A

DISSERTATION REPORT

on

AUTOMATIC TOOL PATH GENERATION FROM B-REP MODEL OF SCULPTURED SURFACES USING CAD SOFTWARE API

Submitted in partial fulfillment of the requirements for the award of degree of

MASTER OF ENGINEERING

IN

CAD/CAM

Submitted by:

JASPREET SINGH NAGRA

Roll No: 801281010

Under the guidance of

Mr. AJAYINDER SINGH JAWANDA

Associate Professor, MED

Mechanical Engineering Department
THAPAR UNIVERSITY
PATIALA-147004, INDIA

JULY 2014

CERTIFICATE

I hereby certify that the thesis entitled "**Automatic Tool Path Generation From B-Rep Model of Sculptured Surfaces Using Cad Software API**" is an genuine record of my own work carried out in completing the thesis requirement for the degree of **Master of Engineering** in **CAD/CAM Engineering** at Thapar University, Patiala, under the guidance of **Mr. Ajayinder Singh Jawanda**, Associate Professor, Mechanical Engineering Department, Thapar University, Patiala.

Date: 17/07/14

Jaspreet Singh Nagra
(801281010)

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Date: 17/7/2014

Mr. Ajayinder Singh Jawanda
Associate Professor, MED,
Thapar University, Patiala
Punjab-147004

Countersigned by: -

Dr. S.K. Mohapatra
Senior Professor and Dean of Academic Affairs,
Mechanical Engineering Department,
Thapar University, Patiala
Punjab-147004

Dr. Ajay Batish
Professor and Head,
Mechanical Engineering Department,
Thapar University, Patiala
Punjab-147004

ACKNOWLEDGEMENT

I express my sincere gratitude to my guide **Mr. Ajayinder Singh Jawanda**, Associate Professor, Mechanical Engineering Department, Thapar University, Patiala, for his invaluable guidance, proper advice, and constant encouragement during the course of my work in this thesis report.

I would also express my gratitude to **Dr. Ajay Batish**, Head, Mechanical Engineering Department for his inspirational guidance and wholehearted corporation which was helped me to present this thesis.

I also thank my friend Mr. Vishwas Mahara for his moral support and useful reviews.

I thank the entire faculty and staff of Mechanical Engineering Department, Thapar University, for their help and moral support.

The author acknowledges the use of facilities of State Initiated Design Centre (SIDC) for woodworking of Mechanical Engineering Department, Thapar University, Patiala, set up under financial support from Handicrafts Division of Ministry of Textile, Government of India.

(Jaspreet Singh Nagra)

ABSTRACT

Conventional CNC machines have been used for machining complex geometries like dies for ergonomically designed parts or complex surfaces. The tool paths generated for these are extremely complex. The traditional paradigm of CAD to machined part uses a specialized CAD package along with an expert CAD designer. This expert gives output of geometry information in the form of 3-D model to a CAM software, which is operated by another expert in CAM. The CNC programmer then converts the tool paths that are generated in CAM software into codes using commands. An expert machinist different from the former experts does the machining which adds to the discontinuity in the flow of data and design requirements. This has inherent problem of multiple data types, multiple softwares, and multiple personnel causing problems in data transfer from one-step to other. The synchronization of these separate entities is a challenge.

This thesis is an attempt to directly generate tool positions on a sculptured surface represented in CAD using B-rep and generate the tool path for a lathe mill, which can be used for tool positioning. This eliminates the CAD and CAM software, the experts required for CAD, CAM and machining in the conventional paradigm for machined part from CAD geometry. The past works use a neutral STL format to read the CAD geometry and generate a tool path for it. Use of STL, which contains faceted model data, has inherited errors like chordal error, which are minimized by increasing the density of triangles but that increases computations and processing time required for generating tool paths.

This thesis makes an attempt at direct generation of the tool path using custom algorithm for pseudo-symmetric parts for the lathe mill directly in CAD software where B-rep geometric model is available. This eliminates STL file errors and subsequent processing of STL. The comparison of generation of the tool path using our developed method that uses B-rep model and the already published “Ball Drop method” [1] that uses STL model is done.

TABLE OF CONTENTS

TABLE OF FIGURES

ABBREVIATIONS

API :	Application Programming Interface
STL:	Standard Tessellation Language
CAD:	Computer Aided Design
CAM:	Computer Aided Manufacturing
CAE:	Computer Aided Engineering
CL:	Cutter Location
CC:	Cutter Contact
B-REP:	Boundary Representation
SCALM:	Single Controlled Axis Lathe Mill
MACRO:	Merged And Correlated Recorded Output
VBA:	Visual Basic for Applications
STEP:	Standard for the Exchange of Product model data
GUI:	Graphical User Interface
TPG:	Tool path generation

CHAPTER 1

INTRODUCTION AND LITERATURE SURVEY

The traditional paradigm involves CAD software, CAM software, CNC programming language and human beings that are experts in CAD, CAM, and operating CNC machine. The primary need is to eliminate these individual entities and facilitating only one person and one software to handle all the work. The secondary need of using API is to generate tool path, which will give us the exact shape of the 3-D model upon machining because the conventional method for generating tool path require 3-D model in faceted form as in STL format. These paradigms involve loss of data, inaccuracies in input formats and complex structures of databases that lead to inaccurate machined parts.

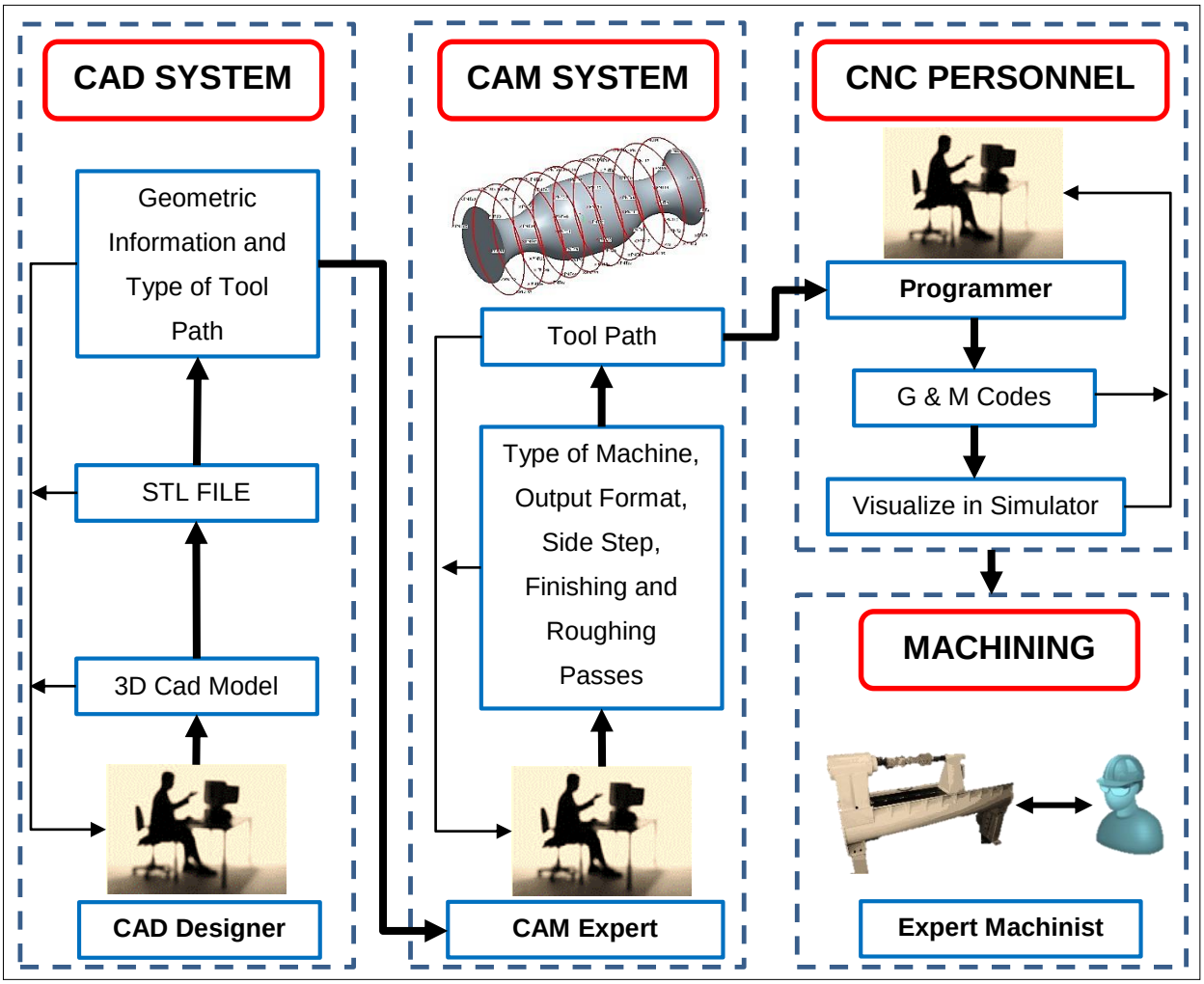


Figure 1.1 Traditional paradigm of CNC machining

1.1 CAM and its problems

Tool path generation using CAM software is computer aided process and is not fully automated. An expert user is required to identify volume of material, which has to removed, who also identify tool that has to be used. The user has to give various inputs such as feed rate, depth of cut, material removal rate (MRR), spindle speed. The traditional CAM requires all the inputs from the expert in use of the CAD package to define all parameters of machine, tool, work piece and all other parameters of the machining. Step-by-step machining is done based on the steps defined by the CAM expert. The CAM expert has to be knowledgeable about the CAD systems. The CAM expert has to translate the model from virtual environment to machined part in the physical environment. Many researchers have done work to simplify this process by removing the CAM software as well as the expertise required. This is done by direct use of CAD for generating tool positions, which is placed along a footprint, and giving the required tool path.

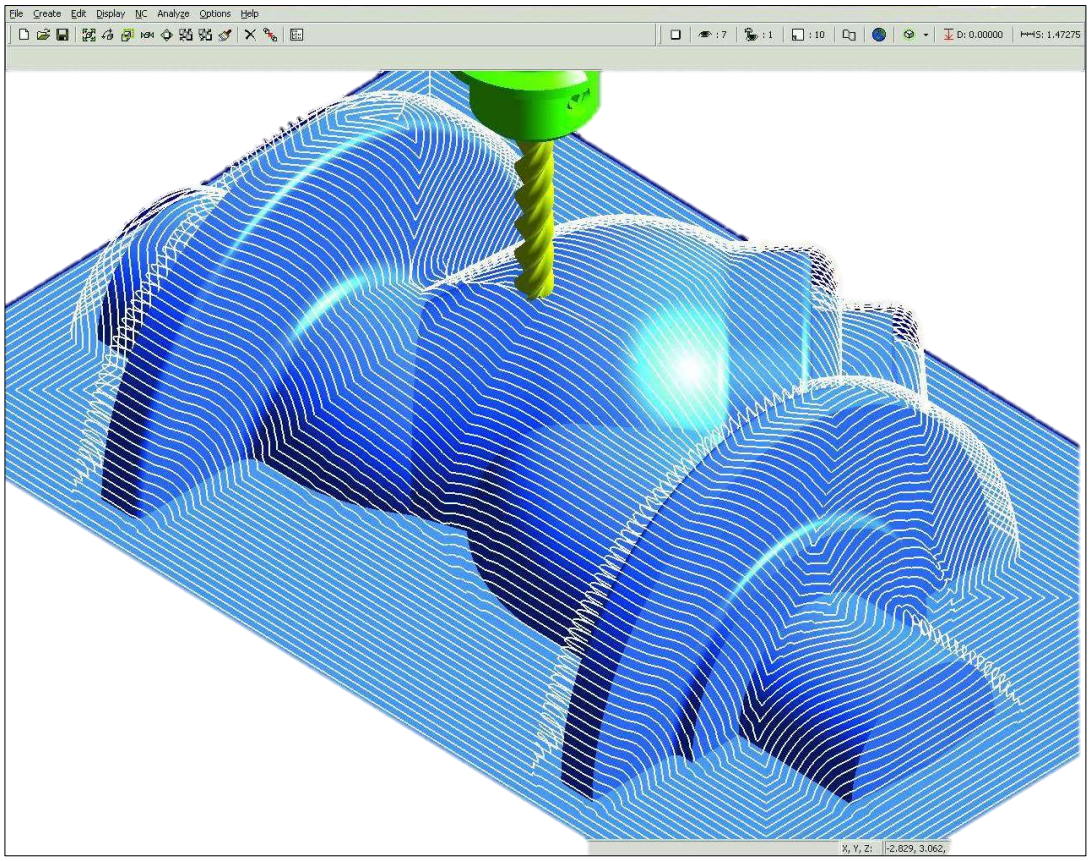


Figure 1.2 Tool path generated in CAM software [Web 14]

1.2 Tool path generation directly from CAD

Form CAD, the tool paths are generated using tool footprint and along the footprint taking increments at which finding the tool positions that are gouge free is called as tool path planning. This strategy is used to eliminate CM from the traditional paradigm. Further, these tool paths are directly fed into the CNC machine, which can position the tool at those positions.

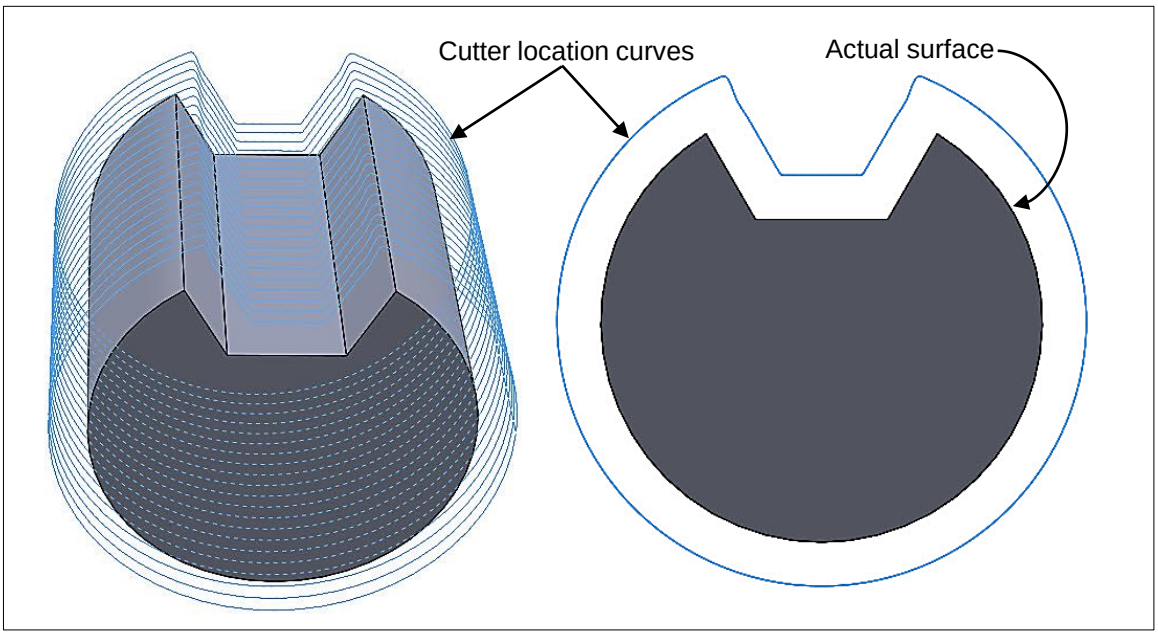


Figure 1.3 Representation of tool path

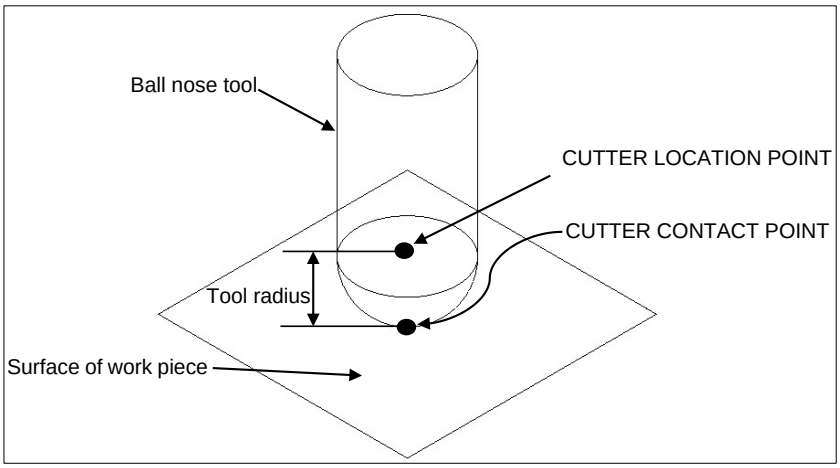


Figure 1.4 Configuration of tool path for ball nose cutter

1.2.1 Basic terminology

CC-point: It is the point where tool makes tangential contact with the surface of work piece.

CL-point: It is the reference point that is used for specifying location of the tool.

CL-surface: It is the surface that is defined by the trajectory formed by CL-points.

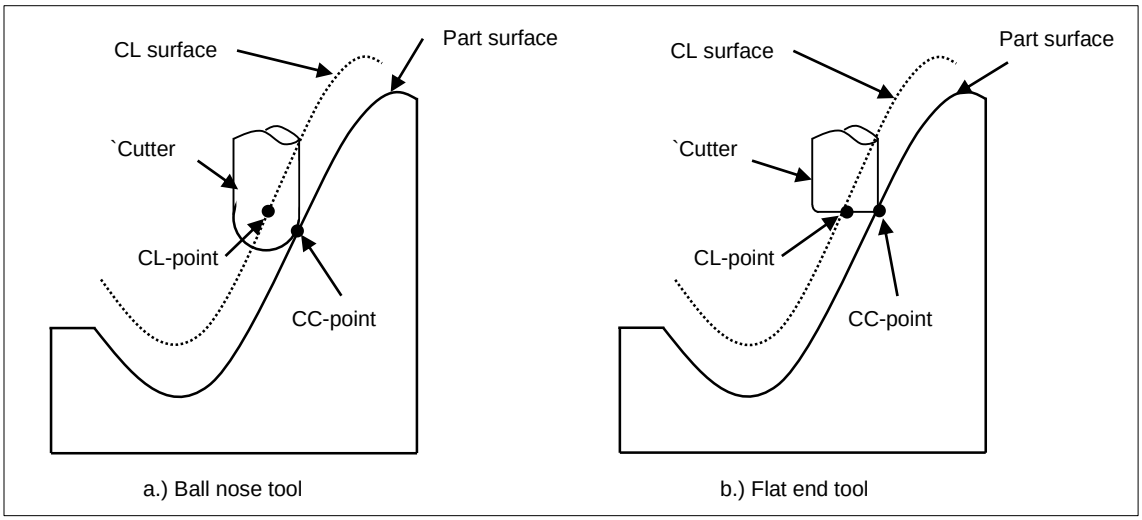


Figure 1.5 Basic tool path terminology

Cutting-condition: It is defined by feed-rate and spindle speed.

NC-path: NC-path is a sequence of CL-points.

Tool-path resolution: The tool path resolution is defined by pitch and increment in z-direction.

1.2.2 Tool path-planning domains

1) **Parameter-domain (PD):** In this domain, the NC path is generated using parametric equations of the curves on the surface of work piece as shown in Figure 1.6 (a) below. The parametric curve is represented by r (u,v).

2) **Guide-plane (GP):** The NC path is first created on a separate plane, which is guiding the cutter. Then the NC path is projected on the actual surface of the 3-D model.

3) **Drive-surface (DS):** This method utilizes the intersections between drive surfaces and r(u, v) to create NC path.

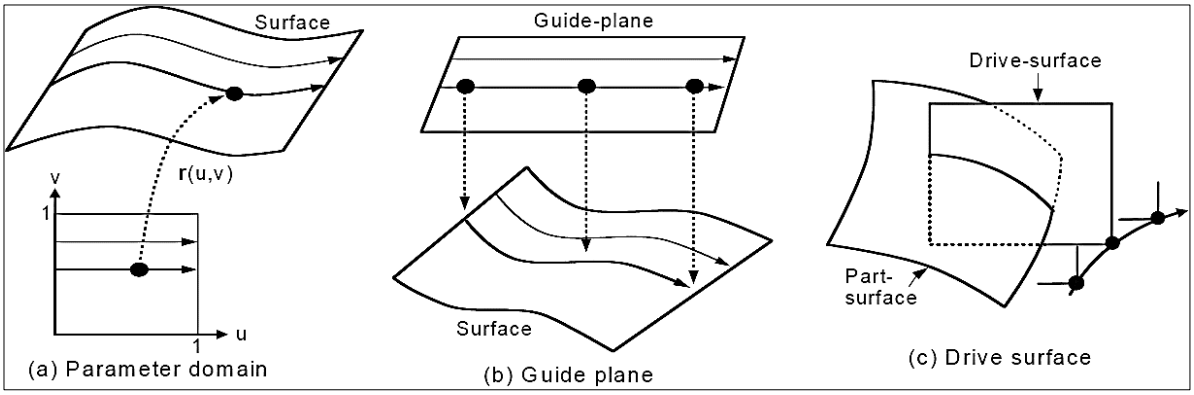


Figure 1.6 Tool path-planning domains [Web 1]

1.2.3 Types of tool path footprints [Web 1]

There are mainly four types of tool-path footprints. These are as following:

1. Serial-pattern footprints.
2. Contour-pattern footprints.
3. Strip-pattern footprints.
4. Radial-pattern footprints.

Serial-pattern footprints: In serial pattern topology, the whole surface is divided in tool footprints, which are created by offsetting curves along x-y direction or along boundary curve or offsetting normal to the boundary curve.

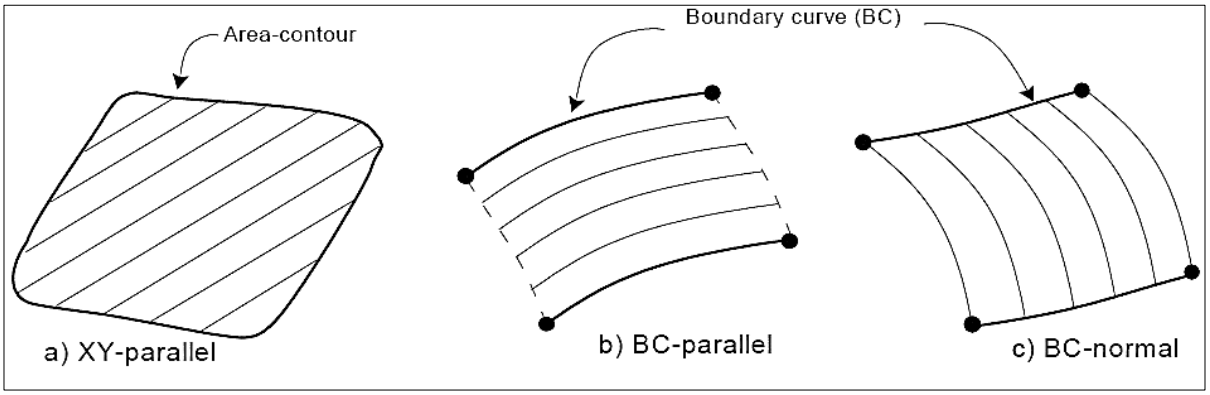


Figure 1.7 Types of serial pattern footprints [Web 1]

Radial-pattern footprints: In radial pattern topology, the tool footprints are offsetted according to radial steps.

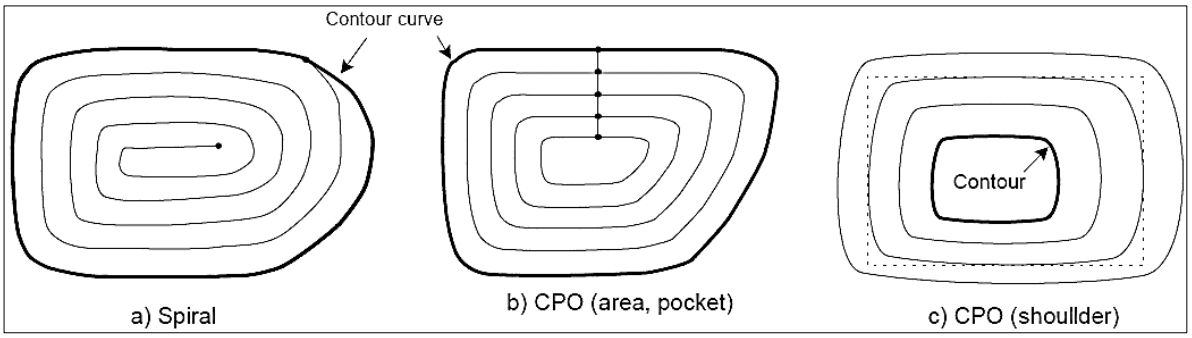


Figure 1.8 Types of radial-pattern topology [Web 1]

Strip-pattern footprints**:** In strip pattern topology, tool footprints are created in form of strips, which are either parallel, or at a fixed distance from each other.

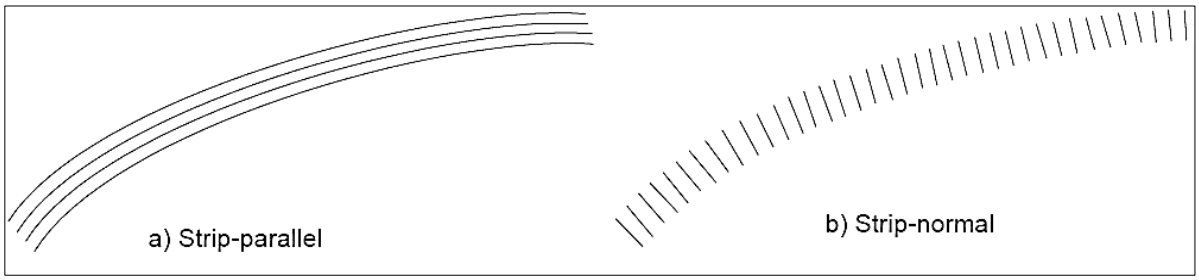


Figure 1.9 Types of strip-pattern topology [Web 1]

Contour-pattern footprints**:** This is most widely used tool path topology. In this, the tool paths are generated from intersection points with the 3-D model. The intersection points can be generated using helical path, circular path that is parallel to z-plane or parallel to boundary curves.

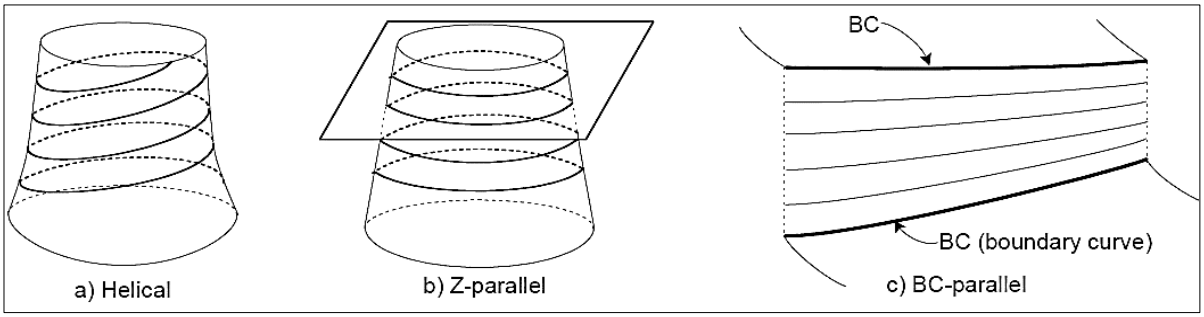


Figure 1.10 Types of contour-pattern topology [Web 1]

1.2.4 Tool path footprint generation methods [Web 2]

The whole tool path footprint generation domain can be classified on following basis:

1. Cutter contact-based methods
2. Cutter location-based methods also called as C-space methods.

Cutter contact based methods:

1. Isoparametric method:

Isoparametric are the curves, which are defined by constant parameter value on parametric surfaces. In this method, cutter contact points are specified along Isoparametric curves on the surface of the work piece. Linear segments approximate the Isoparametric curves. However, if these linear segments are longer, this may lead to under cuts and over cuts in the geometry of model.

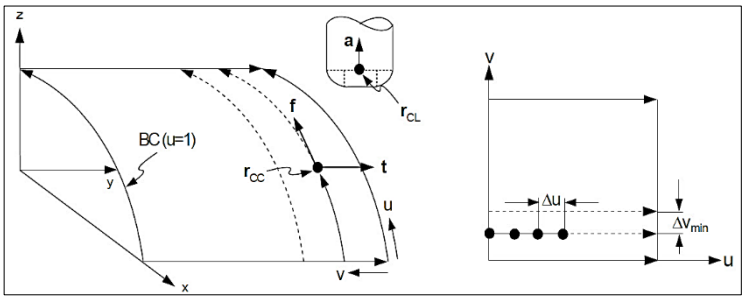


Figure 1.11 Isoparametric method of TPG [Web 2]

2. The Cartesian Method:

In The Cartesian method, the tool paths are generated with the help of guide plane and points are projected onto surface from this guide plane. The cutter contact points are found using intersection algorithm and then cutter location is approximated with respect to the global coordinate system.

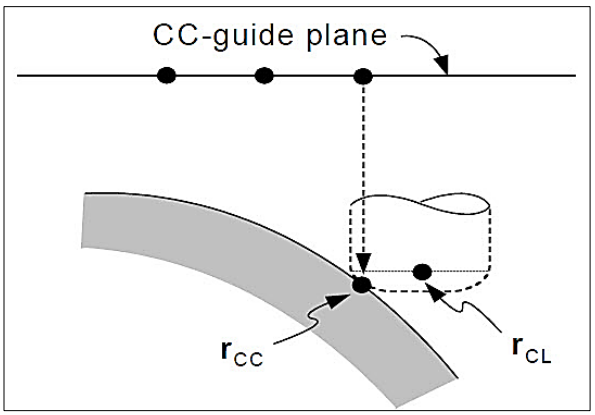


Figure 1.12 Cartesian method of TPG [Web 2]

3. APT-type tool path generation method:

In this method three types of surfaces are used to define the tool path. These are defined as below:

1. Part surface [S1: f(r, s)]: It is the surface that is generated by machining.
2. Drive surface [S2: g(t, u)]: This surface defines the tool path (and path interval).
3. Check surface [S3: h(v, w)]: Check surface is use to define length of step.

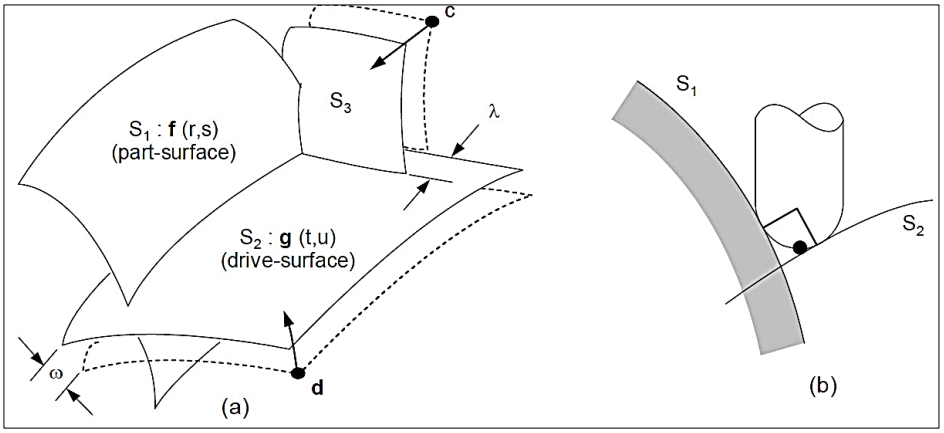


Figure 1.13 APT method of TPG [Web 2]

Tool path footprint generation methods based on cutter location:

It is also known as C-space approach: In this method, firstly Cutter Location-surface of the preform-surface (S_P) is formulated. Then Cutter Location-surface of the design-surface (S_D) is found. The three C-space elements 1.) V_F 2.) V_M 3.) V_G, are computed, which are of volume-type. Finally, tool path is generated from the C-space elements.

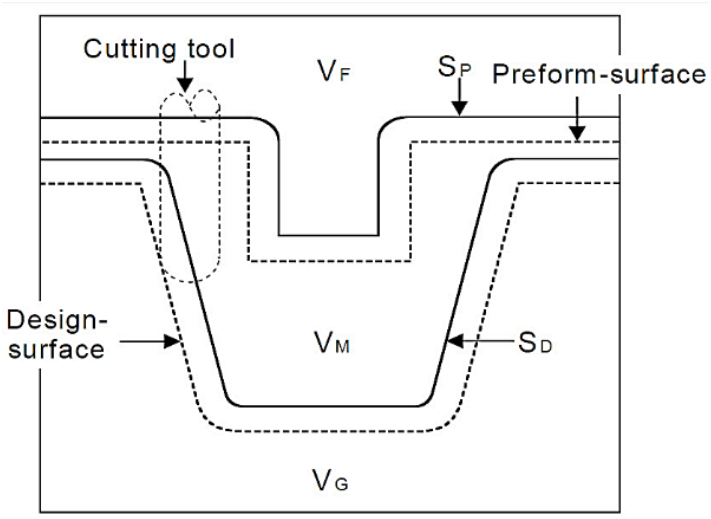


Figure 1.14 C-space approach of TPG [Web 2]

1.2.5 Gouge free cutter location determination along footprint

For formulating gouge free tool path planning, following steps are followed:

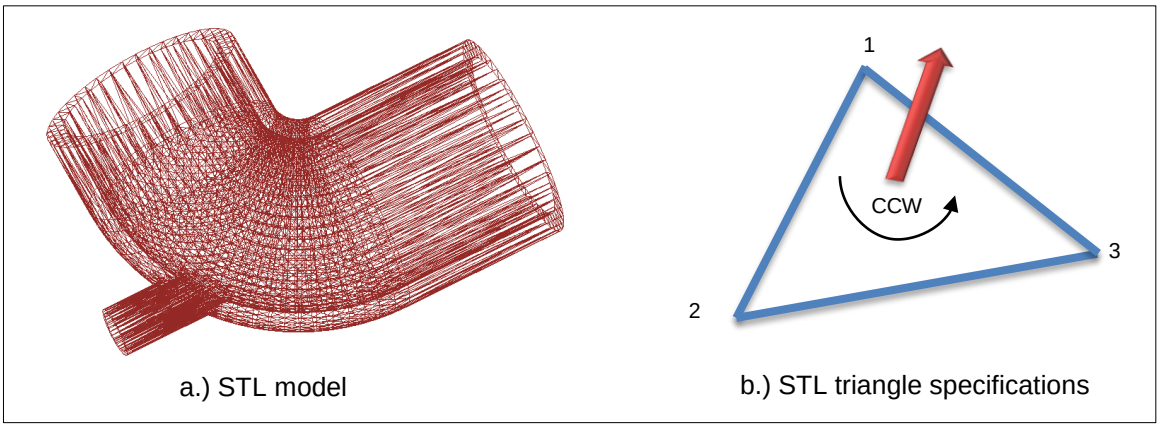
Step 1: Defining type of the footprint.

Step 2: Defining tool positions and finding intersection of tool along the tool footprint.

Step 3: Connecting these intersection points to form a tool path, which may be of zig-zag type, spiral type or any of types of footprints that are defined in section 1.2.3 are used for machining sculptured surfaces.

1.2.6 Study of STL file

STL (Standard Tessellation File) is a neutral file format created by 3-D Systems in 1989. The other name of STL is Standard Tessellation Language. STL (Standard Tessellation File) file is the mostly used in Rapid Prototyping (RP) processes. In the STL file, the whole surface is tessellated logically into a number of small triangles also called as facets. Each facet is described by a normal vector and three coordinate points that represent vertices of the triangle.



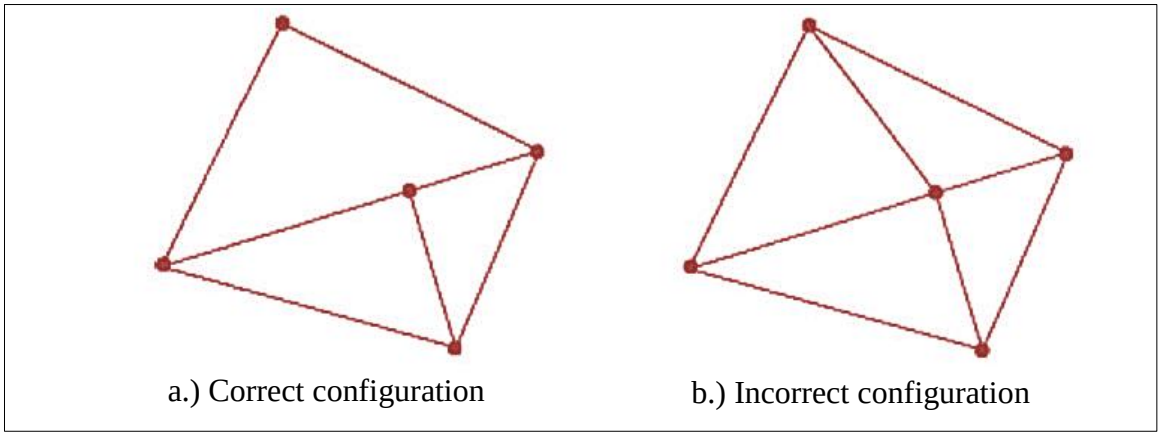
a.) STL model

b.) STL triangle specifications

Figure 1.15 STL file representation

Facet orientation: The triangular facets represent the 3-dimensional surface of model. The orientation of the facet is given by right hand rule. The surface normal is in outward direction when the vertices are presented in counterclockwise order. The surface normal is in inward direction when the vertices are presented in clockwise order.

Vertex-to-vertex rule: In STL file, the each facet share its two vertices with adjoining facets i.e. a vertex of one triangle cannot be left unconnected. The conventional STL file does not contain any other information such as scaling but some CAD packages have modified STL file format in which they save model information for other purposes.



a.) Correct configuration b.) Incorrect configuration

Figure 1.16 Rules for creating STL files

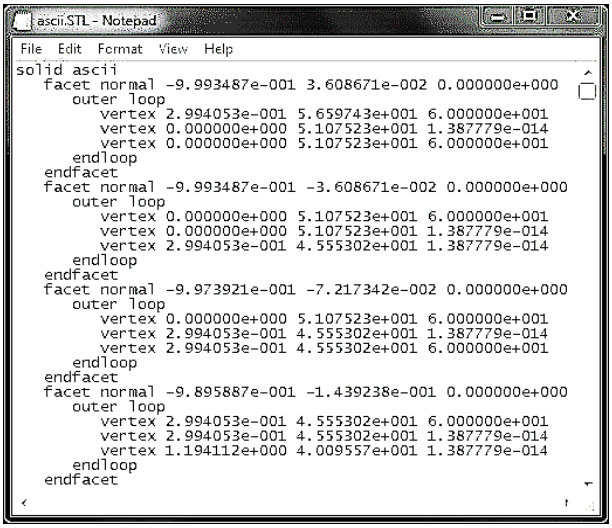
Formats of STL file:

The STL file is represented using two data formats: ASCII and binary. These are described separately below:

ASCII STL file:

solid name
facet normal nx ny nz
  outer loop
    vertex a1x a1y a1z
    vertex b2x b2y b2z
    vertex c3x c3y c3z
  endloop
endfacet
endsolid name

a.) ASCII format



b.) Example STL file for ASCII

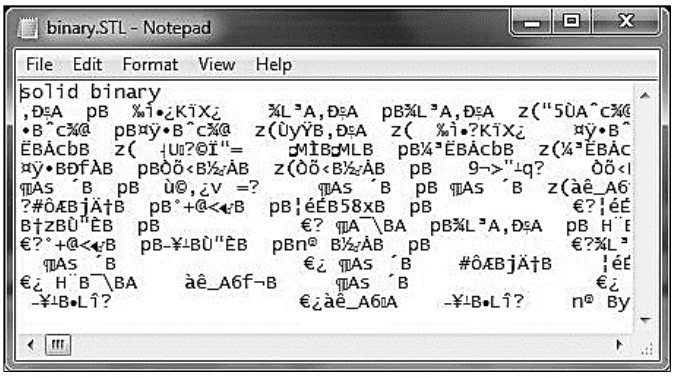
Figure 1.17 ASCII STL file format

The STL file starts with characters “*solid* name of file”. One surface normal and three vertices represent each triangle. The each triangle is ended by character “*endfacet*”. This procedure continues for all the remaining triangles and at last, the STL file is ended by characters “*endsolid* name of STL file”.

STL file-binary format:

Bytes	Data type	Description
80	ASCII	Header. No data significance.
4	unsigned long integer	Number of facets in file
4	float	*i* for normal
4	float	*j*
4	float	*k*
4	float	*x* for vertex 1
4	float	*y*
4	float	*z*
4	float	*x* for vertex 2
4	float	*y*
4	float	*z*
4	float	*x* for vertex 3
4	float	*y*
4	float	*z*
2	unsigned integer	Attribute byte count

a.) Binary format



b.) Example file for binary format

Figure 1.18 Binary STL file format

As the example file depicts, there is no user readable data. The data is stored in machine-readable format in form of small chunks of memory called as bytes. The STL file saved in ASCII format has larger size than the Binary format. The binary format is a machine-readable format. A binary STL file starts with header of 80 characters. The total number of triangles in STL file is given by next 4-byte unsigned integer. The triangles are defined using 32-bit floating-point numbers. This 32-bit data has 3-bytes for surface normal. Consecutively, next 3-bytes represent X coordinate, Y coordinate, and Z coordinate of each vertex of that triangle. Just before the end of STL file, a two byte unsigned 'short' integer is present that is of no use.

1.2.7 Tool path generation using Drop the Ball method [1]

Manos et al. have developed “Drop the Ball” method, which can be used to machine a sculptured surface on three axis CNC lathes [1]. This method can machine sculptures surface by dropping the ball nose cutter with end shaped like a sphere. The cutter is dropped at each position, independent of the type of surface being machined which makes it efficient for machining complex surfaces. The cutter is then moved over the fixed helical footprint to generate gouge- free tool path.

This Drop the Ball method can machine sculptured surfaces efficiently but this method is limited for the use of ball nose end milling cutter with a fixed path along which cutter is moved. In Dropping Method, for each cutter position the tool is dropped from a height along the tool axis at any particular position in X-Y plane. The cutter drops steadily along cutter axis plane and would intersect one of the triangles representing the work piece surface in STL file. The cutter may intersect with STL surface at three different locations:

1. Cutter intersects with planar surface of triangle in Figure 1.19 (a)
2. Cutter intersects with triangle edge in Figure 1.19 (b)
3. Cutter intersects with vertex of a triangle in Figure 1.19 (c).

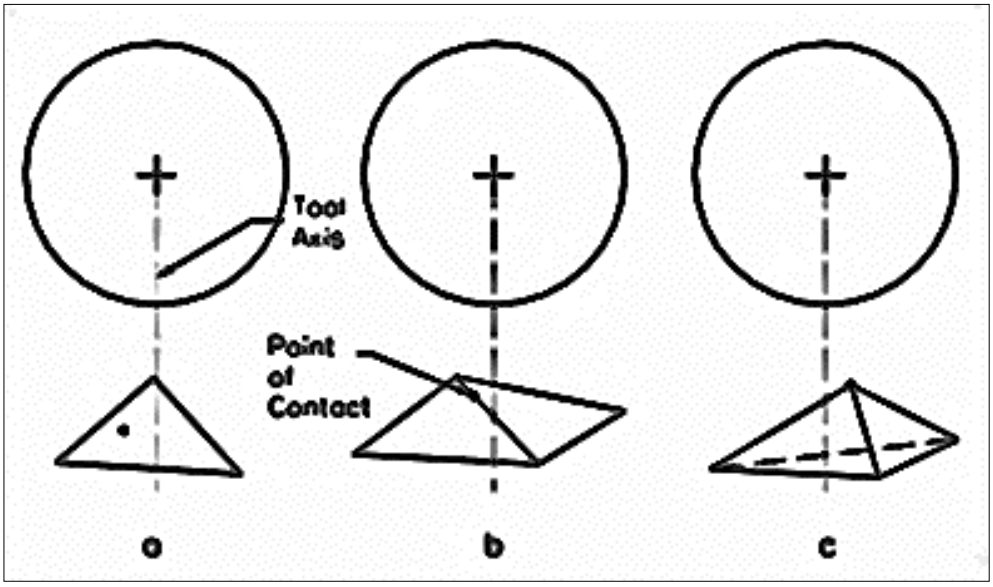


Figure 1.19 Cases for cutter intersection with surface a) with triangular surface b) with triangle edge c) with triangle vertex [1]

Triangle Check:

Each triangle in the shadow of the cutter is considered for three checks and other triangles are neglected. This check is carried out to find whether a ball nose end cutter moving along cutter axis would contact surface of STL triangle. The tangential intersection point of cutter and triangle surface give the cutter location point. According to the geometry, two solutions exists- one where the cutter is located along the positive side of the triangle normal (pointing outwards) and other where the cutter is located on the negative side of the triangle normal. The solution, which represents the contact with the positive side of the triangle normal, is accepted and other is rejected.

The cutter axis is defined by following equation:

$$Cutter(u) = (1-u) \times t_a + u \times t_b \ ,$$

It would intersect surface of triangle within the boundary limits of triangle vertices P1, P2 and P3, where u is the parameter and $0 < \mathrm{u} < 1$ with cutter axis vector ranging from t_a to t_b. Any point on the triangular surface P_c is given as below:

$$P_c = P1 \ + \ (P2 - P1) \ \times \ t \ + \ (P3 - P1) \ \times s$$

t and s parameters are to define the surface of triangular facet. As the mid-plane of the cutter lies along the cutter axis and the cutter intersects tangentially with the surface of the triangle, we get the following equations:

$$(1-u) \times t_a + u \times t_b = \ P1 \ + \ (P2 - P1) \times \ t \ + \ (P3 - P1) \times s + \hat{n} \times R$$

Where:

$$\hat{n} = \frac{(t_b - t_a) \times (E_b - E_a)}{|(t_b - t_a) \times (E_b - E_a)|}$$

Solving above equation as below:

$$\begin{bmatrix} (t_b - t_a)x & (P1 - P2)x & (P1 - P3)x \\ (t_b - t_a)y & (P1 - P2)y & (P1 - P3)y \\ (t_b - t_a)z & (P1 - P2)z & (P1 - P3)z \end{bmatrix} \times \begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{bmatrix} (P1)x - (t_a)x \\ (P1)y - (t_a)y \\ (P1)z - (t_a)z \end{bmatrix}$$

$$\begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{bmatrix} (t_b - t_a)x & (P1 - P2)x & (P1 - P3)x \\ (t_b - t_a)y & (P1 - P2)y & (P1 - P3)y \\ (t_b - t_a)z & (P1 - P2)z & (P1 - P3)z \end{bmatrix}^{-1} \times \begin{bmatrix} (P1)x - (t_a)x \\ (P1)y - (t_a)y \\ (P1)z - (t_a)z \end{bmatrix}$$

After solving this equation for values of u, s, and t. Then putting value of s and t in equation of triangle surface, we get the cutter location point P_c on the surface of the triangle.

Edge Check:

This second check is carried out to find if the cutter would intersect an edge of triangle within edge bounded by its vertices. The edge check is performed for each edge of every STL file triangle. The point at which the cutter intersects with the edge of triangle i.e. cutter contact point is found. The cutter contact point is then analyzed to check whether it lies within the endpoints of the edge. If it is within the edge the intersection point is saved, otherwise, it is rejected and the subsequent check is performed.

The equation of cutter axis vector is:

$$Cutter(u) = (1-u) \times t_a + u \times t_b$$

The equation of edge of a triangle:

$$Edge(s) = (1-s) \times E_a + s * E_b$$

If the tool intersects with an edge of a triangle then following equation must be satisfied:

$$Cutter(u) = Edge(s)$$

$$\text{i.e. } (1-u) \times t_a + u \times t_b = (1-s) \times E_a + s * E_b + \hat{n} \times R$$

Where:

$$\hat{n} = \frac{(t_b - t_a) \times (E_b - E_a)}{|(t_b - t_a) \times (E_b - E_a)|}$$

Vertex Check:

This is third check for finding the intersection of the cutter with the vertices of the triangle. The point of the intersection of the cutter with each vertex is found from the equation below. The equation will give two values for u. If these values are non-real then the cutter does not intersect with triangle vertex. If values are real, then point with maximum radial distance is considered as the cutter location. The equation of vertex check is:

$$V_i - (1-u) \times t_a + u \times t_b = R,$$

Where Vi is vertex of triangle.

1.3 Automated CNC tool path generation using API

As the name suggests, an application-programming interface (API) is a programming canvas that enables users to write their own codes to access already written codes by the software providing company in form of functions, classes, data structures, and commands. The user can build customized add-on packages that provide additional capability to existing functionality of the software. The most commonly used is MS-EXCEL. Some companies like SolidWorks provide the API package free of cost along with main software to their customers but others may charge extra for API package. For CAD applications, SolidWorks provide excellent support and there are enough resources available on internet for a novice user to start customizing SolidWorks. On the other hand, Creo's API requires tedious homework to be done even for installing API package and running it. Furthermore, the resources are limited with limited support.

The user can use CAD API in two ways:

1. Using Visual Basic for Applications (VBA) to write Merged and Correlated Recorded Output (MACRO) that are saved as a single *.swp file.
2. Using programming environment like Visual Basic.Net, Visual C++, Visual in Visual Studio for developing the stand alone applications.

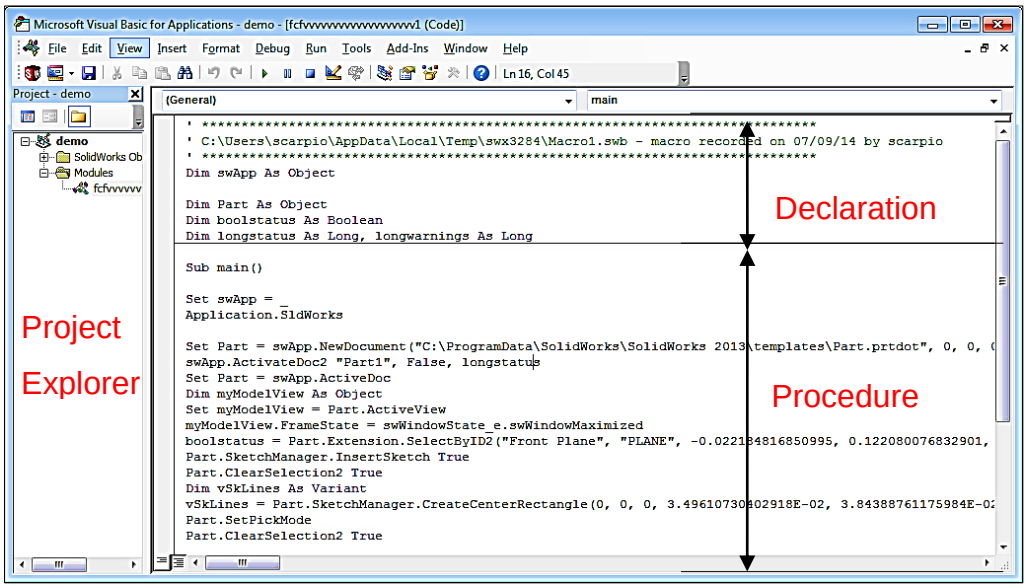


Figure 1.20 Anatomy of SolidWorks API

1.3.1 SolidWorks API

API is a tool to write useful codes in a programming language within an application. As a result, a direct integration can be developed between different applications. BobCAM have developed SolidWorks add in for creating tool paths for 3- axis CNC machines [Web 5]. CAMWorks technologies have also developed milling simulation software using SolidWorks API for 3- axis milling, 2-axis, and 4-axis turning [Web 6]. ModusCAM comes as a SolidWorks add-in, which is used for tool path planning [Web 7]. Vinodhkumar Somavar Muniappan has developed an add-in using SolidWorks API for recognition of machining features, tool path generation, and its simulation [3]. Szu-Hung Cheng et al. used SolidWorks API to simulate the cutting of universal gear generator on a 5-axis CNC machine [7]. Ding Songlin used SolidWorks API for tool path generation for 5-axis NC machining [8]. In his work, the tool and tool holder are modeled by a structured oriented bounding box (OBB), this technique is employed in VR applications for collision detection of 3-D objects, whereas an octree structure approximates the work piece.

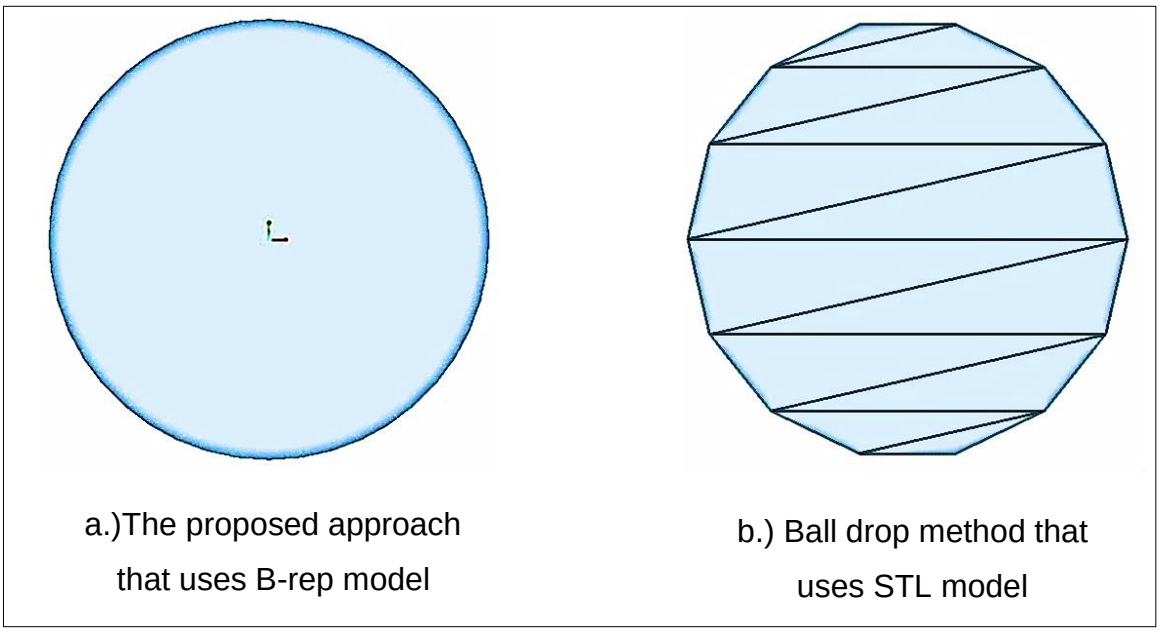
1.4 Gaps in literature

1. There are problems associated with the prevalent methods of tool path generation using CAD. These methods use STL file, which itself, is an approximation of 3-D model. Therefore, there are always loss of data and errors with use STL file. But if the number of triangles are increased, the accuracy of the tool path increases. Therefore, the size of STL file increases which in turn give rise to another problem of longer computations and longer processing of STL file.
2. Tool positioning strategies rely on small spacing along footprint but absolute distance between two positions is not considered.
3. Nobody has done automated tool path generation using CAD API on 3-axis lathe milling machine.

CHAPTER 2
PROBLEM DEFINITION

We want to develop algorithm for automatic part program generation from B-rep model of sculptured surfaces using CAD software API to be machined on CNC lathe mill. Nobody has done automated tool path generation using CAD API on 3-axis lathe milling machine.

From B-rep model using API directly, to generate tool path would actually help in improving the accuracy of the parts is the hypothesis that is proposed in this thesis and is to be explored. We will compare this method with the Ball drop method for accuracy [1]. We will also study STL file for its characteristics of accuracy using different parameters. The two methodologies are differentiated as below:



a.)The proposed approach that uses B-rep model

b.) Ball drop method that uses STL model

Figure 2.1 The two methodologies for tool path generation

Several types of tool footprints are explored in literature survey. The footprint that will be used in this thesis is predefined. The tool footprint will be either helical or parallel cross-section along z-axis. We will define parts that are pseudo-symmetric with z-axis passing through the solid and z-axis never intersects the surface of the part except at the ends. In addition, the surface of the part that we will machine on the lathe mill will never intersect the z-axis.

We can use API of many CAD software such as Creo, AutoCAD, SolidWorks but SolidWorks API is optimal for our use because it has extensive support and resources available free of cost on the internet. The API functions will serve definite purpose in the software and we will have to explore them and use correct one to arrive at what we want. The programming languages that we will use are VBA and VB.NET because SolidWorks API is optimized for VBA.

The problem is defined for 3- Axis Lathe Mill with ball nose milling cutter. The material of work piece will be soft wood because we are not concerned about parameters such as radial step, material removal rate (MRR), cutting forces, time required for machining or kinematics of machining, we are only concerned about position of tool and surface accuracy in present thesis.

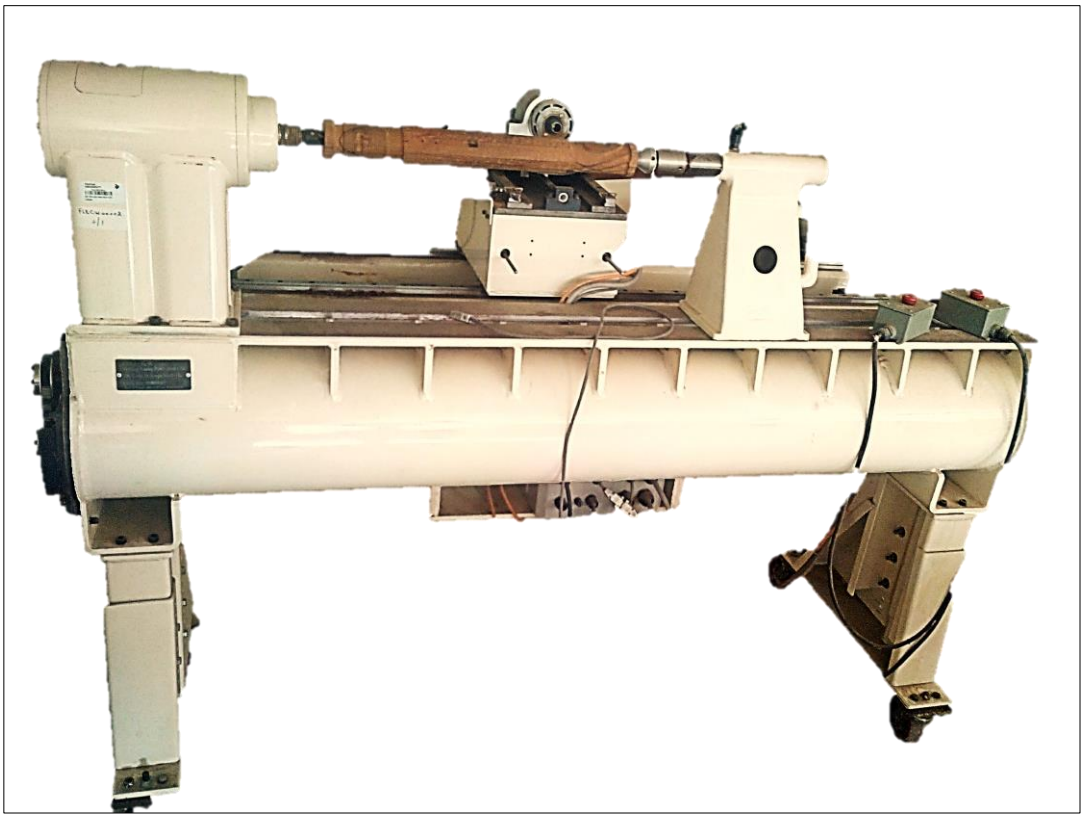


Figure 2.2 3-Axis Lathe-Milling Machine

CHAPTER 3
IMPLEMENTATION

3.1 Effect of saving options on accuracy of surface in STL file

The CAD softwares give user optional settings while saving model as STL file, which affect the quality of the surface of 3-D model in STL file. The most common parameters are "Chord Height" and "Angle Control".

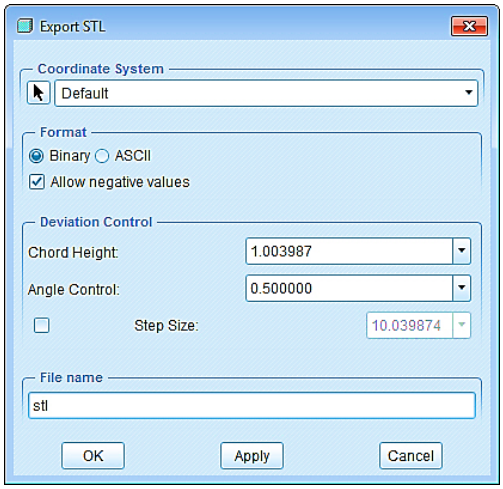


Figure 3.1 STL saving options in Creo

The quality of the surface of the model is controlled by the chordal height and angle control parameters. The figure below represents the effect of these parameters.

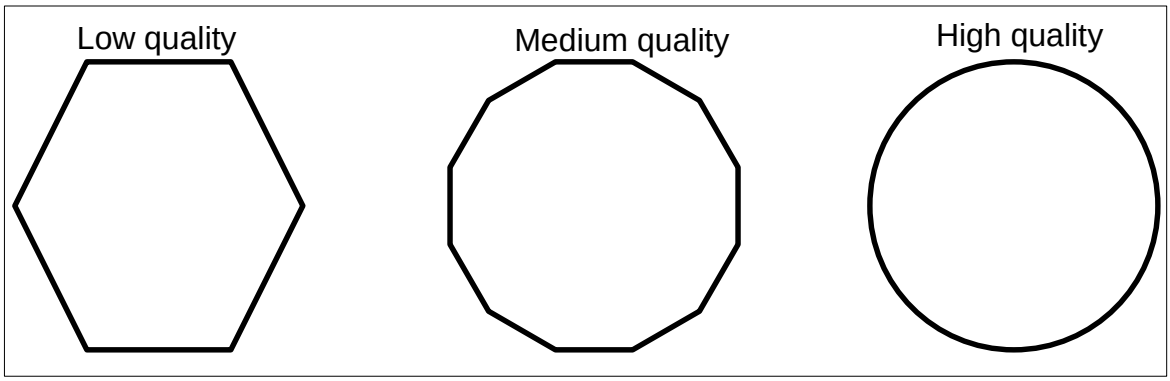


Figure 3.2 Variation in quality of surface in STL file

I. Chord height describes the distance between the actual surface of 3-D model and the triangulated surface of the STL file. Thus, a small value of chordal height will result in high quality of surface of the model but it would require a large number of STL triangles.

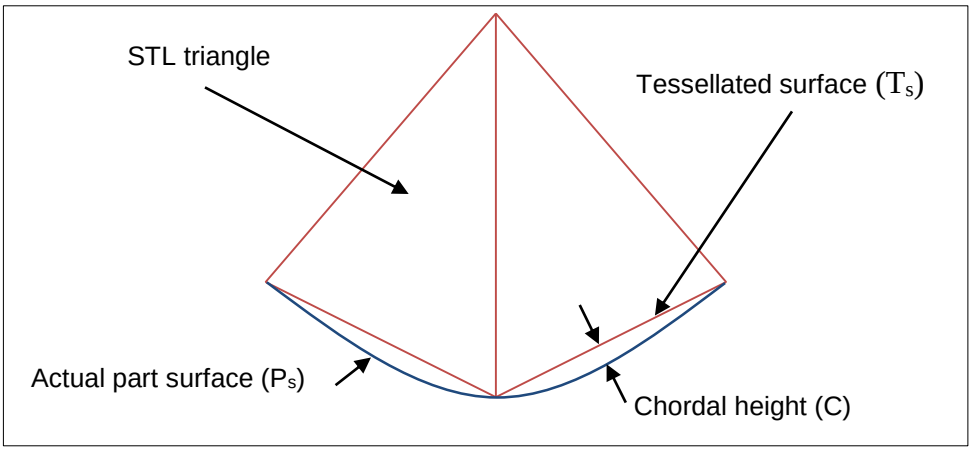


Figure 3.3 Chord height representation

Chord height calculation [Web 3]:

$$C = \frac{M}{(1000 * Q)}$$

Where:

C = Chordal height

P_s = Surface of model

T_s = Triangulated surface

M = Size of model (the length of the diagonal of the bounding box of the model)

Q = Number of elements (recommended value is 0.3, limits 0.1 to 1.0) is input by the designer and this value describes the size of the triangles in areas with high curvature.

The AutoCAD software gives the above formulation in its online help for the chordal height calculation. Nevertheless, each CAD software has its own algorithm for chordal height calculation and there is no standard formula for calculation of chordal height but we can formulate a general trend of the variation of number of the triangles in the STL file that is shown on the next page.

Variation of shape and number of triangles (N) with respect to change in chord height (C) in mm keeping angle constant:

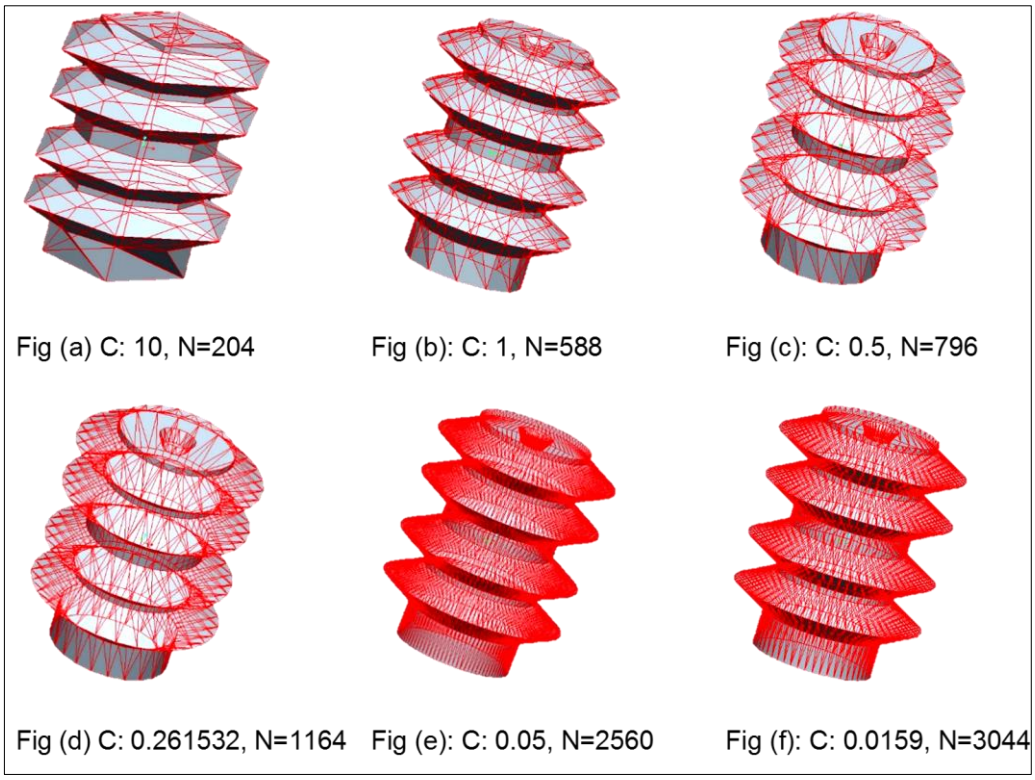
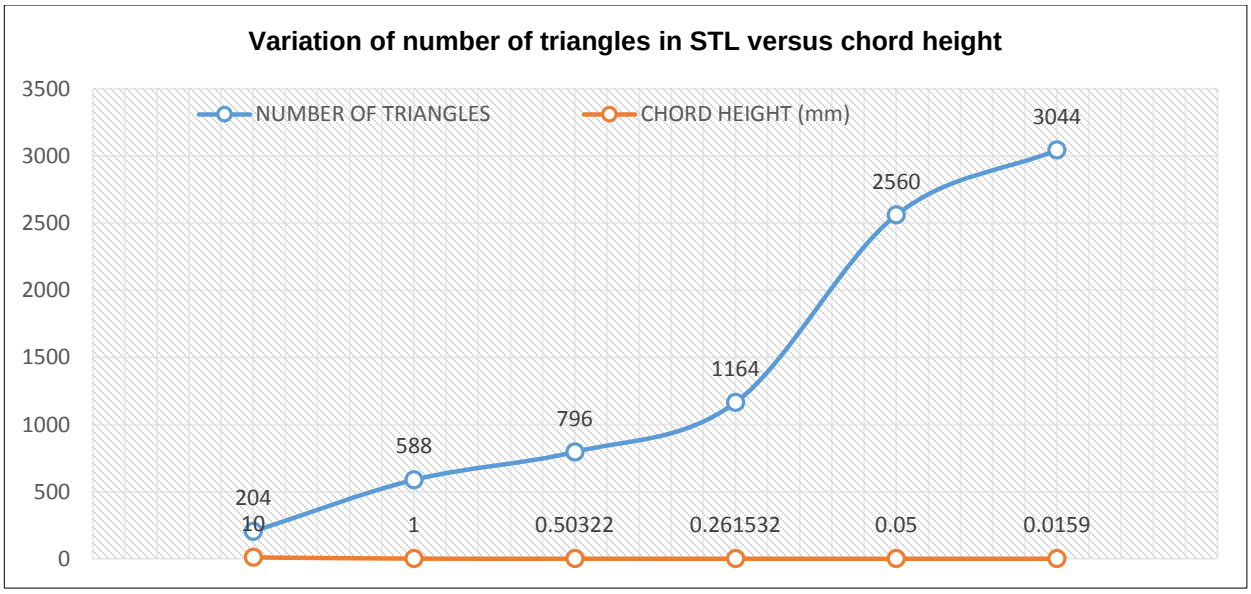


Figure 3.4 Variation in shape with change in chord height



Conclusion: The above models and graph depict, as chord height is decreased, the number of triangles rise exponentially, and model become closer to B-rep model.

II. Angle Control: Angle control controls the smoothness of the surface of the STL model. The more the value of the angle control the surface of the model will be rougher. The previous model was taken with chordal height = 0.5 mm constant throughout and impact of angle control on number of triangles and part was analyzed.

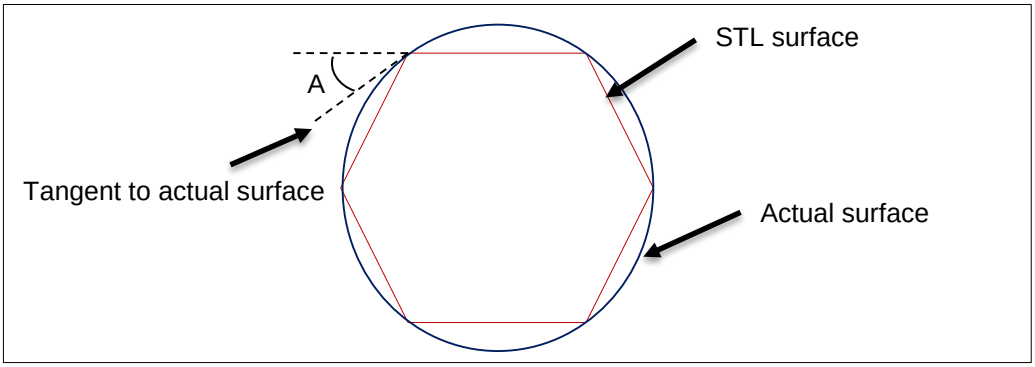
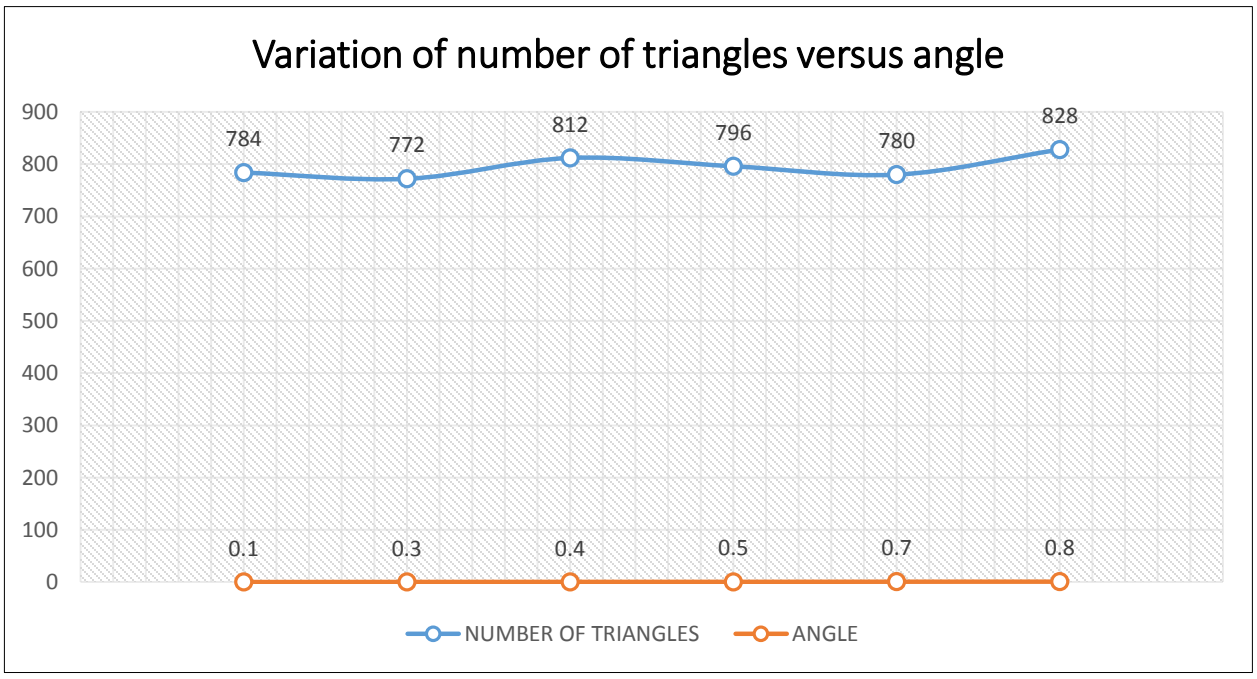


Figure 3.5 Angle control representation

Variation of shape and number of triangles (N) with respect to change in angle (C) in mm keeping chord height constant:



Conclusion: As above graph depicts the variation in angle do not substantially affect the part and number of triangles.

3.2 Tool path generation using Ball drop method

Research work regarding tool path generation using Ball drop method of Manus et al., Kanderpatel, PM Jasra has been studied and their algorithm has been applied to get the cutter location points [1] [5] [4]. In the Ball-Drop Method [1], the cutter moves along a fixed tool footprint. For machining any surface, the tool footprint is discretized into closely spaced points along tool path. The required coordinates of each of these points is determined through three cases for the intersection of cutter and STL surface: The cutter: 1) Intersection with triangular surface, 2) Intersection with triangle edge, and 3) Intersection with triangle vertex. Then the cutter location point with maximum radial distance is chosen.

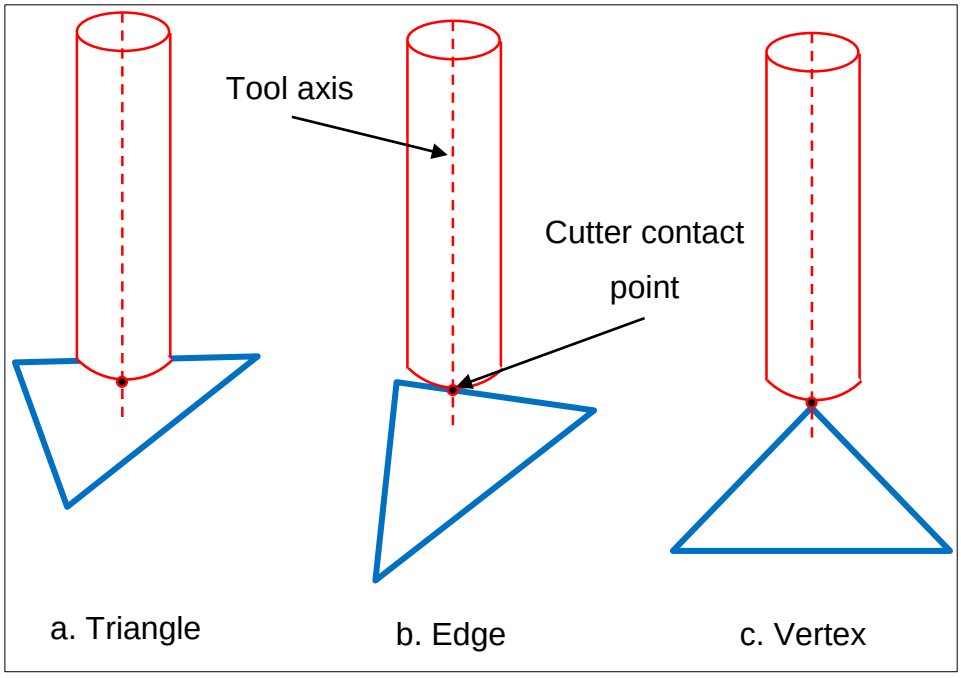
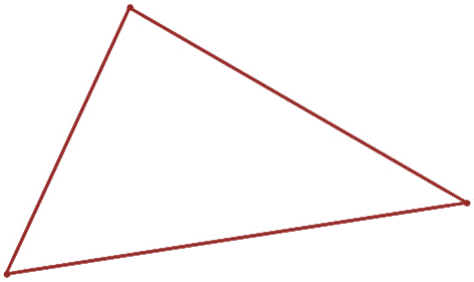


Figure 3.6 Three possible cases of contact of tool and STL surface

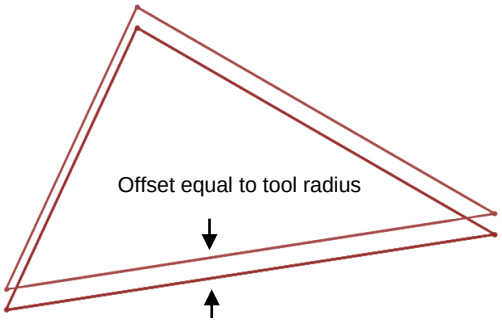
In our work, Ball drop method [1] is slightly modified. The triangle check is done by intersecting tool vector with the offsetted triangle equal to tool radius. The edge check is run to find out intersection of tool vector and edge of a triangle, which is considered as a cylindrical pipe with radius equal to tool radius between defined by two vertices of a triangle as its end points. Then vertex check is also modified by considering a vertex of a triangle as a sphere with radius equal to tool radius. The STL file is taken as input in the

algorithm written in .NET and three checks are performed on the STL data. These are described as following:

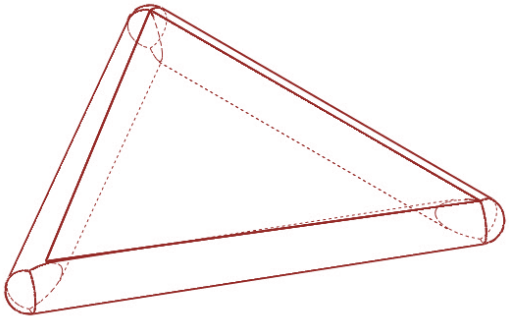
3.2.1 Triangle check



a. Triangle on STL surface



b.Triangle is offset along surface normal



c. 3-D view of offset surfaces of triangle, cylinders for three edge and spheres for three vertices

Figure 3.7 Geometry of Triangle check

The triangle check is carried out to find if a cutter with nose radius (R) moving along cutter axis would intersect with offsetted surface of triangle within the boundary of its vertices P1, P2 and P3, where u is the parameter ($0 < u < 1$), with vector of cutter axis ranging from t_a to t_b.

Equation of cutter vector:

$$Cutter(u) = (1 - u) \times t_a + u \times t_b$$

Any point on the triangular facet P_c is given as below:

$$P_c = P1 + (P2 - P1) \times t + (P3 - P1) \times s$$

t and s parameters are to define the surface of triangle. The midplane of the cutter lies along the cutter axis and the cutter intersects the offsetted plane of the triangle tangentially at a point, we get the following equations:

$$(1-u)\times t_a + u \times t_b = P1 + (P2 - P1)\times t + (P3 - P1)\times s + \hat{n}\times R$$

Where:

$$\hat{n} = \frac{(t_b - t_a)\times(E_b - E_a)}{|(t_b - t_a)\times(E_b - E_a)|}$$

3.2.2 Edge check

This check is carried out to find if the cutter would intersect a cylindrical edge within the limits of the coordinates of vertices that define the edge. The edge check is carried out for each edge of the triangle. Firstly, it must be found that at what point the cutter intersects the cylinder. The contact location P will generate two solutions, but the CL that yields greatest distance from work piece axis is chosen. The cutter location point is then checked to see if it lies within the endpoints of the pipe. If the cutter location point is within the limits of the endpoints, the cutter location is saved, otherwise, it is discarded, and the subsequent check is carried out.

Figure 3.8 Edge-check taking edges as cylindrical pipe

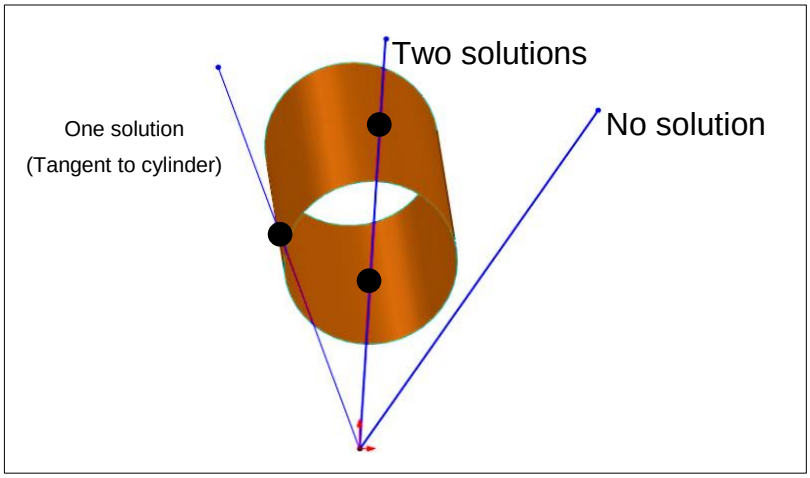


Figure 3.9 Intersection of cutter vector with cylindrical pipe

Intersection of cutter vector and edge considered as cylindrical pipe:

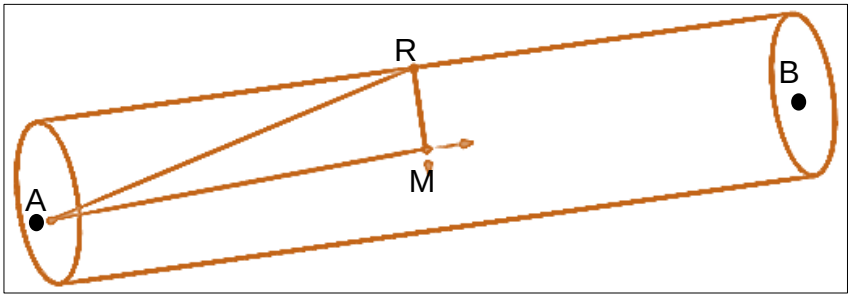


Figure 3.10 Right circular cylinder

Equation of edge taken as cylinder:

Let points be: point A (x1, y1, z1), point R (x, y, z) and point B (x2, y2, z2)

$$|AR| = \sqrt{((x - x1)\ \hat{}2\ +\ (y - y1)\ \hat{}2\ +\ (z - z1)\ \hat{}2)}$$

$$|AB| = \sqrt{((x2 - x1)\ \hat{}2\ +\ (y2 - y1)\ \hat{}2\ +\ (z2 - z1)\ \hat{}2)}$$

Now, AM is projection of AR on line AB

AM = Vector (AR) dot vector (AB)

By using Pythagoras theorem,

$$|AR|\hat{}2 = |AM|\hat{}2 + |RM|\hat{}2$$

$$|AR|^2 - |AM|^2 + |RM|^2 = 0 \ \ldots\ldots (1)$$

Equation (1) is the equation of a right circular cylinder as it gives the locus of point R on surface of cylinder. Equation of cutter vector:

$$cutter(u) = (1 - u) \times t_a + u \times t_b \ldots\ldots \text{(2)}$$

Equating equation (1) and equation (2) give intersection of tool vector and cylinder

$$|AR|\text{^}2 \ - |AM|\text{^}2 + |RM|\text{^}2 \ = \ (1 - u) \times t_a + u \times t_b$$

The above equation can be rewritten in quadratic form as below:

$$ax^2 + bx + c = 0$$

The above equation gives three solutions according to value of $b^2 - 4ac$ (As shown in Figure 3.9).

1. If $b^2 - 4ac < 0$, then the cutter does not intersect the cylinder.

2. If $b^2 - 4ac = 0$, then the cutter is a tangent to the cylinder intersecting it at one point.

3. If $b^2 - 4ac > 0$, then cutter intersects the cylinder at two points. In this case, the cutter location point with maximum radial distance is saved and other is rejected.

The cutter location point can be found by putting value of parameter u in (2), which is given by:

$$u = \frac{-b \ \pm \ \ sqrt(b^2 - 4ac)}{2a}$$

3.2.3 Vertex check

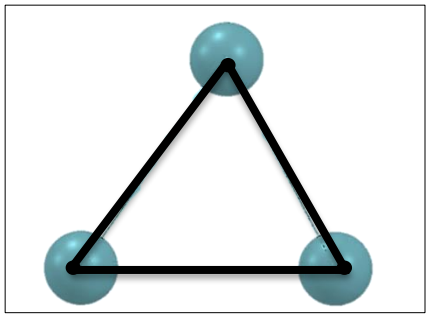


Figure 3.11 Considering vertices of triangle as spheres

In Vertex check, each vertex is assumed to be as a hollow sphere and intersection between a cutter axis and a sphere can be found using the algorithm as suggested by Paul Bourke [4]. Let any point P (px, py, pz) on cutter axis, as shown in Figure 3.12, be defined by:

$$P = P1 + u\,(P2 - P1) \ldots\ldots (3)$$

Alternatively, in each coordinate:

$$px = px1 + u\,(px2 - x1)$$

$$py = py1 + u\,(py2 - y1)$$

$$pz = pz1 + u\,(pz2 - z1)$$

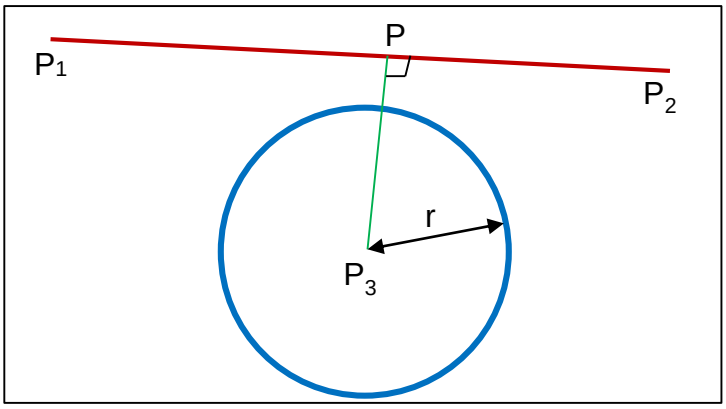


Figure 3.12 Intersection of cutter axis and a sphere [4]

A sphere with center point P_3 (px_3, py_3, pz_3) with radius, r is described by:

$$(px - px3)^2 + (py - py3)^2 + (pz - pz3)^2 = r^2$$

Equating the equation of the cutter axis to the equation of sphere

$$P1 + u\,(P2 - P1) = (px - px3)^2 + (py - py3)^2 + (pz - pz3)^2 - r^2$$

The above equation can be rewritten in quadratic form:

$$au^2 + b\,u + c = 0$$

Where:

$$a = (px2 - px1)^2 + (py2 - py1)^2 + (pz2 - pz1)^2$$

$$b = 2[(px2 - px1)\,(px1 - px3) + (py2 - py1)\,(py1 - py3) + (pz2 - pz1)\,(pz1 - pz3)]$$

$$c = px3^2 + py3^2 + pz3^2 + px1^2 + py1^2 + pz1^2 - 2[px3\,px1 + py3\,py1 + pz3\,pz1] - r^2$$

The above equation gives three solutions according to value of $b^2 - 4ac$.

1. If $b^2 - 4ac < 0$, then the cutter does not intersect the sphere.

2. If $b^2 - 4ac = 0$, then the cutter is a tangent to the sphere intersecting it at one point.

3. If $b^2 - 4ac > 0$, then the cutter intersects the cylinder at two points. In this case, the cutter location point with maximum radial distance is saved and other is rejected.

The cutter location point can be found by putting value of parameter u in (3), which is given by:

$$u = \frac{-b \pm sqrt(b^2 - 4ac)}{2a}$$

3.2.4 Tool path generation algorithm using STL file

Step 1: The algorithm starts with reading model data from STL file.

Step 2: The length and radius of the work piece are found.

Step 3: The tool footprint is then generated using following equations:

$$TX = R * \cos\theta$$

$$TY = R * \sin\theta$$

$$TZ = Zincrement$$

Where TX, TY and TZ are X,Y and Z coordinates of the tool vector.

$$Zincrement = \frac{L}{P * 360}$$

Where L = length of work piece, P = Side step

Step 4: Triangle check: The intersection of tool vector generated in step 3 is checked with all the offset triangles in the STL file. If intersection point is found it is stored in a separate linked list.

Step 5: Edge check: The intersection of tool vector generated in step 3 is checked with all the edges in the STL file. If intersection point is found it is stored in a separate linked list.

Step 6: Vertex check: The intersection of tool vector generated in step 3 is checked with all the vertices in the STL file. If intersection point is found it is stored in a separate linked list.

Step 7: The intersection points found by above three checks are sorted according to the radial distance. The point with maximum radial distance is our final CL-point for that particular tool vector.

Step 8: Step 4 to step 7 is repeated for all the remaining tool vectors generated in step 3.

3.3 Tool path generation using B-rep model in CAD API

In CAD API, built-in functions are available to us to access 3-D model or to create new geometries. Using API the tool path for Single Axis Lathe Milling machine can be generated by two methods:

1. Offsetting entire surface of 3-D model.
2. Offsetting curves found at intersection of plane parallel to Z-axis and 3-D model.

3.3.1 Method-1 Offsetting entire surface

The tool path is found in two steps:

1. Offsetting entire surface of 3-D model radially outward equal to tool radius.
2. Finding intersection of tool vectors with offset surface.

Offsetting surface of 3-D model radially equal to tool radius

In this method, each face of the 3-D model is selected and then the whole surface is offset by tool radius. Now, an offset surface body is available to us to use it for intersection with tool.

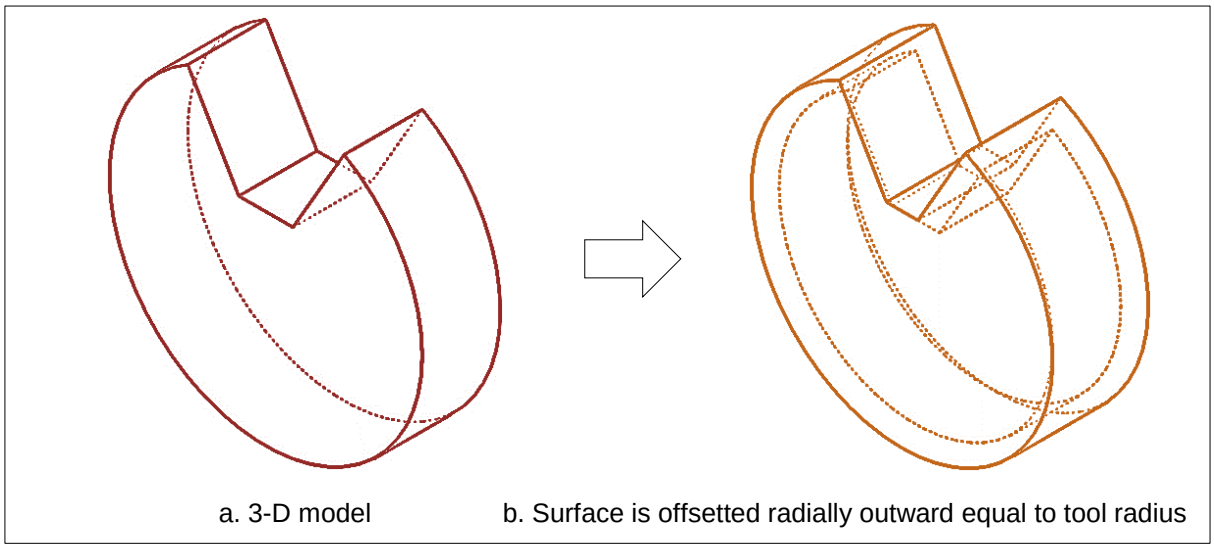


Figure 3.13 Surface offset method

Tool vector intersection algorithm for surface offset method:

Step 1: The algorithm starts with opening 3-D model in SolidWorks.

Step 2: The length of the work piece are then found. The length is found by using ray intersection function "*RayIntersections(parameters)"*. A ray originating from origin is fired along Z-axis and it give two intersection points one at origin itself and other at the end of model, from which length is found [8].

Step 3: Each face of the solid model is selected using "*selectbyID2(parameters)"*." method until all the face are selected [10].

Step 4: The whole surface of solid body is offsetted by tool radius radially outward using insert offset method "*InsertOffsetSurface(parameters)"* [9].

Step 4: In this step tool vectors are generated using predefined footprint. Both helical and z-parallel tool footprints can be used for this method to generate tool vectors.

Step 5: Intersection of all the tool vectors are found with the offset 3-D surface to create CL-points using ray intersection function [8].

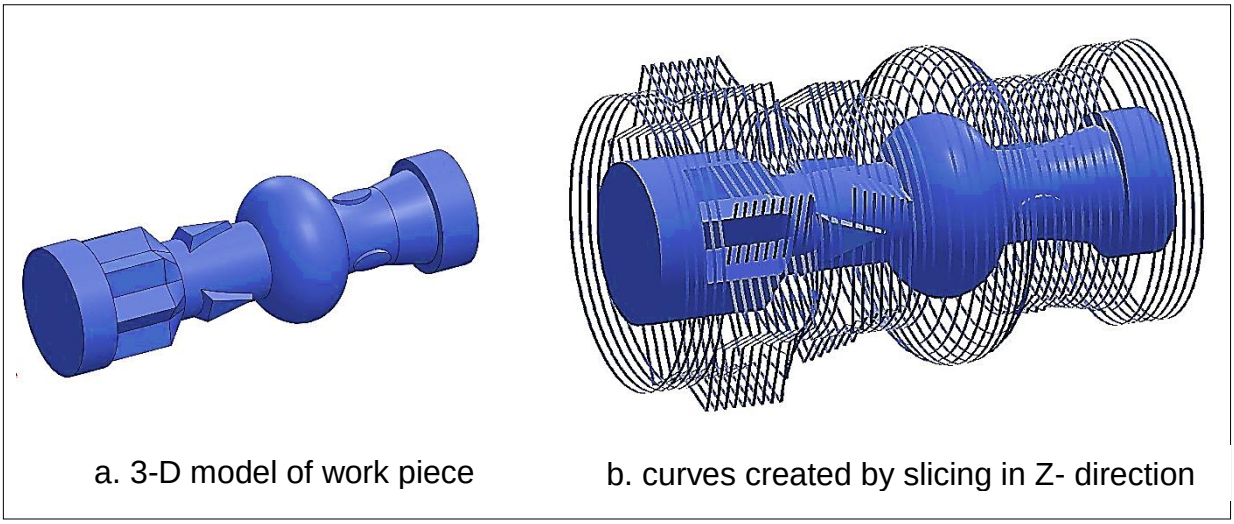
3.3.2 Method-2 Offsetting contour curves

In this method, the tool path generation is done in two steps:

1. Offsetting contour curves created by intersection of plane with 3-D model.
2. Finding intersection of tool vectors with offset curves.

Offsetting contour curves created by intersection of plane with 3-D model

The program takes inputs from the user of tool radius and pitch. The whole model is divided into a number of cross sections parallel to z-axis. On these cross sections, planes are created using inbuilt functions. The contour generated by intersection of plane and the body is offsetted radially outward value equal to tool radius. Now, there are a number of offset sketches available. The intersection of tool vectors and an offset sketches are then found to get cutter location points.



a. 3-D model of work piece

b. curves created by slicing in Z- direction

Figure 3.14 Offsetting contour created by intersection of plane with 3-D model

Tool vector intersection algorithm for contour curve offset method:

Step 1: The algorithm starts with opening 3-D model in SolidWorks automatically.

Step 2: The length of the work piece are then found using ray intersection function "*RayIntersections (parameters*)" [8].

Step 3: Planes parallel to Z-axis are created at a fixed increment in z value using following function "*InsertRefPlane(parameters)" [*11].

Step 4: Intersection of plane created along direction is found using ""*sketch3dintersections(parameters)"*." function with the 3-D model and contour curves are created on this plane [12].

Step 5: The contour curves are the offset outward equal to tool radius-using function "*SketchOffset(parameters)"*." [13].

Step 6: Since the offset curves are created parallel to the z-axis so we cannot use helical tool footprint in this method. Therefore, we use z-parallel tool footprint to create tool vectors.

Step 7: Intersection of tool vectors are found with the offset contour curve using function "*RayIntersections(parameters)"* [9].

3.4 Comparison of surface offset and contour offset method

S.no.	Surface offset method	Contour offset method
1.	The time taken by algorithm of surface offset method to generate one cutter location point is very less.	Time taken by contour offset method algorithm to generate one cutter location point is very high due to creation of large number of sketch entities.
2.	Both helical and z-parallel footprints can be used to find cutter location points.	Only z-parallel tool footprint can be used to find cutter location points.
3.	In cases, where offsetting surfaces lead to volume intersection of solid body, the inbuilt API function fails to cope up with the volume intersection and does not offset the surface as required	This method also fails when offsetted volumes intersect. Therefore, we cannot use roughing tool with large radius for offsetting sketches. In many cases, this problem can be avoided using radius of tool very small. Therefore, we have to use only finishing tool in these cases.
4.	Features like written text in sculptured surfaces has edges that are parallel to z-axis or normal to it. In these cases, we have to use helical tool footprint to avoid high material removal and over cutting.	Features like written text in sculptured surfaces has edges that are parallel to z-axis or normal to it. In these cases, the material removed is far more and overcutting occurs. Since we cannot use helical footprint in this method.

3.5 Adaptive tool vector generation using bisection method

The bisection algorithm is used to improve the tool path by generating more number of tool vectors by using specific conditions. When the radial step is large, the problem of overcutting near convex geometry and problem of undercutting in concave geometry come. So, to avoid these problems to some extent bisection algorithm is used. This method generate tool vectors adaptive where required. When intersection points are found. The angular and radial position of an intersection point is checked with respect to the previously found intersection point; if the difference of both parameters exceeds certain limit then bisection occurs.

3.5.1 Algorithm of bisection method

The following diagram depicts the problem of overcutting and logic to solve this problem.

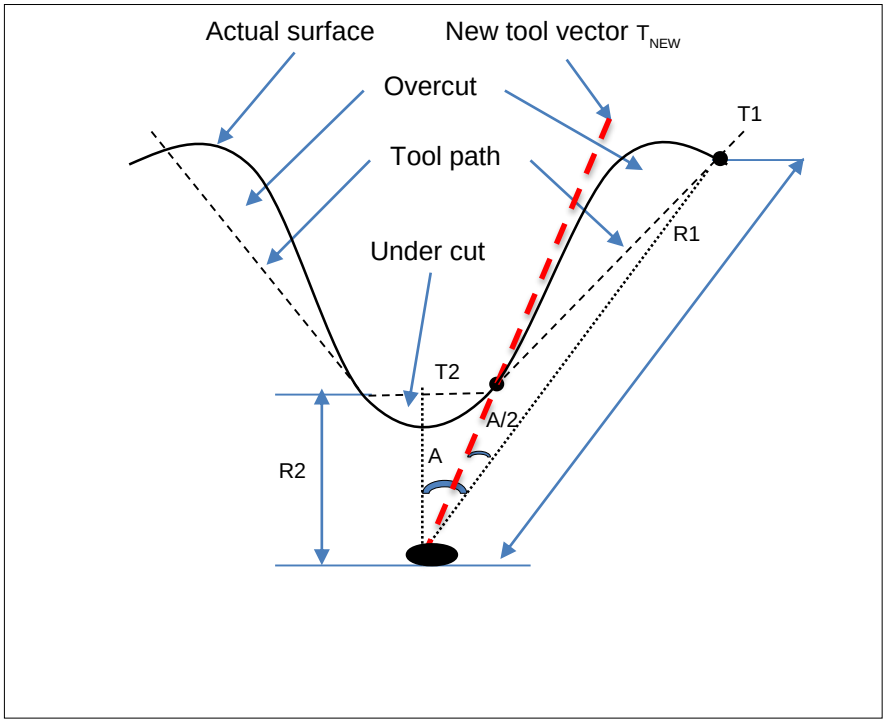


Figure 3.15 Overcutting and undercutting avoided by bisection algorithm

The algorithm of bisection method is explained as following:

Step 1: In case of helical footprint, first intersections are found with all the tool vectors. In case of Z-parallel footprint, intersection are found with all tool vectors in one loop.

Step 2: The bisection algorithm is run which will generate new tool vectors only if following conditions are satisfied at once:

From above figure,

a. If Radial step (R1-R2) > .5 mm

b. if Angular difference (A) > .1 degree

Then tool vector T_{NEW} is created by using following equations:

$$A_{NEW} = \frac{A1 + A2}{2}$$

$$Tx_{NEW} = r \times \cos(A_{NEW})$$

$$Ty_{NEW} = r \times \sin(A_{NEW})$$

$$Tz_{NEW} = \frac{tz1 + tz2}{2}$$

Where:

Tx_{NEW} Ty_{NEW} Tz_{NEW} are x, y, z coordinates of new tool vector.

A_{NEW} is angle of new tool vector.

R1 and R2 are radial distance of intersection points for tool vectors T1 and T2.

Tz1 and Tz2 are z coordinates of tool vectors T1 and T2 as shown in figure.

Step 3: Intersection points are found using new tool vectors and then both tool vector and intersection point are stored in the original linked lists.

Step 4: The linked lists are then sorted according to the angle.

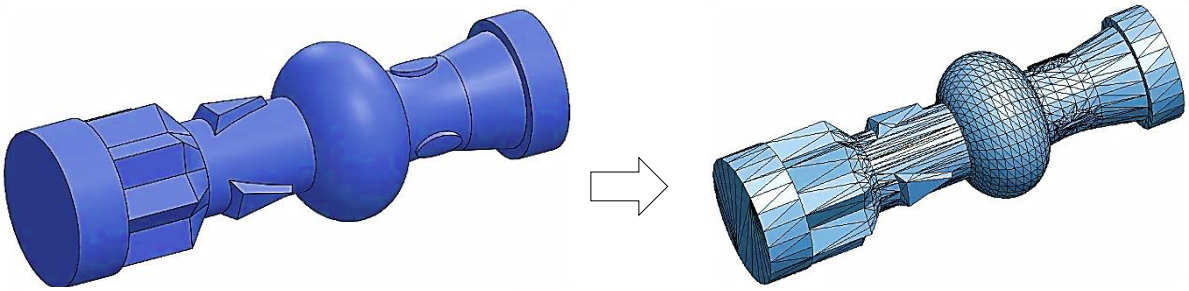
Step 5: Now, step 2 and step 3 are repeated till all the bisections are done.

CHAPTER 4

RESULTS AND DISCUSSIONS

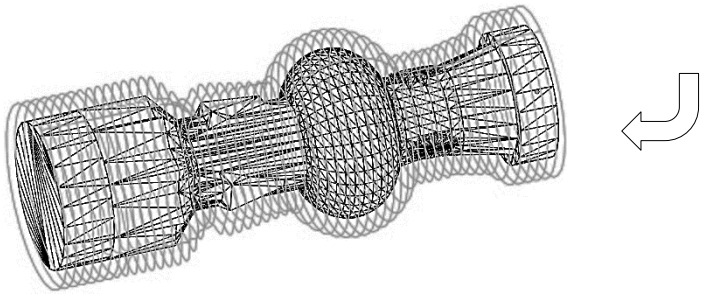
4.1 Simulation results of Ball drop algorithm using STL model:

The algorithm for generating tool path using “Dropping the ball” method has been successfully developed and is verified using CNC Simulator as shown in following figures:

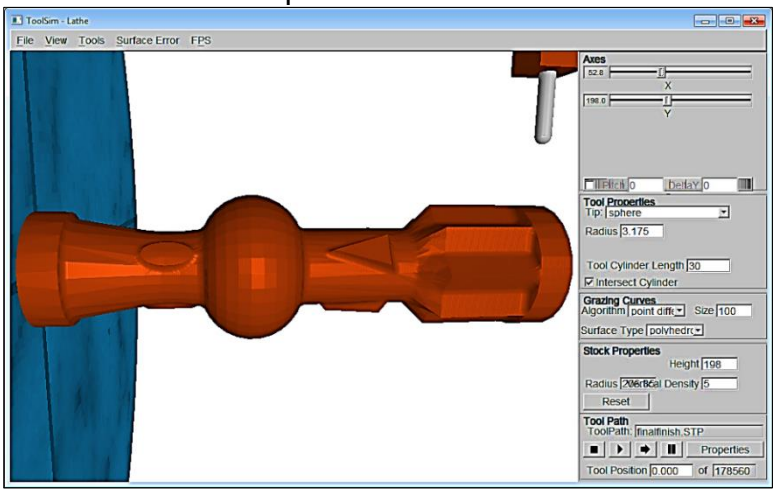


a. 3-D model of work piece

b. STL model of work piece



c. Tool path visualization

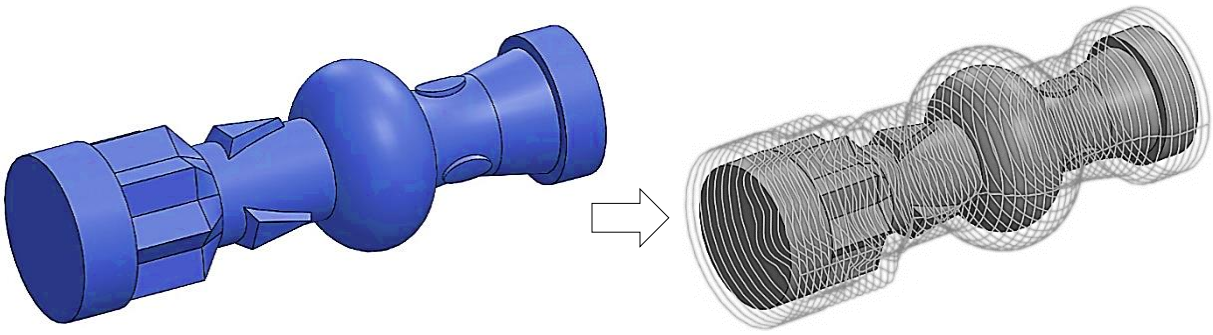


d. Tool path verification in ToolSim software

Figure 4.1 Results of Ball drop method using STL file

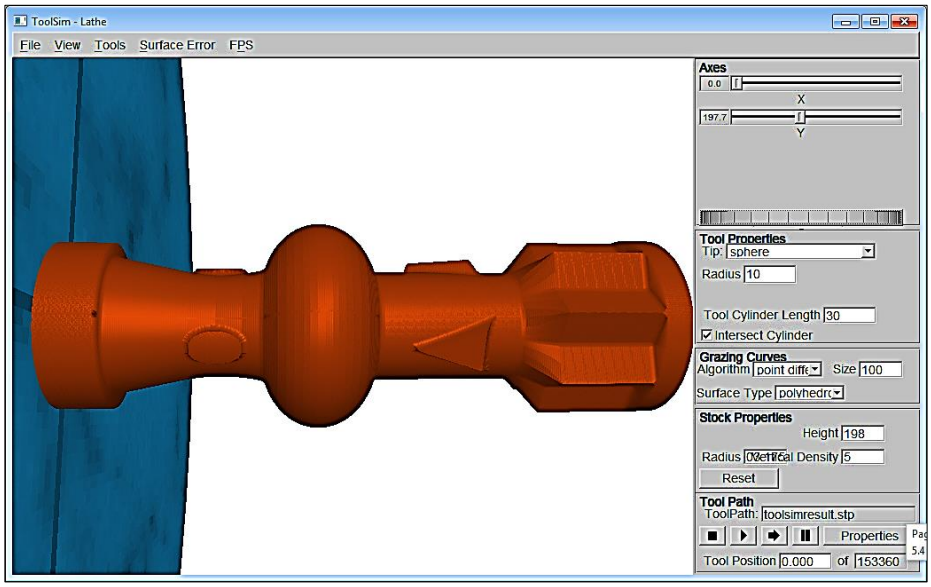
4.2 Simulation results of API method using B-rep model

The algorithm for generating tool path using “B-rep” method” has been successfully developed and is verified using CNC Simulator as shown in following figures:



a. 3-D model of work piece

b. Tool path visualization



c. Tool path verification in ToolSim software

Figure 4.2 Results of API method using B-rep model

4.3 Physical validation and comparison of Ball drop and API methods

The hypothesis that we proposed in problem definition is proven true by physically validation on PBG 2048 3-axis CNC lathe milling machine.

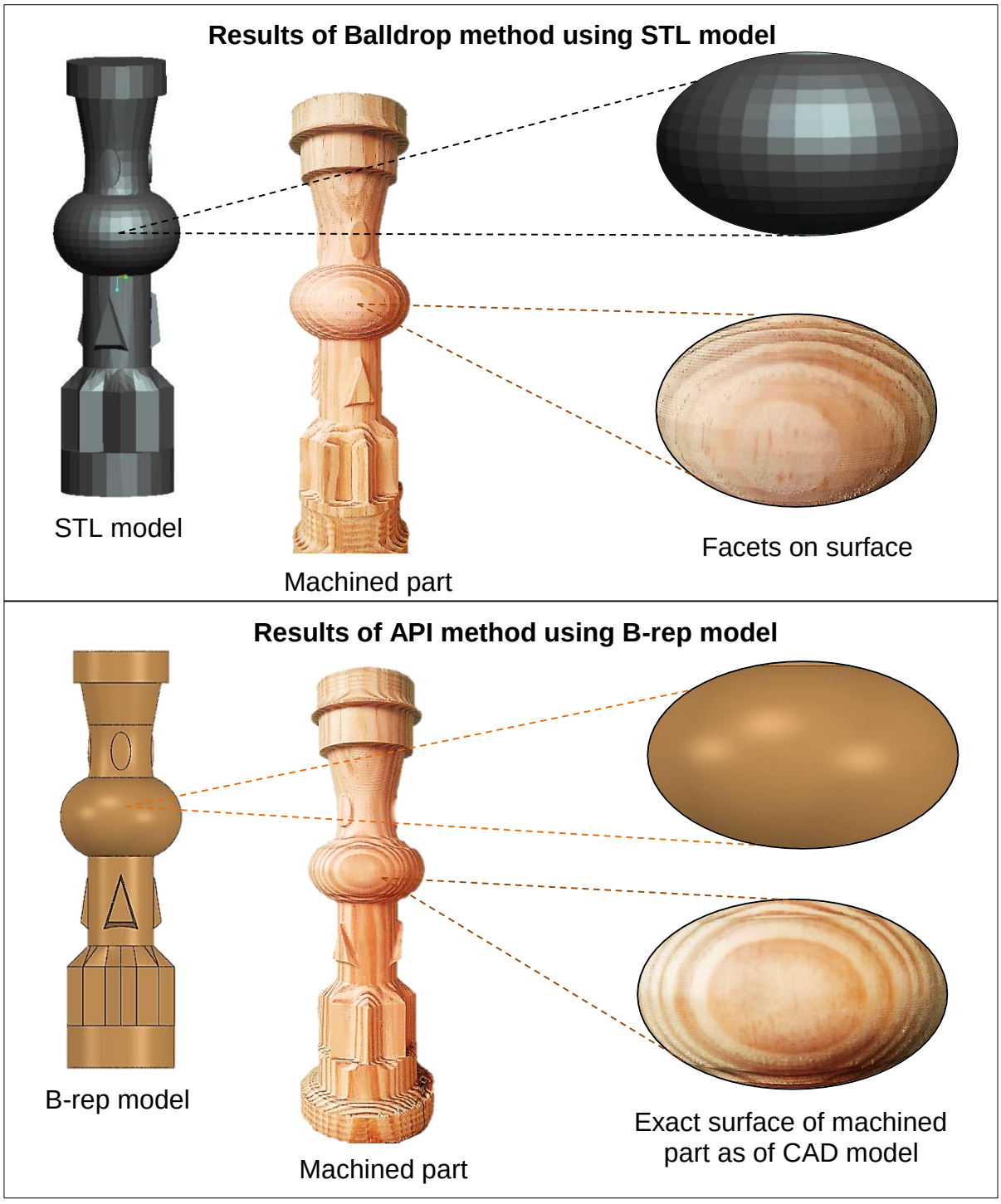


Figure 4.3 Results of physical validation on PBG 2048 3-axis CNC lathe

4.4 Results of bisection algorithm

Both in Balldrop and API method, using bisection algorithm the problems of undercutting and overcutting can be minimised in cases where radial step is large. This is verified by taking a crossectional view of 3-D model and plotting CL points on it in CREO software.

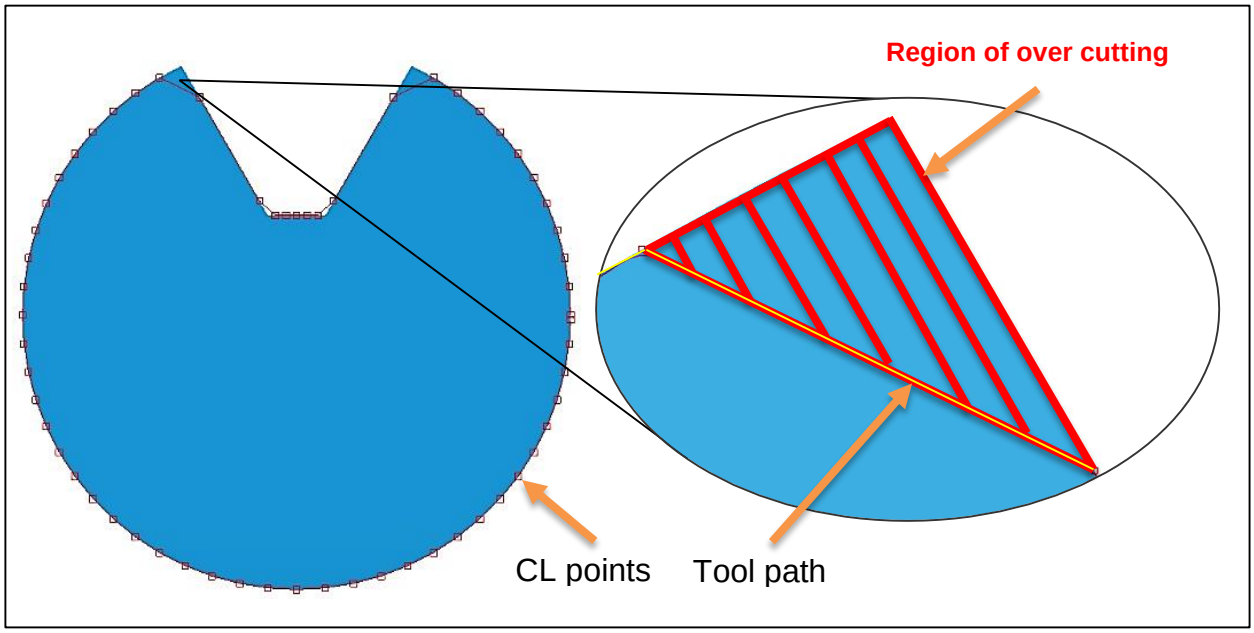


Figure 4.4 Tool path when bisection algorithm is not used

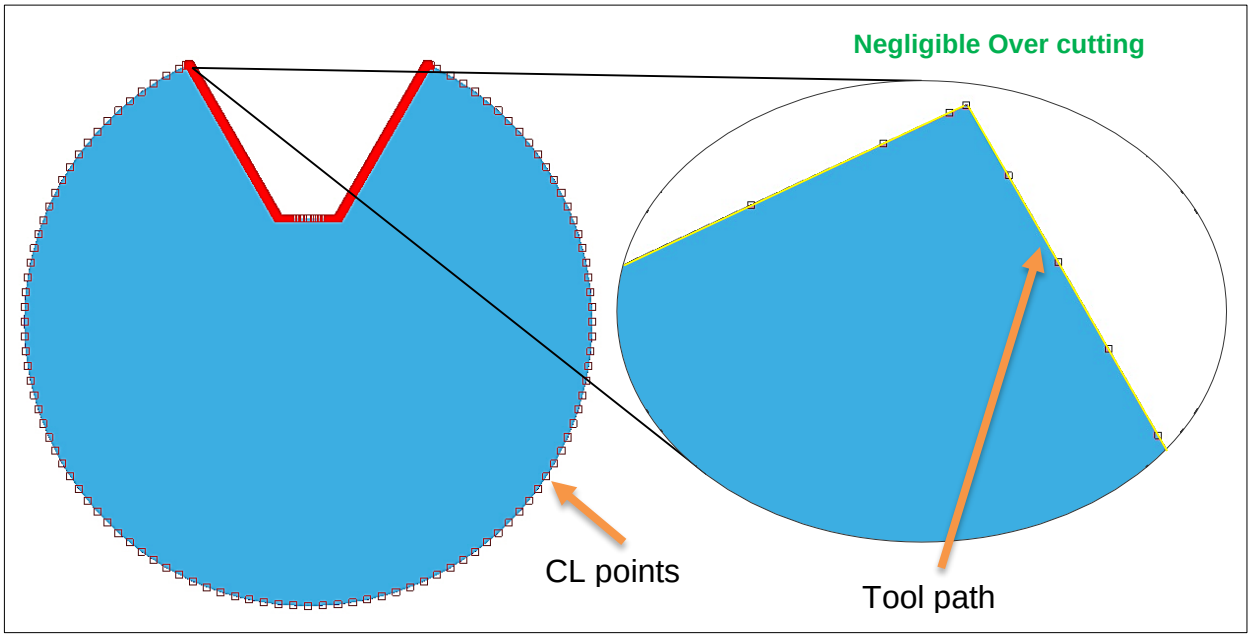


Figure 4.5 Tool path generated using bisection algorithm

4.4.1 Physical validation and comparison of algorithms with and without bisection

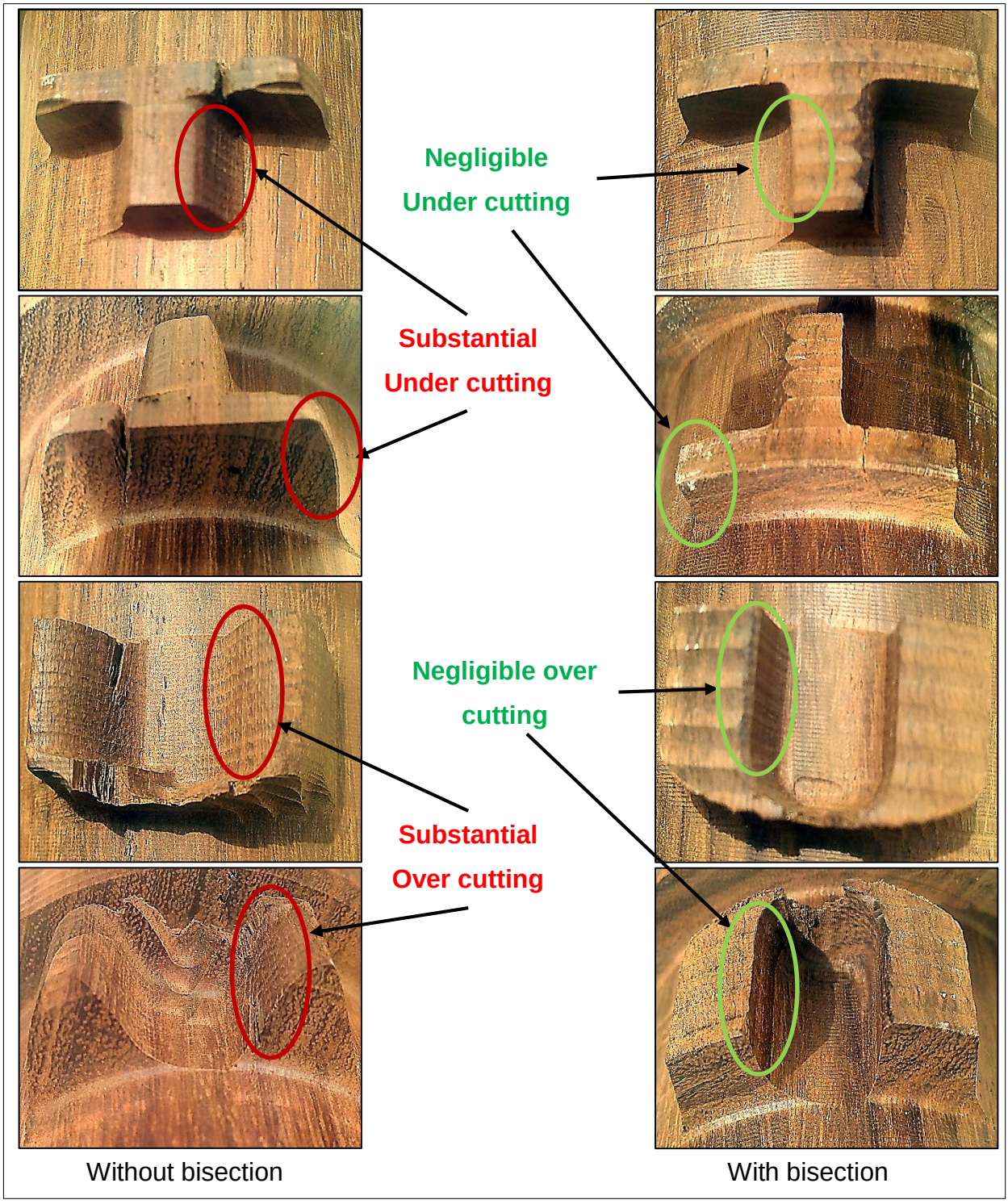


Figure 4.6 Validation done on lathe mill using text characters (T and U)

CHAPTER 5
FUTURE SCOPE

The work done in this thesis is confined to 3-axis lathe milling machine using ball nose tool, using two types of footprints: helical, z-parallel, and using API of SolidWorks only. The future work includes the generation of tool path using other types of tools such as conical, flat end mill, toroidal cutter etc. This work can also be extended to use Application Programming Interface (API) of other CAD packages. These algorithms can be further optimized to generate tool path for 5-axis CNC milling machine or other milling machines. Scallops has been a limitation, so this work can be extended for generating scallops free tool path. In addition, shape of the work piece must be according to rules defined in problem definition. Therefore, these algorithms can be further improved and modified to generate tool paths for other types of shapes such as bow shape or work piece with steep overhangs.

REFERENCES

1. Nick P. Manos et al., "Single Controlled Axis Lathe Mill," International Journal of Advanced Manufacturing Technology, 32(1-2), 2007
2. Juan M Jauregui-Becker et al., Computer-Aided Design and Applications 01/2009, pp.447-459, ISSN 1686-4360.
3. Vinodhkumar Somavar Muniappan, “Automatic Feature Recognition, and Tool path Generation Integrated with Process Planning”. http://uwspace.uwaterloo.ca/bitstream/10012/6979/1/SomavarMuniappan_Vinodhkumar.pdf
4. Paras Mohan Jasra, “Generalized Tool Path Generation Algorithm for Sculptured Pseudo Symmetric Surface Machining”, Master's thesis, Thapar university, Patiala. http://dspace.thapar.edu:8080/dspace/bitstream/10266/1032/3/1032Paras%2880781016%29.pdf
5. Kanderp Patel, “Web Based Automatic Tool Path Planning Strategy for Complex Sculptured Surfaces, 2010”, Master's thesis, University of Waterloo, Canada. https://uwspace.uwaterloo.ca/bitstream/handle/10012/5265/kandarpthesis1.pdf?sequence=1
6. Hsin-Chuan Chen et al., “Computer-Aided Process Planning For NC Tool Path Generation of Complex Shoe Molds”, the International Journal of Advanced Manufacturing Technology Volume 58, Issue 5-8, p 607-619.
7. Szu-Hung Cheng et al., “Study on the Cutting Simulation of Universal Gear Generator”, the Society of Manufacturing Engineers, Taipei.
8. Ding Songlin, “Tool path generation for CNC machining of free-form surfaces”, Ph.D Thesis, Mechanical Engineering, National University of Singapore. http://scholarbank.nus.edu.sg/handle/10635/13910?show=full

WEB REFERENCES

1. [Title: Fundamentals of tool path] ma.gnu.ac.kr/course/2007/acam/06-tpg.pdf (accessed on 9/06/2014).
2. [Title: Tool path generation methods] ma.gnu.ac.kr/course/2007/acam/07-tpgm.pdf (accessed on 9/06/2014).
3. [Title: Importing STL] http://help.autodesk.com/view/MFIA/2015/ENU/?guid=GUID-D99EBE4D-4C49-4BCB-85D2-D58CC10B913C (accessed on 10/06/2014).
4. [Title: Spheres, equations] http://paulbourke.net/geometry/circlesphere/ (accessed on 2/10/2013).
5. [Title: Mill express] http://bobcad.com/products/ (accessed on 12/7/2013).
6. [Title: CAMWorks] http://www.camworks.com/solidworks/ (accessed on 12/7/2013).
7. [Title: CAD-TO-MOTION Application] http://www.fresystems.com/product-ModusCAM-SolidWorks-api.cfm (accessed on 12/7/2013).
8. [Title: Ray intersection function] http://help.solidworks.com/2012/English/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~RayIntersections.html (accessed on 5/6/2013).
9. [Title: Insert offset surface function] http://help.solidworks.com/2013/English/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~InsertOffsetSurface.html (accessed on 1/2/2014).
10. [Title: Select by id function] http://help.solidworks.com/2013/English/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.imodeldocextension~selectbyid2.html (accessed on 2/2/2014).

11. [Title: Insert reference plane function] http://help.solidworks.com/2012/English/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IFeatureManager~InsertRefPlane.html (accessed on 2/2/2014).
12. [Title: Sketch3DIntersections Method] http://help.solidworks.com/2013/English/api/sldworksapi/SolidWorks.Interop.sldworks~SolidWorks.Interop.sldworks.IModelDoc2~Sketch3DIntersections.html (accessed on 3/3/2014).
13. [Title: Offset Sketch Method] http://help.solidworks.com/2013/English/api/sldworksapi/Sketch_Offset_Example_VBNET.htm (accessed on 14/4/2014).
14. [Title: Face Milling] http://cfnewsads.thomasnet.com/images/large/018/18441.jpg (accessed on 1/7/2014).