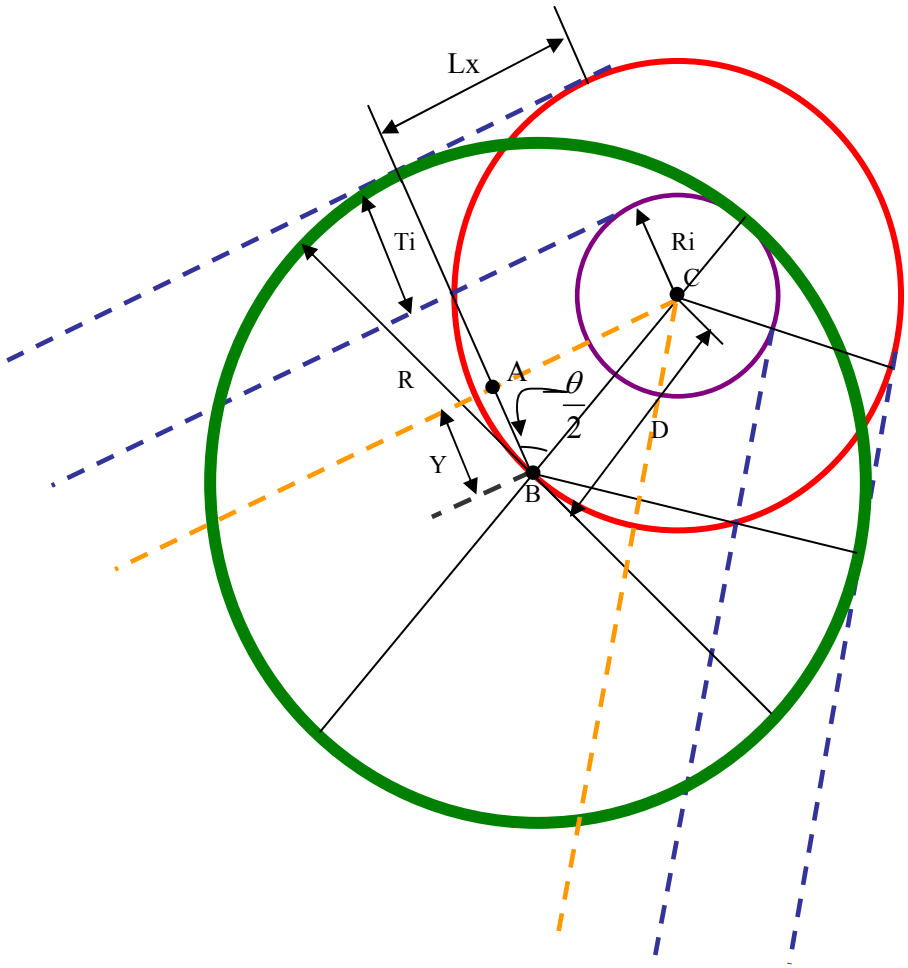


Figure 4.7 Configuration of the circle with maximum radius that can fit within a tolerance envelope

Before explaining the procedure to determine the radius of this circle ,a basic nomenclature is used during the explanation is provided. The nomenclature is followed by the procedure.

Nomenclature:

Radius of the green circle: **R**

Angle subtended by the curve at its center: *'θ'*

Primary tolerance value: **Ti**

Distance between the point where the green circle is tangent to the outer tolerance boundary and the point where the line is tangent to the outer curve: **Lx.**

Distance between the center of the green circle and the purple circle: **D.**

Yellow dashed lines are the construction lines drawn parallel to the legs of the tolerance envelope.

From Figure 4.7, it can be seen that,

R = D + Ri = Ri + Ti + Y.

Also, Lx=AC (since both of them are parallel)

From the triangle ABC,

$$also \ \tan\theta = \frac{Lx}{Y}.$$

$$so, Y \ = \frac{Lx}{\tan\theta}$$

$$\mathrm{R} = \ = \mathrm{Ri} + \mathrm{Ti} + \frac{Lx}{\tan\theta}.$$

So, substituting in the equation (1)

$$\mathrm{R} = (\mathrm{Lx} * (\tan\frac{\theta}{2})^{-1} + \mathrm{Ri} + \mathrm{Ti});$$

$$Lx^2 + Y^2 = D^2$$

$$Lx^2 + \left(\frac{Lx}{\tan\theta}\right)^2 = D^2$$

This further implies that,

$$\mathrm{R} = \sqrt{(Lx(\tan(\frac{\theta}{2}))^{-1})^2 + Lx^2} + \mathrm{Ri} = \mathrm{Lx} * \sqrt{(\tan\frac{\theta}{2})^{-2} + 1} + \mathrm{Ri}; \qquad (2)$$

But from equation (1) We have R = Ri + Ti + Y.

Comparing (1) and (2)

$$\mathrm{Lx} * (\tan\frac{\theta}{2})^{-1} + \mathrm{Ti} + \mathrm{Ri} = \mathrm{Lx} * \sqrt{(\tan\frac{\theta}{2})^{-2} + 1} + \mathrm{Ri}$$

$$\text{So, } \mathrm{Lx} * (\tan\frac{\theta}{2})^{-1} + \mathrm{Ti} = \mathrm{Lx} * \sqrt{(\tan\frac{\theta}{2})^{-2} + 1}$$

This implies,

$$Lx = \frac{Ti}{\sqrt{(\tan(\frac{\theta}{2}))^{-2}+1}-\tan(\frac{\theta}{2})^{-1}}$$

Substituting in equation (1) we get,

$$R = \frac{Ti*(\tan(\frac{\theta}{2}))^{-1}}{\sqrt{(\tan\frac{\theta}{2})^{-2}+1}-(\tan\ \frac{\theta}{2})^{-1}} + Ti\ + Ri \qquad (4.2)$$

Thus the maximum radius of the circle is dependent upon the radius of the inner tolerance envelope and the angle between the lines is:

Testing the Formula:

Consider a mold insert of the following configuration. These values are used to verify all the formulae developed.

Radius of the arc: 0.9 (for any parameter radius or the length of the leg, the units are distance valued (mm, cm, inches, etc.) so only quantity is mentioned for all the parameters without any units.)

Length of the first leg = 8.11

Angle subtended by the first leg = 30 degrees

Length of the second leg= 1.5

Angle subtended by the second leg=330degrees.

Primary and secondary tolerance values= .6

On application of the formula:

$$R=Ri+\frac{Ti*(\tan(\frac{\theta}{2}))^{-1}}{\sqrt{(\tan\frac{\theta}{2})^{-2}+1}-(\tan\ \frac{\theta}{2})^{-1}} + Ti = 0.6+0.6+\frac{0.6*(\tan(30))^{-1}}{\sqrt{(\tan 30)^{-2}+1}-(\tan\ 30)^{-1}} = 5.07$$

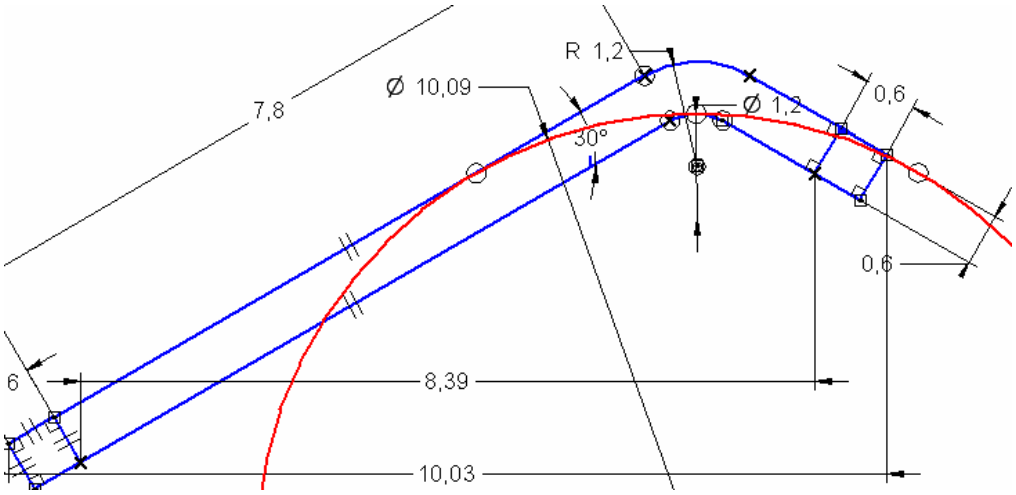


Figure 4.8 CAD model of the tolerance envelope with the arc of the maximum radius

Hence the radius of the largest arc obtained by substituting the parameters of the mold inset matches with the CAD model drawing in the Figure 4.8. So, the formula can be accepted as correct.

Expression to calculate minimum radius of the circle that can fit in the tolerance envelope:

Theoretically, the minimum radius of the circle that can fit within the tolerance envelope is zero. However, in the case of mold inserts, the minimum radius of the arc is determined by the process limitations of the manufacturer. The minimum radius of the arc that can be manufactured depends upon the tolerance value decided by the designer. For a tolerance envelope of tolerance Ti, the minimum radius of the arc that can be manufactured is Ti/2.

Expression to find maximum and minimum distance between end points of the legs .

The second degree of freedom that must be arrested is the angle between the lines. Since the angle between the lines is directly related to the distance between the end points of the mold insert, the maximum and minimum distance between the end points of the legs of the mold insert that can fit within tolerance envelope arrests this second degree of freedom.

For a mold insert to be considered similar, one of the conditions it should satisfy is that the end points of the mold insert should lie within the end tolerance boxes. Hence, the possible maximum and minimum distances between two points that lie on or within the end tolerance boxes is the maximum and minimum distance between the endpoints of the mold insert that can fit within the tolerance envelope. The

configuration needed for the calculation is explained using Figure 4.9which shows the tolerance envelope of a simple mold insert. The rectangular boxes AGHP and EFSD shown in the figure represent the tolerance boxes for the end points. For a mold insert to be considered similar it should lie within the tolerance envelope with its endpoints within these end tolerance boxes. AD and HF represent the maximum and minimum distance between any two given points that lie within or on the end tolerance boxes of the tolerance envelope

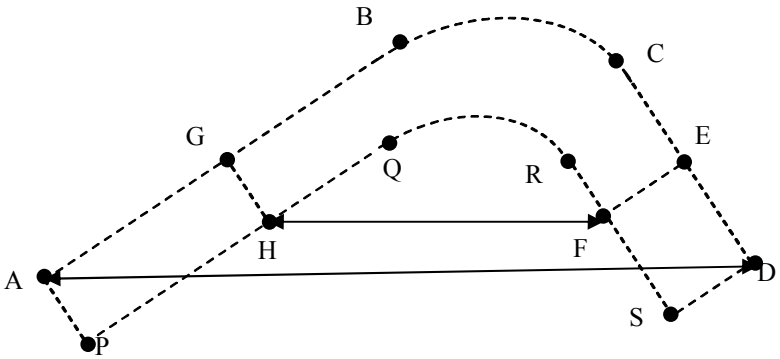


Figure 4.9 Maximum and minimum distance between end points of a similar mold insert

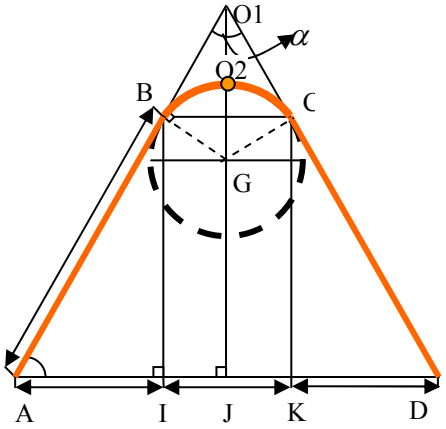


Figure 4.10 Line- Arc –Line mold insert model.

To calculate the maximum distance AD, consider the outer boundary of the envelope as shown in theFigure 4.10. The outer boundary of the tolerance envelope ABCD is represented in orange color. The legs AB and CD are extended to meet at O1. The extensions are shown in black color. Now, the distance between A and D can be considered to be a sum of the distances AI, IK, and, KD.

This implies, AD = AI + IK +KD;

Also, A I = KD, and IK = 2 *JK;

This implies, BF = 2JK + 2KD;

For triangle ABF, the lengths L1, L2 and angle θ is known. Hence, α = 90 – θ/2;

Hence length of BC can be found out. KD = L1 * cosine α;

As well, from quadrilateral, BGCO1, β = 90 - θ/2 = α

Hence, it can be inferred that JK = 2R * sine β.

HI = CE;

Hence, BF = 2L1 * cosine α + 2R * sine β;

Hence, BF = 2 (L1* cosine α + R * sine β);

So, the maximum distance between the two lineL1 and L2 when the included arc of radius R1 subtends an angle α1 and the line make an angle β1 with each other is:

D=2L1 * cosine α1 + 2R1 * sine α1. (4.3)

If the two lines are not of equal size, then the formula would be modified to

D=L1 *cos α1+L2 * cos α2+ R*sin α.2. (4.4)

To find the minimum distance, the parameters L1, L2 and R1 are replaced with the length of the legs and the radius of the arc between them which is given by:

D=(L1-Ti) *cos α1+(L2-Ti) * cos α2+ ((R-Ti)*sin α.2.

Testingthe Formula:

The formula has been verified on a test mold insert of the configuration mentioned below:

Radius of the arc: 0.9 (for any parameter radius or the length of the leg, the units are distance valued (mm, cm, inches, etc.) so only quantity is mentioned for all the parameters without any units.)

Length of the first leg = 8.11

Angle subtended by the first leg = 30 degrees

Length of the second leg= 1.5

Angle subtended by the second leg=330degrees.

Primary and secondary tolerance values= .6

L1max=L1 *cosα+ L2*cosα + 2∗R*sinβ = 8.41∗cos 30+ 1.8 cos 330+.9*sin60=10.04

D-min=7.81∗cos 30+ 1.2 cos 330+.9*sin60=8.40

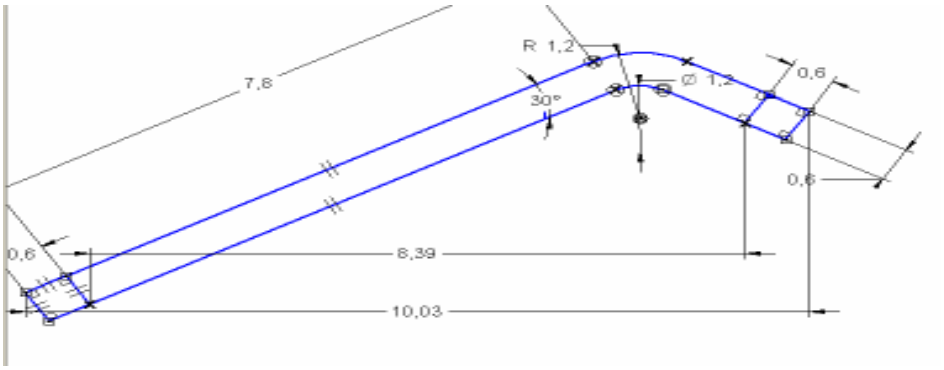


Figure 4.11 CAD model used for reference to check the validity of the formula

Since the values of the maximum and minimum distance between the legs in the CAD model shown in the Figure 4.11matches with value obtained on substituting the parameters in the formula developed, the formula can be accepted correct.

Procedure to find maximum length of the leg that can fit within the tolerance envelope:

To find a general formula for the tangent of maximum length is not possible due to the insufficient number of variables. Hence, a general procedure has been formulated which is presented here. The derivation below shows how to calculate the maximum length of the tangent that can fit within the envelope. The derivation has two steps: assigning a local coordinate system to the envelope and finding the length of the tangent with maximum configuration.

Step1: Assigning local coordinate system to the tolerance envelope

Since, the objective is to find the length which is an absolute quantity and does not depend on the coordinate system; a local coordinate system can be assigned to the mold insert. For convenience, the center of the mold insert that defines the tolerance envelope is fixed to be the origin.

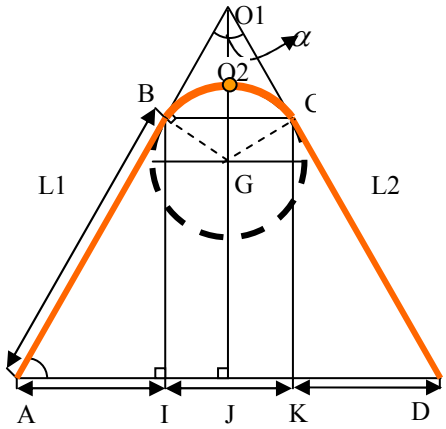


Figure 4.12 Target mold insert of Line profile.

From the Figure 4.12, one assigns the center to be G (0, 0),

Let the radius of the arc be R

Length of the legs be L1=L2=L,

Then, AJ= L* cos α+R*sinβ.

GJ=L*sin α –R *cos β.

So, the coordinates of the A = (-(L* cos α+R*sin β),-(L*sin α–R*cos β)) where B is in the fourth quadrant the negative sign appears in front of both the values.

Similarly, since lengths and angles of all the entities of the model are known, the coordinates of all the required entities can be found.

Step2: Finding the length of the tangent with maximum configuration.

The general procedure to find the equation of the entities and hence the length of the tangent is described. To find the maximum length of the tangent that can fit within the tolerance envelope, it is necessary to consider appropriate configuration of the line-arc-line entities where the length of the tangent that fits with in the envelope is maximum. Consider a target mold insert ABCD and a tolerance envelope drawn around it as shown in the Figure. To differentiate between them easily, the mold insert is represented in red color and its tolerance envelope is represented in black color.

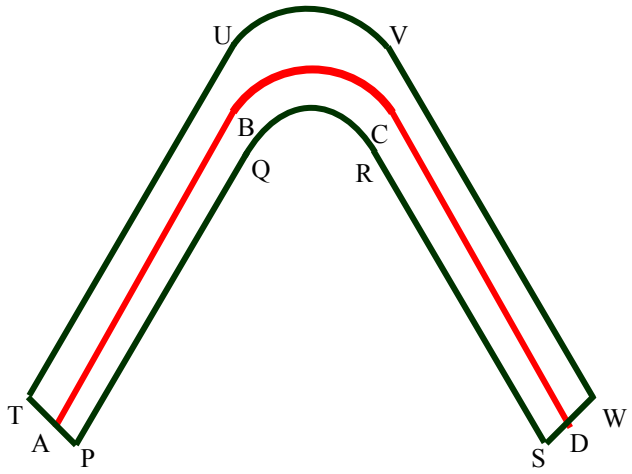


Figure 4.13 Target mold insert and its tolerance envelope

The configuration considered to calculate the maximum length of the leg is shown and explained below

L1- max R-min L2

***L1-max*:** It is the maximum length of the leg that can fit within the tolerance envelope. The configuration is explained using Figure 4.14. The leg starts from W and extending tills the outer boundary of the tolerance envelopes such that it is tangential to the arc of the inner boundary of envelope. If L2 is greater than L1, then the leg starts from the other corner of the outer boundary of the tolerance envelope.

***R-min*:** For the length of the leg to be maximum, the radius of the arc should be as small as possible as greater the radius of the arc, smaller would be the length of the leg. The minimum possible radius of the arc that can be manufactured for an envelope of a given tolerance value Ti is Ti/2 which is fixed by the manufacturing limitations. It may be also noted that the radii in the databases will never approach zero as the database consists of built mold inserts. Also, the arc should be tangential to the outer boundary of the envelope.

***L2*:** The configuration of the leg of maximum length is not dependent on the position and length of L2.

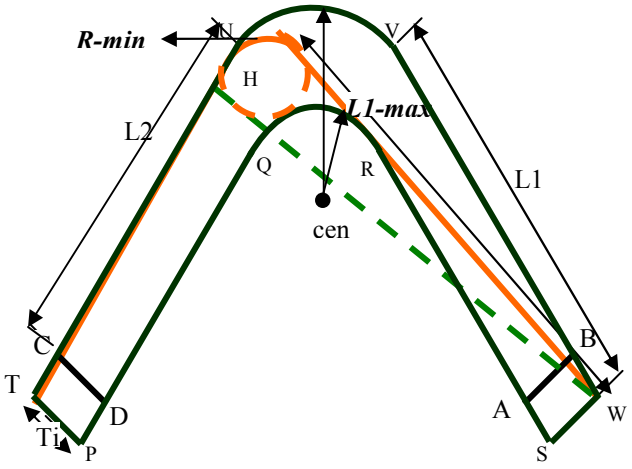


Figure 4.14 Configuration of line arc line entities such that the length of the line is maximum .

Equations of the entities are detailed below. Let (x_1, y_1) be the the external point W, then the equation to the pair of tangents from W to the circle is given by:

$$s_1^2 = s_{11}.s \qquad (4.5)$$

Where s represents the equation of the circle and

$$s_{11} = x^2_1 + y^2_1 - 2*x_1^2 - 2*y_1^2 + c = 0$$
$$s_1 = x_1 * x + y_1 * x - 2*x_1^2 - 2*y_1^2 + c = 0$$
$$s = x^2 + y^2 - 2*x_c * x - 2*y_c * y + c = 0$$

where (x_c, y_c) is the center of the arc of the inner envelope .

Out of the pair of lines represented by the above equation, the tangent which has greater slope is the tangent of our interest. Let $a_1x + b_1y + c_1 = 0$, be the tangent of interest. Fro example, the tangent from W, which is shown in dotted line, is the tangent of our interest. in Figure 4.14. To find the length of the tangent, it is necessary to find the equation of the circle, which is presented below. The locus of the center of the circle is given by either the line parallel to the L2 tangent and at a distance of $\pm T_i/2$ or the circle $x^2 + y^2 = (R_i + T_i/2)^2$. These are shown in Figure 4.15(a) and Figure 4.15 (b) in dotted lines respectively.

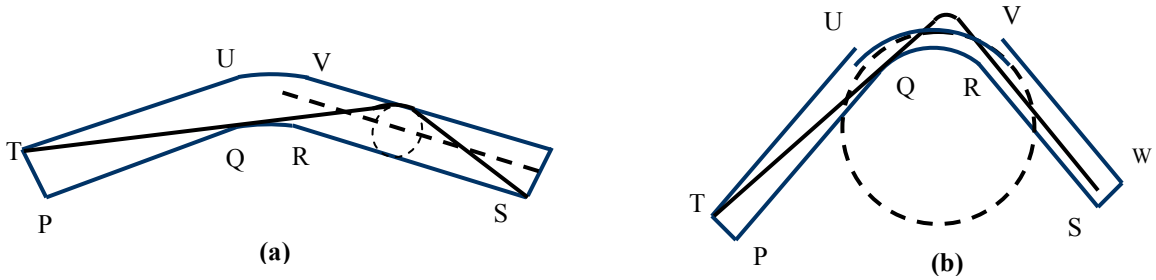


Figure 4.15 Possible Loci for L1max-Rmin-L2 condition

Solving the line ($a_1x + b_1y + c_1 = 0$) with the line and the circle and comparing their lengths with the lengths of the original envelope would give the center of the arc.

Once the locus of the center of the circle is found, the rest of the procedure involves the following steps:

- Find the center of the arc of the configuration
- Solve the equations of the circle and tangent to find the point of tangency.
- Calculate the length of the using Euclidian distance formula.

Testing the Procedure:

A sample calculation of how the length of the largest tangent that can fit within the tolerance envelope is provided. The value thus obtained is verified with the CAD model of the tolerance envelope drawn. Ads the values obtained from both the CAD model and the sample calculation matches, the method is thus validated.

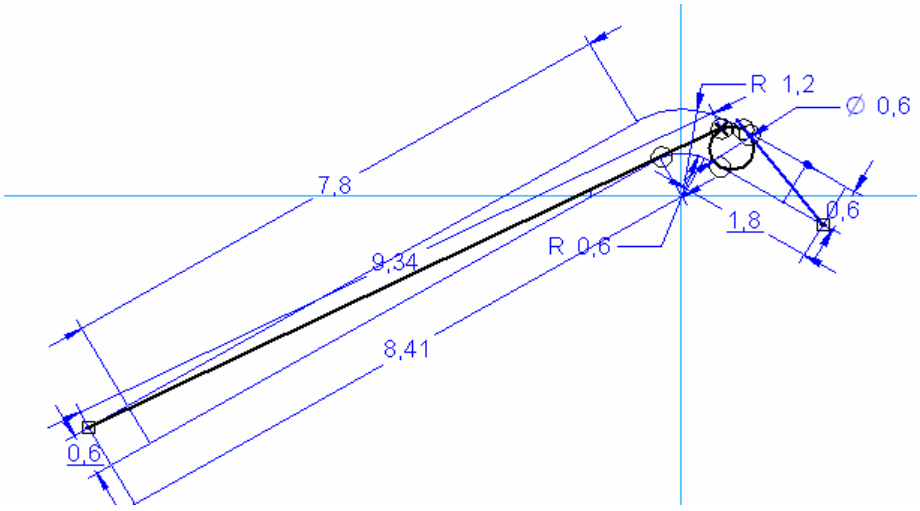


Figure 4.16 CAD model of a tolerance with the insert of the longest possible length that can fit within

The sample calculations are found here.

Step1: Finding the coordinates of the points:

Since out of the two legs, 8.1>1.5, the leg with the largest length lies in the tolerance envelope on the side of leg with length 8.1. Let the center of the arcs of the envelope be the origin.

Hence, coordinates of T= (-(8.4*cos30+1.2*sin30), -(8.4*sin30-1.2 *cos30)=(-7.72, -3.42)

Equation of circle= $x^2 + y^2 = .3^2$

The pair of tangents from T to the circle are given by: (x-2.057y+.68)(x-2.49y-.8)=0

Since the slope of the line (x-2.057y+.68) = 0 is greater between the two, it is the tangent of our interest.

Finding the center of the circle:

Let the center of the arc be (x1, y1).

Solving the equations of the tangent and leg2 of the outer boundary of the envelope, we get the point of intersection as (.67, 1).

Since, the length of the leg2 of the outer boundary until this point is 1.71<(1.5+.3), the locus of the center of arc of the mold insert with the maximum length of leg configuration is a line parallel to the second leg of the outer boundary at a distance of 0.3 from it. Since, the equation of the second leg of the

outer boundary is x+1.73y-1.8=0, Let locus be is x+1.73y-1.8+c =0, where c is a constant. Further, as it passes through the center, it should satisfy the condition,

x+1.73y -1.5=0. (4.4)

Also, perpendicular distance from the center to the tangent is equals to the radius, it should also satisfy the condition, $\frac{x1 - 2.057y1 + .68}{\sqrt{1 + 2.507^2}} = .3$ (4.6)

Solving (4.4) and (4.5), we get the values of (x1, y1) as (0.66, 0.66)

So, the equation of the circle is ($(x - .66)^2 + (y - .66)^2 = 0.3^2$).

So, the length of the largest tangent that can fit within the envelope is obtained by using the Euclidian distance, $(7.72 + .66)^2 + (3.42 + .66)^2 = 9.32$

The procedure of calculating maxima and minima of he three parameters the distance between legs, radius and the length of the tangent together constitute the mathematical model.

Maximum and minimum angle of the leg that can fit within the envelope:

The configuration of the leg that represents the leg with maximum deviation is shown in Figure 4.17. In Figure 4.17, the leg with starts from E and is tangential to the inner arc of the envelope is greatest possible deviation that a leg can have and still fit within the tolerance envelope. Let the deviation between this leg and the angle between the target mold insert be α . The procedure to calculate the deviation is similar to the procedure of calculating the length of the largest leg that can fit within the envelope is similar yo the procedure described to calculate the largest length of the leg that can fit within the envelope. For the arc of the inner tolerance envelope, a pair of tangents can be drawn from point E. The tangent of our interse is one with greater slope. The difference between the angles between this tangent and the angle subtended by the legs of the target mold insert is the maximum possible deviation that a leg that can fit within the tolerance envelope can have from the angle subtended between the legs of the target mold insert.

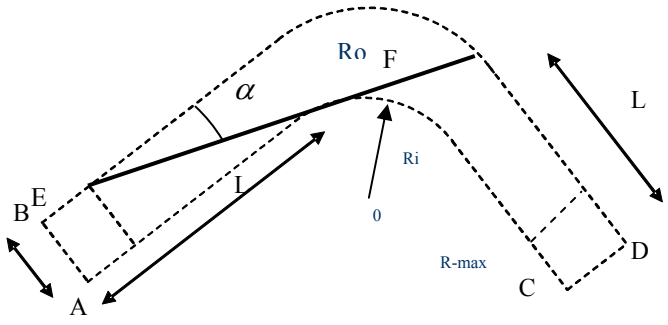


Figure 4.17 Configuration of the leg that has the maximum deviation

Results:

A mold insert is taken from the database and is queried for similar mold inserts. The result obtained from the algorithm is then cross checked using CAD models. Cad models of the tolerance envelope drawn for the target mold insert and CAD models of the retrieved mold inserts are drawn and are superimposed to see if the CAD model fit within the tolerance envelope satisfying the rest of conditions.

Query mold insert considered: Mold insert name in database: ACR13431

Description: 4 1.500001 329.999997 0.600002 1.500000 30.000090 0.599994 6.499998 9.999983 0.599994 2.000021 330.000353 0.

Explanation of the description: 4 is the number of legs, 1.5 is the length of leg, 330(approx) is the angle made by the leg with positive x-axis, 0.6 is the radius of the arc the leg is tangent to, and the length of the next leg is 1.5 and so on. So, the order of the description is the number of legs the mold insert has, the length of the leg, the slope of the leg, and the radius of the arc to which it is tangential.

Result obtained by the implementation of the algorithm developed is shown in the Figure 4.18

```
Enter the number of legs4
Enter the lengths of legs and their corresponding angles and the radius 1.5
330
1.5
30
6.5
10
2
30
radii.6
.6
.6
lamelle that fits in is ACR13432
lamelle that fits in is ACR13431
lamelle that fits in is ACR13428
lamelle that fits in is ACR13427
Press any key to continue_
```

Figure 4.18 Mold inserts retrieved from database using Mathematical Model developed

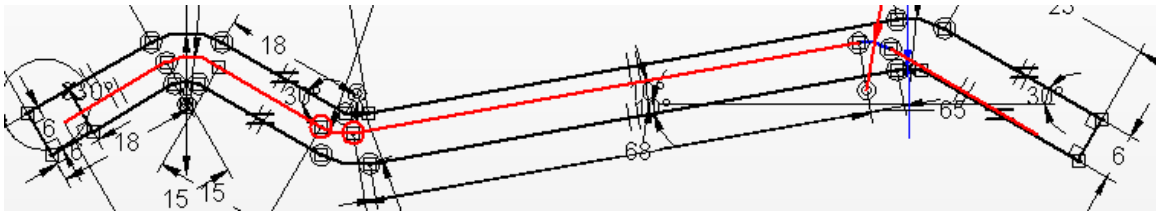


Figure 4.19 Verification of of the retrieved mold insert ACR13432 using CAD model.

Interpretation of the result:

ACR13431 is the target mold insert itself.

ACR13432, ACR13428 are mold inserts rotated through 180 degrees (flipped).

ACR13427. This is an exact match for the query in the database.

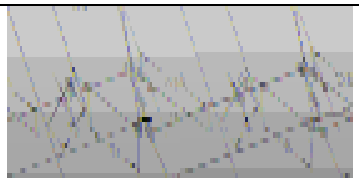
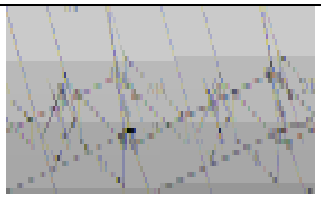
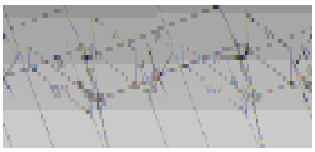
Testing:

To ascertain if the algorithm developed works, 15 different mold inserts are selected at random from the database. The database is then searched for similar mold inserts for each of the randomly selected ones using the developed algorithms. The results are then verified visually for false positives. Also a visual representation of query, matches and false positives is presented. The experiments are conducted on a computer with an Intel Pentium M 1.73 GHz processor and 512 MB of RAM. The Operating System is Windows XP. The code is implemented in C++ and compiled using Intel® C++ Compiler.

Mold insert configuration: ACR13432 4 1.500001 30.000003 0.600002 1.500000 329.999910 0.599995 6.499998 350.000017 0.599994 2.000021 29.999647.

Program output:

```
Enter the number of legs4
Enter the lengths of legs and their corresponding angles and the radius 1.5
30 1.5 330 6.5 350 2 30
radii.6 .6 .6
lamelle that fits in is ACR13432
lamelle that fits in is ACR13431
lamelle that fits in is ACR13428
lamelle that fits in is ACR13427
Press any key to continue_
```

Query	Matches	False positives
 ACR13432	 ACR13428  ACR13431, ACR13427	0

Number of false positives: 0.

Similar mold inserts found: ACR13431, ACR 13428, and ACR13427. (all are replicas).

2) Mold insert configuration: ACR13401 5 0.999999 359.999900 0.999984 1.500002 45.000006 1.000007 3.000000 314.999979 1.000006 1.500000 45.000036 0.999985 1.000001 0.000031 0.000000

Program output:

```
Enter the number of legs5
Enter the lengths of legs and their corresponding angles and the radius 1
0
1.5 45 3 315 1.5 45 1 0
radii1 1 1 1
lamelle that fits in is ACR13468
lamelle that fits in is ACR13401
lamelle that fits in is ACR13364
lamelle that fits in is ACR13232
lamelle that fits in is ACR13083
lamelle that fits in is ACR13012
lamelle that fits in is ACR13570
lamelle that fits in is ACR14837
lamelle that fits in is ACR15392
lamelle that fits in is ACR15500
Press any key to continue_
```

Similar mold inserts found: ACR13468, ACR13401, 13364, 13232, 13083, 13012, 13570, and 14837. All of these mold inserts are replicas of the target mold insert.

False positives: 15500

Figure 4.20 shows a mold insert that has to be verified manually to verify if it is a similar mold insert or a false positive.

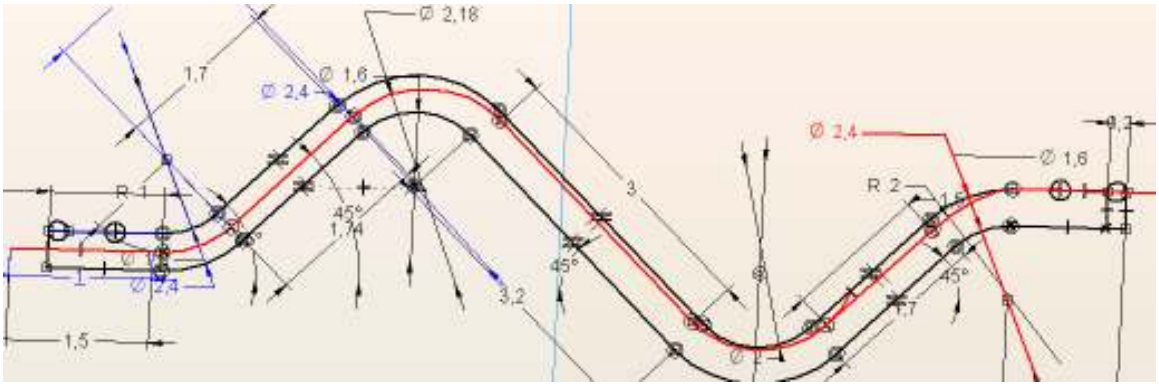
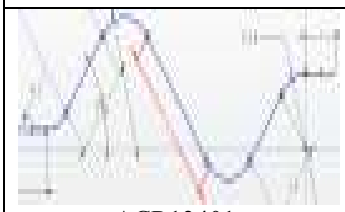
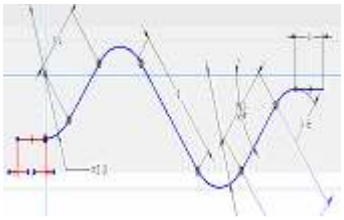
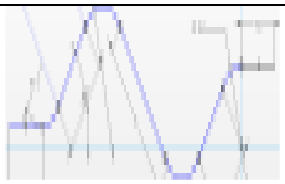


Figure 4.20 Manual verification is needed to know if mold inset LFD15500 is a false positive

Query	Matches	False Positives
 ACR13401	 ACR13468, ACR13083, ACR13012 ACR13570,  ACR13364, ACR13232, ACR14837	 ACR15500

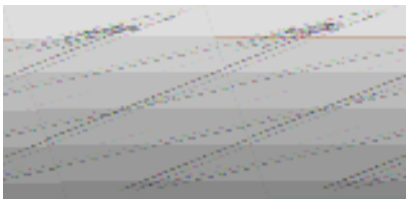
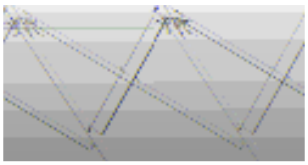
3) Mold insert configuration:: LFD906 2 8.000000 318.000000 0.900000 8.000000 42.000000

Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 8
42
8
42
radii.9
lamelle that fits in is LFD836
lamelle that fits in is LFD906
lamelle that fits in is LFD907
lamelle that fits in is LFD929
Press any key to continue
```

Similar mold inserts found: LFD836, (replica), LFD929.

False Positive: LFD907.

Query	Matches	False positives
 (LFD906)	 (LFD836)  (LFD929)	 (LFD907)

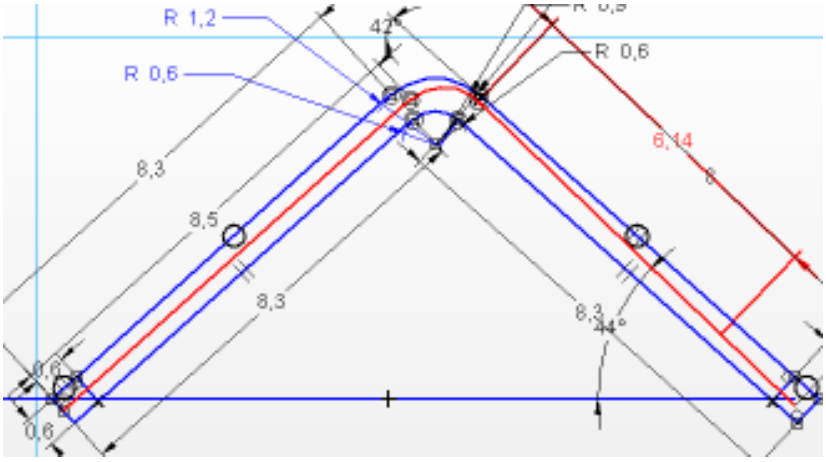


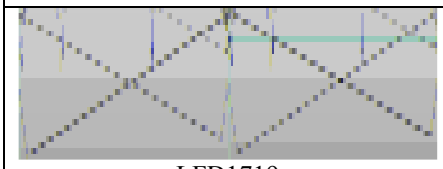
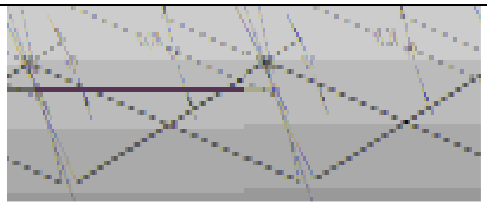
Figure 4.21 Mold insert LFD836 fits within tolerance envelope of LFD906

4) Mold insert configuration: LFD1719 0.600000 2 15.550000 345.000000 0.900000 15.670000 17.000000

Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 15.55
345
15.67
17
radii.9
lamelle that fits in is LFD1146
lamelle that fits in is LFD1719
Press any key to continue_
```

Similar mold inserts found: LFD1146 False Positives: 0

Query	Matche	FalsePositivs
 LFD1719	 LFD1146	0

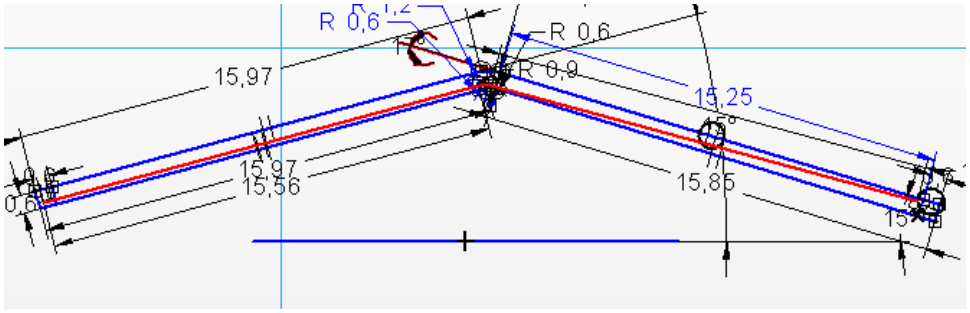


Figure 4.22 Verifying if mold insert LFD1146 is similar to mold insert LFD1719

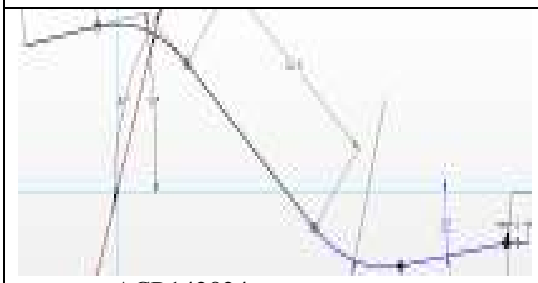
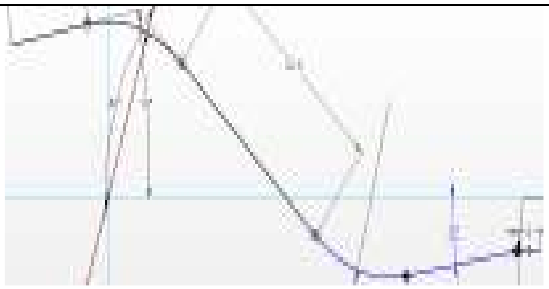
5) Mold insert configuration: ACR14284 0.600000 4 9.899995 9.999968 14.999916 22.800029 39.999999 15.000068 14.599975 10.000200 0.893144 3.000021 355.999524

Program output:

```
Enter the number of legs4
Enter the lengths of legs and their corresponding angles and the radius 9.9
10 22.8 40 14.6 10 3 356
radii15 15 .9
lamelle that fits in is ACR14284
lamelle that fits in is ACR14285
Press any key to continue
```

False Positives: 0

Similar mold inserts found: ACR14285 (replica)

Query	Match
 ACR142824	 ACR142825

5) Mold insert configuration: ACR14582 0.600000 3 4.999999 0.000020 0.900004 15.200026 322.999937 0.899662 4.500030 0.000328

Program output:

```
Enter the number of legs3
Enter the lengths of legs and their corresponding angles and the radius 5
0
15.2
323
4.5
0
radii.9
.9
lamelle that fits in is ACR14582
Press any key to continue_
```

No similar mold inserts found.

6) Mold insert configuration: ACR14568 0.800000 2 24.999992 355.000007 175.000137 17.000030 4.999978

Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 25
355
17
5
radii175
lamelle that fits in is ACR14568
lamelle that fits in is ACR14575
Press any key to continue_
```

Similar mold inserts found: ACR14575 (replica)

False positives present: 0

Query	Match
 ACR14568	 ACR14575

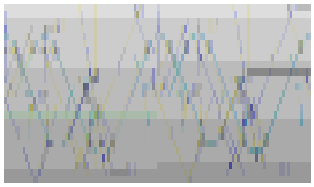
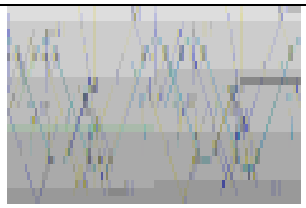
7) Mold insert configuration: ACR18833 0.400000 6 6.999999 359.999996 0.999991 1.500007 45.000145 1.000000 3.000017 314.999631 1.000008 3.000015 45.000291 0.999984 2.999908 314.999176 0.999924 1.500055 44.995898

Program output:

```
Enter the number of legs6
Enter the lengths of legs and their corresponding angles and the radius 7
0 1.5 45 3 314 3 45 3 315 1.5 45
radii1 1 1 1 1
lamelle that fits in is ACR13405
lamelle that fits in is ACR13369
lamelle that fits in is ACR13341
lamelle that fits in is ACR15510
lamelle that fits in is ACR18833
lamelle that fits in is ACR19876
lamelle that fits in is ACR90490
lamelle that fits in is ACR92371
Press any key to continue_
```

Similar mold inserts found: ACR13405, ACR 13369, ACR13341, and ACR90490.

False Positives present: 0

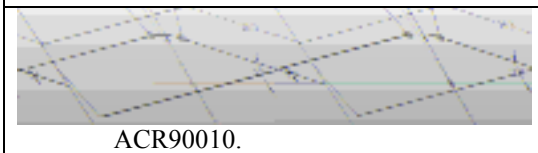
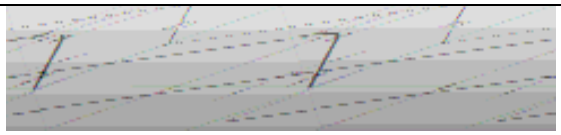
Query	Matches	False Positives
 ACR18833	 ACR13405, ACR13369, ACR13341  ACR90940, ACR92371	

8) Mold insert configuration: ACR90010 2 74.774502 12.630275 10.000230 26.355966 337.421178Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 75
12.6
26.3
337.42
radii10
    max.length,  len_tan_intrest[i]= 82.6832
lamelle that fits in is ACR90010
lamelle that fits in is ACR92194
Press any key to continue
```

Similar Mold inserts found: ACR92194. (replica)

.False positives present: 0.

Query	Match
 ACR90010.	 ACR92194

9) Mold insert configuration: LFD670 2 8.000000 335.000000 1.200000 8.000000 25.000000 0.000000

Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 8
25 8 25
radii1.2
lamelle that fits in is LFD1237
lamelle that fits in is LFD670
Press any key to continue_
```

False positives present: 0

Similar mold insert found: LFD 1237

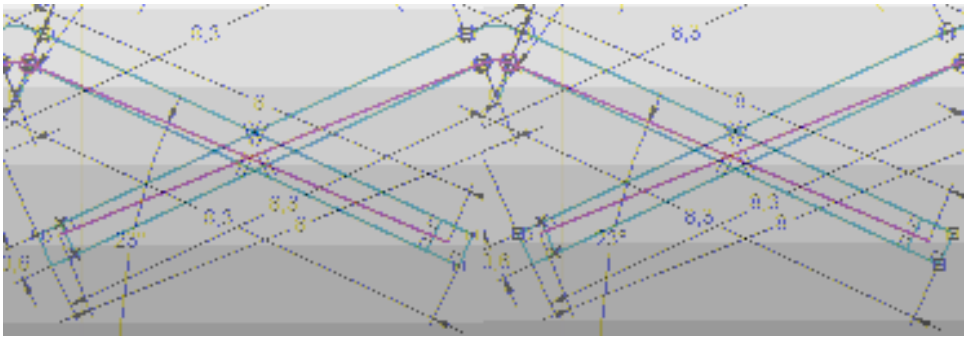


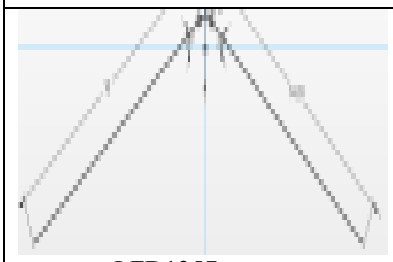
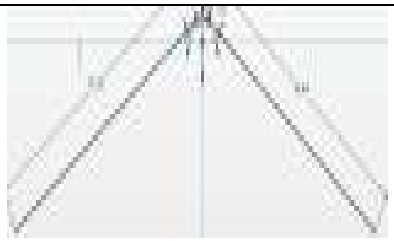
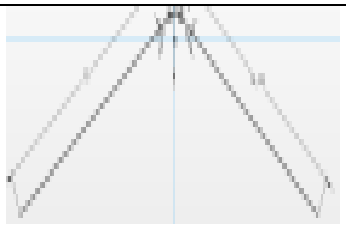
Figure 4.23 Verifying the similarity of LFD1237 with respect to LFD670

10) Mold insert configuration: LFD1357 2 14.080000 330.000000 1.200000 15.000000 28.000000

Program output:

```
Enter the number of legs2
Enter the lengths of legs and their corresponding angles and the radius 14.08
330
15
28
radii1.2
lamelle that fits in is LFD1005
lamelle that fits in is LFD1357
lamelle that fits in is LFD914
Press any key to continue
```

Similar mold inserts found: LFD1005, LFD914

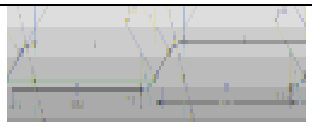
Query	Match	False Positive
 LFD1357	 LFD914	 LFD1005

11) Mold insert configuration: ACR15948 0.600000 3 17.682608 0.000755 3.001811 2.887646 43.823672 3.001413 18.559029 0.00074

Program output:

```
Enter the number of legs3
Enter the lengths of legs and their corresponding angles and the radius 17.68
0 2.88 43.82 18.55 0
radii3 3
lamelle that fits in is ACR15948
lamelle that fits in is ACR15947
Press any key to continue
```

False positives: 0.

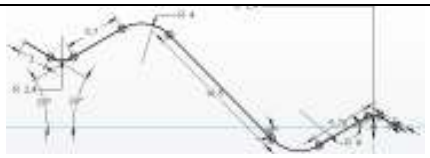
Query	Match
 ACR15948	 ACR15947

12) Mold insert configuration: ACR15743 5 3.000021 330.000205 1.199997 5.769956 29.998277 4.000365 14.795124 315.000233 3.999828 5.770183 30.000702 1.200072 3.000053 329.999929

```
Enter the number of legs5
Enter the lengths of legs and their corresponding angles and the radius 3
330 5.76 30 14.8 315 5.77 30 3 30
radii1.2 4 4 1.2
lamelle that fits in is ACR15213
lamelle that fits in is ACR15455
lamelle that fits in is ACR15156
lamelle that fits in is ACR15190
lamelle that fits in is ACR15743
lamelle that fits in is ACR15817
lamelle that fits in is ACR16701
Press any key to continue
```

Number of false positives found: 0

Similar mold inserts: ACR 15213, ACR15455, ACR15156, ACR15190, ACR15817, ACR16701.

Query	Matches
 ACR15743	 ACR 15213, ACR15455, ACR15156, ACR15190, ACR15817, and ACR16701.

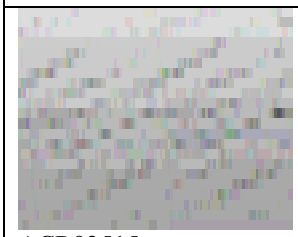
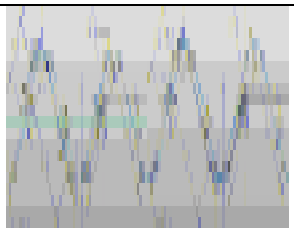
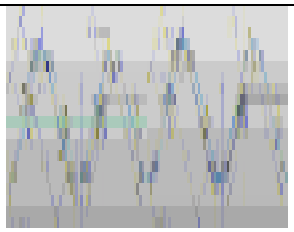
13) Mold insert configuration: ACR92515 0.600000 8 6.274521 0.174096 0.999868 1.956074 49.965487 1.000027 3.697099 304.999720 0.999980 3.698311 55.000216 0.999961 3.698346 304.995162 0.999893 3.698175 55.003520 1.000028 3.698108 304.998731 0.999868 1.846509 55.001550

Program output:

```
Enter the number of legs8
Enter the lengths of legs and their corresponding angles and the radius 6.27
0.17
1.95 50 3.69 305 3.69 55 3.69 305 3.69 55 3.69 305 1.84 55
radii1 1
1 1 1
1
1
lamelle that fits in is ACR15494
lamelle that fits in is ACR90095
lamelle that fits in is ACR90418
lamelle that fits in is ACR92173
lamelle that fits in is ACR92515
Press any key to continue_
```

Number of false positives: 0

Similar mold inserts found: ACR15494, ACR90095, ACR90418, and ACR921714) Mold insert

Query	Match	False positives
 ACR92515	 ACR92515, ACR90418, ACR92515	 ACR90095, ACR92173

14) Mold insert configuration: LFD1010 0.800000 3 18.800000 -0.000000 1.200000 5.470000 320.200000 1.200000 21.370000

Program output:

```
Enter the number of legs3
Enter the lengths of legs and their corresponding angles and the radius
18.8 0 5.47 320 21.37 0
radii1.2 1.2
lamelle that fits in is LFD1010
Press any key to continue
```

No similar mold inserts found

15) Mold insert configuration: LFD1028 0.600000 3 7.000000 320.000000 0.900000 10.000000 22.000000 0.900000 6.000000 320.000000

Program output:

```
Enter the number of legs3
Enter the lengths of legs and their corresponding angles and the radius 7
320
10
22
6
320
radii.9
.9
lamelle that fits in is LFD1028
Press any key to continue_
```

No similar mold inserts found.

Summary of the Experiments:

The results of the experiments thus obtained is summarized in Table 4.5

Table 4-5 Summary of the Experiments

Query	Number of legs	Matches Found	False Positives
ACR1342	4	ACR13428, ACR13431, ACR13427	0
ACR13401	5	ACR13468, ACR13083, ACR13012, ACR13570 ACR13364, ACR13232, ACR14837, ACR15500	ACR15500
LFD906	2	LFD836, LFD929. LFD907	LFD907
LFD1719	2	LFD1146	0
ACR142824	4	ACR142825	0
ACR14568	2	ACR14575	0
LFD1357	2	LFD914, LFD1005	LFD1005
ACR15948	3	ACR15497	0
ACR15743	5	ACR15947, ACR15213 ACR15455, ACR15156, ACR15190, ACR15817, ACR16701	0
1028	3	0	0
1010	3	0	0
ACR92515	8	ACR92515, ACR90418, ACR92515, ACR90095, ACR92173	ACR90095, ACR92173
ACR14582	3	0	0
ACR90010	2	ACR92194	0
LFD 670	2	LFD1237, LFD1341	LFD1341
ACR18833	6	ACR13405,ACR13369, ACR13341,ACR92372, ACR90490,ACR11510, ACR19876	ACR11510, ACR19876

Advantages and Limitations of Mathematical Model:

One major advantage of the mathematical model is that the complexity of the algorithm is low which is of the order O (n). The algorithm is fast when compared to the manual retrieval which takes several days, as reported by the mold designers, to navigate through the database to retrieve similar mold inserts. The algorithm could have been implemented using design exemplar tool but however there is a huge time gain when implemented in a standalone C++ program. This time gain is due to elimination of the requirement in handling spatial data in the C++ program. When the exemplar is implemented in design exemplar tool, much of the computer's processing time has to be dedicated to apply the incident and

tangential constraints on the geometric entities to form a line-arc-line profile. However the important task in the algorithm is to check if the mold insert in the database satisfies the set of conditions developed rather than building a mold insert from line-arc-line entities. Since the step of forming a line-arc-line profile can be completely eliminated in the hard coded program, there is a huge time gain. Also, the algorithm is rotationally and translationally invariant, that is it can identify and retrieve mirror images. The mathematical model has a limitation that it gives only a rough estimate about similar mold inserts and is not very accurate. For example a mold insert with both the largest arc radius and largest length of the leg will not fit within the tolerance envelope but is still a potential candidate for a similar mold insert according to the algorithm. Also, results are satisfying at low tolerance values but as the tolerance values increases, though it can be guaranteed that potential similar mold inserts may not be missed, the number of false positives would also be increasing.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This chapter provides a summary of the accomplished research together with its contributions. Also, some of the limitations of the proposed model were discussed. This chapter ends with a discussion on directions for the future work.

In this thesis, the applicability and suitability of the design exemplar tool for an industrial scenario was investigated with the search and retrieval of mold inserts as a case study. The core research question of this thesis is: *Can design exemplar be implemented to find potential candidates of mold inserts for a given target mold insert?* To address the research question several approaches were proposed to use the design exemplar as a search and retrieval tool. Two approaches proposed are a geometric constraint problem based where a set of conditions are imposed on one geometry to constrain it completely inside or outside another geometry so that conditions of similarity are met. If such a configuration is found then the mold inserts are said to be similar. The limitation of this exemplar approach that prevented the implementation of this approach is the tediousness of authoring exemplars for real world problems. Though authoring general exemplar solution for search and retrieval is possible, 18 different exemplars ought to be authored and networked to achieve this. Another foreseen difficulty was the handling of geometric entities while authoring exemplars. As the complexity of the exemplar increases in terms of the number of conditions imposed and geometric entities to be handled, it may become quite difficult to handle all of them together. During this approach, it was found that interchanging the match and extract parts of the exemplar query may result in reducing the time complexity of the query. However the tediousness in authoring exemplars still exist and hence a different approach was proposed.

In the third approach, a Min-max exemplar approach was proposed. In this approach, a set of maxima and minima were calculated based on the specifications of the target mold insert. The mold inserts present database and fall within these maxima and minima are considered similar. Though the approach gave a favorable result, implementation of the approach through design exemplar tool has an unreasonably high time complexity. Hence it was preferred to code the exemplar approach in C++ as this would remove

the burden of handling the spatial data which decreases the time complexity drastically. Hence the following were some of the limitations identified in the present exemplar technology that should be addressed so that it can be used in an industrial scenario:

- Tediousness of forming exemplars for real world problems.
- High time complexity when searched through large databases.

These issues have to be addressed to implement it as search and retrieval tool. . The following additions to the present system are suggested to enhance its capabilities:

- An automatic exemplar generator which generates exemplars of the geometric models either when selected or dropped down into CAD system could ease a great load of generating exemplars. This feature makes the design exemplar more user friendly and also saves lot of time while building queries.
- Ways to improve the time complexity of exemplars when applied as a search and retrieval tool on huge databases.
- Another suggestion that would help in better use of design exemplar is working on integration of design exemplar into commercially available CAD systems. If this can be achieved, design exemplar would provide the CAD users a good CAD query language that can be customized to their needs.

Therefore for the search and retrieval of mold inserts, in place of the using design exemplar tool, the principle of design exemplar that suites the application at hand was used which gave way to the mathematical model.

The mathematical model developed provides a simple and fast way to retrieve similar mold inserts with a few false positives. Since the main aim of the algorithm developed is to cut down the time taken for the search and retrieval process when compared to the manual search, the performance of the algorithm is satisfactory as it cuts down the number of mold inserts to be checked from a few thousands to a few such as ten in a second. Another advantage the mathematical model has is the property of rotational invariance. This property helps the model to identify similar mold inserts present in the database which are rotated

through an angle. However the mold inserts retrieved should be checked manually as they may also contain a few false positives. The mathematical model is essentially an exemplar which is hard coded to suit the requirements at hand. The declarative nature of the design exemplar still exists in the mathematical model even after hard coding. The set of conditions can rearranged and the result does not depend upon the order of these conditions. Hence the hard coded exemplar is still declarative and not procedural.

Though this model works fine for the application it has been developed for, the mathematical model can not be generalized. The algorithm has gave a satisfactory result because of the absence of great variation in the dimensions of the mold inserts present in the database. For example, the algorithm calculates the maximum length of the leg and the maximum radius that can fit within a given tolerance envelope and the mold inserts that fall within this maxima are considered similar. But it can be seen that a mold insert with both the maximum length of the leg and maximum radius or even close to these extremes can not fit within the tolerance envelope. Since, often there are not many mold inserts present in the database within the range of maxima and minima calculated, only few mold inserts are being retrieved as false positives. Also, if the experiment conducted on the 15 mold inserts which are randomly selected from the database is observed, often the similar mold inserts retrieved from the database are replicas of the target mold insert. Table 5-1shows the number of replicas present in the similar mold inserts retrieved for each mold insert during the experiment.

Table 5-1 Experiment reflecting that the majority of similar mold inserts are replicas of the target

Serial Number	Mold insert Name	Number of mold inserts retrieved from the database	Number of replicas present
1	ACR13432	3	3
2	ACR13401	8	8
3	LFD 906	3	1
4	LFD14284	1	1
5	LFD 1719	1	0
6	LFD92515	4	4
7	LFD 14568	1	1
8	LFD 18833	7	4
9	ACR 92194	2	1
10	ACR15948	6	1
11	ACR15743	5	3
12	ACR92515	0	0
13	LFD 101	0	0
14	LFD1028	0	0

From Table 5-1, it can be seen that 11 of the 15 mold inserts on which experiments are conducted have replicas in the database. Hence, it can be concluded that the database has many mold inserts clustered at certain dimensions rather than dimensions distributed over a range. The number of false positives retrieved by the algorithm would have been very high if the database has a large number of mold inserts with dimensions distributed all over. To generalize the algorithm in order to get accurate results in all circumstances, an optimization criterion introduced may be on the total circumferential length of the mold insert that can fit within the tolerance envelope. Currently for accurate results, the envelope fitting and the normalized distance function algorithms could be used. But they are computationally expensive and implementation of these algorithms needs extensive programming skills. An algorithm that gives accurate results has a variety of applications some of which are mentioned below[13]:

- "To identify the companies with similar growth pattern.
- To determine the products with similar selling patterns.
- Discover stocks with similar price movements
- Find portions of seismic waves that are not similar to spot geometrical irregularities."

Thus, this research has made the following contributions:

Identified the drawbacks of the existing exemplar system which have to be addressed to make it a good commercial CAD query tool.

Provided a new mathematical-based exemplar approach to identify similar mold inserts that can help a company to cut down its tooling expenses.

Closing Thoughts:

The design exemplar as a commercial CAD query tool is still in beginning stages. Though the principle of design exemplar is alluring for a CAD query tool, much effort is needed to bring it into the form of a commercial package. This thesis identifies some of the requirements needed to be fulfilled to achieve this. As this research is not a preplanned to be case study on the design exemplar and was started in search for a tool that can find similar mold inserts, not many areas have been be explored. Some of the questions that should be addressed are:

- If not integrated with the commercial CAD systems, can the design exemplar be used in the industry as a stand alone tool to query CAD models? If no, what aspects should be incorporated into design exemplar to be used as a stand alone CAD query tool?
- How much time would a new user take to learn to use design exemplar and what aspects of design exemplar should be improved to reduce this time?

APPENDIX

ALTERNATIVE METHODS

Also other algorithms are built for the search and retrieval of mold inserts by Wang [16]. In this chapter, these algorithms are discussed and are compared with the developed algorithm.

The Envelope Fitting Algorithm:

In this algorithm, the tolerance envelope along with the tolerance boxes is constructed for a target mold insert. Then the mold insert which has to be queried is positioned such that its starting point lies in the tolerance box at the beginning of the envelope and is translated and rotated in small steps within the 2D space to check if a configuration of the query exists such that the similarity conditions are satisfied. An example of this approach is shown in Figure 0.1. In the figureFigure 0.1, the target mold insert is shown in the red color. Tolerance boxes are constructed at the beginning and ending points of the mold insert. A mold insert from the database which is shown in blue color is translated and rotated to find if a configuration exists which satisfies the conditions of similarity.

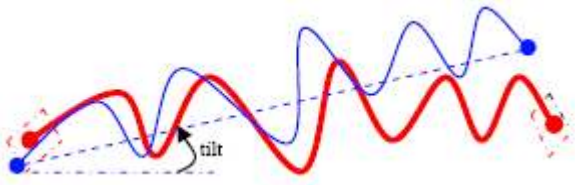


Figure 0.1 Thin query pattern is checked for similarity against bold target pattern. [14]

A match is returned if at least one configuration is found else not. The implementation of the algorithm consists of two steps:

Envelope Construction: In this step, the envelope is approximated into a set of poly-lines using simple offset operations for lower tolerance values. This approach works only if the primary tolerance value (Ti) is less than the radius R of the arc of the target mold insert. Else a more rigorous approach such as Minkowsi addition is applied.

Querying using Collision detection: In querying step, the starting point of the query is placed in the end tolerance box and is checked for the collision detection with the tolerance envelope.

The pseudo code of the algorithm is presented in the Figure 0.2

```
Name:   Envelope Fitting
Input:  1. Two patterns. T for target and Q for query;
        2. The envelope width w
        3. The number of samplings in every Degree Of
Freedom, N
Output: A Boolean indicating whether a match is found.
Procedure:
{
  pattern EN=construct Envelope (T,w);
  pattern EBL=construct End Box (T,w,left);
  pattern EBR=construc tEnd Box (T,w,right);
  range
{[XMin,XMax,YMin,YMax,AMin,AMax]}=findRange(EBL,
EBR);
  Align(EN,(0,0),0);
  Align(EBL,T(0),0);
  Align(EBR,T(1),0);

    for(x=XMin;x<=XMax;x+=(XMax-XMin)/N)
   for(y=YMin;y<=YMax;y+=(YMax-YMin)/N)
```

Figure 0.2 Pseudo code for envelope fitting algorithm[14]

The Normalized Distance Function Algorithm

The normalized distance function does not measure the similarity conditions but gives a measure of dissimilarity between two patterns which is obtained by summing up the distances between corresponding point pairs. A normalized distance function is defined for two patterns as :

$$\mathrm{D}(P,Q) = \frac{\int_0^1 [d(P_t, Q_t)]^2 dt}{[\min(L_p, L_Q)]^2} \qquad (5.1)[14]$$

Where P, Q are two patterns, parameterized according to the variable $t, (t \in [0,1])$.

L_p and L_Q are lengths of the patterns P and Q respectively, where the distance between the two points in two dimensional space is calculated using Euclidian distance formula.

A definition of the distance function is adopted so that sampling of a large number of points can be avoided. The patterns are placed initially before the dissimilarity is measured such that D is small but may not be minimum.

$$D_N(P,Q) = \frac{\sum_{i=0}^{N-1} d(P_{t_i}, Q_{t_i})^2}{[\min(L_p, L_Q)]^2} \qquad (5.2)[14]$$

$$\text{Where } t_i = \frac{i}{N-1}, i = 0,1,2,\ldots,N-1$$

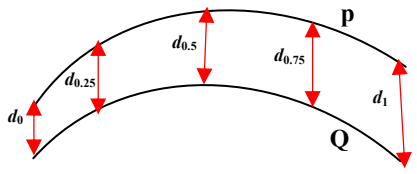


Figure 0.3 Distances between Sample Points, N=5[14]

For example, for the Figure 0.3, shown in ,if N=5 then,

$$D_5(P,Q) = \frac{d_0^2 + d_{0.25}^2 + d_{0.5}^2 + d_{0.75}^2 + d_1^2}{L_Q^2} \qquad (5.3)[14]$$

Name: Normalized Distance Function
Input: 1. two 2-D wireframe patterns P,Q, parameterized according to t.
2. Weighting factor w_t and number of sampling, N .
Output: A non-negative floating number D.
Procedure:

```
{
    Align(P);
    Align(Q);
    D=0;
    for(t=0;t<=N-1;t+=1/(N-1))
```

Figure 0.4 Pseudo code for Normalized Distance Function algorithm[14]

The results obtained with the normalized distance function are then analyzed using cluster analysis. as a measurement of dissimilarity Cluster analysis is a powerful technique in data mining and related areas to separate related items into groups according to some measurement of the similarity (or dissimilarity) between each pairs.

Comparison of the Algorithms:

Each of the proposed algorithms has its advantages and drawbacks. The min-max conditions method (Mathematical Model) is computationally fast, but the development cost is high. The result is not as

rigorous as the envelope fitting method. For example, in an experiment conducted to compare the three algorithms shown in Figure 0.5, the min-max conditions method returns two matches, while only the first one (second from top) fits inside the envelope and the second one (the first from bottom) is a false positive. Furthermore, it is not easily extendable to other scenarios where the definition of similarity is different. The envelope fitting method is computationally complex. Also, it is not flexible. The main advantage of this algorithm is that it is rigorous. The normalized distance function method is both computationally fast and flexible, but not rigorous. It is difficult to correlate the normalized distance with the requirement that the polyline pattern resides within an envelope. The normalized distance function method is the only approach that can be used in the cluster analysis. Although the current result is satisfactory, the threshold needs to be chosen by a more elaborate process which takes three things into account: the intrinsic characteristics of the dissimilarities matrix (statistical measurements), extrinsic characteristics (user supplied rules) and the adaptive process (trying different threshold).

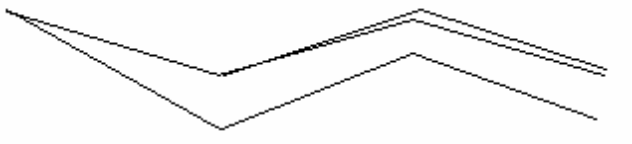


Figure 0.5 Experiment displaying the results of the three algorithms[14]

REFERENCES

1. Veltkamp, R.C. and M. Hagdoorn, *State-of-the-art in Shape Matching*. 1999: Univ.; Niedersächsische Staats-und Universitätsbibliothek.
2. Leizerowicz, W., J. Lin, and M.S. Fox, *Collaborative Design Using WWW*. Proceedings of WET-ICE, 1996. 96.
3. Iyer, N., et al., A Reconfigurable 3D Engineering Shape Search System Part I: Shape Representation. ASME DETC, 2003. 3.
4. Cardone, A., S.K. Gupta, and M. Karnik, *A Survey of Shape Similarity Assessment Algorithms for Product Design and Manufacturing Applications*. Journal of Computing and Information Science in Engineering, 2003. 3: p. 109.
5. McWherter, D., et al., *Database techniques for archival of solid models.* Proceedings of the sixth ACM symposium on Solid modeling and applications, 2001: p. 78-87.
6. Kulak, O., A complete cellular manufacturing system design methodology based on axiomatic design principles. Vol. Volume 48. 2005. 765 - 787
7. Putti, S., Dynamic Networking of Design Exemplars:Towards a Mechanical Design Visual Programming Language. 2007.
8. Putti, S., Dynamic Networking of Design Exemplars: Towards a Mechanical Design Visual Programming Language. 2007, Clemson University: Clemson.
9. Joshua.D.Summers, Development of a Domain and Solver Independent Method for Development of Mechanical Engineering Embodiment Design, in Mechanical Engineering. 2004, Arizona State University: Arizona.
10. Divekar, A. and J.D. Summers, Logical Connectives For A Cad Query Language: Algorithms And Verification, in ", Design Engineering Technical Conferences,, ASME, DETC: Illinois, Chicago, USA.
11. Summers, J.D. and J.J. Shah, EXEMPLAR NETWORKS: EXTENSIONS OF THE DESIGN EXEMPLAR, in Design Engineering Technical Conferences, Computers in Engineering,, DETC-2004: Salt Lake City, UT,.
12. Divekar, A., The Design Exemplar: Foundation for a CAD Query Language. 2004, Clemson University: Clemson.
13. Aggarwal, R., Fast Similarity Search in Presence of Noise, Scaling, and Translation in Time Series Databases, IBM Almaden Research Center.
14. Wang, M. Polyline Similarity Query Algorithms Applied To Search And Retrieval Of Mold INSERTS. in International Design Engineering Technical Conferences & Design Automation Conference. 2007. LasVegas, Nevada,USA: DETC.