Efficient NC Toolpath Generation for 3-axis Machining of 3-Dimensional Freeform Faceted Surfaces Developed from Digital Images

A Dissertation submitted
in partial fulfilment of the requirements
for the degree of

Master of Engineering

in

CAD/CAM Engineering

by

Sumit Sood

Roll No. 801381023

Under the Supervision of

Mr. Ravinder Kumar Duvedi

Assistant Professor

(Mechanical Engineering Department)

MECHANICAL ENGINEERING DEPARTMENT

THAPAR UNIVERSITY, PATIALA

July, 2015

CERTIFICATE

I hereby declare that the thesis titled "**Efficient NC Toolpath Generation for 3-axis Machining of 3-Dimensional Freeform Faceted Surfaces Developed from Digital Images**" is an authentic record of my study carried out as requirements for the award of the degree of **Master of Engineering** in **CAD/CAM Engineering** at **Thapar University, Patiala** under the supervision of **Mr. Ravinder Kumar Duvedi**, Assistant Professor, Mechanical Engineering Department, Thapar University, Patiala during July, 2013 to July, 2015. The matter embodied in this report has not been submitted in partial or full to any other university or institute for the award of any degree.

Date: 14/07/2015

Sumit Sood

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

Mr. Ravinder Kumar Duvedi
Mechanical Engineering Department
Thapar University, Patiala – 147004

Countersigned by

Dr. S. K. Mahopatra
Sr. Professor & Head
Mechanical Engineering Department
Thapar University, Patiala – 147004

Dr. S. S. Bhatia
Dean of Academic Affairs
Thapar University
Patiala – 147004

Dedicated to

My respected parents
Mr. Deepak Sood & Mrs. Rama Sood

ACKNOWLEDGEMENT

I would like to express my profound exaltation and gratitude to my mentor Mr. R. K. Duvedi for his candid guidance, constructive propositions and overwhelming inspiration in the nurturing work. It has been a blessing for me to spend many opportune moments under the guidance of the perfectionist at the acme of professionalism. The present work is testimony to his activity, inspiration and ardent personal interest, taken by him during the course of his work in its present form. I am grateful to Dr. S. K. Mohapatra, Sr. Professor and Head, Mechanical Engineering Department for providing the facilities for the successful completion of this work. I am also thankful to Ms. Nisha Sharma and Mr. Sourabh Dewedi for allowing me to use their photographs in this work.

I would like to express my earnest gratitude to all my friends who directly or indirectly helped me for the successful completion of this report.

SUMIT SOOD

ABSTRACT

In recent years sculptured surfaces have gained a significant importance in engineering as well as in artistic applications. These surfaces are generally machined using Computer Numeric Control (CNC) machines for achieving required surface accuracy and repeatability. For CNC machining of such complex shapes/surfaces one needs accurate toolpath data. Manual computation of such toolpaths is either impossible or it requires a lot of time as well as expertise in relevant area. There is always high probability of error in manual part programing. Thus toolpath generation for the machining of these surfaces is a non-trivial task in itself.

Many commercial solutions are available for automatic generation of Numerical Control (NC) toolpaths for machining of sculptured surfaces. The cost of these software are justified for mass production applications and these software packages generally do not match the need of the small scale artisan or craftsmen because of higher costs and complexity involved in using these software. Thus there is always a need of non-expensive solutions for the automatic generation of toolpaths for such cases. Numerous paradigms have been developed by the researchers to aid this cause. One such solution is proposed in this work.

In present work an effort has been made to develop a windows based console application with a single page Graphical User Interface (GUI) that can capture freeform surfaces from digital images. Free form surfaces are generated in the form of point cloud and Stereo Lithography (STL) model. This application provides separate options to process normal image and human portrait. It can also generate toolpath to carve the image into wooden, metallic or stone plaques using a CNC machine with a ball end mill cutter. An algorithm is also proposed to make the raster toolpaths more efficient by eliminating the unnecessary tool location points by comparing their *z*-map for enabling the CNC machine tool to achieve the higher possible feed rates while machining such parts.

The developed algorithm generates the 3D point cloud data which can be directly used for NC toolpath data generation or the point cloud data can be processed into faceted/ STL format which can be then used for toolpath data generation using a ball end mill cutter of user defined diameter. The developed algorithm is tested on various surface models developed from digital images and the results are given in chapter 3 & 4.

In the later part of this work an improved algorithm for automatic generation of NC toolpath data with user defined scallop height is developed. The new method for constant scallop height control toolpath generation can be used to compute toolpath data for smooth parametric surfaces represented in STL/ faceted format. This algorithm is developed to enhance the machining efficiency by computation of appropriate feed forward and side step values for zigzag toolpath data used for raster machining of the smooth parametric surfaces taken in STL/faceted data format with user defined scallop height as the governing criteria. Thus the new algorithm is capable of generating toolpath data with *bidirectional scallop height control*. Two different Bezier surface part shapes have been used in faceted format to prove the capability of the algorithm. The developed toolpaths for the one of the Bezier surface part models have also been filtered through a customized *cutter location data reduction algorithm* to remove the tool location data points having the same z-height as the neighborhood toolpath locations.

All the toolpaths developed in present work are verified in virtual machining environment by using virtual 3-axis milling machine model in a simulator called ToolSim [1] available at State Initiated Design Center (SIDC) in Mechanical Department of Thapar University, Patiala. The graphical as well as the simulation results from the above algorithms are found satisfactory and explained in the required details in this thesis work.

TABLE OF CONTENTS

NOMENCLATURE

Symbol	Meaning
Y	: Single intensity value of the pixel in grayscale
R	: Intensity value of the red color for the pixel
G	: Intensity value of the green color for the pixel
B	: Intensity value of the blue color for the pixel
x, y, z	: Cartesian coordinates
μ_x , μ_y	: Means in x and y direction respectively
σ_x , σ_y	: standard deviation in x and y direction respectively
$G_0(x, y)$	: Gaussian function
A	: Amplitude
Z	: Mapped depth or height of the considered pixel
I_o	: Intensity value for the considered pixel
Z_{min}	: Minimum depth or height(user specified)
Z_{max}	: Maximum depth or height (user specified)
I_{max}	: Maximum value of intensity (among all pixels)
I_{min}	: Minimum value of intensity (among all pixels)
C_c	: Cutter contact point
C_l	: Cutter location point
u, v,s,t	: Parameters varying from 0 to 1
Δu	: Step size for parameter u
Δv	: Step size for parameter v
$\widehat{N}$	: Normal vector
T_1	: Initial tool position
T_2	: Final tool position
P_1, P_2, P_3 and P_4	: Vertices of triangle / Tool location points
P_5 and P_6	: Tool location points
r	: Radius of ball end mill
I_c	: Intersection point of two spheres
$\hat{u}$, $\hat{v}$, $\hat{w}$	: Direction vectors
M	: Midpoint of two cutter location on neighbouring tool paths
S_s	: Side step value

I_{stl}	: Intersection point on STL surface
h	: Scallop height (Distance between I_c and I_{stl})
F_s	: Forward step value
S_{sl}	: Side step left value
S_{si}	: Side step initial value
S_{sr}	: Side step right value
ϵ	: User defined scallop value
ε	: Limit value for point elimination algorithm

Acronyms

2D	Two Dimensional
3D	Three Dimensional
ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CAM	Computer aided Manufacturing
CL	Cutter Location
CNC	Computer Numeric Control
GUI	Graphical User Interface
NC	Numerical Control
RGB	Red Green Blue
STL	Stereo Lithography

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

Sculpturing or sculpting is a novel art of creating 3D (three dimensional) objects or patterns by carving solid materials, practiced from the time since human learned to use tools. Sculpturing is an essential tool that plays a very vital role in home or office décor. Sculpting using hand tools needs skills, hard work and time. Moreover manual sculpting cannot produce identical as well as very precise shapes. The advancements in the technologies like CNC machines, Computer Aided Design (CAD), Computer aided Manufacturing (CAM) have made it possible to create high quality sculptured artifacts. Also the production time for such complicated artwork has been reduced significantly. The identical copies of these artistic sculptures can be generated in any quantity with great accuracy. This is something which the most skilled hands cannot possibly do.

Though, the aforesaid technologies are extremely fascinating but initial investment in terms of capital in these solutions is very high which can only be justified when used for mass manufacturing. Fierce competition in market has diminished the price of CAD/ CAM software packages, but, they are still out of the reach of a hobbyist. Usually, hobbyists do not require high accuracy in their work which is provided by commercial CNC machines [2]. A number of economic CNC solutions are now available in market like CNC routers or single axis controlled CNC lathe milling machine for 3D sculpturing [2]. These machines do not require special expensive controller, instead they make use of personal computer which could perform the controlling work for these machines. The need is to provide economic CAD/CAM packages that may not provide all features but can still fulfill the major needs of the hobbyist.

There are certain special moments in life which are worth archiving and for that reason many hobbyists spend a lot of time capturing the digital images to memorize select moments. Technology has revolutionized the accessibilities to the high quality digital cameras which are capable of capturing quite good quality images in number of modes and themes. The available smartphones in the market are equipped with digital cameras to help click photographs anywhere and anytime. Most people share these photographs by displaying them at their residences or by posting them on social networking websites to gain admiration from their friends and relatives. They try different things to look classy for which they get their

portraits sketched, etched or carved into plaques to garner attention of their loved ones and proudly display these artworks at their residences.

Many automatic solutions are available these days to convert a digital image into a sketch. Carving out someone's face is believed to be a god gifted skill and the automation solution to carve out 3D impression for a portrait from a digital image is not available so far. 3D scanning coupled with CAD and CAM software packages can be used to capture 3D face and then CNC machines are used to carve it. This process is however very expensive and the cost involved usually cannot be justified for single artifact. Moreover the digitization of the dark gray parts like hair on head and face cannot be achieved. A great amount of research work has been done in the area of conversion of a normal two dimensional digital image into 3D data, but is limited to 3D digital displays which again need some additional tools to realize the desired effect [3–9]. Published literature shows that no solution is available that can automatically generate physical 3D human portraits from 2D images.

In present work an effort has been made to explore the possibilities of automatic generation of accurate 3D portraits while taking coloured or grayscale digital images as input for the algorithm.

1.1 Conversion of Digital Images to 3D Freeform Surfaces

A digital image is a file containing information about number of pixels and colour intensities at each pixel. There are two types of digital images— vector type and raster type. Vector type images are those in which mathematical formulation of graphical primitives like line, circle, spline etc. are used to define the images. These images are very popular in computer graphics. Raster type images, also known as bitmapped images, contains definite numbers of pixels which are stored in a form of a two dimensional array. Vector type images can be converted into raster images but not vice versa. Due to simple architecture, raster type images are popular for image processing applications. In raster image, each pixel stores the numeric value related to colour which represents the brightness of that colour at that pixel position.

Red Green Blue (RGB) model is commonly used to represent the colored image in electronic media in which red, green and blue are the primary colors and any color can be generated by adding these three primary colors.

Grayscale images only contain single value at each pixel which is the numerical value for the intensity of gray color which makes them more suitable for freeform surfaces generation.

Black color is given the lowest intensity value and white color is given the highest intensity value. All intermediate values represent different shades of gray color. Colored images can be converted into a grayscale format with user defined accuracy by approximating the three colors intensity at each pixel with a single intensity value.

Pixels from raster images can be mapped into a mesh in x-y in form of points in physical dimensions. To develop freeform surfaces out of the digital image, a third dimension apart from dimensions in x and y direction is required which can be defined as a function of colour intensity at each pixel in the image

Dark colors always create problems in image processing. Because of low reflection of light from dark regions even 3D scanners cannot detect them. In this present work, an effort is made to process dark hairs to create 3D human portraits.

1.2 Freeform Surface as a Point Cloud Data

Point cloud data is a collection of points defined in a 3D coordinate system. Point cloud data file contains x, y and z coordinates of each point in a point cloud. Points cloud generated from digital image can be used to generate 3-axis CNC toolpaths or for reverse engineering applications. The main advantage of using point cloud data directly for toolpath generation is that it is much faster as compared to toolpath generation using other file formats. Also direct point cloud data usage prevents the toolpath from the errors caused by approximate fitting of the surface through the point cloud [10].

As mentioned earlier, the pixel position and intensity information at each pixel can be utilized to compute physical coordinates for points to develop freeform surface. If the pixel density is high then the points will be very closely packed and thus the freeform surface can be represented by the point cloud itself. If the point cloud density is less the point cloud can be converted into a parametric surface model or into a faceted model.

Plentiful solutions are available to generate toolpath directly from point cloud data [11–17]. But, there are certain things that needed to be considered before generating the toolpath using point cloud data. The point cloud data must be dense enough so that the tool does not gouge in the empty space between adjacent points. To prevent gouging the space between points must not be more than 20% of the tool radius in case of ball end mills.

1.3 Freeform Surfaces in STL Data Format

STL format is a neutral format in which the 3D model is closely approximated by the triangular facets. It is a de facto standard for rapid prototyping as well as for toolpath generation for intricate surfaces. Every STL file contains information about the vertices of the triangles and their surface normal. There are two types of STL files Binary and ASCII (American Standard Code for Information Interchange). Binary STL files are written in binary format, therefore are compact in size but not easy to read. ASCII STL files are written in human understandable text. Although they are large in size as compared to binary ones but they can be easily edited using any text editor and it's very easy to detect any error.

Converting point cloud into parametric surface is not an easy task and there are always errors associated with the process. However, as the points in the point cloud are regularly arranged in x-y plane, they can be easily stitched together to form a faceted/STL surface model. Freeform surface in STL format can be used for toolpath generation as well as for rapid prototyping. STL format provides interference free toolpaths even if the model is enlarged by any factor. Faceted surface model provides best accuracy with ball nosed end mill cutter [18]. Toolpath generation using STL file is a well-established concept. Numerous toolpath generation techniques are available using triangulated models [19–33].

Although, faceted surface models provide interference free toolpath, but, the process of toolpath generation using STL file takes much more time as compared to toolpath generation using point cloud, since to locate the tool at particular position seven checks (one for triangle face, three for edges and three for vertices) needs to be performed for each triangle under the shadow of the tool.

To enjoy the advantages of both STL file and point cloud data, in present work toolpath can be generated using point cloud data when the density of the point cloud is sufficient to avoid gauging. Otherwise, STL model of the freeform surface is used to generate toolpaths.

1.4 Issues in Digital Image to Freeform Surface Conversion

There are certain issues related to the conversion of a digital image into a freeform surface.

- An image may contain some noise induced because of improper or poor lighting conditions during its capture, which may greatly affect the resulting freeform surface.
- While converting the colored image into grayscale there may be different colors that may be approximated by the same grayscale value. This may affect the quality of the freeform surface.

- Sharp colors and regions having lower and higher intensity pixels adjacent to each other will cause the abrupt changes in z dimensions of the corresponding points which are highly undesirable as abrupt changes in surfaces must be avoided in case if the surface is to be used for toolpath generation.
- While processing the image the dark regions are assumed to be the farthest and light regions to be the nearest to the viewer. This is an assumption which is not always true. For example, hairs of a person in image may be of dark color, which will cause craters in place of hairs in the resulting freeform surface. This is totally undesirable.

These problems must be taken care of automatically or with minimum manual effort for the successful conversion of digital image into freeform surface. An effort has been made in this work to counter these problems by applying Gaussian noise filter to counter abrupt changes in the dimensions along *z* direction. The image is processed by segmenting it and then the segments are processed separately to get appropriate results, which are finally clubbed together to obtain single point cloud data.

1.5 Toolpath Planning for 3-axis Machining of Freeform Surfaces

Tool path planning is one of the key aspects for the machining of sculptured surfaces. Machinists prefer efficient and interference free toolpaths which can produce artefacts with reasonable accuracy while keeping machining errors within the limits. A properly planned toolpath must not only ensure the machining accuracy but also ought to decrease the machining time by reducing the toolpath length. For wood carving and other artistic purposes high machining accuracy is mostly not required [2]. Therefore, in such cases fast toolpath is desired which is capable of providing the machining accuracy within specified tolerances. Tool path planning can be widely discretized into three phases [10]. First is to define the proper pattern for the motion of the tool. Second is to specify the points on this path through which the tool must be driven. Third is to check at every specified point that there must be no local or global gouging [10].

In industries ball end mill is mostly used for the machining of complex freeform surfaces [18]. Iso-parametric and iso-planar toolpath planning approaches are widely used for generating 3-axis, interference free toolpaths [10]. Cartesian toolpath generation method is very much similar to the iso-planar method which is a robust approach for generating 3-axis, interference free toolpaths for sculptured surfaces. The trajectory through which the tool is to be moved is generated by projecting a pattern drawn on a plane normal to the tool axis. This

pattern is known as tool footprint shown in Fig. 1.1. Tool footprint can be a zigzag path, parallel contours or space filling curves. Raster and contour toolpath footprints must give best results as the area to be machined is rectangle. Smooth Zigzag toolpaths are more efficient [37]. Therefore, for present work Cartesian toolpath method with zigzag tool footprint is implemented for generating toolpaths for ball nosed end mill cutter which is discussed along with results in chapter 5. In order to keep this report concise, only finishing toolpaths are considered however the same methods can be extended to generate roughing toolpaths in accordance with the roughing toolpath method proposed by Jasra [34].

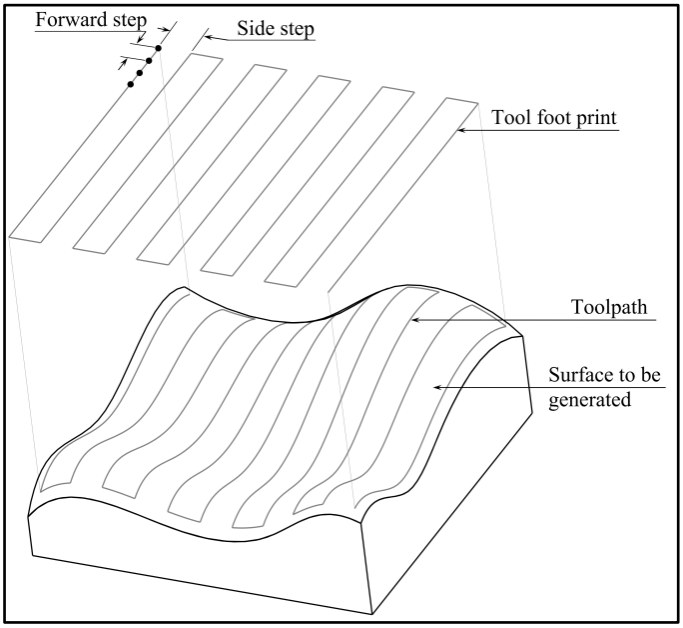


Figure 1.1: Cartesian toolpath method (Zigzag tool footprint)

1.6 Cutter Contact and Cutter Location Points

The cutter contact (C_c) point is the point at which the cutting periphery of tool touches the surface of the workpiece [20]. Cutter contact point varies from position to position and is therefore cannot be used to generate tool paths for CNC machining. Therefore in order to generate a toolpath a point is chosen such that it could be used to drive the tool through the tool path while its position remains fixed on the tool axis. This point is known as cutter location (C_l) point [20]. C_c point and C_l point is shown for ball end mill in Fig. 1.2. The objective of tool path planning is to find C_l points in the space in such a way that C_c points always lie on the surface to be machined. The interpolator in the CNC converts the C_l path to motion commands for all axis in order to achieve desired motion in multi axis machining.

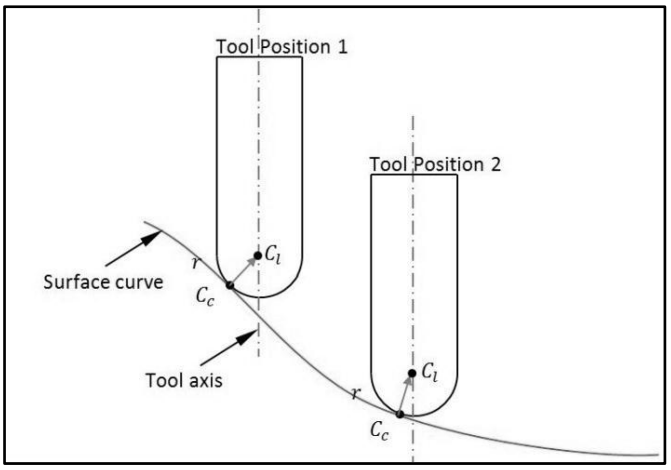


Figure 1.2: Cutter contact (C_c) and Cutter location (C_l) point

1.7 Cutter Location Data Reduction

Regular side steps and feed forwards steps discretization for toolpath generation creates lots of cutter location points. Machine has to move the tool through all these points for which it has to accelerate and decelerate frequently. This increases the machining time drastically [37]. In order to achieve maximum possible feeds the two points must be as far as possible. It is possible for 3-axis machining that the points can be compared for their z heights and if there are more than two adjacent points in toolpath having same z height, the intermediate points can be eliminated. An attempt has been made to develop an algorithm to eliminate such redundant points from densely created toolpath data. This concept cannot be applied to highly sculptured parts. However, it is very much feasible for the parts having flat regions.

While using STL data for toolpath generation, another method to enhance the efficiency can be the one where the scallop height is user defined. Rajiv [35] proposed a method to adjust the side steps for 3-axis raster toolpath by controlling the scallop height by user defined value. Approximation of the surface by triangles causes discontinuity due to which scallop height can never diminish. Therefore a minimum acceptable scallop height is to be specified by the user. The acceptable scallop height must be two times the tolerance specified for creating the STL model in case of machining with flat end mill [36]. Moreover this value cannot be a constant as sculptured surfaces are generally non-continuous. Effect of discontinuities on scallop height has to be studied in detail. An effort has been made to create a bidirectional scallop height control toolpath generation algorithm for STL surface. Effect of discretization of faceted models as well an increase and decrease of side step and feed forward steps with change in user defined scallop height value is studied.

1.8 Present Work

This dissertation work is dedicated to develop a GUI application for windows platform that can be used to generate 3D human portraits and images by developing freeform surface from digital image. Freeform surfaces generated are stored in the form of point clouds and as faceted models in ASCII STL file format. These freeform surfaces are then utilized for the generation of toolpath for plaque carving using ball nosed end mill cutter. Same toolpath can be used for the machining of plaques of various materials like wood, aluminum, marble etc. by changing the machining parameters. An algorithm is also proposed to improve the efficiency of raster toolpaths by reducing the number of tool location points while maintaining the accuracy of the machined part.

In the later part of this work, an improved algorithm for automatic generation of NC toolpath data is proposed for smooth parametric surfaces approximated in STL model. This algorithm calculates the maximum possible side steps and feed forward steps for each tool pass while maintaining constant scallop height in both feed forward as well as side step direction. All the toolpaths developed in present work are then validated in virtual environment by machining simulation.

Chapter 2
LITERATURE REVIEW

Present work is an attempt to appreciate the vision of the generation of 3D human portraits and other freeform surfaces from digital images by using the information available in it in terms of pixel positions and colour intensity. A literature study has been done for finding the appropriate way to convert 2D image data into 3D model and to generate efficient toolpaths for the machining of the same on 3-axis CNC machine. An effort has also been made to incorporate scallop height control in generation of toolpaths. In present chapter the relevant literature available for these aspects have been studied and presented in a suitable way. This study can be divided into three parts, which are discussed one by one in this chapter.

2.1 Digital Image to 3D Model Conversion

Digital image to 3D conversion is not a new concept and is widely used in computer graphics, virtual gaming, film industry etc. Researchers have developed algorithms that can convert a digital image into a 3D image which could be perceived as a 3D object by human eyes. This 2D (two dimensional) to 3D conversions exploits the fact that the images seen by each eye are slightly different in comparison to each other. Human brain can utilize this difference to sense the depth in vision which allows us to see this world in 3D [8]. Rockwood and Winget [3] proposed a method in which a 3D model can be constructed using 2D images of the object and an adaptable mesh. An energy function is defined between images and the mesh. This problem is then posed as a multi variable optimization problem with an objective to minimize the energy function. The mesh is refined at each iteration and when the problem converges, the mesh approximates the shape of the object. Zeng et al. [4] demonstrated a method for generating 3D surface model of the object from the multiple images of the object taken by a common consumer camera at different angles. The whole surface was generated by stitching the surfaces patches together that can be scaled up and down as per the requirement. Cheng et al. [5] proposed a hardware based 2D to 3D conversion system which can automatically convert 2D videos into 3D videos for 3D display. This method utilizes the combination of two depth mapping modules. First is depth from motion and second is depth from geometric viewpoint. Another method of converting 2D video to 3D by using the combination of motion depth and geometric depth methods along with H.264 decoder was addressed by Huang et al. [6]. They claimed improvement in the quality of depth map and reduction in computational time. Cheng et al. [7] showed the

successful conversion of 2D to 3D video using edge detection for depth map calculation. Sneha and Sheela [8] and Yang et al. [9] have also provided different algorithms that can convert a 2D image/video into 3D image/video for the purpose of viewing on 3D display. There are numerous examples and methods for 2D to 3D conversion for 3D viewing but no work has been found that could generate 3D physical model out of a digital image.

2.2 Toolpath Planning for Sculptured Surfaces

Toolpath planning plays a crucial role in CNC machining since the final outcome greatly depends upon the method used for toolpath generation. A review by Lasemi et al. [10] describes the conventional methods used to- generate toolpath, calculate proper tool orientation and select optimal tool geometry for the machining. Recent developments in the field of machining of sculptured or free form surfaces are also highlighted. Sculptured surfaces are generally made up of more than one complex surface. Hence it's very difficult to extract toolpaths directly from the CAD model. Most feasible solution to this problem is to approximate the surface with triangular facets. Recent developments in toolpath generation from STL models and point cloud data are discussed in the consequent subsections.

2.2.1 Point cloud based toolpath generation

Points cloud is available in case of reverse engineering applications in which the model of the product is digitized using coordinate measuring machine (CMM) or digital scanning machine. Toolpath generation from point cloud data is not a new concept. Many researchers have proposed various strategies to generate toolpath directly from point cloud as well as by converting it into surface model. The evident advantage of using point cloud data directly is that it prevents large time consumption for toolpath generation and error caused during the fitting of surface through the points [10]. Lin et al. [11] presented a method of generating tool path from 3D point cloud data, scanned from the physical model using CMM. The scanned points are converted into sectional curves which are parallel to each other and then these curves are then transformed into triangulated surface. This approximated surface is then offset to generate the cutter path. Lin and Liu [12] proposed a memory efficient algorithm in which first the point cloud data is stored using Z-map model. Roughing tool paths are then generated slice by slice for different depths by finding the intersecting points from point cloud and then identifying cut and uncut area out of the whole for a particular depth. Chuang et al. [13] provided a technique to merge multiple point clouds scanned out of same model and then toolpath is generated by converting final point cloud into STL model. Teng et al. [14] devised a new technique to generate tool path for efficient machining directly from 3D

scanned point cloud. The points are compared using numerical iterative method to divide them into two sectors as per their complexity. Different cutters are then used to machine different sectors. Efficiency as well as accuracy is assured by selecting optimum tools from standard set. Kayal [15] proposed a method in which machining error is calculated as a function of chordal error, scallop height and slope of curvature. 3D data points are used to generate grids of variable size which are used to generate tool path using inverse offset method in order to ensure efficient and accurate tool path of variable side step and forward step size while keeping the machining error within the tolerance range. Makki et al. [16] also proposed a method of generating toolpath directly from the 3D scanned data. For 3-axis machining the Z level scheme is used by incorporating voxel method. This method is then extended to 5-axis machining in which first of all, all the feasible orientations are computed and then the points are segmented so as to have minimum orientations to machine the model. The toolpath is then generated for each orientation. Zhang et al [17] conferred an approach to generate finishing tool path straight from 3D point cloud data. This method incorporates the calculation of forward steps based on the curvature of the tool path curve generated by intersection of the driving plane and the surface plane, specified by moving least square method. The developed algorithms calculate the forward interpolation steps as well as side steps based on the curvature while maintaining the chordal error within tolerance limit.

2.2.2 STL based machining of free form surfaces

Today most of the commercial CAD packages support STL file format. It is one of the most popular file formats among the machinists because of its simplicity and ease by which it can be used as an input to toolpath planning algorithms [19]. Main advantage of using STL format is that only linear operations are required for toolpath computation [20]. Even defected or incomplete STL models can also be used for toolpath generation [29,30]. STL model can be generated from parametric surfaces [21,22] or point cloud data [23]. Iso-planar toolpath generation method is prevalent for the machining of STL models [10,21,24,25,28]. In this method the toolpath is generated by intersecting the tessellated surface with a set of parallel planes. The gap between the planes is governed by the side step value which depends on the permissible scallop height value. Yuwen et al. [26] suggested an iso-parametric method for toolpath generation for the machining of triangulated freeform surfaces where the facets are parameterized by a harmonic function. This method claimed to have better efficiency and accuracy in terms of generated toolpath since the toolpath is generated along the boundary of the model. Sortino et al. [27] devised a novel approach to improve the

accuracy of the machined part by providing compensation for the geometric errors. The toolpath is first generated using original STL file. This toolpath is then used to machine a test part. This part is then scanned into a 3D model which is further used to modify the original STL model. Toolpath generated using this modified STL is believed to have higher accuracy. Chen and Shi [28] conferred the procedure for the generation of toolpath for 3-axis milling using vertex offset method to generate offset triangular surface to generate toolpath curves using iso-planer method.

2.2.3 Tool positioning strategies for 3-axis milling

Tool positioning is an essential step towards the generation of a toolpath since it decides how the tool must be placed at different positions along the tool trajectory so as to calculate the corresponding tool location points. Conventionally, surface offset method is used to generate the cutter location surface that is intersected with the tool driving planes in order to generate the tool driving curves [20]. Offsetting the surface is not a trivial task and sometimes it may cause self-intersection of the surface. Calculation of intersection curve also includes complex calculations if the surface is a freeform surface. Moreover, the offset method is suitable for ball nose end mill cutters and for other cutters like toroidal end mill the degree of complexity further increases. This problem can be simplified by using STL model. STL surface offset method is very popular among the researchers and is used explicitly to generate toolpath for ball mill [11,21–23,22,30,32,33]. Though this method is widely used but it is not computationally efficient.

Another method is devised by Manos et al. [2] known as “Ball drop” cutter location method for ball nose end mill cutter. This method is applicable to surfaces consisting of points governed by vectors in three dimensional space or STL models. This approach first generates a two dimensional grid is x-y plane and then for each tool location the z-height is calculated by dropping the ball, of radius equals to that of the cutter’s, along the axis of the tool and then computing the first contact point of the ball with the surface. For STL models the method for finding the tool location can be discretised in four steps—Shadow check, triangle check, edge check and vertex check.

This method can be easily extendable to other shapes of the tool like flat end mill cutters or fillet end mill cutters. A doughnut is dropped instead of a ball in case of filleted end mills. This approach ought to be a robust way of generating toolpath and is implemented by many other researchers [18,34–36].

2.3 Toolpath Efficiency Improvement

Scallop height in machining governs the surface finish and accuracy of the part produced. Iso-planar and iso-parametric toolpath techniques utilizes constant side step value which cannot always ensure the efficiency of the toolpath. Small side steps results in smaller scallops but longer toolpath claiming more machining time. Larger side steps results in lesser machining time but does not guaranty desired machining accuracy and surface finish. Thus, side step size needs to be optimized to balance between machining efficiency and machining accuracy.

Iso-scallop method is the answer to this problem. Feng and Li [31] presented toolpath generation method for constant scallop in which two offsets of original design surface are used. One is scallop surface which is used for generating constant scallop curves to drive the tool and another is tool centre surface which helps in calculating tool location points for each tool pass. Lee et al. [32] devised an approach to generate toolpaths with constant scallop height using parametric mesh. The surface is first discretized for generating the offset mesh and then offset triangulated surface is generated. Using the first drive plane, driving curve is obtained. Tool path is generated using that curve and then the distance of next drive surface is calculated so as to keep scallop height constant and the process goes on. Yang and Feng [33] conferred a method to generate finishing tool path for 3-axis machining. The machining error is controlled by upper and lower limit offset surfaces. Z-map method is used to remove self-intersections of the triangulated facets. To ensure constant scallop height another offset surface is used to compute the side step value.

Patel et al. [18] formulated a novel approach to choose optimal tool shape for the machining of free form surface. The overall material removal for machining with each tool shape is calculated. The tool that removes more material for same toolpath with minimum scallop height is then selected.

Rajiv [35] proposed a new method for toolpath generation for 3-axis vertical milling machine in which the side step is optimized using bisection method to keep scallop height within permissible limit. Side step is decreased if the calculated scallop height value is more than user specified maximum scallop height to ensure desired machining accuracy and is increased if the calculated scallop height value is less than the user specified value to reduce the toolpath length. This results in toolpath with adaptive side steps while keeping feed forward steps constant.

Kim and Choi [37] developed mathematical model for the computation of machining time which considers acceleration and deceleration of the machine drives for machining time calculation. A comparison between different footprints which showed that smooth zigzag toolpath is most efficient.

2.4 Conclusion

This literature survey has provided the insight that numerous algorithms have been developed for a normal digital image to 3D image conversion for 3D display, but no literature is available for 3D human portrait generation from single digital image.

Numerous methodologies are available to generate toolpath from point cloud data as well as STL data out of which Cartesian toolpath planning method is best suited for present work. Ball end mill gives best machining results for sculptured machining [19]. Therefore ball end mill is selected for machining.

Available literature shows that for improving toolpath efficiency, toolpath length is considered as the prime factor which is kept as small as possible by varying side step values while keeping the scallop height constant. Lengths of toolpath line segments, i.e. spatial distance between tool lactation points have huge effect on machining time [37]. No approach has been found that can eliminate needless intermediate tool location points from automatically generated toolpaths for sculptured machining.

Toolpath generation algorithm proposed by Rajiv [35] for controlling scallop height is unidirectional. Thus, it optimizes side steps only. This method can be extended for bidirectional control of scallop height to compute side steps as well as feed forward steps so as to generate efficient toolpaths for surface machining.

In present work, a single page GUI application for Windows based computers to generate 3D portraits and freeform surfaces from digital images is developed. Toolpath generation method for such artefacts is discussed and an algorithm is also proposed to reduce the machining time by eliminating unsought intermediate points without compromising the machining efficiency. Bidirectional algorithm based on Rajiv's [35] methodology is also proposed to generate efficient toolpath from STL model for 3-axis machining. Toolpaths generated for various parts are then successfully verified in milling simulation software.

Chapter 3
DIGITAL IMAGE TO FREEFORM SURFACE CONVERSION

A digital image contains a lot of information in terms of color intensity at every pixel. This paradigm treats every pixel position as a physical position in space and uses the color intensity at each pixel to extract *z* depth/height information. Thus, the final 3D model essentially depends on each pixel's color and its intensity. Figure 3.1 depicts the process of Digital image to freeform surface generation in present work.

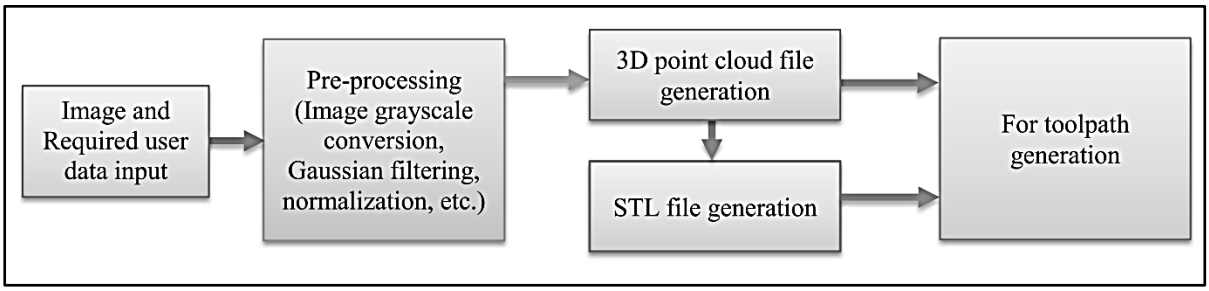


Figure 3.1: Basic process flow of digital image to freeform surface conversion

Based on this approach, a Microsoft windows console application with a single page GUI is developed. This application is simple and user friendly. The whole process of the application can be divided into three parts— Digital image to 3D freeform surface generation, toolpath generation, toolpath efficiency improvement. Figure 3.2 shows the process flows chart for the conversion of digital image to 3D freeform surface using the application developed. Features of this application are discussed with examples in the subsequent sections. Visual C++ 2010 is used as the programming tool for this console application. OpenCV 2.4.10 is used for image processing. OpenCV is an open source computer vision library containing thousands of computer vision algorithms available for free, to use under open source Berkeley Software Distribution license [39].

3.1 Digital Input Image

Raster type digital image in various formats like jpeg, png, bmp, gif are supported by the developed application and can be taken as input for the developed program. Input Image can be colored or grayscale. The final output of the developed approach depends greatly upon the quality of the input image since the 3D model is generated using the information available in the image. There are several factors like image size/ resolution, image brightness etc. that are needed to be considered for selecting an image for freeform surface generation so as to get best possible results. These factors are discussed one by one in subsequent subsections.

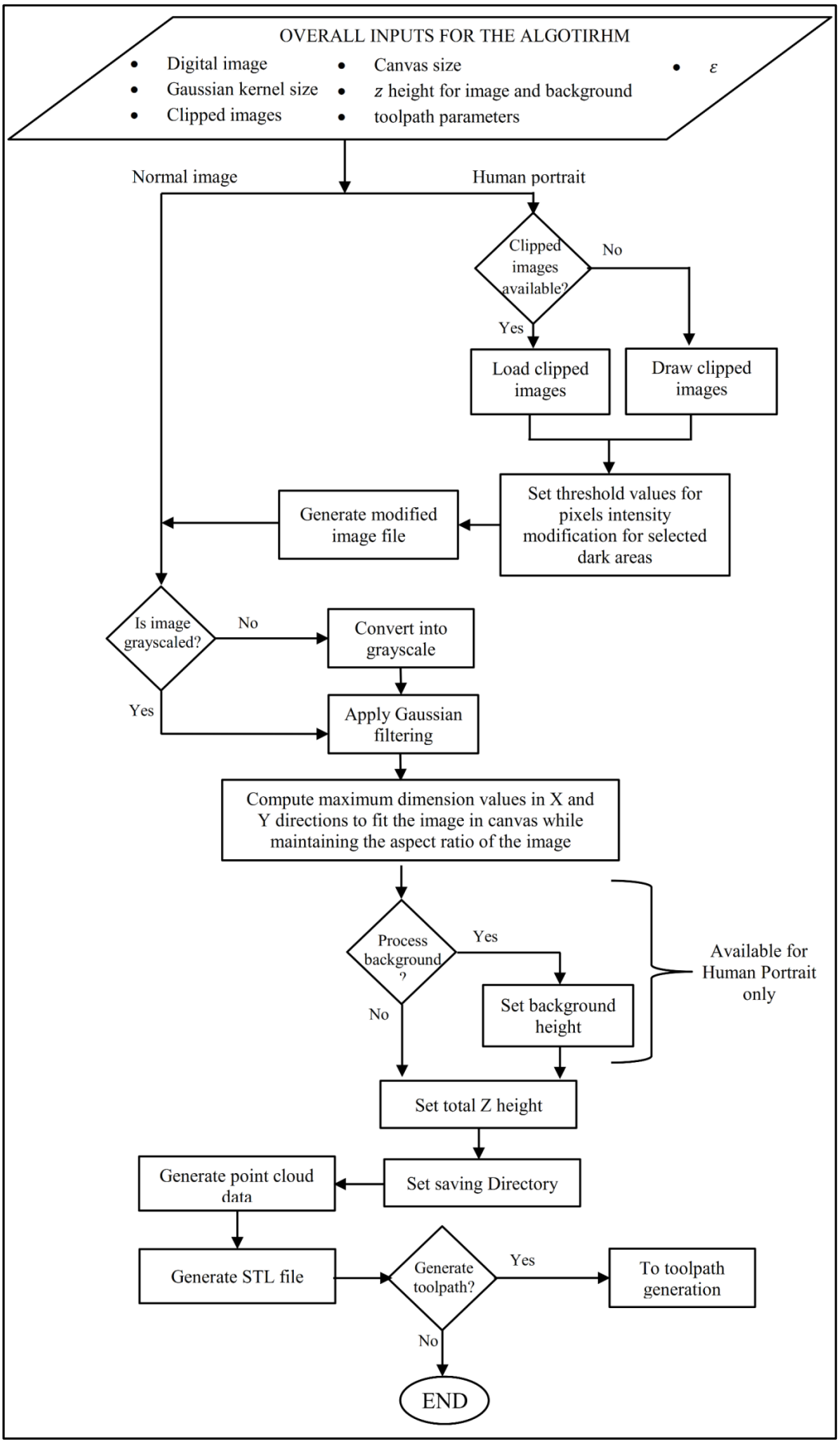


Figure 3.2: Digital image to 3D model conversion algorithm flowchart

3.1.1 Image resolution

The resolution of a digital image can be defined in terms of number of rows and number of columns of pixels in an image. For example a resolution of 800×600 means that there are 600 rows and 800 columns of pixel in the image constituted of 480000 pixels. In the proposed algorithm each pixel corresponds to a single point in a 3D space. Hence, larger the number of pixels, denser will be the point cloud data.

Maximum size of the image that can be processed using this application is limited by the memory available for dynamic allocation, storage memory and constraints imposed by the used open source image processing algorithms. Image of 10000×10000 pixels have been successfully tested with this application which is much higher than the resolution usually required for the practical application of this methodology.

3.1.2 Image brightness

Image brightness or image intensity plays a key role in depth calculation in presented approach. The brighter region in the image is considered closer to camera while the darker region is believed to be the farthest. Improper lighting and environmental conditions may cause shades in the image.

Figure 3.3 shows (a) image with improper lightening causing shade on the right side of face, thus, making it to appear darker in comparison to the left, (b) image taken in low lighting conditions making image absurd as an input for this approach and (c) image uniform illumination over the face.

Figure 3.3: Images with different illumination: (a) image with improper lighting, (b) image with low lighting and (c) image with proper lighting

For the developed methodology, properly illuminated image is always desired. Since, there exists no provision for taking care of unnecessary dark regions in the image at present. The problem of improper illumination can be countered first handedly while clicking it.

3.1.3 Image colors

Colors in the image plays a significant role in freeform surface generation since the color intensity at each pixel color intensity is used to compute *z* dimension. In case of 8-bit grayscale images black color is considered as base color having 0 intensity value. White color holds the highest value i.e. 255. All the other colors have grayscale intensity value between 0 and 255 when they are converted into 8-bit grayscale image.

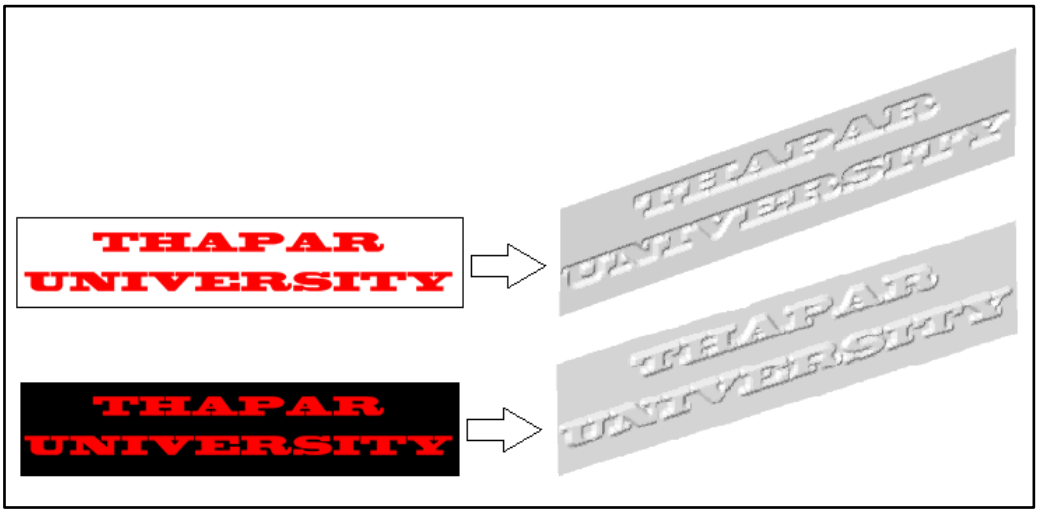


Figure 3.4: Effects of colors on 3D model

Colour selection in the image greatly affects the final output, based on which engraved or embossed text letters or patterns can be generated using presented methodology. Figure 3.4 shows an example in which colors decide whether the letters in the image will be engraved or embossed.

3.2 Methodology for Digital Image to 3D Freeform Surface Conversion

Two different approaches have been implemented to convert digital image into 3D freeform surface. First approach treats the image as normal image and the second approach provides the option to process human portraits. Both of these approaches are discussed later in this chapter. There are certain common steps involved in both the approaches which are discussed in subsequent sections. Moreover, to use the canvas size efficiently, the application automatically calculates the best fit to use the maximum area of the canvas size provided by the user while maintaining the height to width ratio of the image unaltered. All output files are stored using image file name as prefix so that they can be easily identified. Interface of the developed application is shown in Fig.3.5.

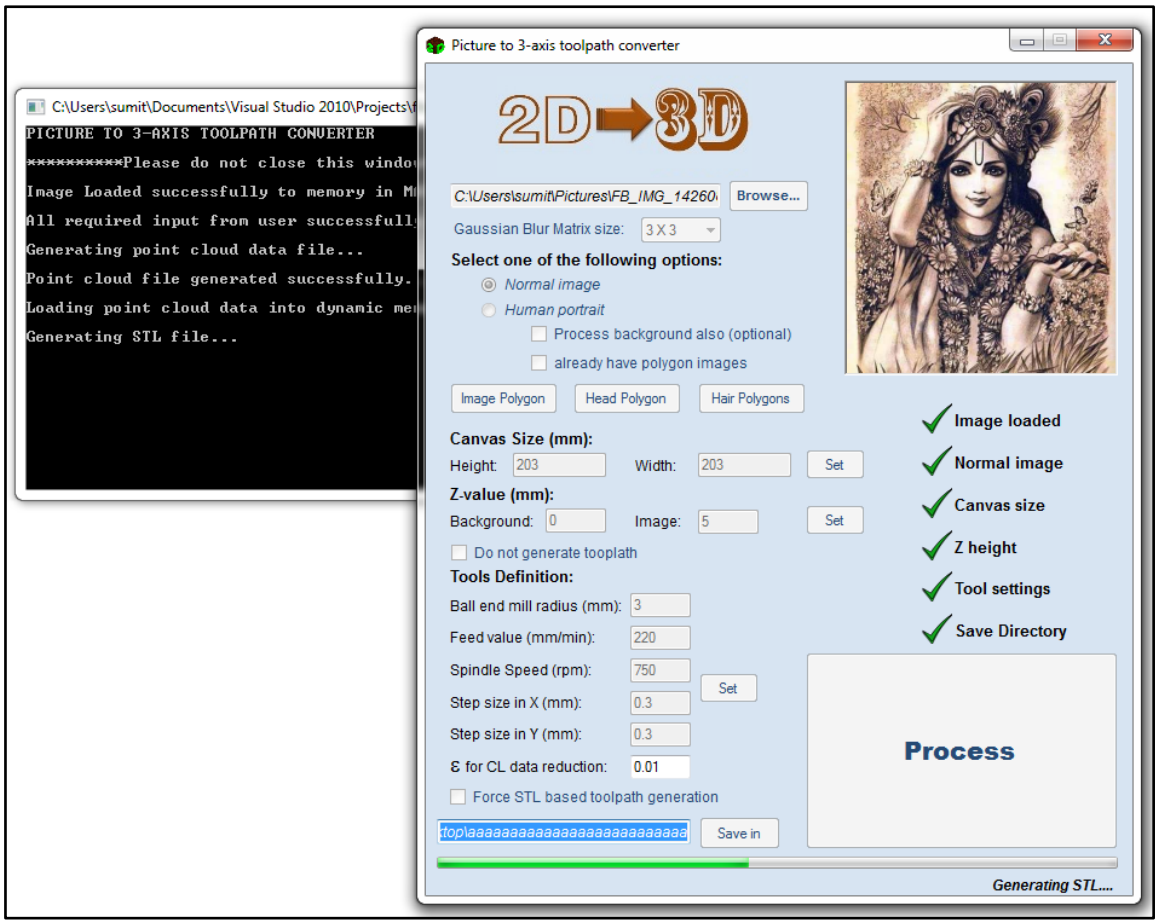


Figure 3.5: Console application interface

3.2.1 Colored image to grayscale conversion

Grayscale image is a single channel image having single brightness value at every pixel. Grayscale images are widely used in image processing and computer vision and it is suitable for depth information extraction. An 8-bit grayscale image have (2^8-1) i.e. 255 discreet levels to represent the brightness which are adequate for the present study.

Any colored image with RGB model can be converted into grayscale image by taking the weighted average of the intensities of three colors using the formula [39]

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \qquad (3.1)$$

where Y is the single intensity value of the pixel in grayscale, R is the intensity value of the red color for the pixel, G is the intensity value of the green color for the pixel and B is the intensity value of the blue color for the pixel. Figure 3.6 shows the colored image and its corresponding grayscale image which is generated by using the above mentioned formula.

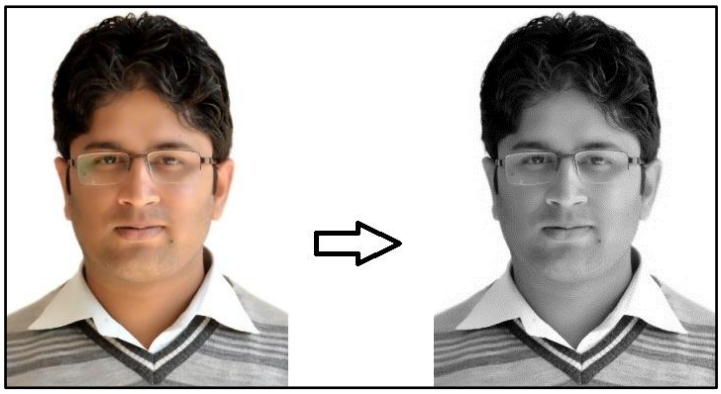


Figure 3.6: RGB to grayscale conversion

3.2.2 Gaussian smoothing (blurring)

For toolpath generation, the abrupt changes in adjacent tool locations are undesirable. In an image file the brightest as well as darkest region can exist adjacent to each other. This results in a toolpath containing abrupt changes in z direction since the brightest region represents the highest z value and the darkest region represents the lowest z value. To overcome this problem, Gaussian blurring method should be implemented. It's an application of mathematical Gaussian function in which the neighbor pixels of the concerned pixel are used to modify the intensity of that concerned pixel. Gaussian function is calculated by using the formula [39]

$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}+\frac{-(y-\mu_y)^2}{2\sigma_y^2}} \tag{3.2}$$

Where $G_0(x,y)$ is the Gaussian function, A is the amplitude, μ_x and μ_y are the means in x and y direction respectively, σ_x and σ_y are the standard deviation in x and y direction respectively. The Gaussian filter also makes the image smooth and reduces unnecessary details which make the artifact more machining friendly. Figure 3.7 shows the original image and the image obtained after applying Gaussian filtering. In the present work, built-in function of OpenCV (open source computer vision library) is used for performing Gaussian blurring on input images.

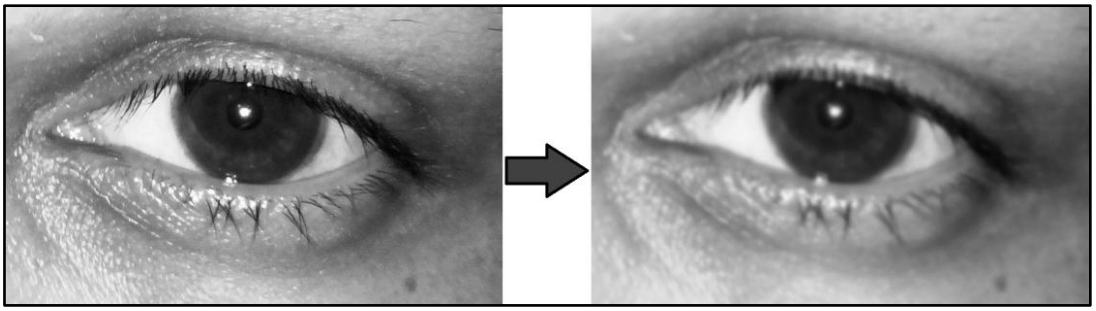


Figure 3.7: Gaussian blurring effect: Original image (left), image after Gaussian blurring (right)

3.2.3 Gaussian kernel size

Kernel is a matrix that is used for image convolution. Gaussian kernel decides how many neighboring pixels are to be considered to normalize the brightness value of the pixel using equation 3.2. Kernel size of 3×3 means that 9 pixels are considered for the calculation including the pixel itself. Larger kernel size induces more blur to the image. In the present application kernel size is kept variable and its value can be selected out of the options available. Thorough study of the effect of kernel size is required to select the best suited kernel size for the image to be processed. For present work kernel size is used as 3×3 and 5×5 which gives satisfactory results.

3.2.4 3D point cloud generation

3D Point cloud is generated by evenly distributing the pixels over the canvas area in x-y plane which is be covered. x and y position for each pixel is then computed as the location of the corresponding point in the point cloud. Dimension in z direction is calculated using the intensity value. The intensity value in the range of 0-255 is mapped to $Z_{min} - Z_{max}$ range. This is done by using a simple mapping formula

$$Z = (I_c - Z_{min}) \times \frac{Z_{max} - Z_{min}}{I_{max} - I_{min}} + Z_{min} \tag{3.3}$$

Where Z is the mapped depth or height of the considered pixel, I_c is the intensity value for the considered pixel, Z_{min} is the minimum depth or height (user specified), Z_{max} is the maximum depth or height (user specified), I_{max} is the maximum value of intensity (among all pixels) and I_{min} is the minimum value of intensity (among all pixels). Figure 3.8 shows the point cloud generated from an image without Z mapping and point cloud generated when Z_{max} is taken as 5mm and Z_{min} is taken as 0.

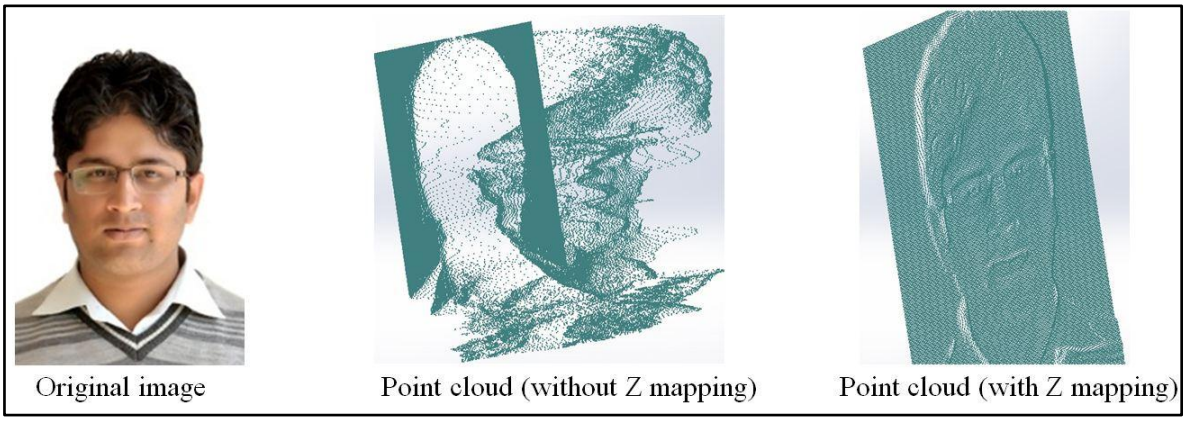


Figure 3.8: 3D point cloud from digital image

3.2.5 STL model generation

STL model can be conveniently crafted from point cloud data generated in present work as all the points in the point cloud are properly arranged in x-y plane. A simple stitching algorithm has been used to stitch the points into ASCII STL file. Figure 3.9 shows a 3D point cloud and its corresponding STL file.

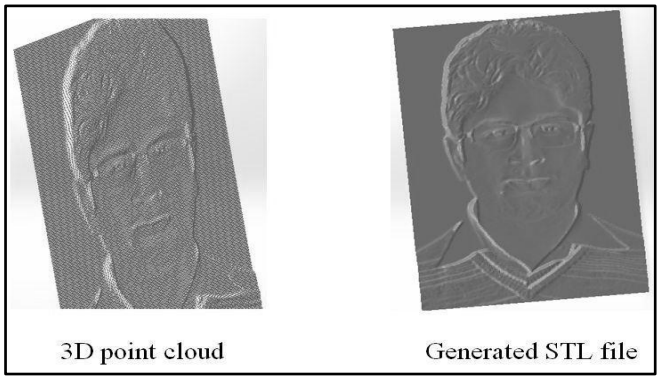


Figure 3.9: STL file from point cloud data

3.2.6 Image processed as normal image

This option treats the input image as an ordinary image and processes it by the basic methodology discussed earlier. The final output solely depends on the colors and their intensity in the image. Any image in which the positioning of different regions of an image must be only relative to each other in order to make sense out of the 3D model can be processed in this mode. This mode is therefore appropriate for converting the textures, patterns, artistic sketches, images of deities, complex images containing nature scene etc. to freeform surface. Figure 3.10 shows one such application for 3D carving of an artistic image.

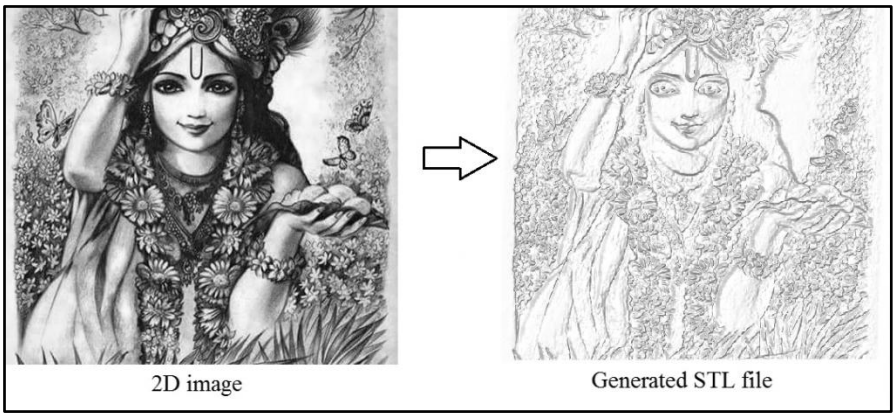


Figure 3.10: STL file generated using normal image processing

This method is however not suitable for processing human portrait images. Due to dark color of human hair, the region containing hairs would get depressed as compared to the face area which results in an unpleasant look of the 3D model. To overcome this problem, a separate approach is presented to process the human portrait.

3.2.7 Image processed as human portrait

This method is not fully automatic and requires user's intervention at various steps. Under this option, application allows the user to select the human figure in the image to separate it from the background. This is done by using the polygon selection tool to select the image as shown in Fig. 3.11. Once the selection is done, application generates an image of same size as that of input image's where the human portrait area is marked as black and the rest of the area is marked as white. Likewise, two other additional images are generated. The first is for the selected head hairs and the second is for the selection of facial hairs and the details that are needed to be raised as compared to other bright regions. Multiple objects/regions can be selected by drawing multiple selection polygons.

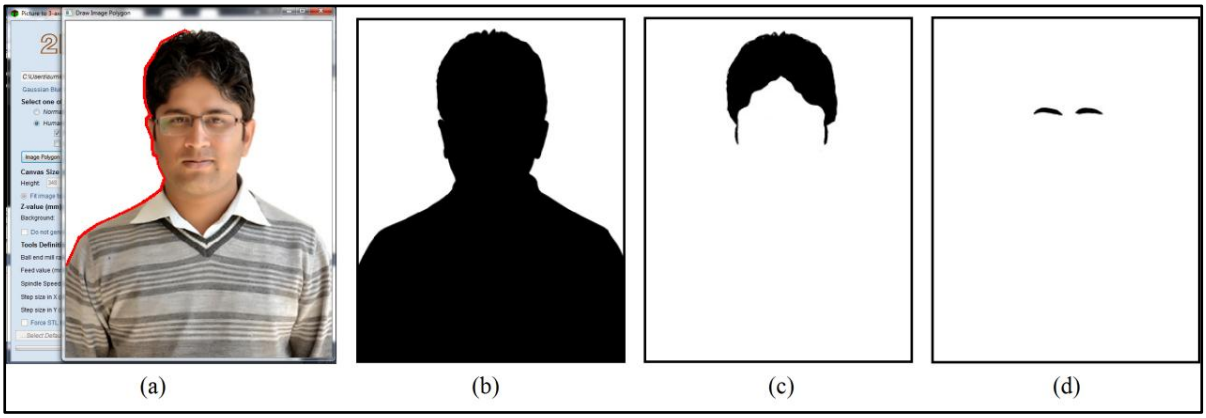


Figure 3.11: Human portrait processing: (a) Image region selection, (b) Image area selected, (c) Head hair area selected and (d) Other facial hairs area selected

The three images generated at this step are stored in computer's storage memory so that they can be used directly if the user wants to process the same image again. The selection tool is bit difficult to use. Therefore, these additional image files can also be generated using any image editing software and then they can be used as direct input to the application. The background area can be discarded or processed as per user's choice. When the background area is required to be considered, it is mapped to user defined background's z value and the human portrait is lifted above the background.

To elevate the dark region specified by the user, the intensity of the pixels in that area is modified. The proportion by which the intensity of pixels is modified is variable and can be adjusted by the user as per requirement as shown in Fig. 3.12. After this modification, image is processed using the same approach as that is used for normal image processing.

Figure 3.12: Pixels intensity modification

3.3 Procedure to Use the Application

This application is very easy to use as at every step, only relevant options are available for the user to select. Step-by-step procedure to convert digital image into freeform surface is explained with the help of snapshots to provide a clear perspective of the application.

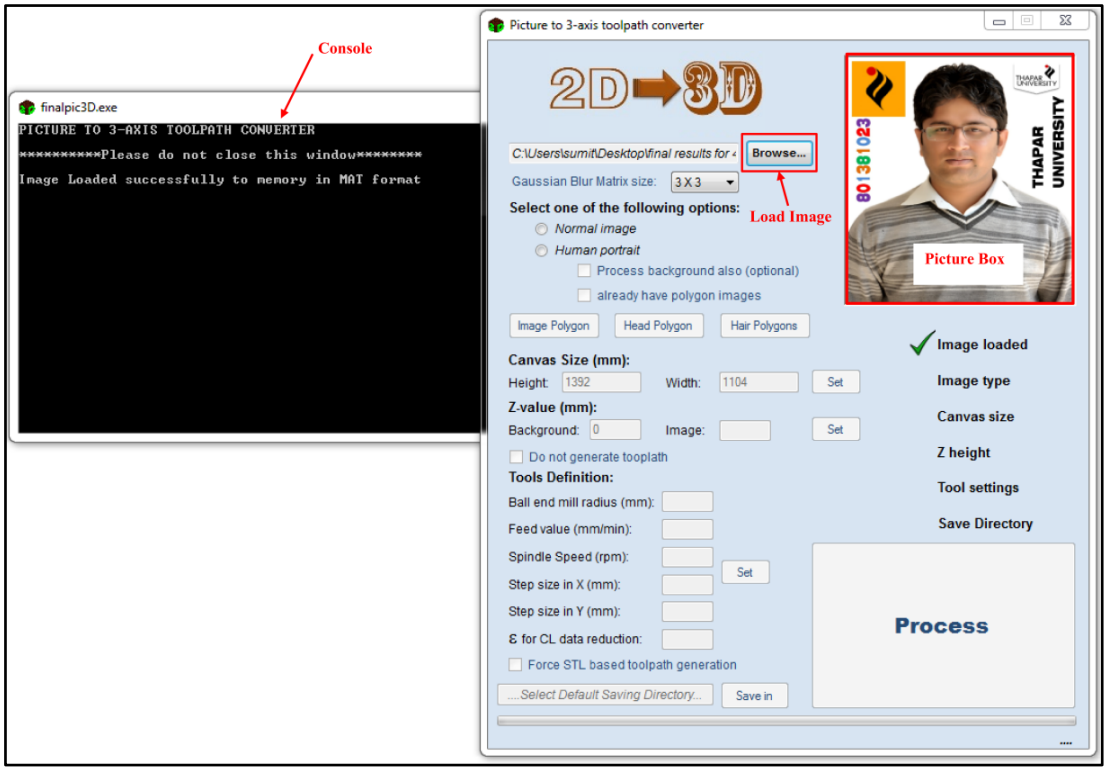


Figure 3.13: Load image and set kernel size

A black console, shown in Fig 3.13, also appears behind the application that displays relevant messages and important information so as to help user understand the process going on.

Step 1: Load the image by clicking on "Browse" button. Different image file formats like jpeg, png, bitmap etc. are supported. Once the image is successfully loaded it is displayed in the picture box as shown in Fig. 3.13. A tick appears against "Image loaded" text. Once the image is loaded select the Gaussian Kernel size from the drop down menu.

Step 2: Select the mode in which image is to be processed. If the normal image mode is selected, go to step 8.

Step 3: When human portrait mode is selected, it enables the options shown in Fig. 3.14 that are not available in normal image mode. User can opt for processing background of the human portrait or to omit it from the final 3D model. Tick the checkbox "Already have polygon images" and go to step 4, if image polygon and hair polygon images are available in form of images. Else, go to step 5.

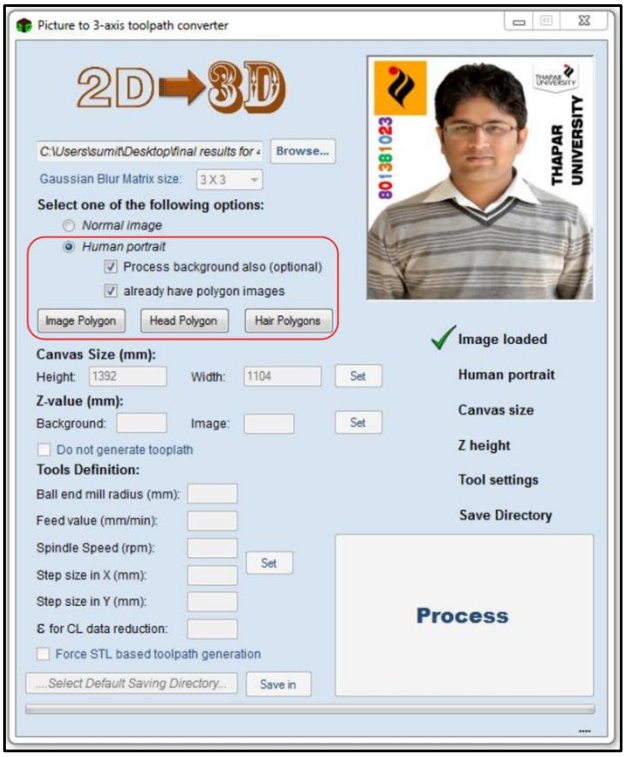


Figure 3.14: Options for human portrait processing

Step 4: Load the required three images one by one using relevant loading buttons and go to step 8. Images must be of same dimensions as that of the original image.

Step 5: Draw the polygon to mark the boundary of the portrait. Use left mouse button to select the corner points of image polygon. Use mouse right click to undo a previously

selected point. Hit "Enter" button on keyboard when finished selecting image polygon. Once the selection is done the selected portion is shown in black color on white background.

Step 6: Select hairs on head using head polygon option. Procedure is same as that in step 5.

Step 7: Select facial hairs like eyebrows, moustache etc. Multiple selection polygons can be used. Use mouse middle click to close current polygon and start another. Press "Enter" button on keyboard to finish the selection.

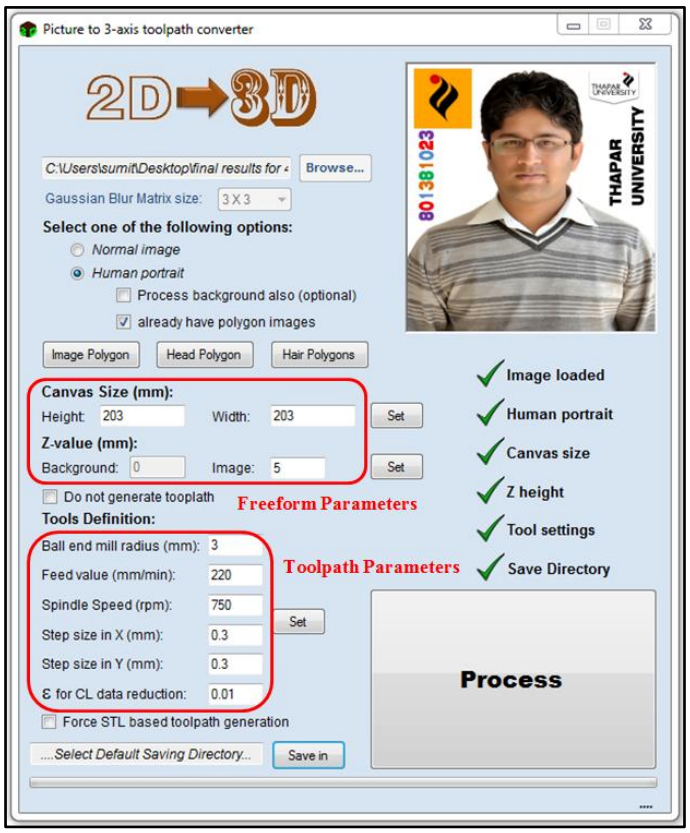


Figure 3.15: User defined parameters

Step 8: Input the size of the plaque on which the image is to be carved. Height is considered as the dimension in x direction and width is considered as the dimension in x direction.

Step 9: Set the maximum z dimension in mm for 3D surface to be generated as Z-value for image. Option to set background height is available only when human portrait mode is selected and "Process background also" checkbox is checked.

Step 10: Select "Do not generate toolpath" to disable toolpath generation and go to step 12.

Step 11: Enter cutter radius, feed value, spindle speed and step size in x and y direction. Select "Force STL based toolpath generation" to force application to use STL file for toolpath generation. ε value for cutter location data reduction algorithm also needs to be entered here.

Step 12: Select the directory to save the output files.

Step 13: Press "Process" button to start the processing. Go to step 14 if the human portrait mode is selected else wait till the application completes the processing.

A message box will appear showing the message "Job done" on the successful execution of all steps.

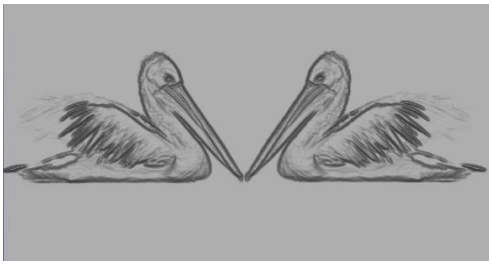
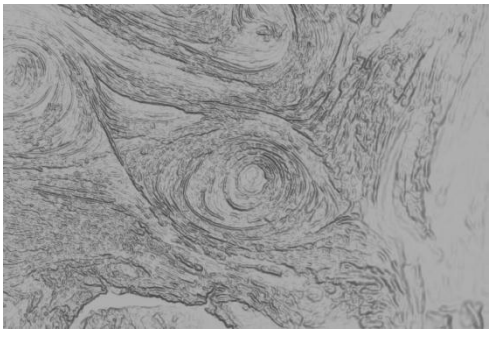
Step 14: Adjust the threshold values using sliders for modifying the intensity of the hair pixels as shown in Fig 3.12. Close the active window and then wait till the application completes the processing. A message box will appear showing the message "Job done" on the successful execution of all steps as shown in Fig.3.16.

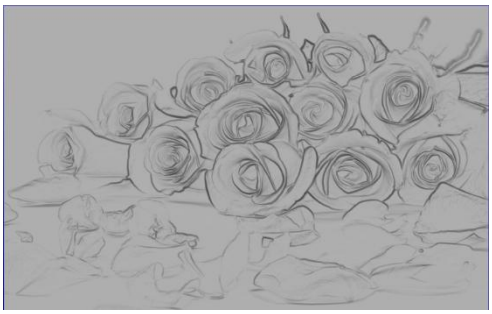
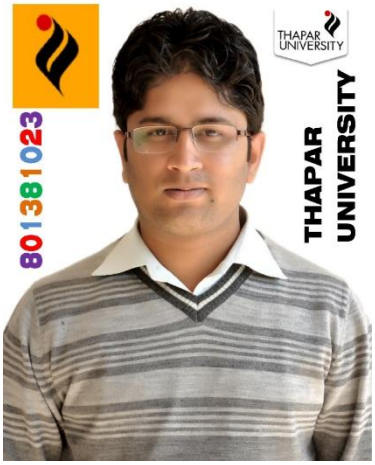
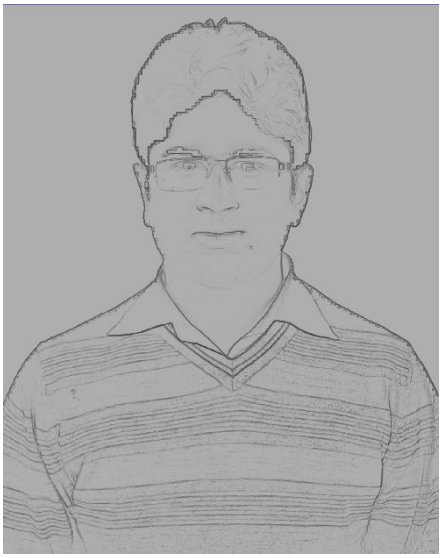
Figure 3.16: Successful execution of the program

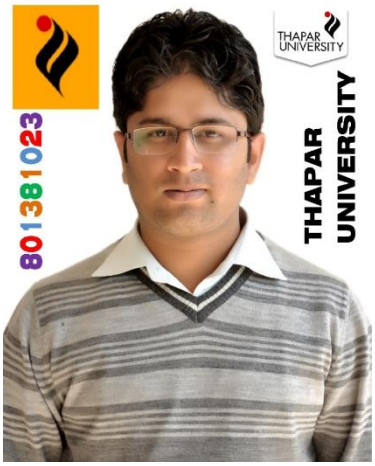
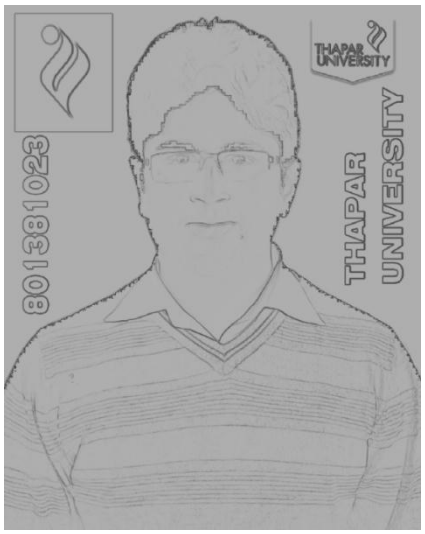
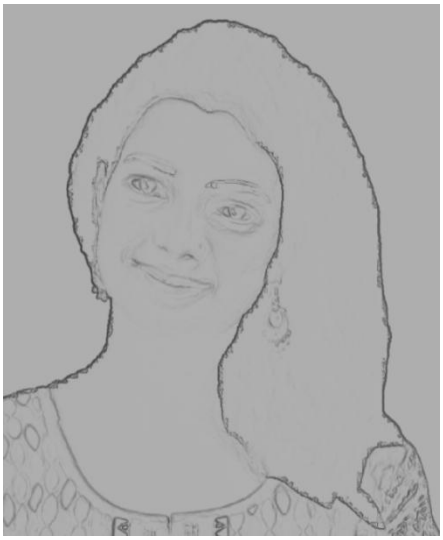
3.4 Results and Discussion

Three dimensional freeform surfaces captured from two dimensional images can be used for toolpath generation for creating 3D photo frames, engraving texture or text and creating beautiful artifacts for decoration purpose. Faceted models can also be used for rapid prototyping. Table 3.1 shows the results obtained in form of STL files by processing various images using developed application. Gaussian kernel size is kept 5×5 and canvas size as 203mm×203mm for all the images.

Table 3.1: Results of digital image to freeform surface conversion

Original Image (Image resolution) pixel×pixel	**Z-value (in mm)**	**Processing mode**	**STL file Snapshot (Maximum Dimensions X×Y×Z mm^3)**
 (618×328)	10	Normal image	 (107.74×203×10)
 (480×480)	5	Normal image	 (203×203×5)
 (900×600)	5	Normal image	 (135.33×203×5)

 (1024×640)	5	Normal image	 (126.88×203×5)
 (640×480)	5	Normal image	 (152.25×203×5)
 (1104×1392)	3	Human Portrait without background	 (203×161×3)

 (1104×1392)	3 for back ground & 5 in total	Human Portrait with background	 (203×161×5)
 (700×855)	5	Human Portrait without background	 (203×166.2×5)

It is clear from Table 3.1 that the developed application can successfully capture freeform surfaces from digital images. These freeform surfaces can be used to carve 3D impression on any plaque to create a 3D artifact for that image. As mentioned earlier, toolpath generation feature has been integrated into the application. Another cutter location data reduction algorithm is also integrated. Next chapter is dedicated for the discussion of this toolpath generation algorithm and its results.

Chapter 4
METHODOLOGY FOR EFFICIENT TOOLPATH GENERATION

Toolpath planning is very crucial for achieving desired results in CNC machining of freeform surfaces. In industries ball end mill is mostly used for the machining of complex freeform surfaces. For present work Cartesian toolpath method with zigzag tool footprint is implemented for generating toolpaths for ball nosed end mill cutter.

4.1 Cutter Location Method Used

"Ball drop" method by Manos et al. [2] is one of the robust methods for calculating the tool locations that ensures gauge free toolpaths for STL surfaces. This method was initially developed for Single Controlled Axis Lathe Mill (SCALM) that uses helical toolpaths. Algorithm for "Ball drop" method for 3-axis machining had been successfully implemented for 3-axis toolpath generation [18,34,35]. This method is also extended to other tool shapes and 5-axis machining [19,38]. To find cutter location point while using ball nosed end mill, a sphere of radius equals to the radius of ball end mill is dropped along the tool axis at tool location and its first contact with the surface is recorded. Based on this first contact with surface, tool's centre height is calculated to position the tool at that particular tool location. Similarly tool height for each tool location is calculated. For STL models a tool can contact the surface in three ways as shown in Fig. 4.1.

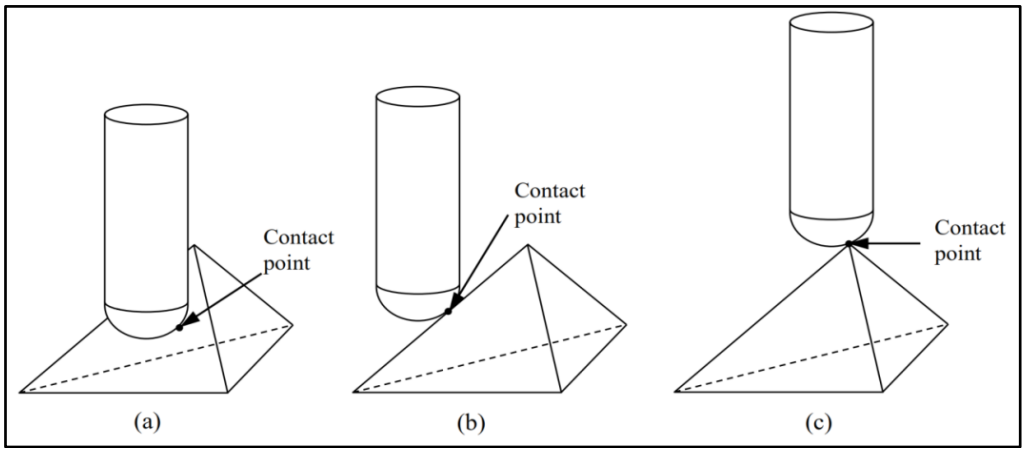


Figure 4.1: First contact with STL surface: (a) with face, (b) with edge and (c) with vertex

Tool can touch the face of any of the triangular facets of surface, it can touch any edge shared by two triangles or it may touch any vertex shared by two or more triangles. The mathematical formulation and algorithm for finding tool's centre height for 3-axis machining using "Ball Drop" is already well developed. It is made available at SIDC woodworking lab, by Mr. R. K. Duvedi, Assistant Professor, Mechanical Engineering department, Thapar

University, Patiala. Dissertation work of Jasra [34] and Rajiv [35] can be referred for its implementation.

However, a new shadow check method is implemented for generating toolpath for 3D plaques using STL as input. In this method whole STL file is not considered at once. Instead of using STL file as an input, the point cloud file, from which the STL file is generated, is used. Positions of the points are stored in a 2D array in the computer memory. Points lying under the shadow of the tool are identified using this array and the triangles are then generated using these identified points. These triangles are the suitable candidates for ball drop check for that position. Because of this, toolpath computation time decreases considerably since at every point we do not have to check all the triangles to identify the suitable candidates but instead, the concerned triangles are generated and checked on demand. Once the triangles under the tools are identified, checks are performed to compute tool centre height.

4.2 Toolpath Generation Method for 3D Freeform Surfaces

Figure 4.2 shows the algorithm for toolpath generation which has been implemented in the present work. The application can smartly switch between point cloud and STL file to generate toolpath based on the quality of the input available. Point cloud is preferred to generate toolpath if it's dense enough such that the distance between adjacent points in x and y direction is not more than 20% of cutter radius, although, user can override this to choose toolpath generation using STL model in any case. It must be noted that the present implementation is a special case as the space between neighboring points is constant along x as well as in y direction.

This application compares the dimensions of the surface in x and y direction. Direction with larger dimension is chosen as feed forward direction. Tool positioning in the case of point cloud can be considered as a special case of "Ball drop" method where the tool only touches the vertices of the triangles. Since, there is no need of the triangle check and the edge check, the toolpath generation using point cloud is much faster. Toolpath generated using this methodology is very rich in number of tool location points. With decrease in either of the step value, feed forward step value or both, the number of cutter location points increases and vice versa.

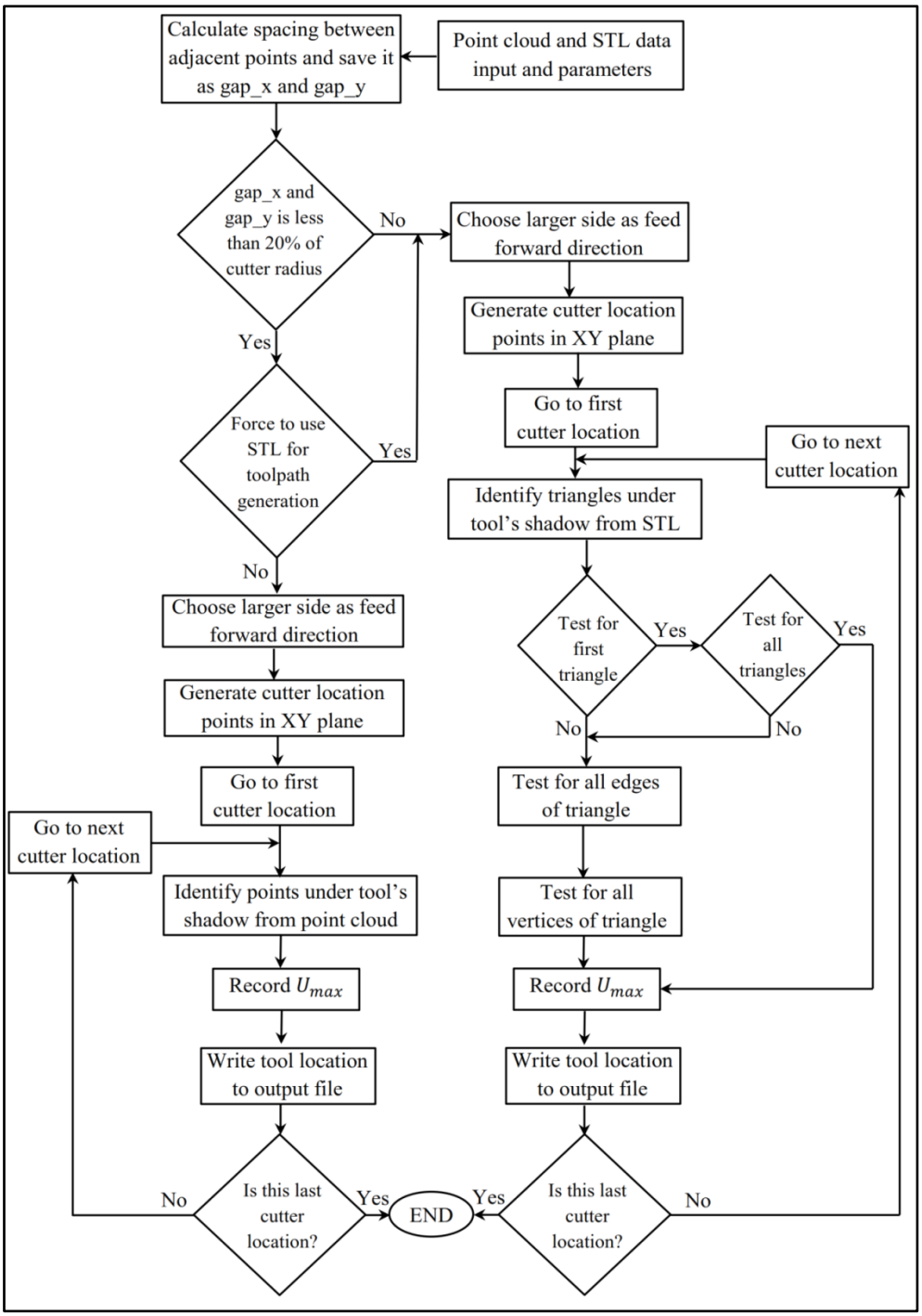


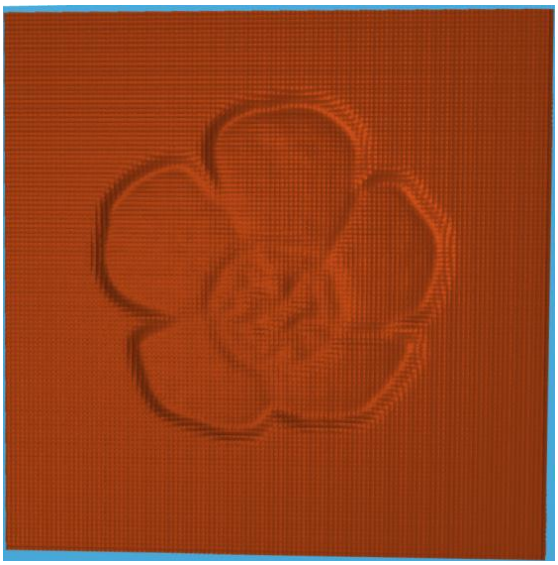
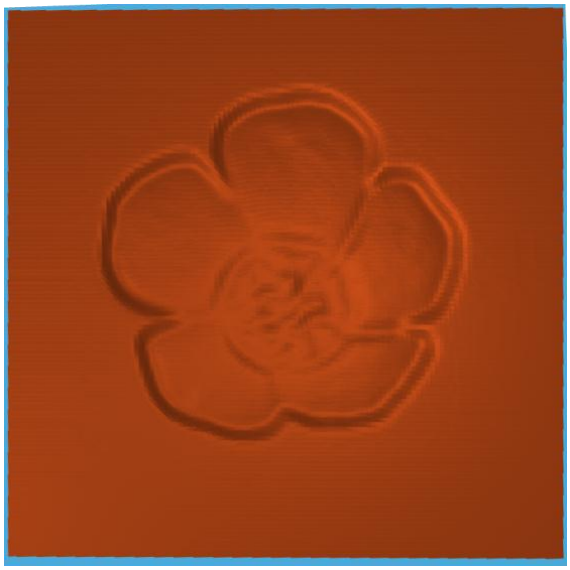
Figure 4.2: Toolpath generation algorithm for the application

4.3 Point Cloud versus STL for Toolpath Generation

Toolpath generation using point cloud data is much faster as compared to toolpath generation using STL file. Point cloud data can be used effectively to generate toolpath only if the point cloud is dense enough so that tool does not dip in the space between the points. Table 4.1 shows the comparison of the two different file inputs for the toolpath generation of the same image. In both the cases, canvas size is 203mm×203mm, tool radius is 3mm and side step and feed forward step size is 0.3mm. It can be clearly seen from the simulation results that in the

case of point cloud data the tool moves in the space between the points, thus creating the grainy pattern which does not exist in the case of toolpath generated from STL file.

Table 4.1: Point cloud data versus STL

Point cloud input	STL input
Number of Points in point cloud data is 14161	Number of triangles in STL file is 27848
Space between adjacent points is 1.72 mm in x and 1.72 mm in y	-
Toolpath generation time is 43 seconds	Toolpath generation time is 3 min 16 sec
	

4.4 Cutter Location Data Reduction Algorithm

The length of the toolpath depends on the dimensions of the workpiece to be machined, radius of the tool, side step, feed forward step and the number of tool passes. For a plaque of 203mm length and 203mm width to be machined by the tool of 3mm radius, 0.3mm side step and 0.3mm feed forward step, there will be more than 400000 tool location points through which the tool have to move. This results in an increased machining time since the machine has to interpolate through large number of points. The size of toolpath files thus obtained are also large. These files are needed to be copied to CNC controller's internal memory before the execution. With advancement in technology and reduction in digital memory cost, new CNC machines are equipped with large internal memories and for them large toolpath files is not a problem. But, for older machines the toolpath needs to be as small as possible.

Therefore, an algorithm is proposed to reduce the toolpath file size as well as to increase the machining efficiency, by eliminating the unnecessary points. This algorithm takes advantage of the fact that in most of the cases of sculptured machining, high accuracy is not desired. The point can be omitted from the toolpath for which the z height is very close to that of its adjacent points. Figure 4.3 shows how such points are eliminated.

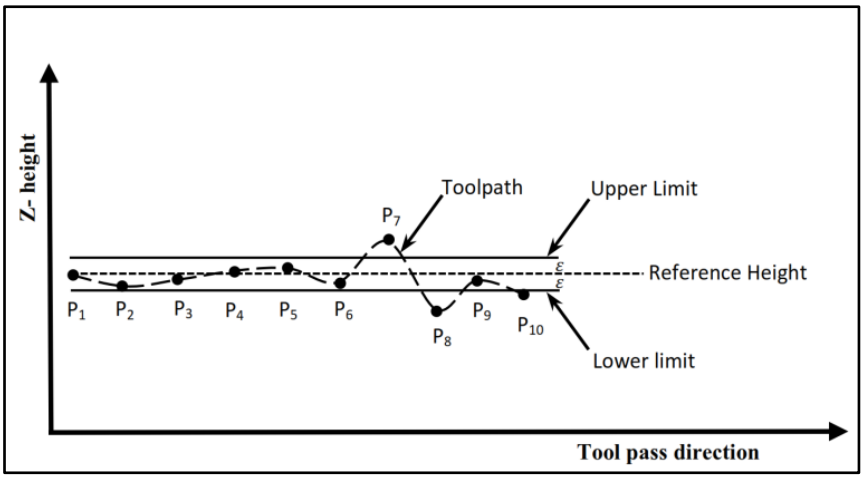


Figure 4.3: Cutter location points

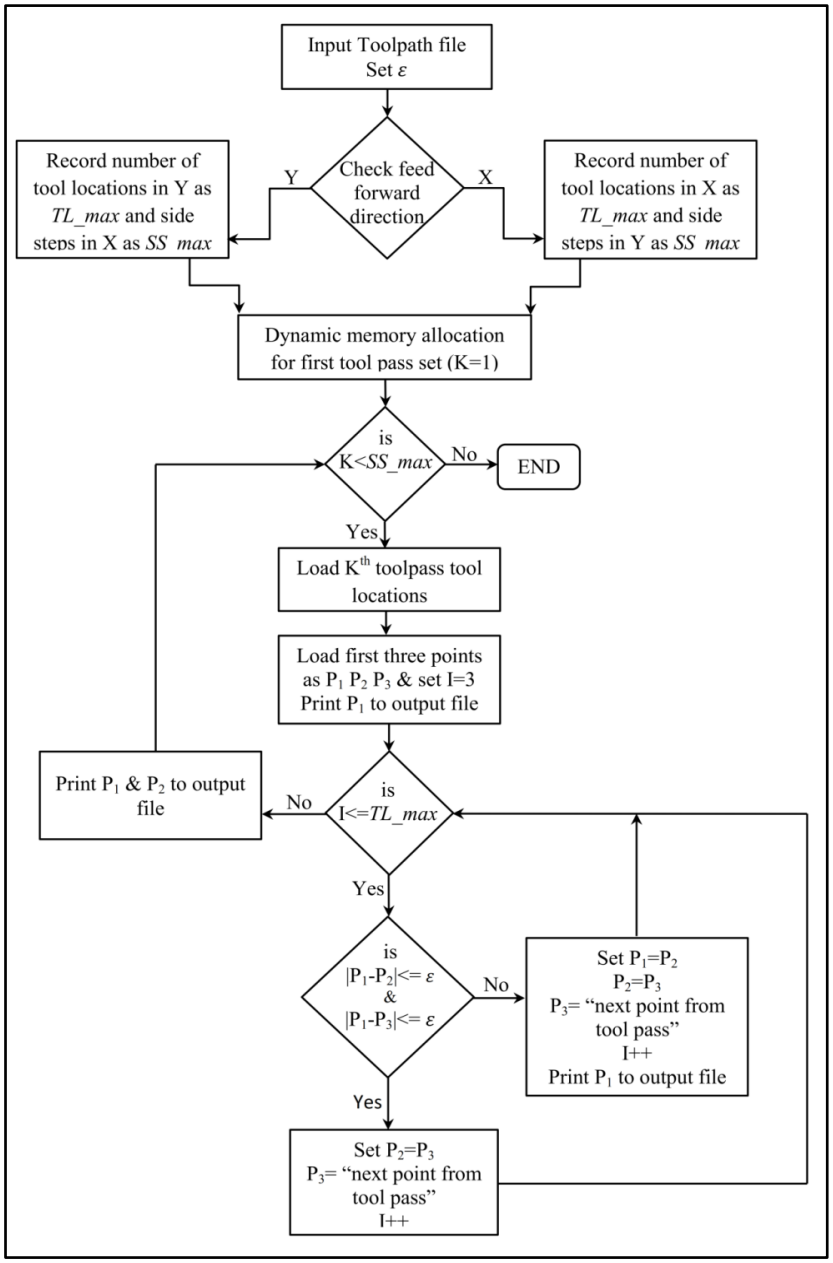


Figure 4.4: Cutter location data reduction algorithm

Here ε is the tolerance value that is specified by the user depending upon the level of accuracy required. Height of first point $P_1$is taken as reference. Height of next point P_2 is compared to that of P_1. As the difference between the two heights is within tolerance, P_2 can be eliminated. But, elimination of P_2 will depend on P_3. Difference of the height between P_1 and P_3 is calculated. As this difference is also within tolerance limit, therefore, the machine can be moved directly from$P_1$to P_3. Hence, there is no need of point P_2. Similarly, as P_4, P_5 and P_6 lie in the same tolerance band when compared to P_1, points P_3, P_4, P_5 will be eliminated. Figure 4.4 shows the algorithm for eliminating undesirable tool location points for raster toolpaths. These algorithms have also been incorporated in the main application for generating efficient toolpaths.

4.5 Results and Discussion

To validate the effectiveness of the developed cutter location data reduction algorithm, the image shown in Fig. 4.5 is used to generate the toolpath.

Figure 4.5: Test image for points elimination algorithm

Table 4.2 shows the result for different values of ε. It can be clearly seen that this methodology can significantly decrease the total number of tool location points so as to achieve higher feed rates while machining.

Table 4.2: Validation of unsought cutter location points elimination algorithm

Value of ε	0.0 mm	0.02 mm
Tool location points in original toolpath	456976	456976
Tool location points in new toolpath	76366	67705
No. of points eliminated	380610	389271
Processing time (for cutter location data reduction)	19.906 sec	18.938 sec
Part size	203×203×5 mm^3	203×203×5mm^3

Figure 4.6 shows the simulation results for original toolpath and two other toolpaths generated by the present methodology for different values of ε. Simulation results shows that there is no change visually in the machined parts but the number of tool location points are significantly decreased. Some minor variations can be observed in the three images shown in

the Fig. 4.6. This is because of the slight differences in the orientation and the lighting while taking the snapshots from the simulator.

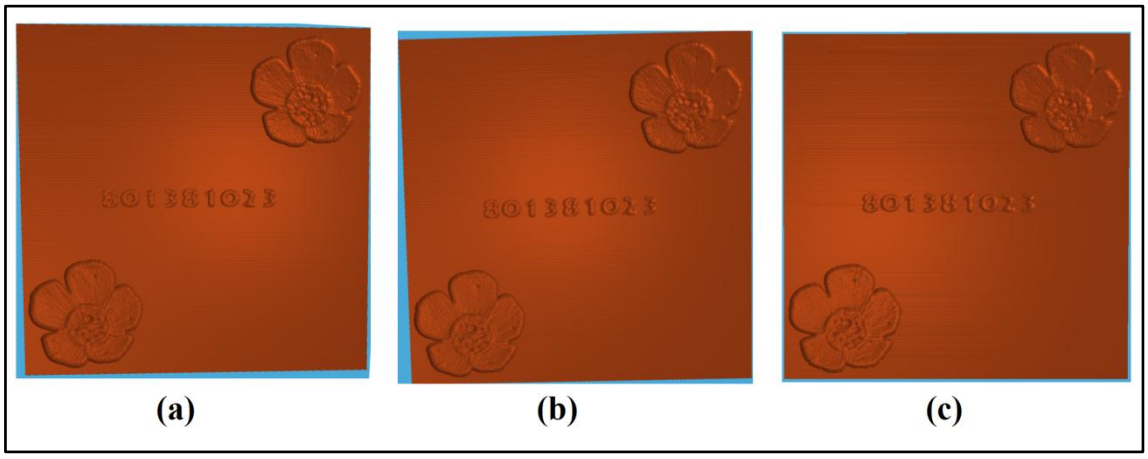


Figure 4.6 Simulation results for point elimination algorithm: (a) original toolpath, (b) toolpath with ε=0.0 and (c) toolpath withε=0.02

Table 4.2 shows the simulation results of the toolpath generated using STL file data for two different images. Ball nosed end mill radius is taken as 3mm, side step and feed forward step size is taken as 0.3mm each. Simulator results are shown in Fig. 4.7.

Table 4.3: Toolpath simulation results (using STL)

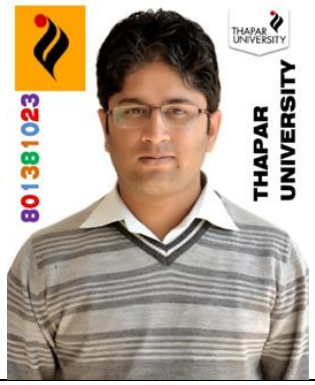
Input image		
Image size	480×480 pixels	1104×1392 pixels
Image processing mode	Normal Image	Human Portrait with background
No. of point in point cloud	230400	1536768
No of triangles in STL	458882	3068546
Toolpath generated using	STL	STL
Initial number of Cutter location (CL) points	456976	363549
ε value	0.01mm	0.01mm
CL points after reduction	429887	243295
No. of CL points omitted	27089	120254
Toolpath generation time	13 min 7 sec	66 min 38 sec
Total Processing Time	16 min 14 sec	72 min 58 sec

Figure 4.7: Machining simulation results (STL)

It can be observed from table 4.3 that for highly sculptured part there is less reduction in the number of tool location points.

Simulation results for the toolpaths generated using the point cloud data for various images are shown in table 4.4.

Table 4.4: Toolpath simulation results (using point cloud)

Image size (in pixels)	297×210
Image processing mode	Normal Image
No of points in point cloud	202704
No of triangles in STL	403518
Toolpath generated using	Point cloud
Initial no. of CL points	242684
ε value	0.01
CL points after reduction	77240
No. of CL points omitted	165444
Toolpath generation time	52 sec
Total Processing Time	2 min 56 sec

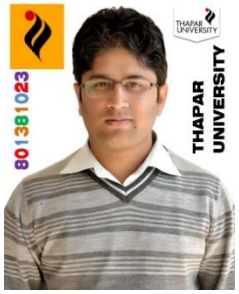


Image size (in pixels)	1104×1392
Image processing mode	Human portrait without background
No of points in point cloud	1536768
No of triangles in STL	3068546
Toolpath generated using	Point cloud
Initial no. of CL points	363549
ε value	0.01mm
CL points after reduction	211636
No. of CL points omitted	151913
Toolpath generation time	3 min 34 sec
Total Processing Time	10 min 45 sec

Image size	1024×640 pixels
Image processing mode	Normal Image
No of points in point cloud	655360
No of triangles in STL	1307394
Toolpath generated using	Point cloud
Initial no. of CL points	286371
ε value	0.01mm
CL points after reduction	192104
No. of CL points omitted	94267
Toolpath generation time	1 min 51 sec
Total Processing Time	5 min 32 sec

Results shows that toolpath generation using STL file is bit slower that the toolpath generation using point cloud data. Moreover, Cutter location data reduction algorithm works well for different types of sculptured surfaces toolpaths.

Chapter 5
BIDIRECTIONAL CONTROLLED SCALLOP HEIGHT TOOLPATH

Iso-scallop toolpath method is an improvement over iso-planer and iso-parametric methods, since it ensures the constant scallop height throughout the toolpath. This also optimises the length of the toolpath as maximum side step is taken between adjacent tool passes while ensuring the same scallop height throughout the toolpath [35]. One such iso-scallop height based toolpath generation method is proposed by authors [35] for smooth surfaces, approximated into STL models, which use bisection method to iterate for the best side step value between two adjacent tool locations in side step direction to maintain the constant scallop height. This method generates raster toolpaths and uses "Ball drop" method for positioning the tool at different cutter locations. Although, this method efficiently generates the toolpath, but this method adjusts the toolpath in one direction only i.e. in side step direction. In present chapter the modification of this method is proposed which performs bidirectional adjustment of the toolpath.

5.1 Method Used to Determine Scallop Height

Scallop height is defined as the maximum height of the uncut material left between two adjacent tool feed forward directions measured generally along the surface normal. In present work scallop height is measured using method proposed by authors [35]. In this method the scallop height is calculated along the ray projected through the point M which is the centre of the line joining the cutter location points $C_{L11}$and C_{L21}(which are S_s distance apart) and the point I_c which is generated by the intersection of the two spheres, representing the tool at two adjacent locations, projected on the plane perpendicular to feed forward direction and passing through $C_{L11}$and C_{L21} as shown in Fig. 5.1. The ray hits the surface at I_{stl}. The distance between I_c and I_{stl} gives the value of scallop height, when measured along the projected ray. I_c is calculated as:

$$I_c = M + \hat{w}r \tag{5.1}$$

Here, r is the radius of the circle of intersection between two spheres with their centres S_s apart as shown in Fig. 5.2. It is obtained using relation:

$$r = \sqrt{R^2 - l^2} \tag{5.2}$$

Where l is calculated as:

$$l = |C_{L11} - M| \tag{5.3}$$

$\hat{w}$ is the unit vector along the projected ray and is calculated usingfollowing relations:

$$\hat{u} = \frac{M - C_{L11}}{|M - C_{L11}|} \tag{5.4}$$

$$\hat{v} = \frac{\hat{u} \times \hat{t}}{|\hat{u} \times \hat{t}|} \tag{5.5}$$

$$\hat{w} = \frac{\hat{u} \times \hat{v}}{|\hat{u} \times \hat{v}|} \tag{5.6}$$

Where,

$$\hat{t} = \{0,0,1\}$$

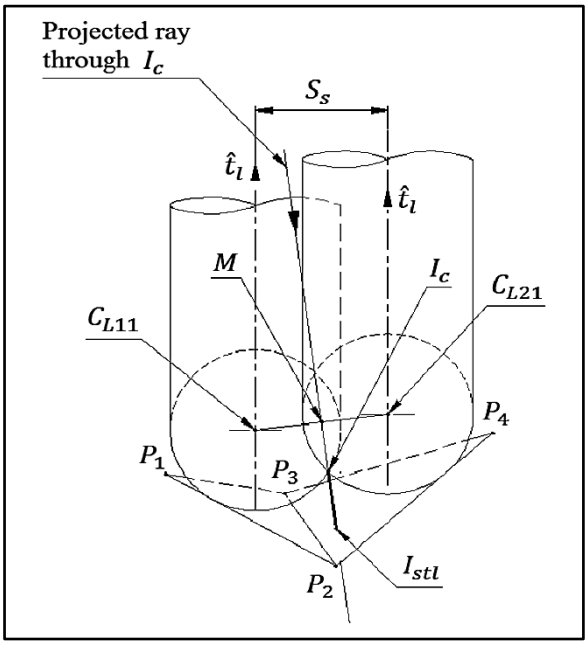


Figure 5.1: Scallop height calculation [35]

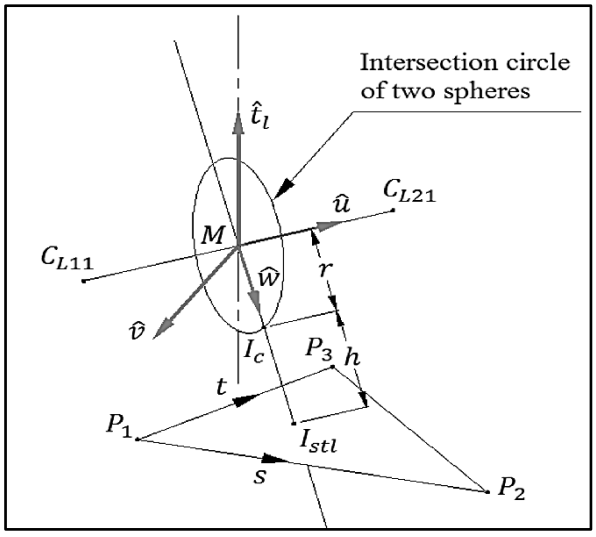


Figure 5.2: Intersection circle of two spheres [35]

From Fig. 5.2 it can be noticed that the scallop height is h. To compute h first I_{stl} needs to be computed, which is the point at which the ray from I_c along unit vector $\hat{w}$ intersects with the triangle $P_1P_2P_3$. In parametric form I_{stl} can be expressed as:

$$I_{stl} = P_1 + s(P_2 - P_1) + t(P_3 - P_1) \tag{5.7}$$

Where

$$0 \leq (s, t, s + t) \leq 1$$

Also,

$$I_c + h\widehat{w} = I_{stl} \tag{5.8}$$

Equating equations (5.7) and (5.8) gives:

$$I_c + h\widehat{w} = P_1 + s(P_2 - P_1) + t(P_3 - P_1) \tag{5.9}$$

$$\Rightarrow h\widehat{w} + s(P_1 - P_2) + t(P_1 - P_3) = P_1 - I_c$$

or

$$h\widehat{w} + sA + tB = C \tag{5.10}$$

where,

$$A = (P_1 - P_2), B = (P_1 - P_3), C = (P_1 - I_c)$$

Equation (5.10) can be written in matrix form in terms of scalar components as:

$$\begin{bmatrix} \widehat{w}_x & A_x & B_x \\ \widehat{w}_y & A_y & B_y \\ \widehat{w}_z & A_z & B_z \end{bmatrix} \begin{bmatrix} h \\ s \\ t \end{bmatrix} = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{5.11}$$

or

$$\begin{bmatrix} h \\ s \\ t \end{bmatrix} = \begin{bmatrix} \widehat{w}_x & A_x & B_x \\ \widehat{w}_y & A_y & B_y \\ \widehat{w}_z & A_z & B_z \end{bmatrix}^{-1} \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{5.12}$$

If the ray intersects with the triangle, the values of s and t must satisfy the condition that

$$0 \leq (s, t, s + t) \leq 1$$

If this condition is satisfied the value of h is considered as a solution. All the triangles that can intersect with ray are identified and then for each triangle h is computed. The largest value of h is the value of maximum scallop height.

5.2 Bisection Method for Side Step Adjustment

To optimize the side step in order to achieve the user defined scallop height value ε, bisection method (interval halving method) is used by authors [35]. We start with initial side step value S_{si} and iterate between limits S_{sl} and S_{sr} to find the side step value which gives scallop height value equal to ε as shown in Fig. 5.3. If there exist a function f which is continuous within interval $[S_{sl}, S_{sr}]$ provided that $f(S_{sl})$ and $f(S_{sr})$ are of opposite sign, then, there exist $S_{si} \in (S_{sl}, S_{sr})$ for which $f(S_{si}) = 0$. Starting with $S_{si} = \frac{S_{sl}+S_{sr}}{2}$, interval is reduced at every iteration to find the solution by comparing the function values until the interval reduces to almost zero. It is assumed that only one such solution exists within the considered interval.

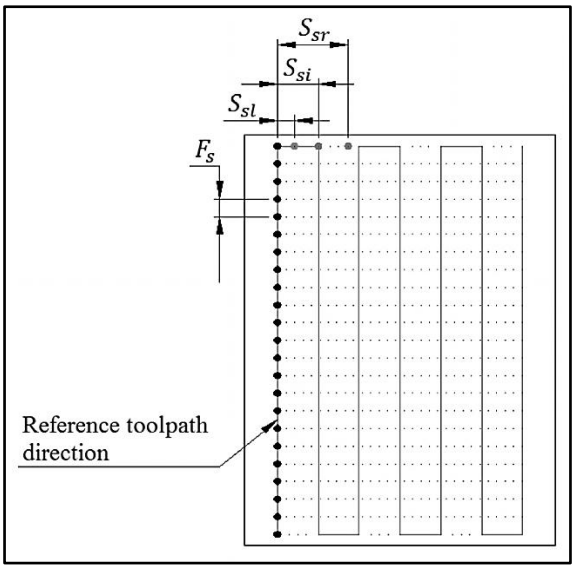


Figure 5.3: Side step adjustment using bisection method [35]

5.3 Bidirectional Controlled Scallop Height Toolpath

Raster path topology has been used for generating toolpaths. Method proposed by authors [35] only optimizes side steps for controlling scallop height. In proposed method, side steps and feed forward steps both are optimized. This whole process can be segmented into two parts. In first part, author's [35] method is implemented to optimize the steps in first direction. For this, first of all initial feed forward direction is selected. In final toolpath this direction will be the side step direction. After deciding the initial feed forward direction, first line of toolpath is generated using constant feed forward step value. This line is then used as reference for the next line.

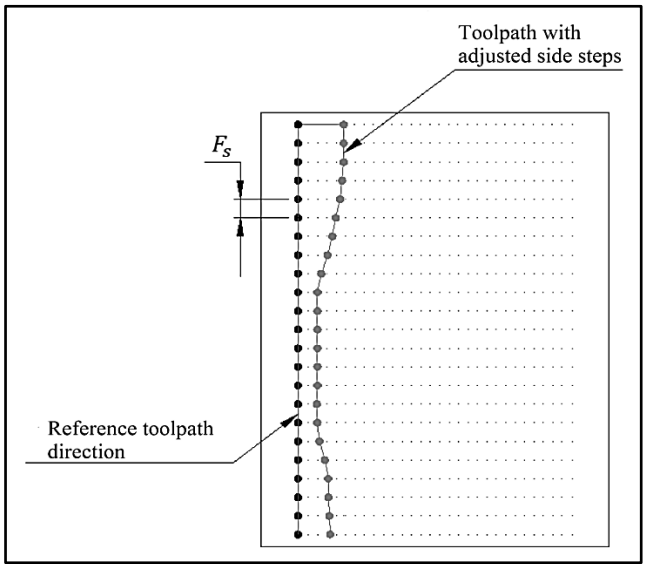


Figure 5.4: Side step adjustment for constant scallop height [35]

Next line is generated using by adjusting the side steps for maintaining the constant scallop height between two passes as shown in Fig. 5.4. Feed forward step value remains same.

Minimum side step value from this line is stored in a file and then selected to generate straight toolpath line parallel to the previous one. This new toolpath line is then used as reference for the next one. This process goes on till the last toolpath line is generated. Finally the toolpath with constant feed forward steps and varying side steps is achieved. At this point only optimised side step values are required. Therefore, the toolpath generated in first part is discarded. This finishes the first part of the algorithm.

Bidirectional scallop height controlled toolpath is generated in second part of the algorithm. Directions are now switched, i.e. the direction which is previously considered as side step direction is now taken as the new feed forward direction. Instead of using constant feed forward steps, previously stored side step values are used for positioning the tool in feed forward direction. The first line of the toolpath is generated using new feed forward step values and by taking this toolpath line as reference, next line is generated while computing new side step values for maintaining the constant scallop height between two passes. Minimum side step value from this line is selected to generate straight toolpath line parallel to the previous one. This new toolpath line is then used as reference for the next one. This process goes on till the last toolpath line is generated. Finally, this results in toolpath with optimised feed forward as well as side steps as shown in Fig. 5.5. This whole process can be represented in the form of a flowchart as shown in Fig. 5.6.

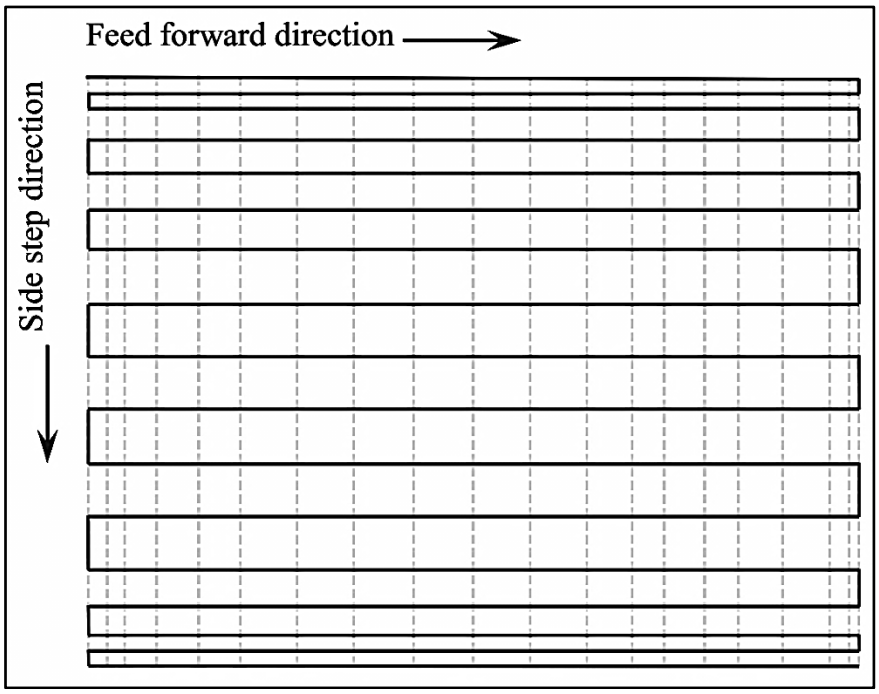


Figure 5.5: Toolpath with varying feed forward steps and side steps

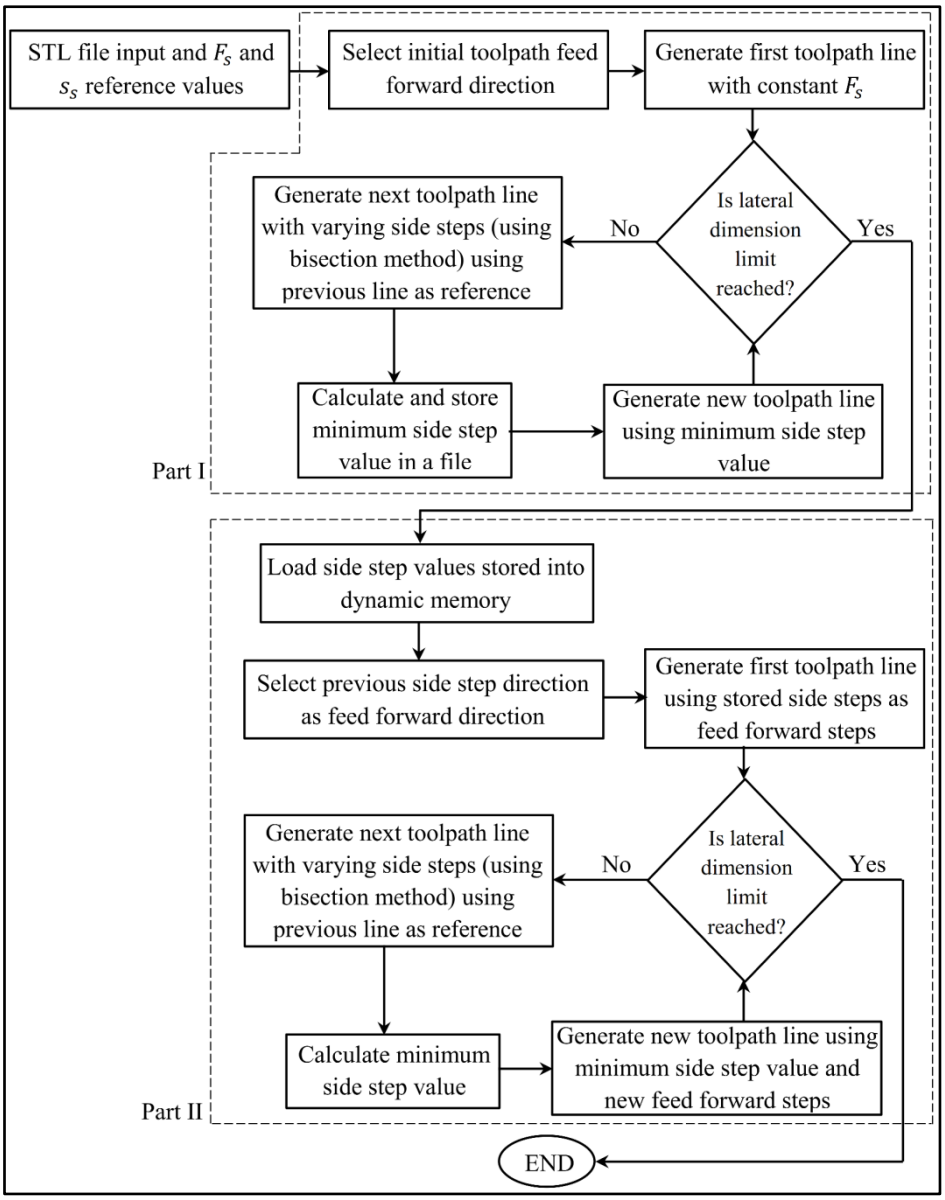


Figure 5.6: Toolpath with bidirectional scallop height control

5.4 Results and Discussion

Maintaining constant scallop height throughout the toolpath is not a trivial task. Hence, toolpath with minimum side step and feed forward steps values is generated which keeps the scallop height smaller than the user specified value ϵ throughout the surface. Number of tool passes depends momentously on ε value. If the value of ε is decreased, the number of tool passes increases and vice versa. Minimum scallop height specified by the user cannot be less than the surface tolerance which is used to generate the input STL surface [35]. Results of the developed approach for various input surfaces are discussed in the succeeding sections.

5.4.1 Bezier test part with Δu and Δv 0.05

Test part considered for the analysis of the variation of scallop height in present work is shown in Fig. 5.7. The dimensions of the part are 50×100×25mm containing 886 triangles.

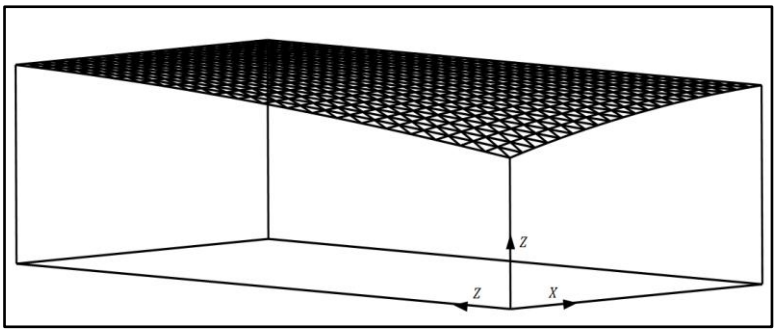


Figure 5.7: Bezier test part with Δu and Δv 0.05

Control points for this Bezier test surface are shown in table 5.1. Δu and Δv are taken as 0.05mm.

Table 5.1: Control points for Bezier test part

Points	P_{00}	P_{01}	P_{02}	P_{10}	P_{11}	P_{12}	P_{20}	P_{21}	P_{22}
x-coordinates	0.0	0.0	0.0	25.0	25.0	25.0	50.0	50.0	50.0
y-coordinates	0.0	50.0	100.0	0.0	48.0	100.0	0.0	50.0	100.0
z-coordinates	19.0	24.0	25.0	24.0	24.5	25.0	25.0	25.0	25.0

For the toolpath generation, x direction is taken as initial and y direction as final feed forward direction. Figure 5.8 shows the x-y plot for the initial toolpath generated in x direction for calculating the final feed forward steps for ϵ equal to 0.08mm.

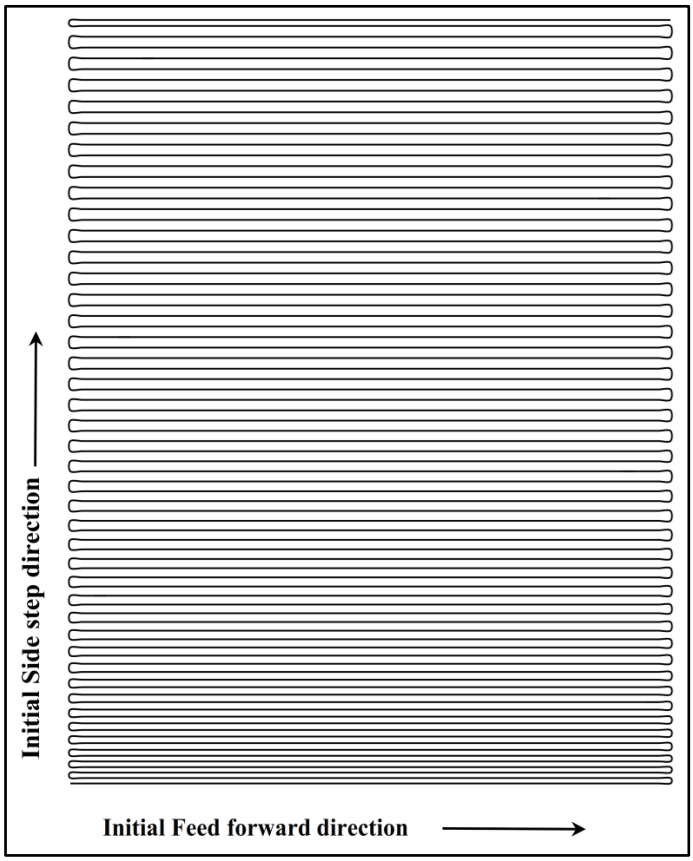


Figure 5.8: Initial toolpath for Bezier test part

x-y plot for the final toolpath generated using developed approach is shown in Fig. 5.9.

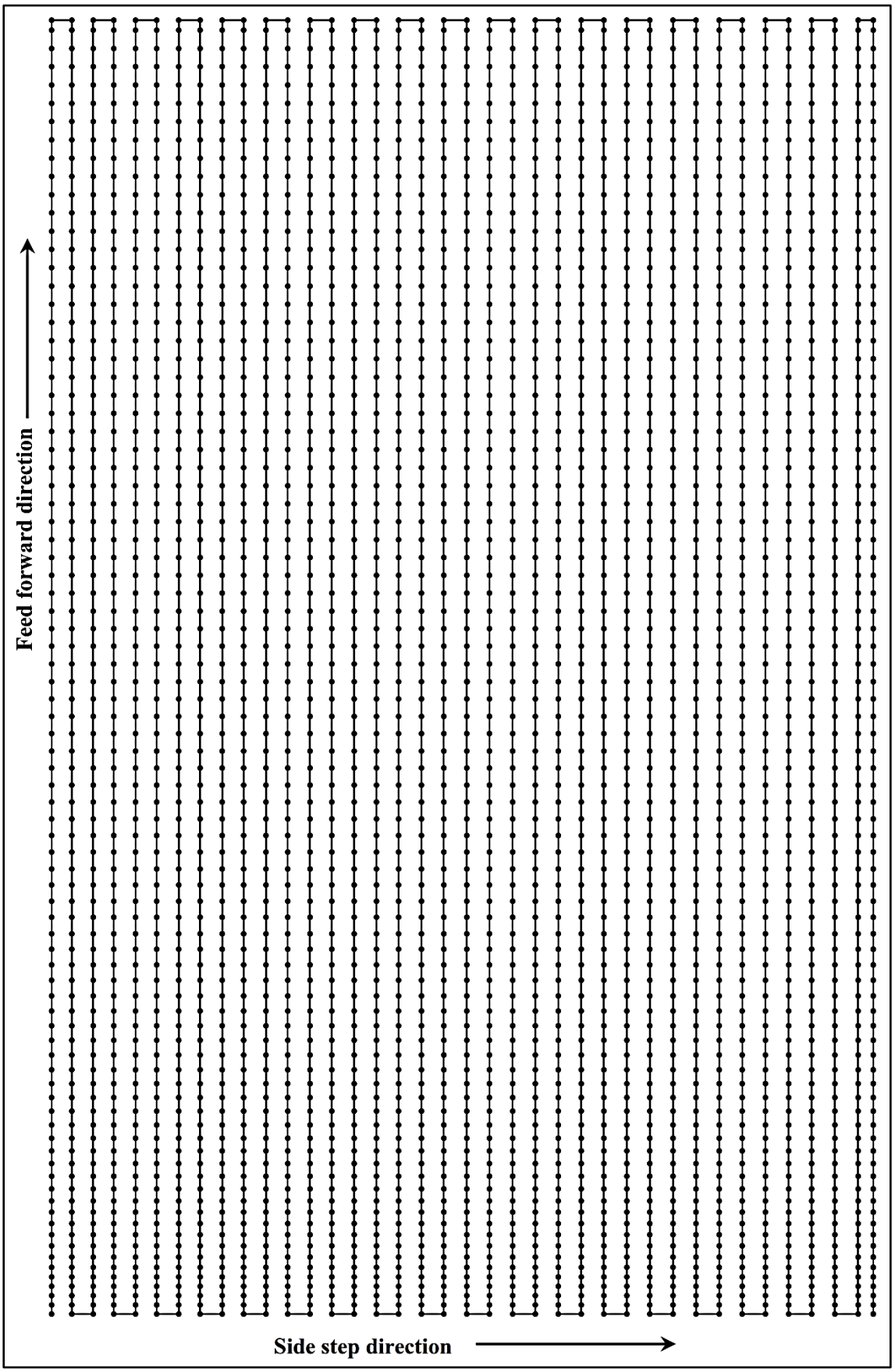


Figure 5.9: Final toolpath for Bezier test part, ϵ =0.08 mm

Figure 5.10 shows the simulator results generated by using “Toolsim” simulator.

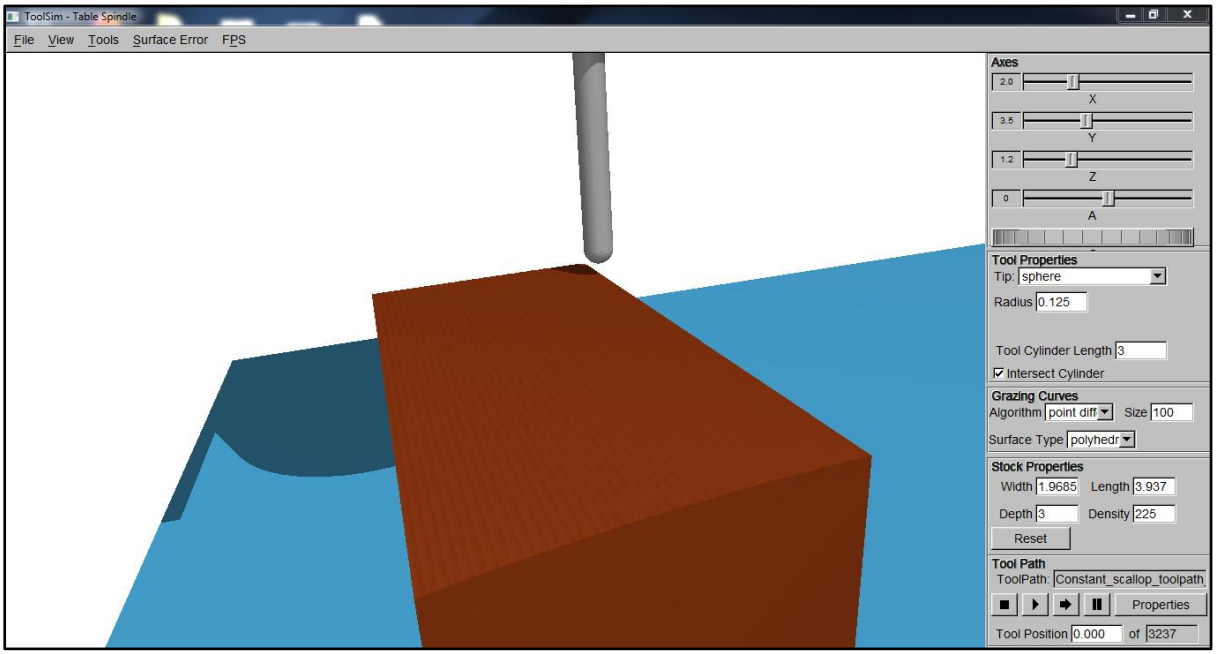


Figure 5.10: Simulation result for Bezier test part, ϵ =0.08 mm

5.4.2 Controlled scallop height plots for test part

To capture the scallop height Z-buffer method is implemented. Obj file is generated by the simulator for the virtually machined part which is then compared to the STL model to compute the scallop height at different places. Figure 5.11, 5.12 and 5.13 shows the plots of scallop height for the bidirectional scallop height control. For constant y value, scallop height is plotted against x for ϵ equal to 0.08mm.

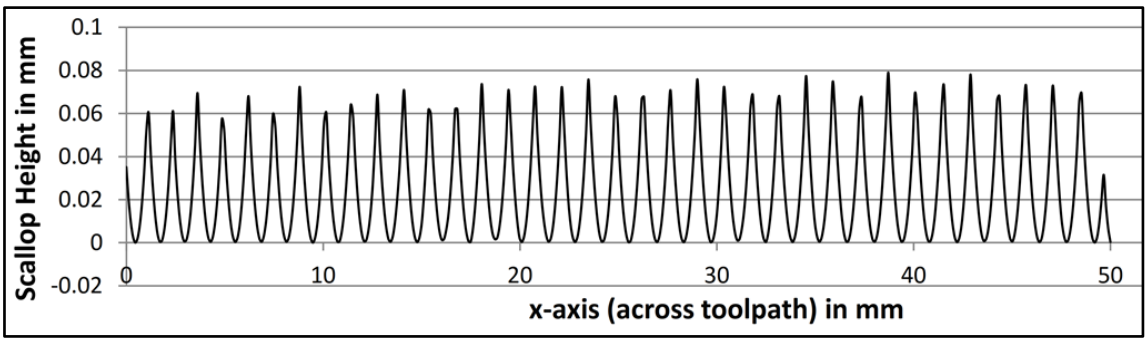


Figure 5.11: Controlled scallop height plot at y=12.3025mm for ϵ =0.08mm

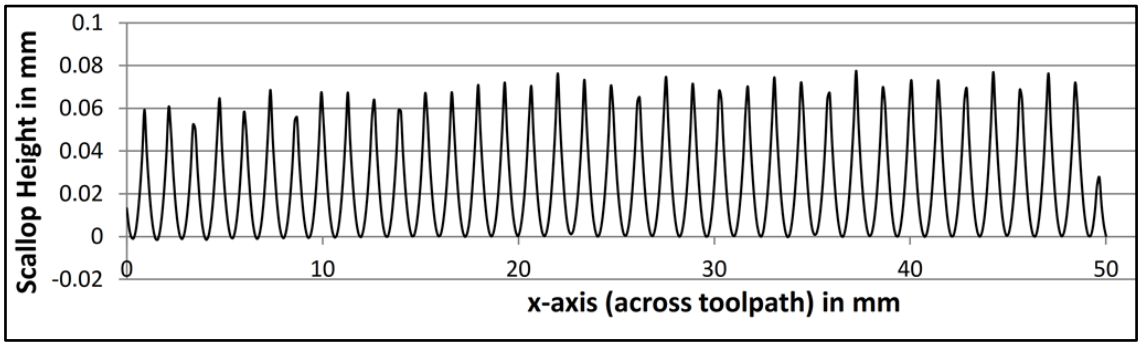


Figure 5.12: Controlled scallop height plot at y= 34.8758mm for ϵ =0.08mm

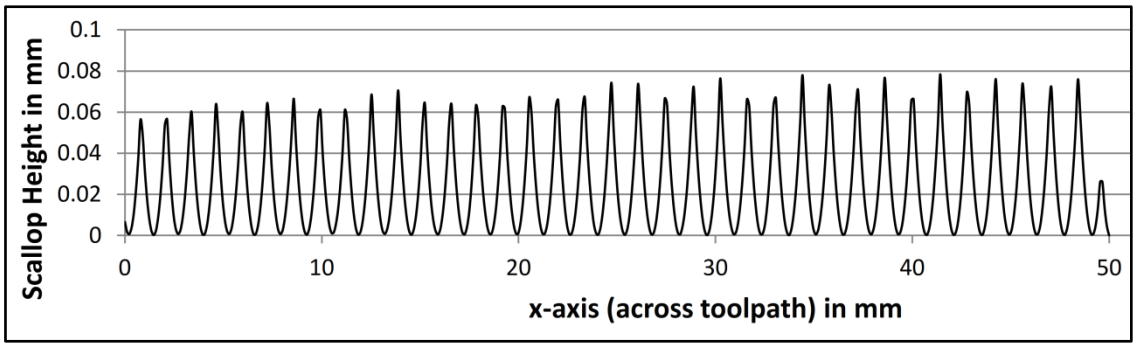


Figure 5.13: Controlled scallop height plot at y= 46.1624mm for ϵ =0.08mm

Figure 5.14, 5.15 and 5.16 shows the plots of the scallop height for the bidirectional scallop height control. For constant y value, scallop height is plotted against x for ϵ equal to 0.125mm. It can be seen from the plots that the scallop height is less that the specified value.

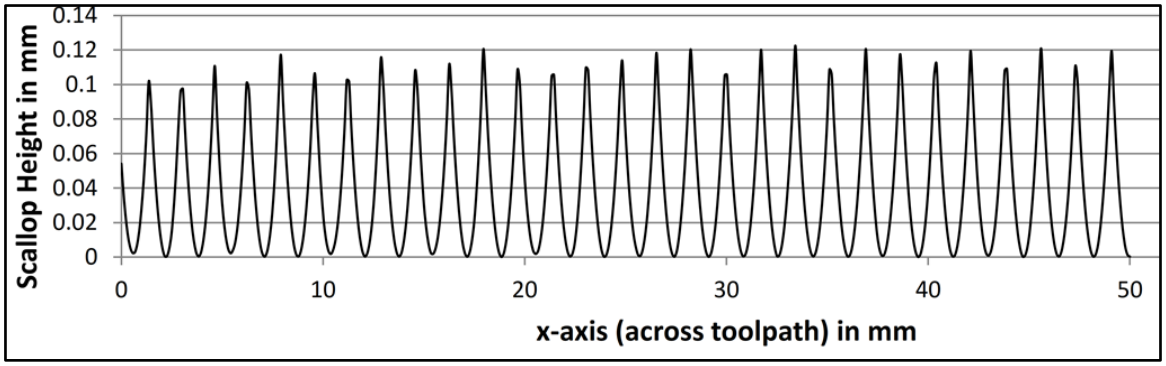


Figure 5.14: Controlled scallop height plot at y= 1.0158mm for ϵ =0.125mm

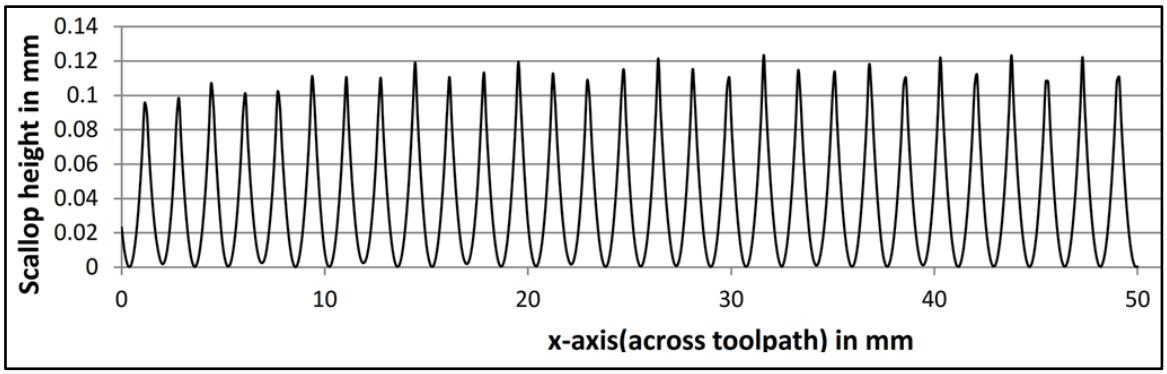


Figure 5.15: Controlled scallop height plot at y= 23.5891mm for ϵ =0.125mm

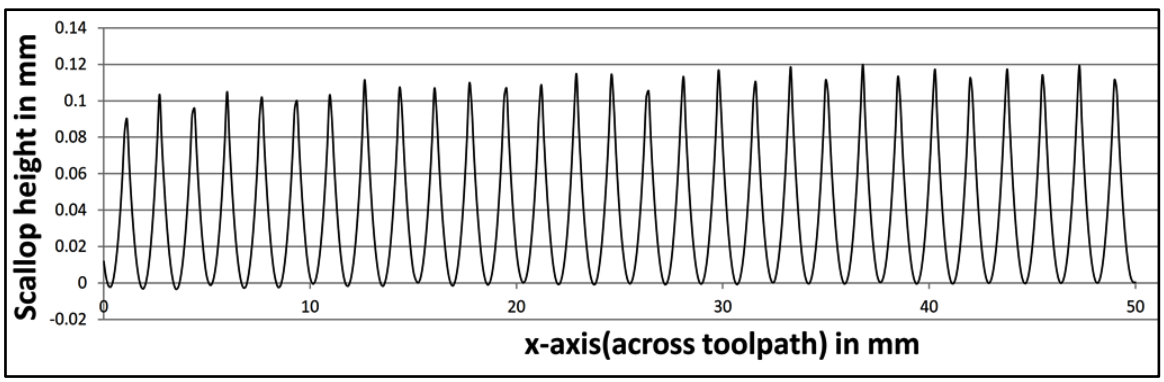


Figure 5.16: Controlled scallop height plot at y= 34.8758 mm for ϵ =0.125mm

Table 5.2 shows the number of tool location points and processing time for the toolpaths generated for the considered Bezier test surface. This methodology does not work for ϵ <0.06mm for the considered test part. Below this value, the side step size becomes too small in order to maintain the specified scallop height. Such side steps are practically not acceptable.

Table 5.2: Toolpath data for Bezier surface with Δu and Δv 0.05

ϵ value	No. of tool locations	N0. of tool passes	Processing time
0.05 mm	-	-	-
0.06 mm	6795	45	53.765 sec
0.08 mm	3237	39	29.25 sec
0.1 mm	2485	35	24.599 sec
0.125 mm	1953	31	20.467 sec

5.4.3 Bezier test part with Δu and Δv 0.025

Increasing the number of facets results in the greater smoothness of the surface. Bezier test surface with Δu and Δv equal to 0.025 contains 3366 facets, which is shown in Figure 5.17.

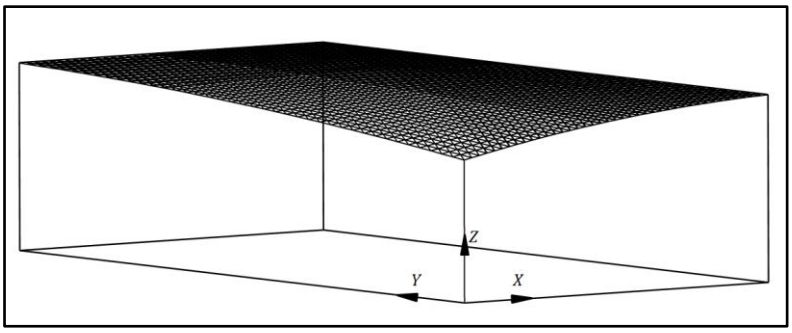


Figure 5.17: Bezier test part with Δu and Δv 0.025

Table 5.3 shows that even after refining the mesh the methodology does not work for ϵ <0.06. Except for ϵ equals to 0.06mm the results are same for both the test parts.

Table 5.3: Toolpath data for Bezier surface with Δu and Δv 0.025

ϵ value	No. of tool locations	N0. of tool passes	Processing time
0.05 mm	-	-	Does not work
0.06 mm	4905	45	136.73 sec
0.08 mm	3237	39	97.49 sec
0.1 mm	2485	35	80.36 sec
0.125 mm	1953	31	68.85 sec

5.4.4 Convex Bezier test part with Δu and Δv 0.05

To check the methodology on other parts a convex Bezier test surface is considered with Δu and Δv as 0.05. This test part is of 50×50×25 mm^3 size and contains 886 triangular facets which is shown in Fig. 5.18.

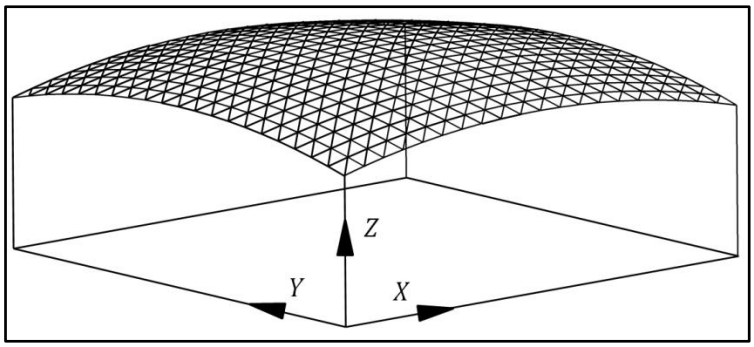


Figure 5.18: Convex Bezier test part with Δu and Δv 0.05

Control points for the considered surface are shown in table 5.4. This approach does not work on the considered part for ϵ <0.08mm as the step size in such cases is very small.

Table 5.4: Control points for convex Bezier test part [35]

Points	P_{00}	P_{01}	P_{02}	P_{10}	P_{11}	P_{12}	P_{20}	P_{21}	P_{22}
x-coordinates	0.0	0.0	0.0	25.0	25.0	25.0	50.0	50.0	50.0
***y*-coordinates**	0.0	25.0	50.0	0.0	25.0	50.0	0.0	25.0	50.0
***z*-coordinates**	15.0	21.0	15.0	21.0	25.0	21.0	15.0	21.0	15.0

Table 5.5 shows the toolpath data for the convex Bezier part for different values of ϵ. x-y plot of the toolpath for ϵ =0.08mm and its simulation result are shown in Fig. 5.19.

Table 5.5: Toolpath data for convex Bezier surface with Δu and Δv 0.025

ϵ value	No. of tool locations	N0. of tool passes	Processing time
0.07 mm	-	-	Does not work
0.08 mm	7921	89	59.08 sec
0.1 mm	3025	55	27.89 sec
0.125 mm	1849	43	19.48 sec
0.15mm	1295	37	14.81 sec

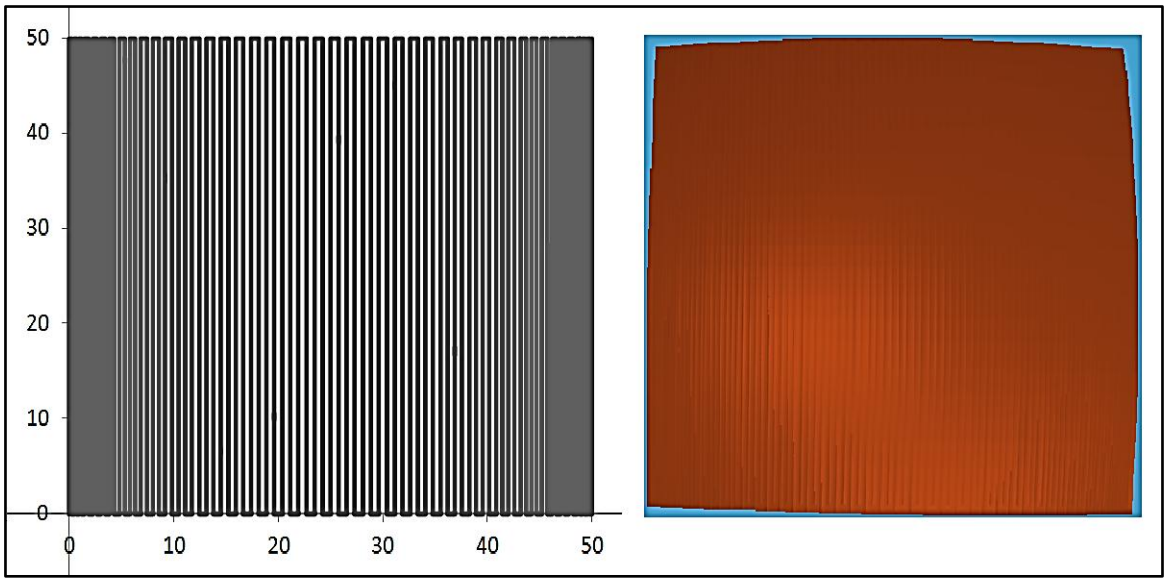


Figure 5.19: Toolpath and simulation result for convex Beazer test part for ϵ =0.08mm

5.4.5 STL surface versus 3D plot for cutter location points

3D plots for the tool location points generated with ϵ =0.08 mm using bidirectional scallop control algorithm for Bezier test part and convex Bezier test part are shown in Fig. 5.20 and Fig.5.21, alongside to their STL surfaces plot. One can observe from the figures that the 3D plots of tool location points generate the same shape as that of the original STL surface. This means the toolpaths generated using present approach is correct and can be used to machine such parts.

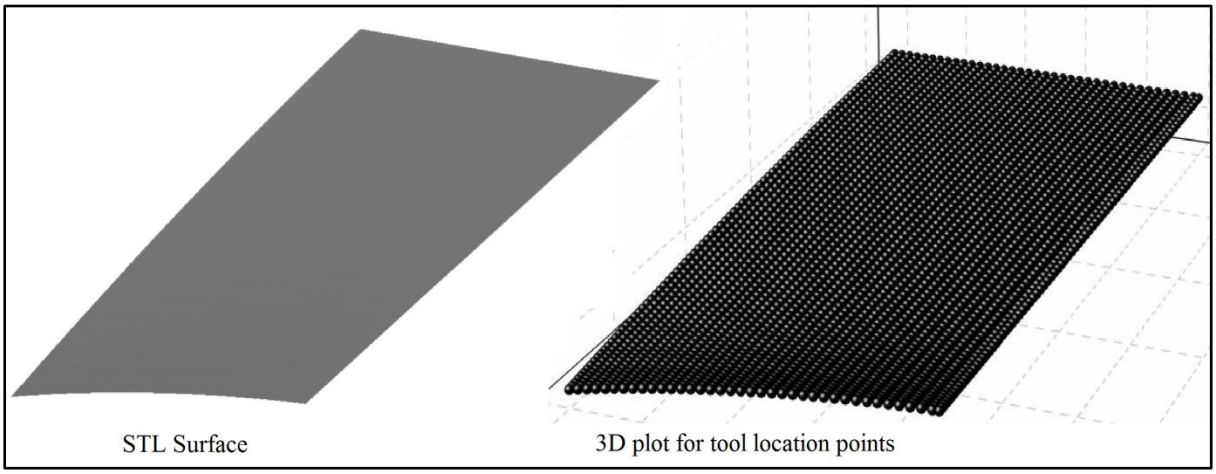


Figure 5.20: Comparison of STL surface and tool location points plot for Bezier test part

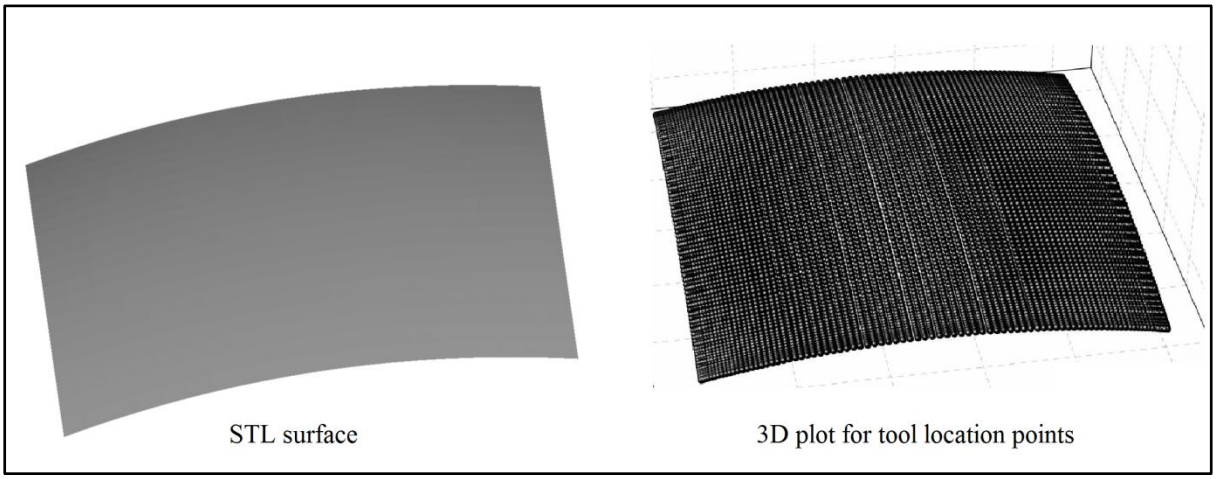


Figure 5.21: Comparison of STL surface and tool location points plot for convex Bezier test part

5.4.6 Application of cutter location data reduction algorithm for Bezier test part

For Bezier test part shown in Fig. 5.7, x-y plot of original toolpath is given in Fig. 5.9. Point elimination algorithm discussed in previous chapter is also applied to this toolpath. x-y plots for toolpaths generated for different values of ε are shown in Fig. 5.22–5.25.

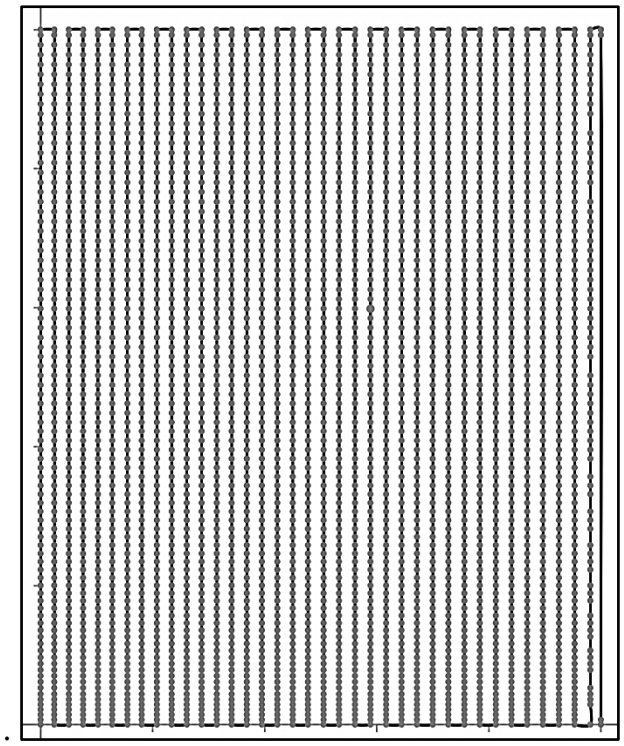


Figure 5.22: New toolpath generated for $\varepsilon = 0.0$mm

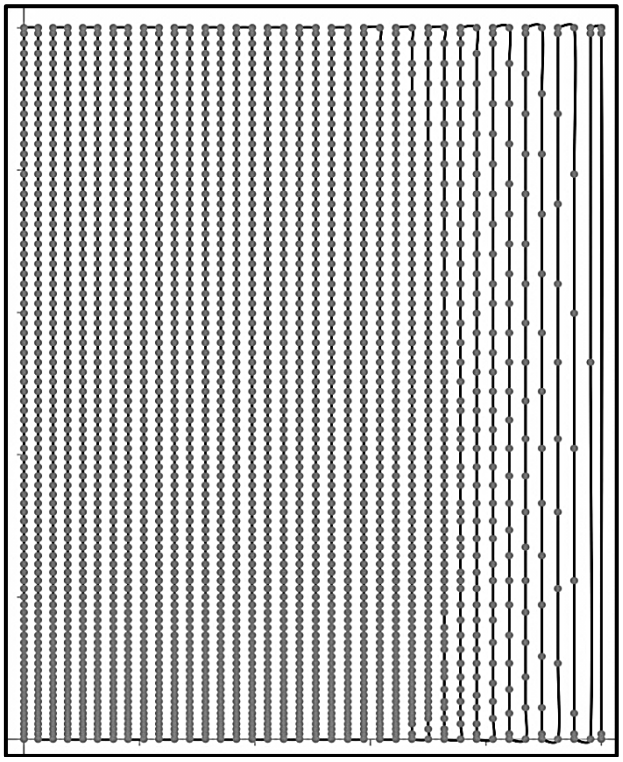


Figure 5.23: New toolpath generated for $\varepsilon = 0.02$mm

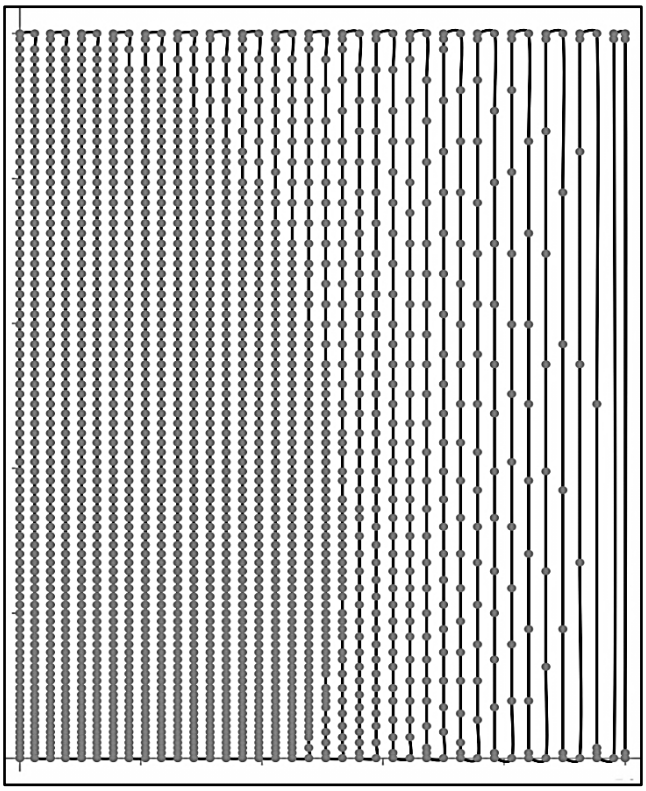


Figure 5.24: New toolpath generated for $\varepsilon = 0.05$mm

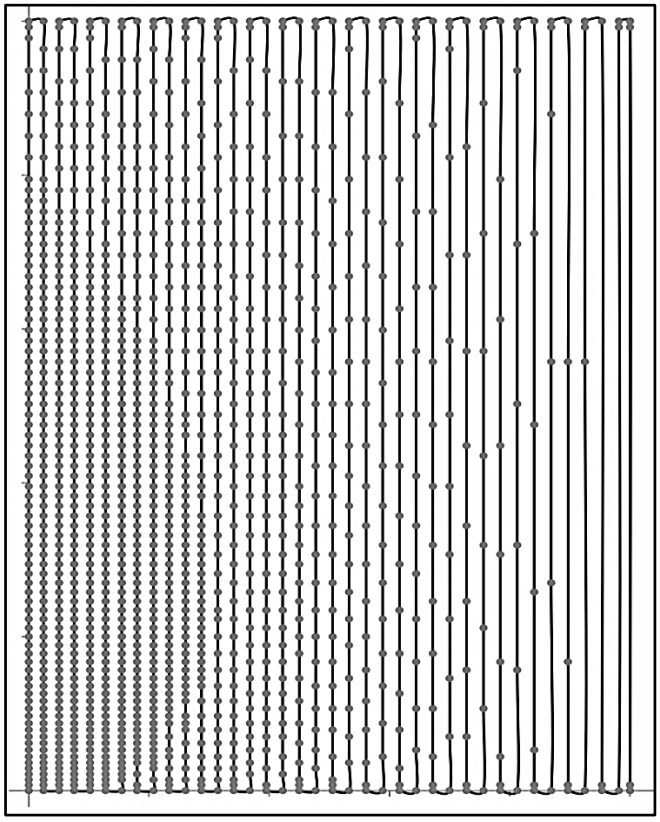


Figure 5.25: New toolpath generated for $\varepsilon = 0.1$mm

Figure 5.26 shows the simulation results of the new toolpaths generated by cutter location data reduction algorithm for various values of ε.

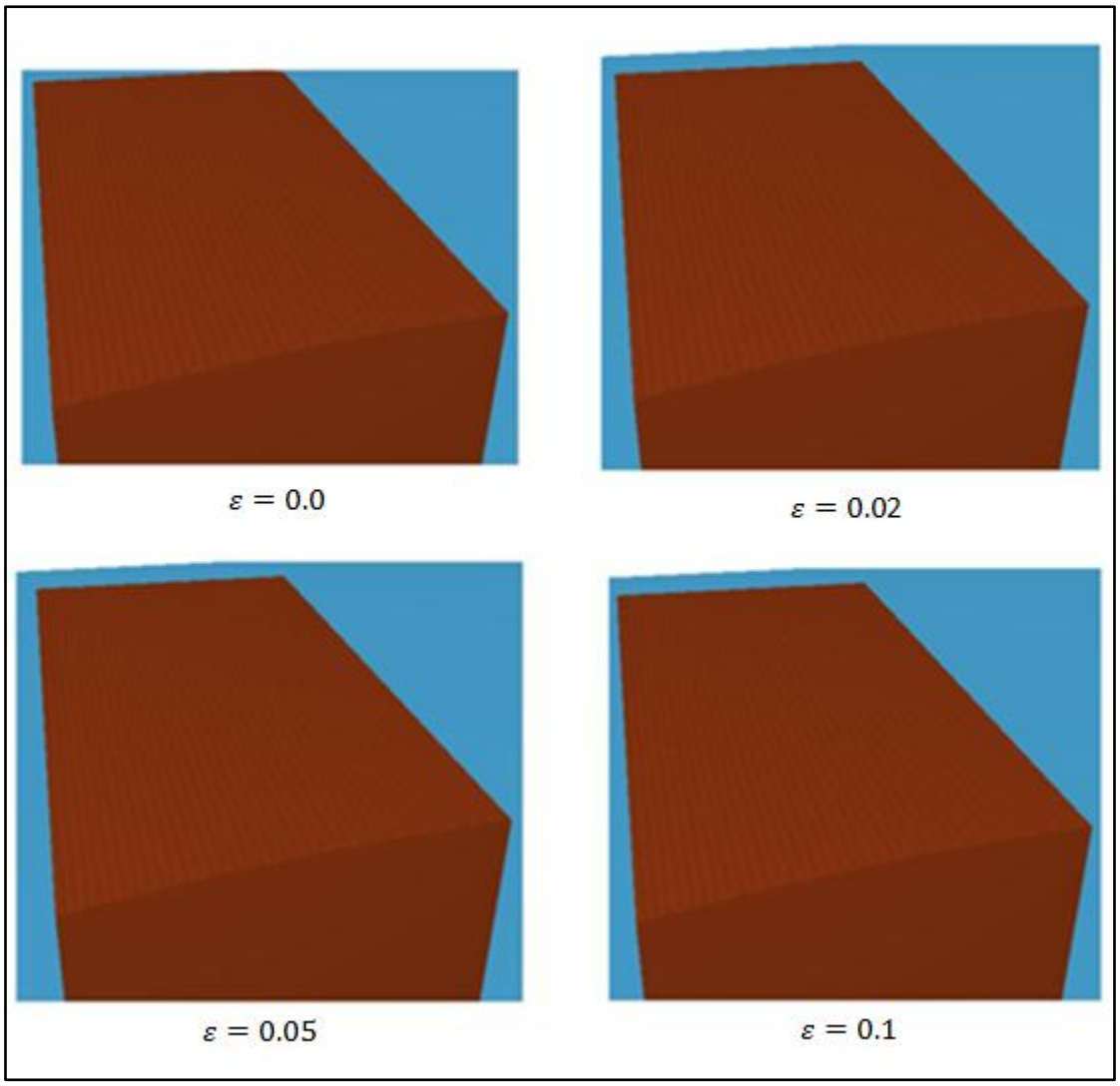


Figure 5.26: Simulation results for various values of ε for Bezier test surface

Results shows that the cutter locations data reduction algorithm can be used successfully to further reduce the cutter location points in a toolpath generated using bidirectional scallop height control algorithm.

Chapter 6
Conclusion and Scope for Further Study

The work presented in this dissertation can be broadly discretised into three parts. First of all a Microsoft windows based single page GUI application is developed to create point clouds and STL models of 3D human portraits and other freeform surfaces from digital images which can be used for carving using 3-axis machining or for creating 3D portraits using rapid prototyping. Toolpath planning strategy for 3-axis machining of such surfaces using ball end mill cutter is discussed with examples. A need was felt to reduce the cutter location points data for reducing the machining time, for which an algorithm is proposed and discussed. Last part of this work signifies a bidirectional approach to control scallop height during toolpath generation for smooth parametric surfaces approximated to STL models. This approach is based on the unidirectional approach for controlling the scallop height developed by authors [35].

6.1 Conclusion

- An application is developed to generate 3D human portraits from digital images. In presented method, an attempt has been made to process selected dark regions like head hair and facial hair. Normal images can also be processed and converted into 3D freeform surfaces.
- This Application supports different image file formats. Image size up to 10000×10000 pixels is successfully tested. There is no restriction on the size of the image. Therefore, theoretically image of any size can be processed using this application.
- The dimensions for the captured freeform surface for user defined canvas size are automatically calculated to make best out of the available space.
- Finishing toolpaths are generated for 3-axis machining of the developed freeform surfaces by ball nosed end mill using Cartesian method with raster toolpath topology. Algorithm automatically selects between point cloud data and STL file for toolpath generation depending on the density of the point cloud.
- A methodology is also proposed to eliminate unnecessary tool location points by comparing their z map so that the machining time for the machining of developed surfaces can be reduced.
- Another algorithm is proposed that generates toolpaths with bidirectional scallop height control. This algorithm works for smooth parametric surfaces only which are

approximated by triangles. Scallop height is controlled in both x and y directions using bisection method based on the maximum scallop height value specified by the user. Side steps as well as feed forward steps are adjusted to generate efficient toolpath while maintaining the scallop height within the limit.

- Scallop height plots confirm the effectiveness of this methodology for bidirectional scallop height control.
- Results are verified at different stages by machining the test parts in virtual environment using "Toolsim" simulator.

6.2 Future Scope

- Edge detection and other advanced image processing algorithms can be implemented for identifying different objects in digital image so that a proper 3D model can be generated which will be equivalent to what our eyes perceive from the image.
- Methodologies developed in present work are only tested for the ball end mill. Implementation for the other tool shapes can be studied.
- At present, bidirectional scallop height control methodology only supports smooth surfaces. It can be further tuned to make it work on intricate surfaces.

REFERENCES

[1] Mann, S., Bedi, S., Israeli, G., & Zhou, X. L. (2010). Machine models and tool motions for simulating five-axis machining. *Computer-Aided Design*, *42*(3), 231–237.

[2] Manos, N. P., Bedi, S., Miller, D., & Mann, S. (2007). Single controlled axis lathe mill. *The International Journal of Advanced Manufacturing Technology*, *32*(1-2), 55–65.

[3] Rockwood, A. P., & Winget, J. (1997). Three-dimensional object reconstruction from two-dimensional images. *Computer-Aided Design*, *29*(4), 279–285.

[4] Zeng, G., Paris, S., Quan, L., & Sillion, F. (2007). Accurate and scalable surface representation and reconstruction from images. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, *29*(1), 141–158.

[5] Cheng, C. C., Li, C. T., Huang, P. S., Lin, T. K., Tsai, Y. M., & Chen, L. G. (2009, January). A block-based 2D-to-3D conversion system with bilateral filter. In *Consumer Electronics, 2009. ICCE'09. Digest of Technical Papers International Conference on* (pp. 1–2). IEEE.

[6] Huang, X., Wang, L., Huang, J., Li, D., & Zhang, M. (2009, November). A depth extraction method based on motion and geometry for 2D to 3D conversion. In *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on* (Vol. 3, pp. 294–298). IEEE.

[7] Cheng, C. C., Li, C. T., & Chen, L. G. (2010, January). A 2D-to-3D conversion system using edge information. In *Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on* (pp. 377–378). IEEE.

[8] T., S., & Sheela, T. (2013). 2D TO 3D Conversion for images using hough transform. *International Journal of Emerging Technology and Advanced Engineering*, *3*(1), 482–486.

[9] Yang, N. E., Lee, J. W., & Park, R. H. (2013). Depth map generation using local depth hypothesis for 2D-to-3D conversion. *International Journal of Computer Graphics & Animation*, *3*(1), 1–15.

[10] Lasemi, A., Xue, D., & Gu, P. (2010). Recent development in CNC machining of freeform surfaces: A state-of-the-art review. *Computer-Aided Design*, *42*(7), 641–654.

[11] Lin, C. Y., Hwang, Y. Y., & Lai, J. Y. (1997). Interference-free cutting-path generation based on scanning data. *The International Journal of Advanced Manufacturing Technology*, *13*(8), 535–547.

[12] Lin, A. C., & Liu, H. T. (1998). Automatic generation of NC cutter path from massive data points. *Computer-Aided Design*, *30*(1), 77–90.

[13] Chuang, C. M., Chen, C. Y., & Yau, H. T. (2002). A reverse engineering approach to generating interference-free tool paths in three-axis machining from scanned data of physical models. *The International Journal of Advanced Manufacturing Technology*, *19*(1), 23–31.

[14] Teng, Z., Feng, H. Y., & Azeem, A. (2006). Generating efficient tool paths from point cloud data via machining area segmentation. *The International Journal of Advanced Manufacturing Technology*, *30*(3-4), 254–260.

[15] Kayal, P. (2009). Inverse offset method for adaptive cutter path generation from point-based surface. *International journal of CAD/CAM*, *7*(1).

[16] Makki, M., Lartigue, C., Tournier, C., & Thiebaut, F. (2008). Direct duplication of physical models in discrete 5-axis machining. *Virtual and Physical Prototyping*, *3*(2), 93–103.

[17] Zhang, D., Yang, P., & Qian, X. (2009). Adaptive NC path generation from massive point data with bounded error. *Journal of Manufacturing Science and Engineering*, *131*(1), 011001-1–011001-13.

[18] Patel, K., Bolaños, G. S., Bassi, R., & Bedi, S. (2011). Optimal tool shape selection based on surface geometry for three-axis CNC machining. *The International Journal of Advanced Manufacturing Technology*, *57*(5-8), 655–670.

[19] Duvedi, R. K., Bedi, S., Batish, A., & Mann, S. (2014). A multipoint method for 5-axis machining of triangulated surface models. *Computer-Aided Design*, *52*, 17–26.

[20] Yau, H. T., Chuang, C. M., & Lee, Y. S. (2004). Numerical control machining of triangulated sculptured surfaces in a stereo lithography format with a generalized cutter. *International journal of production research*, *42*(13), 2573–2598.

[21] Lai, J. Y., & Wang, D. J. (1994). A strategy for finish cutting path generation of compound surfaces. *Computers in industry*, *25*(2), 189–209.

[22] Chen, T., & Shi, Z. (2008). A tool path generation strategy for three-axis ball-end milling of free-form surfaces. *journal of materials processing technology*, *208*(1), 259–263.

[23] Lin, C. Y., Hwang, Y. Y., & Lai, J. Y. (1997). Interference-free cutting-path generation based on scanning data. *The International Journal of Advanced Manufacturing Technology*, *13*(8), 535–547.

[24] Hwang, J. S., & Chang, T. C. (1998). Three-axis machining of compound surfaces using flat and filleted endmills. *Computer-Aided Design*, *30*(8), 641–647.

[25] Park, S. C. (2004). Sculptured surface machining using triangular mesh slicing. *Computer-Aided Design*, *36*(3), 279–288.

[26] Yuwen, S., Dongming, G., & Haixia, W. (2006). Iso-parametric tool path generation from triangular meshes for free-form surface machining. *The International Journal of Advanced Manufacturing Technology*, *28*(7-8), 721–726.

[27] Sortino, M., Belfio, S., Motyl, B., & Totis, G. (2014). Compensation of geometrical errors of CAM/CNC machined parts by means of 3D workpiece model adaptation. *Computer-Aided Design*, *48*, 28–38.

[28] Chen, T., & Shi, Z. (2008). A tool path generation strategy for three-axis ball-end milling of free-form surfaces. *journal of materials processing technology*, *208*(1), 259–263.

[29] Park, S. C., & Chang, M. (2010). Tool path generation for a surface model with defects. *Computers in Industry*, *61*(1), 75–82.

[30] Lee, D. Y., Kim, S. J., Kim, H. C., Lee, S. G., & Yang, M. Y. (2006). Incomplete two-manifold mesh-based tool path generation. *The International Journal of Advanced Manufacturing Technology*, *27*(7-8), 797–803.

[31] Feng, H. Y., & Li, H. (2002). Constant scallop-height tool path generation for three-axis sculptured surface machining. *Computer-Aided Design*, *34*(9), 647–654.

[32] Lee, S. G., Kim, H. C., & Yang, M. Y. (2008). Mesh-based tool path generation for constant scallop-height machining. *The International Journal of Advanced Manufacturing Technology*, *37*(1-2), 15–22.

[33] Yang, D. O., & Feng, H. Y. (2008). Machining triangular mesh surfaces via mesh offset based tool paths. *Computer-Aided Design and Applications*, *5*(1-4), 254–265.

[34] Jasra, P. M. (2009). *Generalized tool path generation algorithm for sculptured pseudo symmetric surface machining* (M.E. dissertation, Thapar University Patiala).

[35] Rajiv. (2014). *Controlled scallop height tool path generation for 3-Axis vertical CNC machining of STL surfaces* (M.E. dissertation, Thapar University Patiala).

[36] Lauwers, B., Kiswanto, G., & Kruth, J. P. (2003). Development of a five-axis milling tool path generation algorithm based on faceted models. *CIRP Annals-Manufacturing Technology*, *52*(1), 85–88.

[37] Kim, B. H., & Choi, B. K. (2002). Machining efficiency comparison direction-parallel

tool path with contour-parallel tool path. *Computer-Aided Design*, *34*(2), 89–95.

[38] Duvedi, R. K., Bedi, S., Batish, A., & Mann, S. (2015). Numeric implementation of drop and tilt method of 5-axis tool positioning for machining of triangulated surfaces. *The International Journal of Advanced Manufacturing Technology*, *78*(9-12), 1677–1690.

[39] *The openCV reference manual* (2.4.9.0 ed.). (2014).