OFF-LINE NOMINAL PATH GENERATION OF 6-DOF ROBOTIC MANIPULATOR FOR EDGE FINISHING AND INSPECTION PROCESSES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MAHMOUD NEMER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

JUNE 2016

Approval of the thesis:

OFF-LINE NOMINAL PATH GENERATION OF 6-DOF ROBOTIC MANIPULATOR FOR EDGE FINISHING AND INSPECTION PROCESSES

Submitted by **MAHMOUD NEMER** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** ____________________

Prof. Dr. R. Tuna Balkan
Head of Department, **Mechanical Engineering** ____________________

Assoc. Prof. Dr. E. İlhan Konukseven
Supervisor, **Mechanical Engineering Dept., METU** ____________________

Examining Committee Members:

Prof. Dr. Reşit Soylu
Mechanical Engineering Dept., METU ____________________

Assoc. Prof. Dr. E. İlhan Konukseven
Mechanical Engineering Dept., METU ____________________

Assoc. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Engineering Dept., METU ____________________

Asst. Dr. Ali Emre Turgut
Mechanical Engineering Dept., METU ____________________

Prof. Dr. Arif Adlı
Mechanical Engineering Dept., Gazi University ____________________

Date: 23/06/2016

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Mahmoud Nemer
Signature:

ABSTRACT

OFF-LINE NOMINAL PATH GENERATION OF 6-DOF ROBOTIC MANIPULATOR FOR EDGE FINISHING AND INSPECTION PROCESSES

Nemer, Mahmoud

Ms., Department of Mechanical Engineering

Supervisor: Assoc. Prof. İlhan E. KONUKSEVEN

June 2016, 157 pages

This thesis deals with the development of a Computer Aided Robotic Machining Process Planning package. The main aim of the package is to generate an efficient, collision-free, nominal tool path needed for edge finishing and inspection processes by utilizing a 6-DoF robotic arm.

Automation of edge deburring and chamfering consists of two main parts. First part is generating the overall nominal tool path. While the second part focuses on controlling the material removal. The overall nominal tool path planning involves analyzing the geometry of the workpiece, determining and designing an efficient collision-free tool path and generating the tool path data for the robot and finally verifying it. The generated tool path can also be used for different robotic machining processes.

One of the most popular PC-based CAD software, SolidWorks, is chosen as the user interface platform. A software package programmed in the application programming interface (API) of SolidWorks generates tool path data for the robot. The ABB IRB2000 robot is chosen for executing the generated tool path. The programming language used for developing this software is Visual Basic. Ultimately, such path is to be utilized as the nominal tool path by any control strategy present in the literature for a complete automatic edge finishing process.

Keywords: Offline programming, CAD-based tools, Edge deburring, Edge scanning, Path generation.

ÖZ

SON IŞLEM TAŞLAMASI VE YÜZEY MUAYENESI AMACIYLA, 6 SERBESTLIK DERECELI ROBOTIK MANIPULATÖR KULLANARAK ÇEVRIMDIŞI YÖRÜNGE ÇIKARIMI

Nemer, Mahmoud

Ms., Department of Mechanical Engineering

Supervisor: Assoc. Prof. İlhan E. KONUKSEVEN

Haziran 2016, 157 sayfa

Bu tezde Bilgisayar Destekli Robotik İşleme Prosesi Planlama pakteti irdelenmiştir. Bu paketin ana amacı, kenar işlemesi ve muayenede kullanılan, verimli, ve çarpışma riski olmayan, takım ucunun izlediği nominal yolu, 6 serbestlik derceli bir robot ile geliştirmektir.

Malzemenin kenarındaki çapakların alınmasında ve pah kırma işlemlerinin otomasyonu iki bölümden oluşmaktadır. İlki, bütün işlem için bir nominal yol geliştirmektir. İkinci bölümde ise temel amaç malzeme kaldırma miktarını kontrol etmektir. Aletin takibi için geliştirilen nominal yolu planlamak için birkaç işlemin gerçekleşmesi lazım: parçanın geometrisinin analizi, çarpışmayı verimli bir şekilde önlemek, robotun kullanması için aletin yol verilerinin oluşturulması ve son olarak işlemin doğrulanması.

Arayüz platform olarak bilgisayar tabanlı CAD yazılımlarının en popülerlerinden olan SolidWorks kullanılmıştır. Robotun yol verilerini üretmek için SolidWork'ün API'yı ile bir yazılım paketi geliştirilmiştir. Bu yolu takip etmesi için ise ABB IRB2000 robotu seçilmiştir. Programlama dili olarak Visual Basic kullanılmıştır. Sonuç olarak bu yol tamamen otomatik bir kenar işleme prosesi için aletin takip etmesi gereken nominal yol olarak kullanılabilir.

Anahtar kelimeler: Çevrimdışı programlama, CAD tabanlı araçlar, Çapak alma, Çapak tarama, Yörünge üretimi.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Assoc. Prof. Dr. E. İlhan Konukseven for providing me this research opportunity, guiding me throughout the study, and his patience. I am also indebted him for his precious evaluations and feedbacks in writing this thesis. I would like to express my gratitude to Asst. Prof. Dr. Bugra Ahmet Koku for making me a better engineer. I owe many thanks to my friends who study in the METU Mechatronics Laboratory for their precious support and presence. To my family, I really appreciate their endless support and patience.

To my family,

with the deepest gratitude

TABLE OF CONTENTS

CHAPTERS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

2D	Two Dimensional.
3D	Three Dimensional.
ADLP10	ABB Data Link Protocol.
API	Application programming interface.
ARAP	ABB Robot Application Protocol.
a_i	Length of link i, distance between $\vec{u}_3^{(i-1)}$ and $\vec{u}_3^{(i)}$ along $\vec{u}_1^{(i-1)}$.
CAD	Computer-Aided Design.
CAM	Computer-Aided Manufacturing.
$C^{(i-1,i)}$	Rotation matrix of link i with respect to link i-1.
c_{ij}	Element of the C matrix in ith row and jth column.
DoF	Degree of freedom.
$\vec{d}$	Offset vector from an edge.
d_i	Offset of link i, distance from the origin of frame i-1 to $\vec{u}_1^{(i)}$ along $\vec{u}_3^{(i-1)}$.
l_i	Length in the direction of the major axis of link i.
O_0	Origin of the global coordinate system.
RF	Reference frame.
$\vec{r}$	Position vector of the tip-point with respect to the origin of global coordinate.
$\vec{r}_{i-1,i}$	Relative position of the origin of frame i with respect to frame i-1.
TSP	Traveler Salesman Problem.
$\vec{w}$	Position vector of the wrist with respect to the origin of global coordinate.
$\vec{u}_1^{(n)}$	Unit vector of frame n in the direction of x-axis.
$\vec{u}_2^{(n)}$	Unit vector of frame n in the direction of y-axis.

$\vec{u}_3^{(n)}$	Unit vector of frame n in the direction of z-axis.
α_i	Twist of link i, angle from $\vec{u}_3^{(i-1)}$ to $\vec{u}_3^{(i)}$ about $\vec{u}_1^{(i)}$.
γ	Safety level, in the direction of z-axis of the global coordinate.
Δx	Difference in the x-axis direction.
Δy	Difference in the y-axis direction.
Δz	Difference in the z-axis direction.
θ_i	Angle of joint i, angle between $\vec{u}_1^{(i-1)}$ and $\vec{u}_1^{(i)}$ about $\vec{u}_3^{(i-1)}$.

CHAPTER 1

INTRODUCTION

1.1 Overview

Deburring processes have been identified as the bottleneck in many machine industries. The burr removal methods can induce dimensioning errors to the workpiece if improperly executed. Burrs are caused by many machining process including milling, drilling, turning, and broaching. Edge finishing like chamfering is important for several reasons, sharp edges may pose personal hazardous, since they can cause injuries to worker or user. Part mating may be more difficult due to clearance restriction caused by burrs. High stress concentration at sharp corners can cause product failures, reduce tool life during hard finishing. Presently, manual finishing accounts for 12% of the total labor cost [1].

Chamfering is performed at the final stage of manufacturing, where parts have their highest added value, quality control is absolute necessity. Despite this requirement, even in today's most fully automated factories it is still a common sight to see dozens of worker manually chamfer produced parts. Edge finishing is typically performed manually using hand held power tools with brushes, abrasive tips, or rotary files or by manual files and knives. The techniques employed with these tools are not well documented and inspection of these chamfered edges is not quantitatively defined. Typically the worker runs the finger over the edge to inspect the work. Improving both the efficiency and quality of chamfering is a major concern. Chamfering is labor intensive and can represents a significant portion of the expense of manufacturing machined parts. In addition, chamfering is frequently a dirty, noisy, and undesirable job and high turnover in terms of personnel. Training personnel in proper chamfering technique coupled with high turnover rate adds to the overall expense of the chamfering. Variation in skill level of chamfering personnel causes variation in the quality of the part. Errors encountered in the chamfering operation which causes the part to be scrapped are costly, as the part is near the end of its manufacturing cycle. Consequently, quality control and part inspection are key processes in the lifecycle of a product. These processes are able to verify product quality; and can provide essential feedback for enhancing other processes. No change is made to product during inspection, in order to increase its value. Time and resources are spent on these processes, without a gain in profit, making the reduction of the time spent on these processes an attractive concept to manufacturer.

An amazing transformation of edge finishing has occurred over the past 50 years trying to mimic adaptive nature of human intelligence in order to replace manual deburring. Anthropomorphous robots are the best state of the art compromise between performance and flexibility for automated deburring tasks [2]. They provide larger work volumes, safety and efficiency at a lower cost than CNC machines. Also they provide a greater reachability and working capabilities on the complex paths of the deburring tasks.

1.2 Literature Survey

Two main topics were considered in the literature survey, namely, path generation of the deburring process and path generation of laser scanners.

Starting with the first paper, Valente and Oliveira [3] define three steps for creating and controlling a tool path that leads to the desired deburring results. The first step mentioned was the offline programming, in this stage the user manually generates a nominal path with the use of the 3D CAD model that the tool center point (TCP) will be following. The second step called contact evaluation module. In this stage the contact between the tool and the workpiece during the execution of the nominal path is evaluated using a modified version of Malkin’s model for grinding, which determines the grinding power needed to insure efficient deburring. The final step is called the active path control. This stage monitors and control the robot path during the actual deburring process in order to maintain the power signal as close as possible to its target level. This control strategy is based on two signals, Acoustic Emission and Power. This signal is compared with two limits, a lower one (T1) and a higher limit (T2), and hence the nominal path is modified by keeping the signal between T1 and T2, see the figure below.

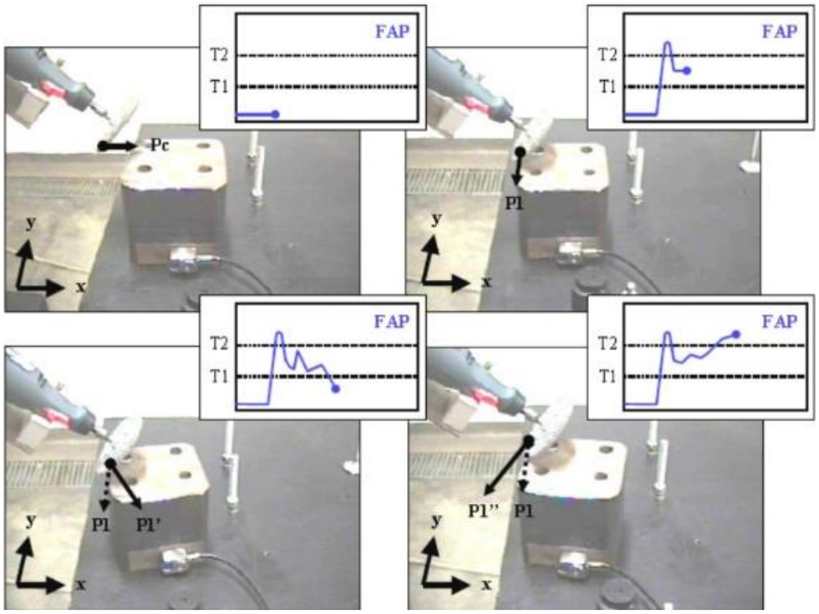


Figure 1-1: Strategy for tool path control.

In their paper, Murphy et al. [4] offer a technique to automate robot programming. The technique uses CAD geometry data to automatically generate robot deburring path and then corrects it using a force sensor attach to the tool head. Using a graphics interface, an operator specifies the edges on a part to be deburred, the deburring tools to be used, the speeds, feed rates and contact force desired. After that, a robot path planner generates an initial deburring path for the robot. The initial path generation is done in two steps. First step is edge organization. The deburring data developed by the graphic interface is an unordered list of edges. Edges with the same deburring tool and roughly the same deburring parameters are grouped in deburr-paths. The path planner then divides each deburr-path into one or more loops (a loop is a set of edges that can be deburred without lifting the tool from the part). The second step is pose generation. Once edges are organized, the path planner creates a sequence of poses which define the trajectory for each deburr-path (the poses contain position and orientation information in the part’s coordinate frame) during deburring the robot moves between these poses in straight-line motion. There are two types of poses in a deburr-path: Vertex pose, at which the tool makes contact with the part, and go-to pose, used for approaching and depart trajectories. After that these position are adjusted using a force sensor to compensate for offset errors.

Leali et al. [2] discusses an off-line programing (OLP) approach to overcome the disadvantage of the point-to-point teaching method. The main idea is to use a 3D CAD model of the workpiece and define the teaching points on it. The first step concern the analyses of the industrial problem and the workpiece geometry and material, aiming at clarifying the robotic tasks. The main robot operations are clarified, addressing finishing tools, manufacturing parameters. The next macro-phase is the CAD-based OLP. The model must contain all the information needed to simulate the robot work-cycle. In a CAD-based OLP two types of software can be used. The first is based on general purpose software platforms. They create the work path from the mathematical description of the CAD features. Such “Paths from the Math” approach is very intuitive and quick but not manufacturing oriented. The second one represents an extension of the typical CAM simulation, where the robot is typically regarded as a 5 + 1 axes tool machine. After robot programming all the designed 3D modular models and subprogram modules are assembled in a virtual workcell layout. A workcell virtual prototype is then available to simulate and virtually optimize the process performances and generate the robot code. A workcell calibration process is finally required to define the exact position of the reference frames.

A teaching method using teaching support devices was developed by Sugita et al [5] for a deburring robot. The main idea is to teach the robot using devices other than the robot itself, so that there will not be a down time and the programming can be done off-line. Two kinds of teaching support devices were developed to prepare the tool path. One is a three-wire type teaching support device. The structure of the device is shown in Figure 1-2 a). This is a device that can measure a position in 3D coordinates, and it is composed of a position measuring unit and a posture measuring unit. These units are connected with a wire to measure the distance between them, and the device can measure the position and direction vector of the dummy tool on the tip of the posture measuring unit in the device's coordinates. Figure 1-2 a) shows the structure of the second teaching support device. This device is composed of two arms and a wrist, and has six axes of freedom. Optical encoders are adopted as angle detectors for each link in order to reduce the influence of noise and temperature fluctuation to the accuracy. The detecting resolution of each link is 17 bit. This method can solve the wire-damping problem in positioning the device.

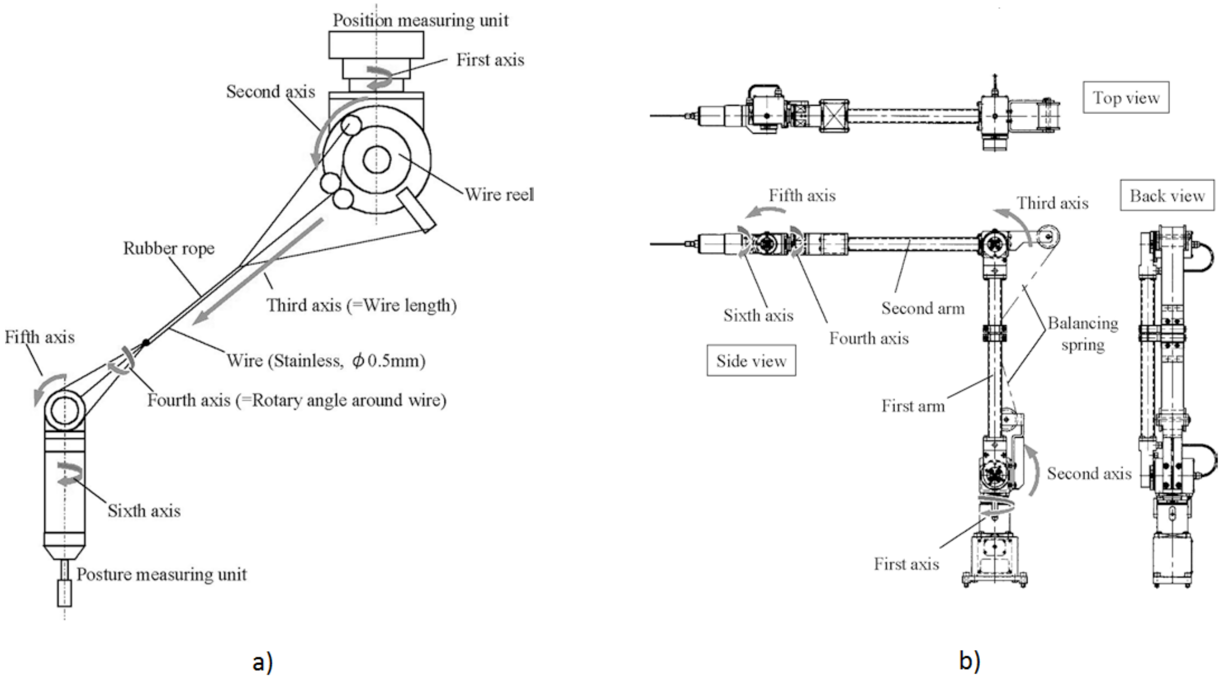


Figure 1-2: a) Three-wire type teaching support device
b) Structure of arm-type teaching support device.

El-Bestawi et al. [6], presented an approach that focused more on the off-line planning part. In particular, a “hypothesis and test” method approach was adopted, that is, random discrete poses of the robotic hand were generated between the start and goal positions while checking for collisions in these random poses. Afterwards, a smoothening algorithm is applied to the collision-free poses to yield a continues path. Nonetheless, it was pointed out that a presence of a user is required, hence not fully automated, and that this procedure takes about 3 hours on a normally sized workpiece.

Asakawa et al. [7] offers an approach to automate the nominal path needed for deburring a hole on free-curved surface on the basis of CAD data. The 3D curve of the edge is equally divided by points P, depending on a chamfering condition. A normal vector N_{pn} is defined at the point P_n, F_n is the vector directing from P_n to P_{n1}, respectively. An outer product of N_{pn} and F_n corresponds to the tool axis vector D, see Figure 1-3. Doing this for all the P points then transforming these information into robot control commands results in a nominal path to be followed by the end-effector. The paper also adds a touch sensor in order to control the initially generated path in real-time to make sure the end result of the chamfering process is as desired.

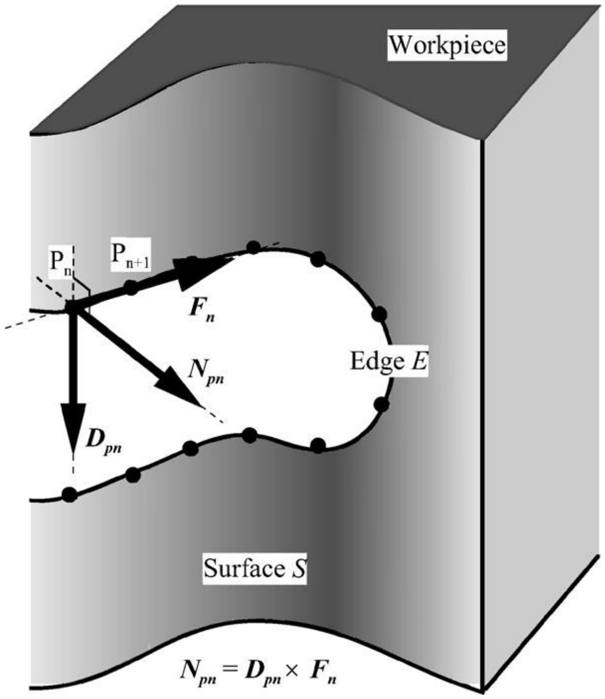


Figure 1-3: Generation of chamfering points.

Zhang and his team [8] proposed a method for on-line path generation for robot deburring of cast aluminum wheels. The automated robot path generation system is developed to automatically generate a 6 DOF robot path based on the vision, force and position sensor fusion. Before generating a path to deburr a wheel, the tool path is manually marked on the surface of the wheel. The robot tool is controlled to continuously follow the center of the marked tool path on the surface while the tool tip is kept continuous contact with the surface using a force control strategy. Thus the z coordinate of the tool tip in the tool frame can be directly obtained. From the images captured by the camera, the x and y coordinates and the roll orientation can be controlled using the visual input. The other two orientations are determined by finding the local surface normal based on the (x, y, z) coordinates in a local region which are obtained by moving the tool in a zigzag pattern. The position and orientation are recorded and a path is generated that follows the feature and is perpendicular to the local surface. The generated path is then smoothed to obtain a final tool path. The robot is then automatically programmed to perform a task.

Lee et al. [9] worked on enhancing the teaching and playback method of path generation for deburring process. By compressing and smoothing the initial path using Douglas-Peucker (DP) algorithm the path becomes less noisy and vibrations that may arise due to the physical limitations of the robotic manipulator's joints are damped. Moreover, due to the unexpected shape of the burrs, this research presents a method that protects the tool from damage by reducing the tool velocity based on the measured reaction force, which was measured and utilized at the tool with a force sensor, and it offered information on the estimated size of the burrs. That is, when the burr was big, the measured force was also large. The size of the tangential force at the tool had to be controlled so that the tool would not be damaged. The larger the burrs were, the more slowly the manipulator had to move in the tangential direction to maintain a relatively constant tangential force so as to tear off the plotted burrs; and when it encountered a burr, the slower speed of the endpoint along the surface implied a constant volume of material that had to be removed for every unit of time, which made the force constant in the tangential direction. The tangential force was found to have been linearly related to the tool velocity, so that even though the material removal rate decreased, the operator had to slow down the tool velocity to overcome this problem.

Song et al. [10] proposes a hybrid off-line path generation method for minimizing the error offset that may arise due to imperfections, Furthermore, impedance control is used to avoid applying excessive contact force in real-time. Tool path generation based on matching between the tool paths from the CAD model to teaching point is proposed to minimize the position and orientation errors of the workpiece. The basic tool path for deburring can be generated by the CAM software with a CAD model by extracting the G-code and converting its commands into the initial tool path. Using direct teaching based on impedance control, some contact points between the tool and the actual workpiece are manually selected as the teaching points, which are the minimum number of points to feature the shape of the workpieces. Based on matching the tool path extracted from the CAD model to the teaching points, the transformation matrix reflecting the position and orientation errors of the workpiece can be achieved. Then, the tool path can be modified to minimize the position and orientation errors. Therefore, the proposed method can take the advantages of both the teaching method and the CAD/CAM approach.

A paper written by Ziliani et al. [11] deals with the implementation of a mechatronic methodology for the robotic deburring of planar workpieces with an unknown shape performed by an industrial manipulator. The approach is based on the use of a hybrid force/velocity control law and on a correlated suitable design of the deburring tool by utilizing a contour tracking method that aims to control the normal force and the tangential velocity of the robot probe along the normal and tangential directions on the contacting point, respectively.

Princely and Selvaraj [12] developed a teaching-less robot system for deburring planar workpieces having burrs on the edges using image processing system. This system does not require the contour shape data from the CAD profile or by manual entry of the data by the robot operator. In this work a vision sensory system is used to capture the image of the workpiece. This image of the workpiece is then processed to acquire the edges to be deburred by segmentation of the edges into straight lines. The robot language program for each workpiece is generated automatically from the workpiece shape data and finishing condition data. The main advantage of this method is that it provides the orientation, position, and shape of the workpiece on the deburring workstation in a short time, overcoming the offset errors that may arise from mounting the workpiece on pallets and/or working tables.

For laser scanning related topic, no paper that utilizes a point laser scanner discussed the generation of the scanning path itself for piece inspection. [13] and [14] show an example of such papers where the main concern is the algorithm used for scanning and organizing the collected data from the scanner itself. On the other hand, line laser scanners are widely used for freeform surface inspection. Xi et al. [15] proposed path planning method that utilizes the CAD model of the considered workpiece. By determining the surface profile, the method considers how to set the field of view of the scanner in order to achieve a maximum coverage. Son et al. [16] developed an automated laser scan planning system for the multi-patched freeform surfaces. The scan plan includes scan directions, scan regions and the corresponding scan paths. Morozov and his team are developing a custom MATLAB toolbox in an ongoing project [17]. Their goal is to achieve an automated path generation system for non-destructive tests. However, none of these research conceder the objective of scanning the edges of a given workpiece.

1.3 Thesis Motivation and Objective

Going through the literature of the automatic edge finishing one can notice that the major concern is directed towards the real-time path planning of the end-effector of the robot while performing the action of deburring. Less attention is given for the initial (nominal) path or motion of the robot, which is done before the actual finishing process. Furthermore, the motion needed for scanning the edges of a part shares many features with that of the edge finishing motion, making it possible to develop one algorithm capable of generating both kind of motions.

Hereby, the main objective of this study is to utilize a 6-DoF industrial manipulator and generate a nominal collision-free path for a given workpiece from its CAD model. This path is then used for either scanning or finishing the wanted edges on that workpiece. However, if the path is generated for edge finishing processes then a real-time control strategy has to be applied on the generated path in order to achieve the desired end results on the workpiece.

It is necessary to point out that this study aims to proof the concept of generating an off-line path for an automatic edge scanning/finishing. On the other hand, this does not mean that the offered solution is the absolute optimal one. There are other algorithms for both collision-free path generation and solving the Traveler Salesman Problem (TSP) that can be applied to this problem. Nonetheless, the proposed method in this thesis significantly reduces the complexity of the problem. Yet it offers an efficient overall solution. With one disadvantage that it might overlook some edges, if found unreachable by the robot manipulator.

ABB IRB2000 model is assumed throughout the thesis as the 6-DoF manipulator. SolidWorks 2014 is used as the CAD software. A software developed in the API environment of SolidWorks and written in visual basic programming language serves as the main program to perform the main goal of the study. Finally, the generated path is converted into motion commands and sent to the ABB IRB2000 by an independent software that is developed in C++ language.

1.4 Thesis Outline

In this thesis, the main components of the development of an off-line path generation system are introduced and how they are constituted is described.

Chapter 2 includes basic concepts of robot kinematics of ABB IRB2000. Forward and inverse positional kinematic analyses are derived. Implementations of these analyses to the computer program is also discussed.

Chapter 3 explains the path planning procedure of the tip point of the robotic arm. Collision detection test is also discussed in this chapter.

Chapter 4 describes the developed programs covering all parts of the software except for those explained in the previous chapters.

Chapter 5 discusses the outcome of experimental tests to verify the working procedure of the study.

Chapter 6 concludes the thesis by summarizing the work done and discussing possible future work

CHAPTER 2

KINEMATIC ANALYSES

Kinematics is the science of motion. ABB IRB2000 is considered as a series of links connected by joints. Joints of robots have one degrees of freedom. The user/programmer is interested in the position and orientation (pose) of the end-effector. However, the robot is controlled by the joint actuators and actuators controls the joints in terms of angles. There are two main parts of these analyses, forward and inverse kinematics. In the kinematic analyses, the translational and rotational relations between adjacent links must be described. Hartenberg and Denavit proposed a matrix method for this purpose. First HD convention parameters will be expressed and position analyses will be done accordingly.

2.1 Hartenberg-Denavit (HD) Convention

A systematic technique for establishing the displacement matrix for each two adjacent links of a mechanism was proposed by Hartenberg and Denavit in 1955. The same convention will be used in this investigation [18].

The HD convention is mainly implemented in robot manipulators, which consist of an open kinematic chain in which each joint contains one degree of freedom and the joint is either revolute or prismatic. The HD convention is implemented through the following steps:

1. Number the links and joints, starting at the base. The stationary base is denoted as link 0 and the end effector is link m, as demonstrated in Figure 2-1. Link n moves in respect to link n-1 around (for revolute) or along (for prismatic) joint i.

2. Establish links' coordinate system for each of the joints.

3. Define the joint parameters, which are the four geometric quantities $\square_{n+1}$, d_{n+1}, a_{n+1}, $\square_{n+1}$.

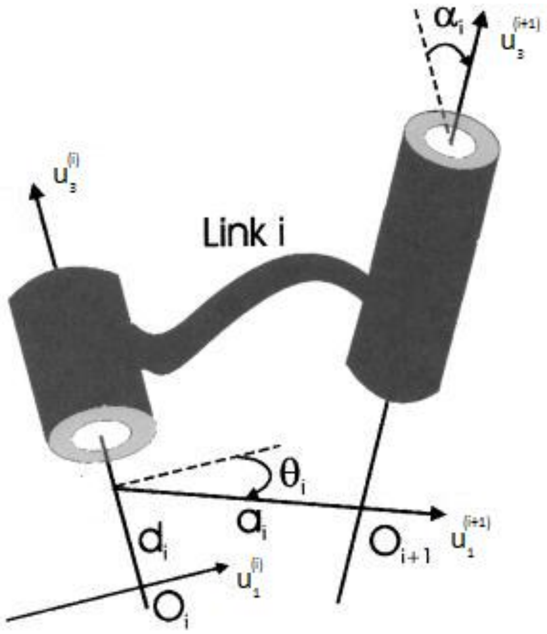


Figure 2-1 Schematic of the Denavit-Hartenberg convention parameters of a link.

Using these parameters, the orientation matrix $\widehat{\mathrm{C}}^{(\mathrm{n-1,n})}$ and the relative position of link n with respect to link n-1 is given by (2.1) and (2.2), respectively:

$$\widehat{\mathrm{C}}^{(\mathrm{n-1,n})} = \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_\mathrm{n}}\mathrm{e}^{\tilde{\mathrm{u}}_1\alpha_\mathrm{n}} \tag{2.1}$$

$$\vec{r}_{n-1,n} = \mathrm{d_n}\vec{u}_3^{(n-1)} + \mathrm{a_n}\vec{u}_1^{(n)} \tag{2.2}$$

Where,

$$\tilde{n} = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \quad \text{if} \quad \bar{n} = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$$

And,

$$\mathrm{e}^{\tilde{\mathrm{u}}_1\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \mathrm{e}^{\tilde{\mathrm{u}}_2\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \mathrm{e}^{\tilde{\mathrm{u}}_3\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

According to the robot's links and distance definitions in Figure 2-2, the HD parameters are as shown in table 2-1:

Table 2-1: Denavit-Hartenberg parameters of ABB IRB2000 manipulator.

Link	a_i $[mm]$	α_i	d_i $[mm]$	θ_i
1	0	$\alpha_1 = -\pi/2$	$d_1 = 750$	θ_1
2	$a_2 = 710$	0	0	θ_2
3	$a_3 = 125$	$\alpha_3 = \pi/2$	0	θ_3
4	0	$\alpha_4 = -\pi/2$	$d_4 = 850$	θ_4
5	0	$\alpha_5 = \pi/2$	0	θ_5
6	0	0	$d_6 = 100$	θ_6

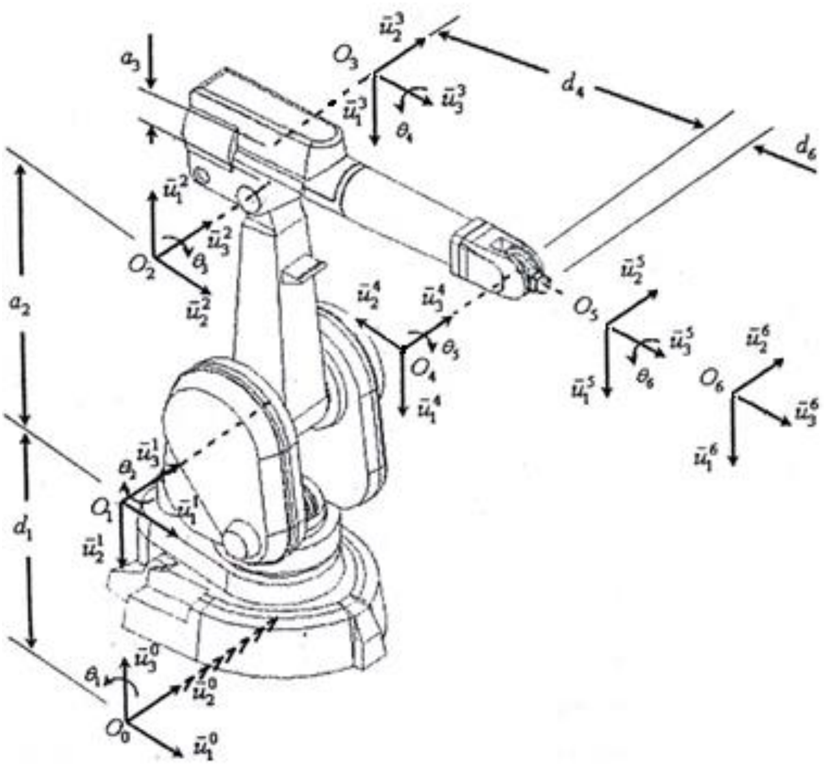


Figure 2-2: Reference frames of ABB IRB2000 Manipulator.

The orientation matrices between each two consecutive links can be found using (2.1) as:

$$\hat{C}^{(0,1)} = e^{\tilde{u}_3 \theta_1} e^{-\tilde{u}_1 \frac{\pi}{2}} \tag{2.3}$$

$$\hat{C}^{(1,2)} = e^{\tilde{u}_3 \theta_2} \tag{2.4}$$

$$\hat{C}^{(2,3)} = e^{\tilde{u}_3 \theta_3} e^{\tilde{u}_1 \frac{\pi}{2}} \tag{2.5}$$

$$\hat{C}^{(3,4)} = e^{\tilde{u}_3 \theta_4} e^{-\tilde{u}_1 \frac{\pi}{2}} \tag{2.6}$$

$$\hat{C}^{(4,5)} = e^{\tilde{u}_3 \theta_5} e^{\tilde{u}_1 \frac{\pi}{2}} \tag{2.7}$$

$$\hat{C}^{(5,6)} = e^{\tilde{u}_3 \theta_6} \tag{2.8}$$

The location of each link's origin with respect to the previous link's origin is calculated as follows:

$$\bar{\mathrm{r}}_{0,1}{}^{(0)} = \mathrm{d}_1\overline{\mathrm{u}_3} \tag{2.9}$$

$$\bar{\mathrm{r}}_{1,2}{}^{(1)} = \mathrm{a}_2 \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_2}\overline{\mathrm{u}_1} \tag{2.10}$$

$$\bar{\mathrm{r}}_{2,3}{}^{(2)} = \mathrm{a}_3 \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_3} \mathrm{e}^{\tilde{\mathrm{u}}_1 \frac{\pi}{2}}\overline{\mathrm{u}_1} \tag{2.11}$$

$$\bar{\mathrm{r}}_{3,4}{}^{(3)} = d_4\overline{\mathrm{u}_3} \tag{2.12}$$

$$\bar{\mathrm{r}}_{4,5}{}^{(4)} = \overline{0} \tag{2.13}$$

$$\bar{\mathrm{r}}_{5,6}{}^{(5)} = \mathrm{d}_6\overline{\mathrm{u}_3} \tag{2.14}$$

2.2 Position Analyses

2.2.1 Forward Position Analysis

The position and orientation of the end-effector is determined using joint angles. This is named as forward position analysis. This analysis is done symbolically. Found position and orientation elements are used in other kinematic analyses. In robotic applications, generally inverse kinematic analyses are used, because, generally the pose (position & orientation) of end-effector is known values but joint angles are unknown values. The orientation can be found first, because, part of the position is found using orientation.

Orientation of the end-effector is found by multiplying all rotation matrices, because, the lengths of the links and offsets cannot affect the orientation. The orientation matrix is then:

$$\hat{C}^{(0,6)} = \hat{C}^{(0,1)}\hat{C}^{(1,2)}\hat{C}^{(2,3)}\hat{C}^{(3,4)}\hat{C}^{(4,5)}\hat{C}^{(5,6)}$$

$$\begin{aligned}
\hat{C}^{(0,6)} &= (e^{\tilde{u}_3\theta_1}e^{-\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_3\theta_2})(e^{\tilde{u}_3\theta_3}e^{\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_3\theta_4}e^{-\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_3\theta_6}) \\
&= (e^{\tilde{u}_3\theta_1})(e^{-\tilde{u}_1\frac{\pi}{2}}e^{\tilde{u}_3(\theta_2+\theta_3)}e^{\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_3\theta_4})(e^{-\tilde{u}_1\frac{\pi}{2}}e^{\tilde{u}_3\theta_5}e^{\tilde{u}_1\frac{\pi}{2}})(e^{\tilde{u}_3\theta_6}) \\
&= (e^{\tilde{u}_3\theta_1})(e^{\tilde{u}_2\theta_{23}})(e^{\tilde{u}_3\theta_4})(e^{\tilde{u}_2\theta_5})(e^{\tilde{u}_3\theta_6})
\end{aligned}$$

Then,

$$\hat{C}^{(0,6)} = e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}e^{\tilde{u}_3\theta_6} \qquad (2.15)$$

Where, $\theta_{23} = \theta_2 + \theta_3$. Also recall that, $e^{-\tilde{u}_1\frac{\pi}{2}}e^{\tilde{u}_3\theta}e^{\tilde{u}_1\frac{\pi}{2}} = e^{\tilde{u}_2\theta}$

Using (2.2), the equation of the tip point position relative to the base is:

$$\vec{r}_{0,6} = \vec{r}_{0,1} + \vec{r}_{1,2} + \vec{r}_{2,3} + \vec{r}_{3,4} + \vec{r}_{4,5} + \vec{r}_{5,6}$$

$$\begin{aligned}
\bar{r} = \bar{r}_{0,6}{}^{(0)} = \bar{r}_{0,1}{}^{(0)} + \hat{C}^{(0,1)}\bar{r}_{1,2}{}^{(1)} + \hat{C}^{(0,2)}\bar{r}_{2,3}{}^{(2)} + \hat{C}^{(0,3)}\bar{r}_{3,4}{}^{(3)} + \hat{C}^{(0,4)}\bar{r}_{4,5}{}^{(4)} \\
+ \hat{C}^{(0,5)}\bar{r}_{5,6}{}^{(5)}
\end{aligned}$$

Substituting from (2.3)-(2.14),

$$\begin{aligned}
\bar{r} = d_1\overline{u_3} + e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_2}a_2\overline{u_1} - e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}a_2\overline{u_1} + e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}d_4\overline{u_3} + \\
e^{\tilde{u}_3\theta_1}e^{\tilde{u}_2\theta_{23}}e^{\tilde{u}_3\theta_4}e^{\tilde{u}_2\theta_5}d_6\overline{u_3} \qquad (2.16)
\end{aligned}$$

2.2.2 Inverse Position Analysis

Inverse position analysis is to find joint angles from given pose of the end-effector. First we must determine rotation & translation matrices with given position and orientation.

After the matrices are formed, using the detailed expressions of the elements, we can find the joint variables of the robot. The elements in the position matrix are independent, but in rotation matrix, only 3 of 9 elements are independent. This means, there are 6 independent equation for 6 unknown joint variables.

The first thing to do is to convert tip point position to wrist point position. This can be done by subtracting $\vec{r}_{5,6}$ from $\vec{r}_{0,6}$:

$$\bar{w} = \bar{r} - \mathrm{d}_6 \hat{\mathrm{C}}^{(0,6)} \overline{\mathrm{u}_3} = \mathrm{d}_1 \overline{\mathrm{u}_3} + \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_1} \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_2} a_2 \overline{\mathrm{u}_1} - \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_1} \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} a_2 \overline{\mathrm{u}_1} + \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_1} \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} d_4 \overline{\mathrm{u}_3}$$

Let $\bar{w}^* = \bar{w} - \mathrm{d}_1 \overline{\mathrm{u}_3} = \mathrm{e}^{\tilde{\mathrm{u}}_3 \theta_1} \left(\mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_2} a_2 \overline{\mathrm{u}_1} - \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} a_2 \overline{\mathrm{u}_1} + \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} d_4 \overline{\mathrm{u}_3} \right)$ (2.17)

θ_1 can be found by Pre-multiplying (2.17) by $\overline{\mathrm{u}_2}^t \mathrm{e}^{-\tilde{\mathrm{u}}_3 \theta_1}$, as explained below:

First, pre-multiply (2.17) by $\mathrm{e}^{-\tilde{\mathrm{u}}_3 \theta_1}$:

$$\mathrm{e}^{-\tilde{\mathrm{u}}_3 \theta_1} \bar{w}^* = \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_2} a_2 \overline{\mathrm{u}_1} - \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} a_2 \overline{\mathrm{u}_1} + \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} d_4 \overline{\mathrm{u}_3} \qquad (2.18)$$

Next, post-multiply by (2.18) by $\overline{\mathrm{u}_2}^t$:

$$\overline{\mathrm{u}_2}^t \mathrm{e}^{-\tilde{\mathrm{u}}_3 \theta_1} \bar{w}^* = w_3 = \overline{\mathrm{u}_2}^t \left(\mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_2} a_2 \overline{\mathrm{u}_1} - \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} a_2 \overline{\mathrm{u}_1} + \mathrm{e}^{\tilde{\mathrm{u}}_2 \theta_{23}} d_4 \overline{\mathrm{u}_3} \right) = 0$$

On the other hand,

$$\mathrm{e}^{-\tilde{\mathrm{u}}_3\theta_1}\overline{w}^* = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 - d_3 \end{bmatrix} = \begin{bmatrix} w_1\cos\theta_1 + w_2\sin\theta_1 \\ w_2\cos\theta_1 - w_1\sin\theta_1 \\ w_3 - d_3 \end{bmatrix}$$

Then, $w_2\cos\theta_1 - w_1\sin\theta_1 = 0$

Consequently,

$$\theta_1 = atan2(\sigma_1 w_2, \sigma_1 w_1), where\ \sigma_1 = \pm 1 \tag{2.19}$$

In this thesis σ_1 will always be taken as +1. The reason is that the parts to be processed will be positioned in the global coordinate system such that $\theta_1\ \epsilon\ [-90^o, 90^0]$.

Now θ_2 can be found from (2.18)

$$\mathrm{e}^{-\tilde{\mathrm{u}}_3\theta_1}\overline{w}^* = \mathrm{e}^{\tilde{\mathrm{u}}_2\theta_2} a_2\overline{\mathrm{u}_1} - \mathrm{e}^{\tilde{\mathrm{u}}_2\theta_{23}} a_3\overline{\mathrm{u}_1} + \mathrm{e}^{\tilde{\mathrm{u}}_2\theta_{23}} d_4\overline{\mathrm{u}_3}$$

$$= a_2\begin{bmatrix} \cos\theta_2 \\ 0 \\ -\sin\theta_2 \end{bmatrix} - a_3\begin{bmatrix} \sin\theta_{23} \\ 0 \\ \cos\theta_{23} \end{bmatrix} + d_4\begin{bmatrix} \cos\theta_{23} \\ 0 \\ -\sin\theta_{23} \end{bmatrix}$$

So, the first and third components of $\mathrm{e}^{-\tilde{\mathrm{u}}_3\theta_1}\overline{w}^*$ are:

$$w_1\cos\theta_1 + w_2\sin\theta_1 = a_2\cos\theta_2 - a_3\sin\theta_{23} + d_4\cos\theta_{23} \tag{2.20}$$

$$w_3 - d_1 = -a_2\sin\theta_2 + a_3\cos\theta_{23} + d_4\sin\theta_{23} \tag{2.21}$$

Multiply both (2.20), (2.21) and add them:

$$\begin{aligned}(w_1\cos\theta_1 + w_2\sin\theta_1)^2 + (w_3 - d_1)^2 \\ &= a_2^2 + a_3^2 + d_4^2 + 2a_2d_4(\cos\theta_2\sin\theta_{23} - \sin\theta_2\cos\theta_{23}) \\ &- 2a_2a_3(\cos\theta_2\cos\theta_{23} + \sin\theta_2\sin\theta_{23}) \qquad (2.22)\end{aligned}$$

Utilizing the trigonometric identities, $\cos\alpha\cos\beta + \sin\alpha\sin\beta = \cos(\alpha - \beta)$ and $\sin\alpha\cos\beta + \cos\alpha\sin\beta = \sin(\alpha + \beta)$, (2.22) can be simplified into:

$$\begin{aligned}(w_1\cos\theta_1 + w_2\sin\theta_1)^2 + (w_3 - d_1)^2 \\ &= a_2^2 + a_3^2 + d_4^2 + 2a_2d_4\sin\theta_3 - 2a_2a_3\cos\theta_3 \qquad (2.23)\end{aligned}$$

Dividing (2.23) by $2a_2\sqrt{a_3^2 + d_4^2}$ and rearranging,

$$\begin{aligned}\frac{(w_1\cos\theta_1 + w_2\sin\theta_1)^2 + (w_3 - d_1)^2 - a_2^2 - a_3^2 - d_4^2}{2a_2\sqrt{a_3^2 + d_4^2}} \\ &= \frac{d_4}{2a_2\sqrt{a_3^2 + d_4^2}}\sin\theta_3 - \frac{a_3}{2a_2\sqrt{a_3^2 + d_4^2}}\cos\theta_3 \qquad (2.24)\end{aligned}$$

Let $\cos\gamma = \frac{d_4}{2a_2\sqrt{a_3^2+d_4^2}}$ and $\sin\gamma = -\frac{a_3}{2a_2\sqrt{a_3^2+d_4^2}}$

Then (2.24) can be rewritten as:

$$\sin(\theta_3 + \gamma) = \frac{(w_1\cos\theta_1 + w_2\sin\theta_1)^2 + (w_3 - d_1)^2 - a_2^2 - a_3^2 - d_4^2}{2a_2\sqrt{a_3^2 + d_4^2}}$$

So,

$$\theta_3 = \sin^{-1}\left(\frac{(w_1\cos\theta_1 + w_2\sin\theta_1)^2 + (w_3 - d_1)^2 - a_2^2 - a_3^2 - d_4^2}{2a_2\sqrt{a_3^2 + d_4^2}}\right) - \gamma \qquad (2.25)$$

Where, $\gamma = \text{atan}_2\left(-\frac{a_3}{2a_2\sqrt{a_3^2+d_4^2}},\ \frac{d_4}{2a_2\sqrt{a_3^2+d_4^2}}\right)$

Note that (2.25) gives two possible solutions for θ_3; however, since only an upper arm configuration is needed throughout the motion the solution that gives θ_3 in the second or third quadrant is taken. That is, $\theta_3 \in [90°, 270°]$.

Finding can be done by going back to (2.20) and (2.21). Rewriting them in matrix format,

$$\begin{bmatrix} a_2 + d_4\sin\theta_3 - a_3\cos\theta_3 & d_4\cos\theta_3 + a_3\sin\theta_3 \\ d_4\cos\theta_3 + a_3\sin\theta_3 & -a_2 - d_4\sin\theta_3 + a_3\cos\theta_3 \end{bmatrix}\begin{bmatrix}\cos\theta_2 \\ \sin\theta_2\end{bmatrix} = \begin{bmatrix} w_1\cos\theta_1 + w_2\sin\theta_1 \\ w_3 - d_1 \end{bmatrix}$$

$$\begin{bmatrix}\cos\theta_2 \\ \sin\theta_2\end{bmatrix}$$
$$= \begin{bmatrix} a_2 + d_4\sin\theta_3 - a_3\cos\theta_3 & d_4\cos\theta_3 + a_3\sin\theta_3 \\ d_4\cos\theta_3 + a_3\sin\theta_3 & -a_2 - d_4\sin\theta_3 + a_3\cos\theta_3 \end{bmatrix}^{-1}\begin{bmatrix} w_1\cos\theta_1 + w_2\sin\theta_1 \\ w_3 - d_1 \end{bmatrix}$$
$$= A\begin{bmatrix} -a_2 - d_4\sin\theta_3 + a_3\cos\theta_3 & -d_4\cos\theta_3 - a_3\sin\theta_3 \\ -d_4\cos\theta_3 - a_3\sin\theta_3 & a_2 + d_4\sin\theta_3 - a_3\cos\theta_3 \end{bmatrix}\begin{bmatrix} w_1\cos\theta_1 + w_2\sin\theta_1 \\ w_3 - d_1 \end{bmatrix}$$

Where, $A = \frac{-1}{(a_2+d_4\sin\theta_3 - a_3\cos\theta_3)^2 + (d_4\cos\theta_3 + a_3\sin\theta_3)^2}$

Notice that the denominator of A can never be zero.

Then,

$$\cos\theta_2 = \frac{(a_2 + d_4 \sin\theta_3 - a_3\cos\theta_3)(w_1\cos\theta_1 + w_2\sin\theta_1) + (d_4\cos\theta_3 + a_3\sin\theta_3)(w_3 - d_1)}{(a_2 + d_4\sin\theta_3 - a_3\cos\theta_3)^2 + (d_4\cos\theta_3 + a_3\sin\theta_3)^2}$$

$$\sin\theta_2 = \frac{(d_4\cos\theta_3 + a_3\sin\theta_3)(w_1\cos\theta_1 + w_2\sin\theta_1) + (a_2 + d_4\sin\theta_3 - a_3\cos\theta_3)(w_3 - d_1)}{(a_2 + d_4\sin\theta_3 - a_3\cos\theta_3)^2 + (d_4\cos\theta_3 + a_3\sin\theta_3)^2}$$

So, θ_2 can be found as $\theta_2 = \mathrm{atan}_2(\sin\theta_2\,, \cos\theta_2\ )$ (2.26)

The remaining three joint angles will be found from the orientation of the end-effector. At this point θ_1, θ_2 and θ_3 are known; hence, the following operation can be performed on the orientation matrix $\hat{\mathrm{C}}^{(0,6)}$:

$$\hat{\mathrm{C}}^{(0,6)} = \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_1}\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_{23}}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}$$

Let $\hat{\mathrm{C}}^* = \mathrm{e}^{-\tilde{\mathrm{u}}_2\theta_{23}}\mathrm{e}^{-\tilde{\mathrm{u}}_3\theta_1}\hat{\mathrm{C}}^{(0,6)} = \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6} = \begin{bmatrix} c_{11}^* & c_{12}^* & c_{13}^* \\ c_{21}^* & c_{22}^* & c_{23}^* \\ c_{31}^* & c_{32}^* & c_{33}^* \end{bmatrix}$

To find θ_5 pre-multiply $\hat{\mathrm{C}}^*$ by $\overline{\mathrm{u}_3}^t$ and post-multiply by $\overline{\mathrm{u}_3}$:

$$\overline{\mathrm{u}_3}^t\hat{\mathrm{C}}^*\overline{\mathrm{u}_3} = \left(\overline{\mathrm{u}_3}^t\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\right)\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\left(\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_3}\right) = \overline{\mathrm{u}_3}^t\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\overline{\mathrm{u}_3} = \cos\theta_5$$

Then, $\theta_5 = \mathrm{atan}_2\left(\sigma_5\sqrt{1-\cos^2\theta_5}\,, \cos\theta_5\right) \quad where\ \sigma_5 = \pm 1$ (2.27)

Note that σ_5 points to a wrist flip ambiguity, meaning that the physical configuration of the end-effector in both cases are the same. Hence, σ_5 has no effect on the physical orientation of the end-effector. For the sake of completeness, it is to be taken as +1 in this work.

Similarly, θ_6 can be found as follows:

$$\overline{\mathrm{u}_3}^t \hat{\mathrm{C}}^* \overline{\mathrm{u}_1} = \left(\overline{\mathrm{u}_3}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\right)\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\left(\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_1}\right) = \overline{\mathrm{u}_3}^t \mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_1} = -\sin\theta_5\cos\theta_6 = c^*_{31}$$

$$\overline{\mathrm{u}_3}^t \hat{\mathrm{C}}^* \overline{\mathrm{u}_2} = \left(\overline{\mathrm{u}_3}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\right)\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\left(\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_2}\right) = \overline{\mathrm{u}_3}^t \mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_2} = \sin\theta_5\sin\theta_6 = c^*_{32}$$

So, $\theta_6 = \mathrm{atan}_2(\sigma_5 c^*_{32}, -\sigma_5 c^*_{31}) \quad , \sin\theta_5 \neq 0$ (2.28)

Same procedure to find θ_4:

$$\overline{\mathrm{u}_1}^t \hat{\mathrm{C}}^* \overline{\mathrm{u}_3} = \left(\overline{\mathrm{u}_1}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\right)\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\left(\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_3}\right) = \overline{\mathrm{u}_1}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\overline{\mathrm{u}_3} = \sin\theta_5\cos\theta_4 = c^*_{13}$$

$$\overline{\mathrm{u}_2}^t \hat{\mathrm{C}}^* \overline{\mathrm{u}_3} = \left(\overline{\mathrm{u}_2}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\right)\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\left(\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6}\overline{\mathrm{u}_3}\right) = \overline{\mathrm{u}_2}^t \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_2\theta_5}\overline{\mathrm{u}_3} = \sin\theta_5\sin\theta_4 = c^*_{23}$$

So, $\theta_5 = \mathrm{atan}_2(\sigma_5 c^*_{23}, \sigma_5 c^*_{13}) \quad , \sin\theta_5 \neq 0$ (2.29)

In case $\sin\theta_5 = 0$ this means $\theta_5 = 0 \; or \; \pm\pi$. However θ_5 cannot be $\pm\pi$ due to physical constraints. Therefore, one scenario is possible, where $\hat{\mathrm{C}}^*$ becomes:

$$\hat{\mathrm{C}}^* = \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_2 0}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6} = \mathrm{e}^{\tilde{\mathrm{u}}_3\theta_4}\mathrm{e}^{\tilde{\mathrm{u}}_3\theta_6} = \mathrm{e}^{\tilde{\mathrm{u}}_3(\theta_4+\theta_6)} \qquad (2.30)$$

(2.30) means that at this singularity it is not possible to find θ_4 and θ_6 separately, but only their sum. Fortunately, this will not affect the collision check test, as will be further explained in the next chapter. Nonetheless, for $\sin\theta_5 = 0$, $\theta_4 + \theta_6$ are found as shown below:

$$\hat{\mathrm{C}}^* = \mathrm{e}^{\tilde{\mathrm{u}}_3(\theta_4+\theta_6)} = \begin{bmatrix} \cos(\theta_4+\theta_6) & -\sin(\theta_4+\theta_6) & 0 \\ \sin(\theta_4+\theta_6) & \cos(\theta_4+\theta_6) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{11}^* & c_{12}^* & c_{13}^* \\ c_{21}^* & c_{22}^* & c_{23}^* \\ c_{31}^* & c_{32}^* & c_{33}^* \end{bmatrix}$$

Then, $\theta_4 + \theta_6 = \mathrm{atan}_2(c_{21}^*, c_{11}^*)$ (2.31)

CHAPTER 3

PATH GENERATION

This chapter is the heart of the thesis. Here, the overall path of the end-effector is going to be generated in order to be later executed by the robotic manipulator. Starting with the preparation stage, the edges needed to be scanned or deburred are selected. Next, for each selected edge the corresponding needed motion of the end-effector in order to process that specific edge is generated. Such motions will be referred to as processing motions. Then, all possible combination of motions that connect separated processing motions are calculated and generated. Such motions will be referred to as non-processing motions. After that, a search algorithm is utilized to select a subset of the non-processing motions that yield an efficient connected path along with the processing motions. Lastly, this overall path will be translated into motion commands and sent to ABB IRB2000

3.1 Preparation Stage

This section discusses the parameters that the user need to input to the program. These parameters will serve as the foundation or basis that the software will use to generate the overall motion of the robot.

3.1.1 Setting the Global Coordinate System

The user is expected to define a datum reference frame. The frame is to be an orthogonal, right-handed and isoscaled “equally scaled in the coordinate axes” coordinate system. All the paths, edges and points will be expressed in this coordinate system throughout the study. Such reference coordinate system can be easily defined using the reference geometry tool provided by the software, which can be reached from “Features/Reference Geometry/Reference Coordinate System” as shown in Figure 3-1. Note that these defined coordinates corresponds to a given position on the working space.

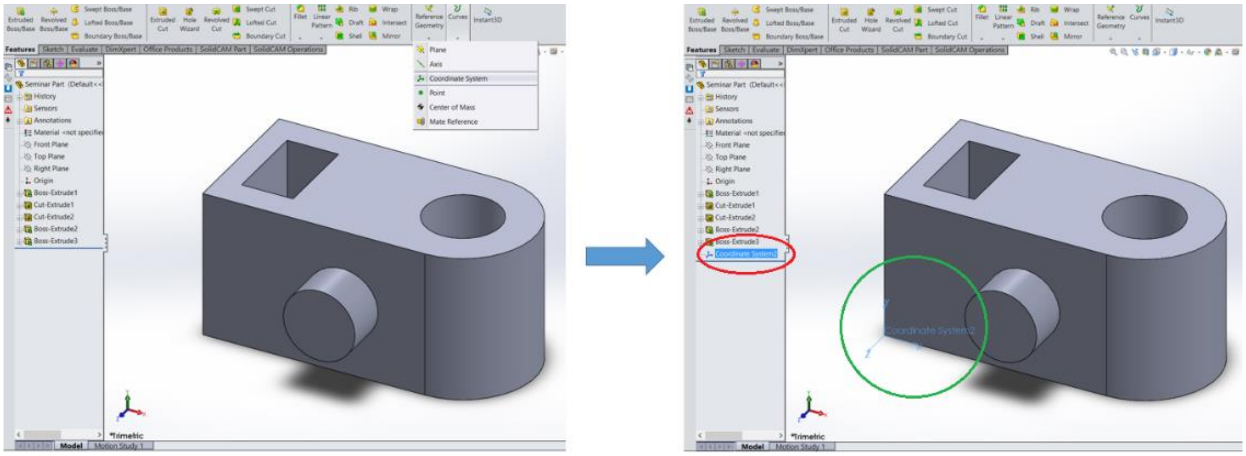


Figure 3-1: Defining the location and orientation of the part with respect to the global coordinate system.

The necessary transformations can be performed using the reference frame (RF) to correctly position the workpiece with respect to the global coordinates of the robot.

Viewing RF from the global coordinates as a surface represented by its xy-plane with a normal parallel to z-axis, the following reasoning can be followed to easily create the needed rotation matrix.

First, replace the x-axis components of RF in the first column, which result in rotating the axis to the global x-axis. Then, to make the x’y’-plane lie on the global xy-plane, the unit vector perpendicular to the x’y’-plane is to be oriented towards the global z-axis. This means that the components of z’-axis should be placed in the third column of the matrix. Finally, in order to assure that all axes are perpendicular to each other the third axis is the result of the cross product of z’-axis and x’-axis “in the mentioned order”, resulting in the y’-axis to be placed in the second column of the rotation matrix. This procedure is illustrated in the following formulae:

$$x' = x_1' \, \hat{\imath} + x_2'\hat{\jmath} + x_3'\hat{k} \tag{3.1}$$

$$y' = y_1' \, \hat{\imath} + y_2'\hat{\jmath} + y_3'\hat{k} \tag{3.2}$$

$$z' = z_1' \, \hat{\imath} + z_2'\hat{\jmath} + z_3'\hat{k} \tag{3.3}$$

$$R = \begin{bmatrix} x_1' & y_1' & z_1' \\ x_2' & y_2' & z_2' \\ x_3' & y_3' & z_3' \end{bmatrix} \tag{3.4}$$

Where x’, y’ and z’ are the axes of RF viewed from the global coordinates and R is the rotation matrix.

Such rotation matrix R can be converted or represented in three basic rotations. Here these rotations are taken about the three global axes in the sequence z-y-x. Note that such sequence is arbitrarily chosen and other sequences can be used to end up with the same overall rotation. Therefore, the multiplication of the rotation matrices will be equal to R as shown in equation (3.5).

$$R = \begin{bmatrix} x_1' & y_1' & z_1' \\ x_2' & y_2' & z_2' \\ x_3' & y_3' & z_3' \end{bmatrix} = R_z R_y R_x$$

$$= \begin{bmatrix} \cos\emptyset & \sin\emptyset & 0 \\ -\sin\emptyset & \cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} \cos\phi\cos\theta & \sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi \\ -\sin\phi\cos\theta & \cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi \\ \sin\theta & -\sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix}$$

Where R_z, R_y and R_x are the rotation matrices about z, y and x axes, respectively.

From the above equation, the three rotation angles can be found as follows,

$$x_3' = \sin\theta$$

$$\theta = \arcsin x_3' \tag{3.5}$$

Note that equation (3.5) will yield two possible solutions for θ. Both are valid however, once an angle is chosen to be the correct angle it will affect the other two angles. Here the angle between [-90°, 90°] is chosen. Next,

$$\sin\phi = \frac{-x_2'}{\cos\theta} \quad ,\cos\theta \neq 0$$

$$\cos\phi = \frac{x_1'}{\cos\theta} \quad ,\cos\theta \neq 0$$

$$\phi = atan_2(\sin\phi, \cos\phi)$$

Similarly, for ψ,

$$\sin\psi = \frac{-y'_3}{\cos\theta} \quad ,\cos\theta \neq 0$$

$$\cos\psi = \frac{z'_3}{\cos\theta} \quad ,\cos\theta \neq 0$$

$$\psi = atan_2(\sin\psi, \cos\psi)$$

In case $\cos\theta$=0 the solution discussed above is not valid and hence this case must be treated specially. However, in such case θ is directly found and is equal to either -90^o or 90^o, and from (3.5) this ambiguity is eliminated. That is, if x'$_3$ is 1 then θ is 90^o and -90^o if x'$_3$ is -1. Physically, this case means that the x'-axis is parallel to z-axis and by rotating around y-axis by either 90^o or -90^o x'-axis will be brought to x-axis. From the last sentence, it can be deduced that after rotating about y-axis, rotating about x-axis in a specific angle will align RF to the global coordinates without the need to rotate around z-axis. That is ψ = 0^o. Note that this case is actually a singularity point.

$$R = \begin{bmatrix} x_1' & y_1' & z_1' \\ x_2' & y_2' & z_2' \\ x_3' & y_3' & z_3' \end{bmatrix} = R_z R_y R_x$$

$$= \begin{bmatrix} \cos 0 & \sin 0 & 0 \\ -\sin 0 & \cos 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta\sin\psi & -\sin\theta\cos\psi \\ 0 & \cos\psi & \sin\psi \\ \sin\theta & -\sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta & \sin\theta\sin\psi & -\sin\theta\cos\psi \\ 0 & \cos\psi & \sin\psi \\ \sin\theta & -\sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix} \tag{3.6}$$

So, ψ can be determined from (3.6),

$$\sin\psi = z_2'$$

$$\cos\psi = y_2'$$

$$\psi = atan_2(\sin\psi, \cos\psi)$$

Lastly, after rotating the part by these angles, It will be translated linearly in each axis to bring the origin of RF to the (0, 0, 0) point of the global coordinates by the simple following equations,

$O = (O_x, O_y, O_z)$

$\Delta x = -O_x$

$\Delta y = -O_y$

$\Delta z = -O_z$

Where *O* is the RF origin expressed from the global frame point of view and Δx, Δy and Δz are the linear translation along the x, y and z axes, respectively. The relative distance of the RF with respect to the global coordinate system, *O*, is entered by the user through the graphical user interface (GUI) of the software. A complete explanation about the GUI is presented in chapter 4.

3.1.2 Selecting the Edges to be Processed

Before starting with path generation, first the edges on the part that are to be processed (scanned or finished) is to be selected. The user need to first define the planar faces that will be in direct contact with the working table or fixture, such faces will be referred to by Ground Faces. Figure 3-2 below shows an example of a selected ground face.

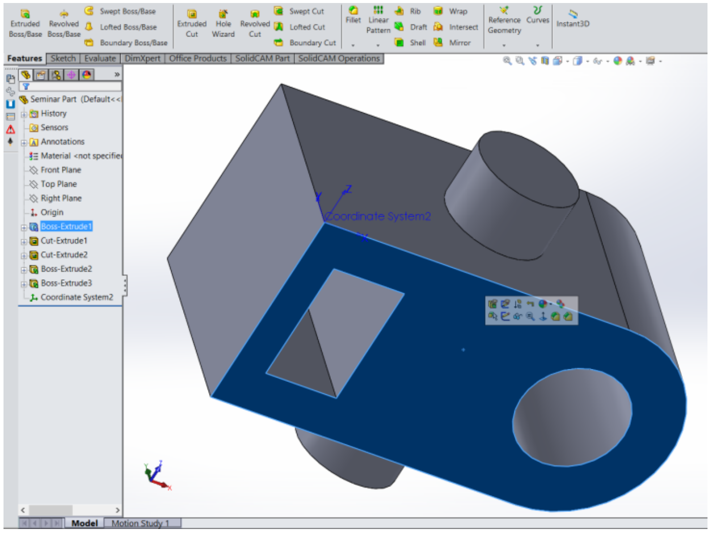


Figure 3-2: Selecting a ground face by the user

In edge finishing and scanning it is the convex edges that needs to be processed. Using Power Select Utility in SolidWorks, a filter can be set in order to select only convex edges out of all the edges in a part, this is done automatically by the software. After selecting the convex edges, the ones laying on the ground faces are recognized and eliminated. The reason for it is that the tool will not be able to reach such edges due to the physical constraints imposed by the working table.

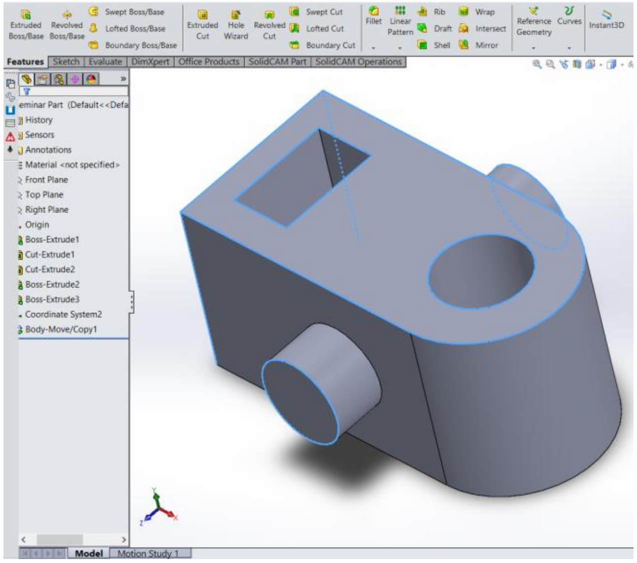


Figure 3-3: Result of the automated convex edge selection

3.2 Processing Motion Generation

As discussed before, the overall motion of the robot consists of two parts, namely, processing and non-processing motions. Each processing motion corresponds to one edge on the part; therefore, these motions are fixed, as will be seen in this subsection. For that reason, each of these motions needs to be checked to be collision-free.

An edge is said to be reachable by a robotic manipulator, if the robot and the tool holder do not collide with the part throughout the needed motion to process that edge. Checking the reachability of an edge can be done in two steps. First, calculate the processing motion for the selected edge. Secondly, simulate the calculated motion in SolidWorks and check if there are any collisions. In case no collisions occurs then that edge will be marked as reachable, otherwise it will be discarded from the process.

The robot manipulator's links are modeled as cuboids as shown in Figure 3-4. Such model is conservative, which can accommodate for some deviations. Also, it is faster to maneuver cuboids in SolidWorks which will require fewer memory and time compared to using an actual geometric model.

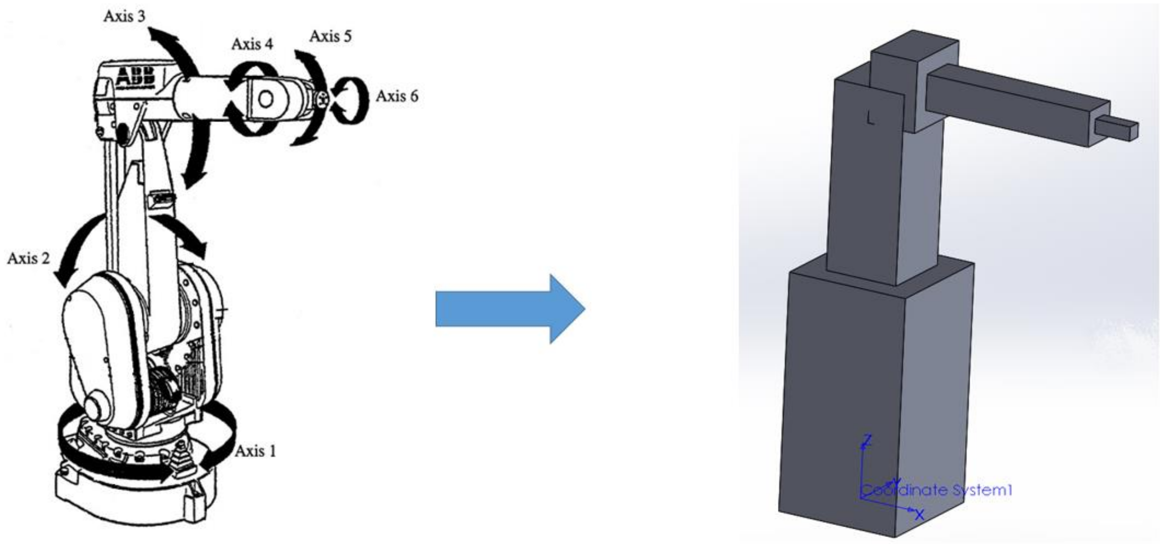


Figure 3-4: the geometric model of the robotic arm used in the study for performing the collision detection test.

The following sub-sections describes the detailed procedure for generating the processing motions that corresponds to each of the previously selected edges.

3.2.1 Calculating Processing Motions

Two types of edges are considered in this study, namely, Straight line and Circular edges.

In case of a straight line, the motion can be sufficiently defined by the start and end positions of the end-effector. These positions can be found by taking the desired offset value and direction (approach angle) from both ending vertices of that edge. The offset value is the relative distance between the edge and end-effector during the processing motion, such parameter is numerically entered by the user. While the offset direction can be chosen as either the average of the adjacent faces' normals, check Figure 3-5 a), or normal to the direction of the formed burr, Figure 3-5 b). Generally, the burr forms in the direction normal to both the normal vector of the later machined face between the two adjacent faces and the direction of the edge itself [19], as shown in Figure 3-6.

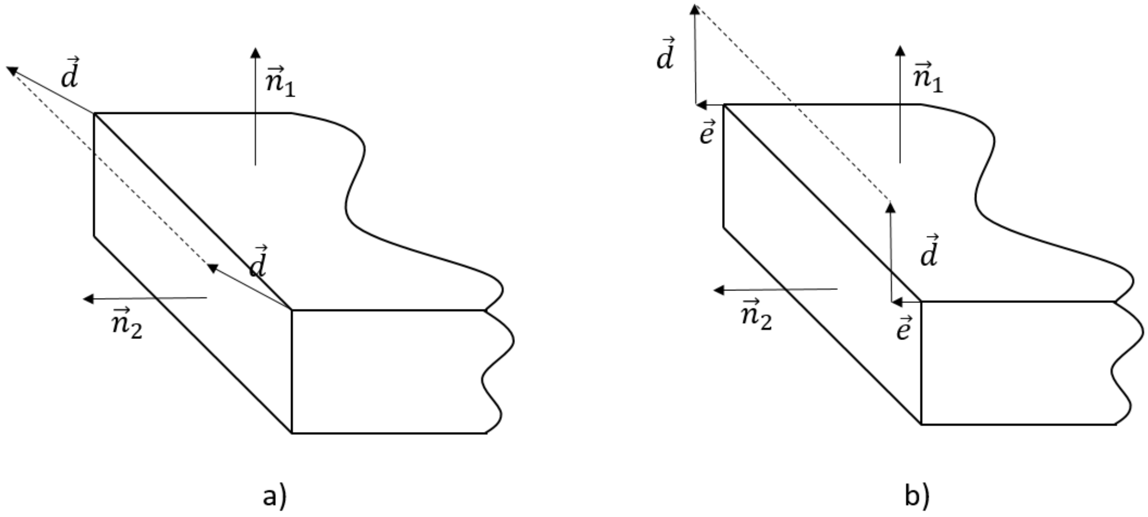
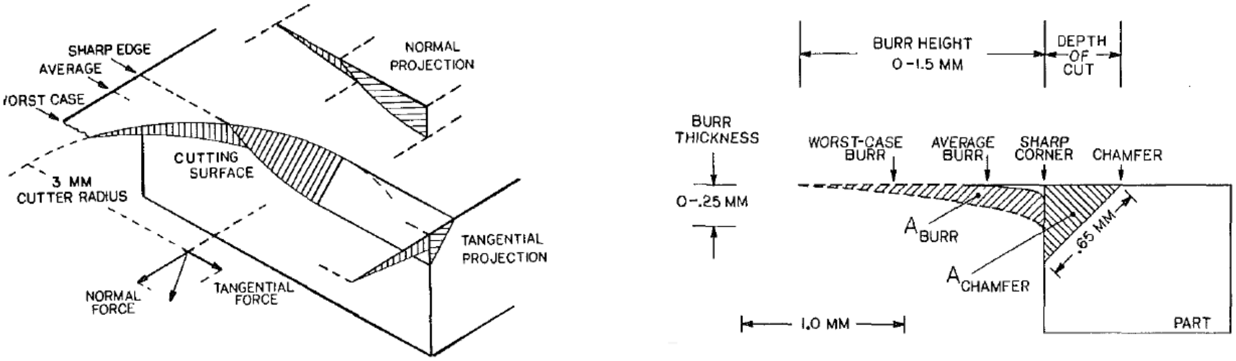


Figure 3-5: a) Illustration of an offset direction, $\vec{d}$, that is equal to the average of the two normals of the adjacent faces of the edge, $\vec{n}_1$ and $\vec{n}_2$ b) Illustration of an offset direction, $\vec{d}$, in the normal direction to the formed burr.



a) Assymetric View of a Formed Burr

b) Tangential Edge Projection of a Formed Burr

Figure 3-6: Illustration of formed burrs direction [19].

Mathematically, $\vec{d}$ can be found for the two case as follows:

- For the average direction case,

$$\vec{d} = d\frac{\vec{n}_1 + \vec{n}_2}{\|\vec{n}_1 + \vec{n}_2\|} \tag{3.7}$$

Where, d is the offset value.

- Normal to the burr formation direction,

$$\vec{d} = d\,\vec{n}_1 \tag{3.8}$$

Where, d is the offset value and $\vec{n}_1$ is the normal of the face machined last.

Note that in the second case a small offset $\vec{e}$ is introduced in order to position the tool tip on the burr itself instead of the edge attached to it, the value and direction is calculated as shown below,

$$\vec{e} = \begin{cases} e\frac{\vec{n}_1 x\vec{t}_1}{\|\vec{n}_1 x\vec{t}_1\|} & , \quad \left(\frac{\vec{n}_1 x\vec{t}_1}{\|\vec{n}_1 x\vec{t}_1\|}\right).\vec{n}_2 > 0 \\ -e\frac{\vec{n}_1 x\vec{t}_1}{\|\vec{n}_1 x\vec{t}_1\|} & , \quad \left(\frac{\vec{n}_1 x\vec{t}_1}{\|\vec{n}_1 x\vec{t}_1\|}\right).\vec{n}_2 < 0 \end{cases} \tag{3.9}$$

Where, e is equal to half of the tool head's width, $\vec{t}_1$ is the direction of the edge in space and $\vec{n}_1$ is the normal of the face machined last. Note that $\vec{t}_1$ can have two opposite directions; therefore, the correct direction is the one that yield $\vec{e}$ which have an angle between [-90^0, 90^0] with $\vec{n}_2$.

Now, the position of end-effector is defined by its location in space, $\vec{r}$, and orientation, $C^{(0,6)}$. The location can be calculated for both starting and ending positions by simply adding the position vector of the corresponding vertex and the previously calculated offset vector, $\vec{d}$. See Figure 3-7.

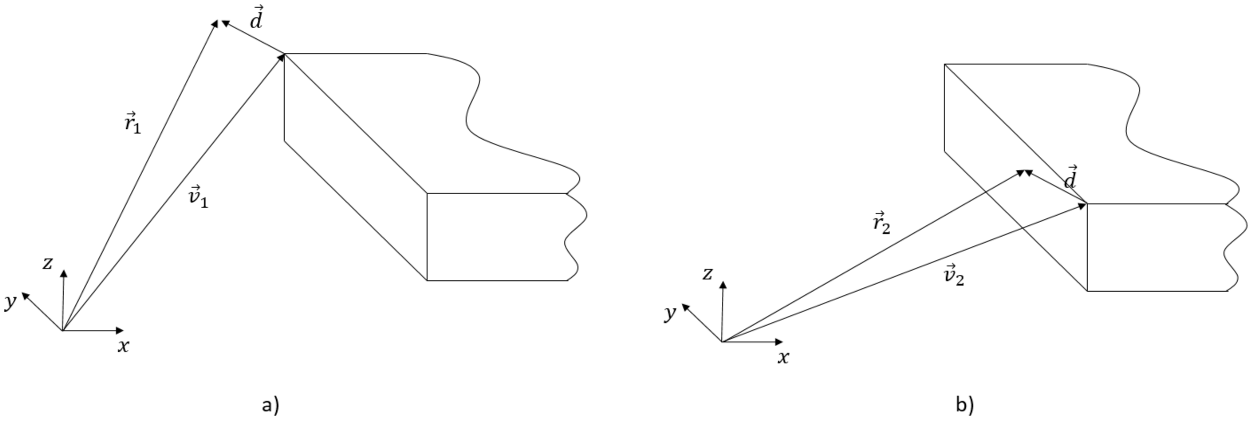


Figure 3-7: a) The location of the end-effector's starting position b) The location of the end-effector's ending position.

As for the tip point orientation, it is found as follows. The opposite direction of the offset vector, $\vec{d}$, is taken to be the z-axis of the end-effector. The y-axis is oriented such that it is parallel to the edge. The sign of the direction of y-axis is decided such that the x-axis will be pointing downwards with respect to the global coordinates, using the right-hand rule. Figure 3-8 gives a good example of such calculation.

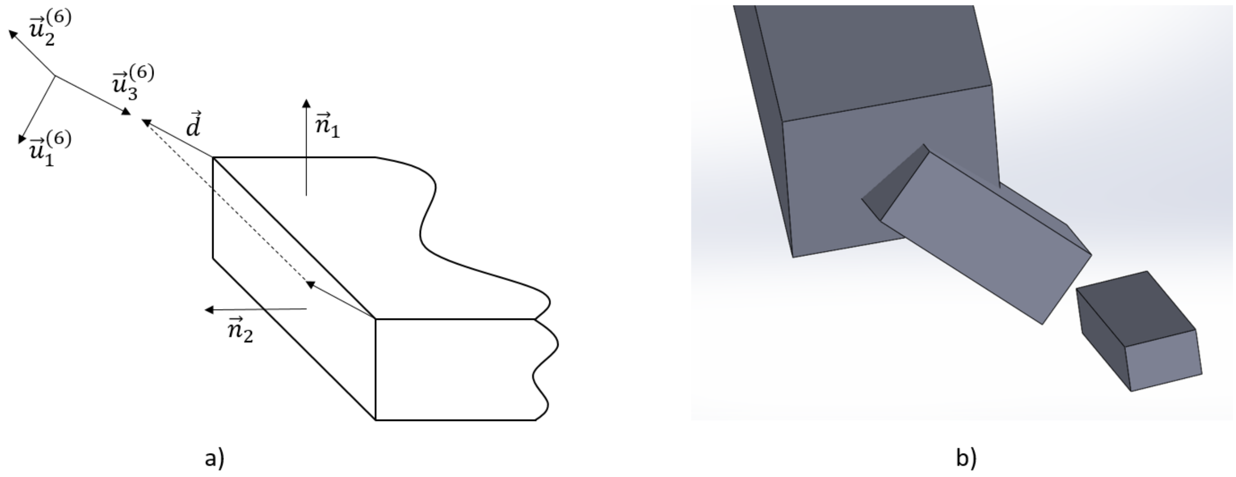


Figure 3-8: a) Taking the offset desired from a straight edge for generating the corresponding processing motion. d is the offset vector, n_1 and n_2 are the adjacent faces normals, the shown coordinates is the orientation of the end-effector of the ABB IRB2000 b) The CAD simulation of the corresponding motion in a).

In the case of a circular edges, the edge will either be a full circle or an arc. Since, ABB IRB2000 can create arc motions that are less than 180^{o} at a time, full circles are going to be divided into four equal arc segments. Partial arcs will be divided into two equal segments. Each arc segment motion needs to be represented by three positions of the end-effector, the start and end positions along with an intermediate position. This position is used by the robot to decide on the direction of circulation.

Similar to the straight edge scenario, the offset vector is added to the edge location to give the location of the tip-point of the robotic hand. As for the orientation, the z-axis will be in the opposite direction of the offset vector. The y-axis points towards the tangent at the given point. Note that y-axis can have two opposite direction, the one that gives an x-axis pointing downwards, according to the right-hand rule, is selected.

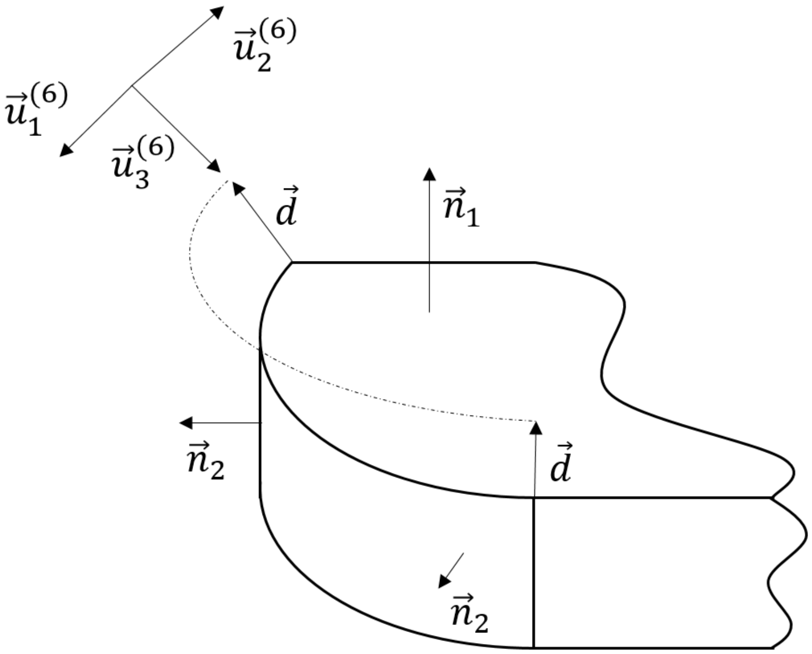


Figure 3-9: Taking the offset desired from a circular edge for generating the corresponding processing motion

Note that the angle between adjacent edges needs to be considered while calculating the processing motions. A small modification on the starting and/or ending positions of a processing motion is to be performed, in case the corresponding edge has an adjacent edge that forms an angle less that 180^o with it. For instance, the edge given in Figure 3-10 has two adjacent edges that both creates an angle of 90^o. Therefore, due to the physical thickness of the tool, both the starting and ending positions of the corresponding processing motion will be shifted by an amount equal to half the width of the tool in the direction of the edge, see Figure 3-10.

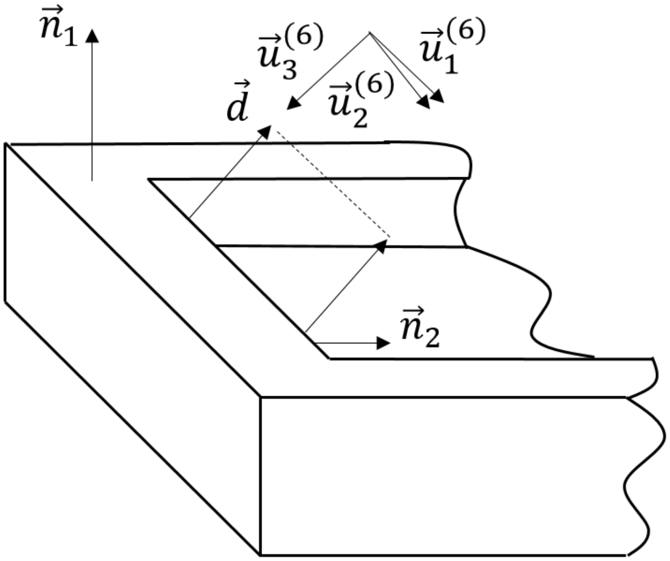


Figure 3-10: An example of a straight line edge that has adjacent edges that form angles less than 180° with it and how it affects the corresponding processing motion.

Similar approach is also considered for circular arc edges. The main difference from a straight edge is that the performed shift is an angle instead of a pure translation, see Figure 3-11. The angle α is found by $\alpha = \arctan\frac{w}{R}$. Where w is half the width of the tool and R is the radius of the arc motion.

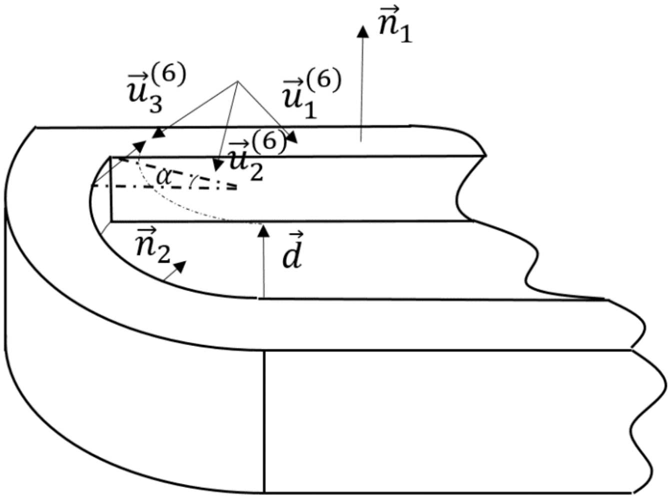


Figure 3-11: An example of a circular arc edge that has an adjacent edge that form an angle less than 180° with it and how it affects the corresponding processing motion.

This concludes the calculation procedure for each selected edge to find its corresponding processing motion. Resulting in one processing motion for each selected edge.

3.2.2 Collision Detection Test

In the previous subsection, a processing motion for each of the selected edges was calculated. Now, these motions need to be verified to be collision-free. Note that such motions are necessary to perform the task, either scanning or deburring. Hence, if any of these motions turns to have a collision it means that the corresponding edge cannot be processed. Therefore, such edges are going to be marked as unreachable.

Each of these motions are divided into discrete poses of the robotic manipulator. In other words, a discrete motion approach is adapted for the test of collisions. At any given posture the position of each link of the manipulator is going to be calculated, using inverse kinematics, and then check if any link collide with the part at the given posture. Such approach although not continuous in nature, it actually covers the whole range of motion as far as collision is concerned.

The procedure for discretizing the straight and circular motions are somewhat different. For the straight line motion, note that the orientation of the end-effector is actually constant. Hence all of the intermediate poses will also have the same orientation. The number of poses can be directly found using equation (3.11). Basically, the equation divides the straight line into equally spaced parts.

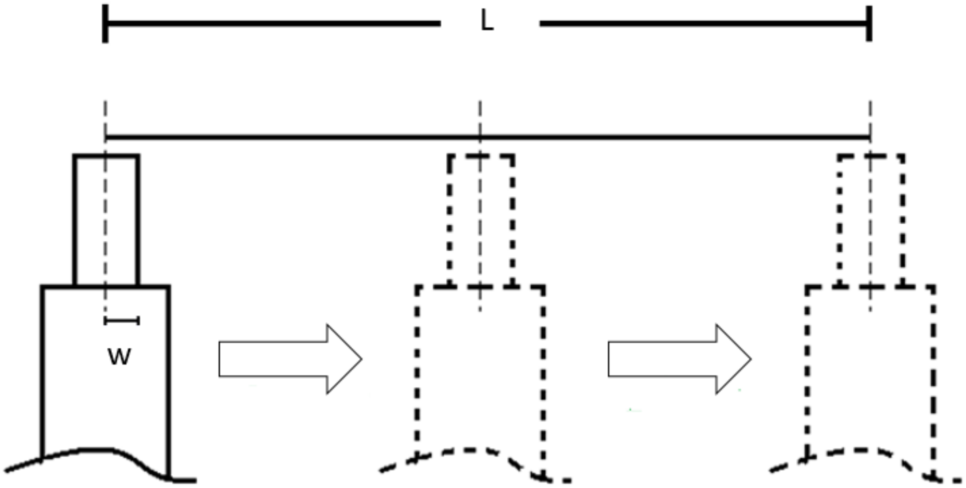


Figure 3-12: An illustration of discretizing the processing motion corresponding to a straight edge

$$\text{The number of poses needed} = 2 + \left\lceil \frac{L}{2w} - 1 \right\rceil \qquad (3.10)$$

Where L is the length of the straight motion and w is half the width of the tool holder.

In case of an arc edge, the swept angle of the edge is divided again into equal increment angles using equation (3.11). The orientation of each pose is found by rotating the orientation matrix of the previous pose along the axis of the arc edge by an angle α as given in (3.12).

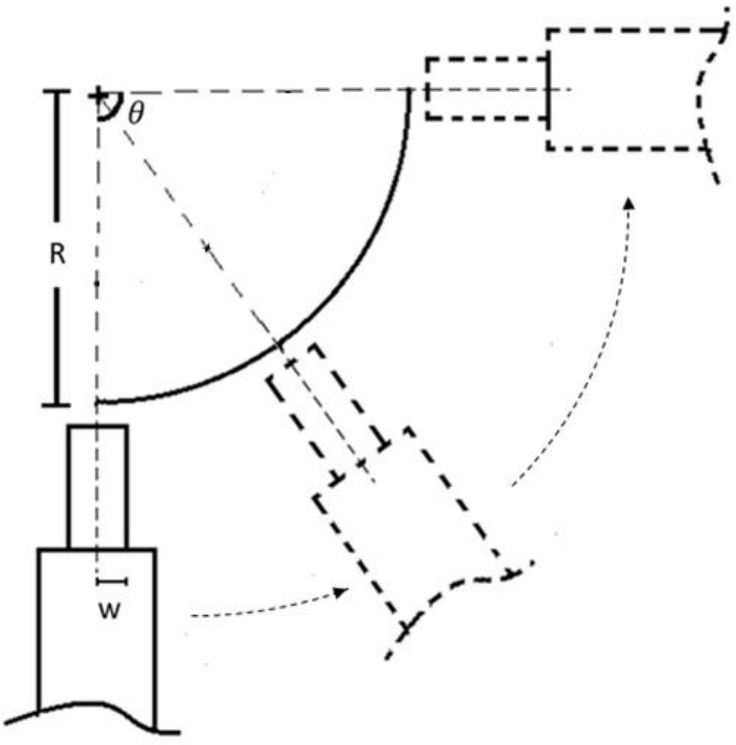


Figure 3-13: An illustration of discretizing the processing motion corresponding to an arc edge

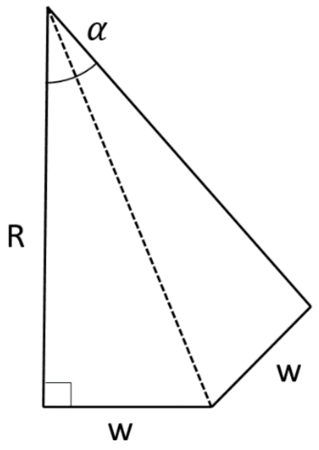


Figure 3-14: The trigonometry used to derive the formulas below.

$$\text{The number of poses needed} = 2 + \left\lceil \frac{\theta}{\alpha} \right\rceil \tag{3.11}$$

$$\text{Where } \alpha = 2\ tan^{-1}\frac{w}{R} \tag{3.12}$$

Where θ and R are the swept angle and radius of the arc motion, respectively, and w is half the width of the tool holder.

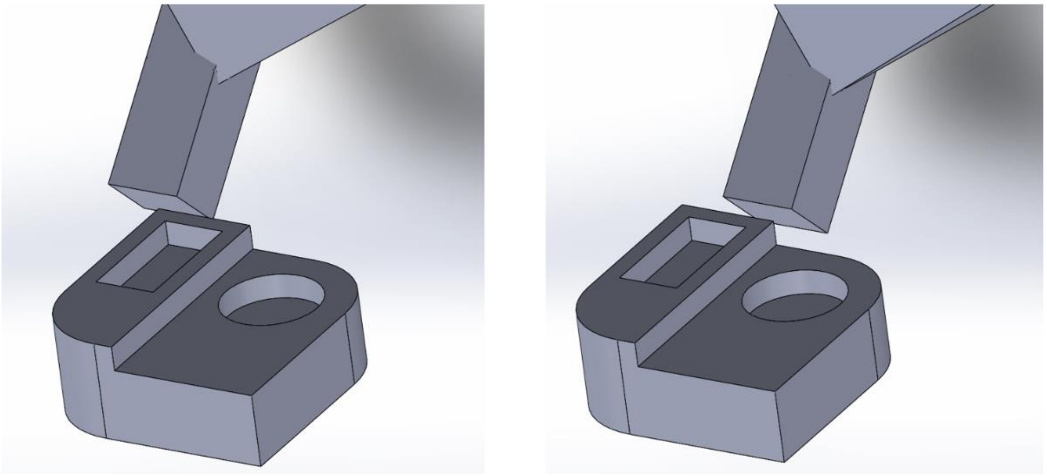


Figure 3-15: Example for a straight edge collosion detection test

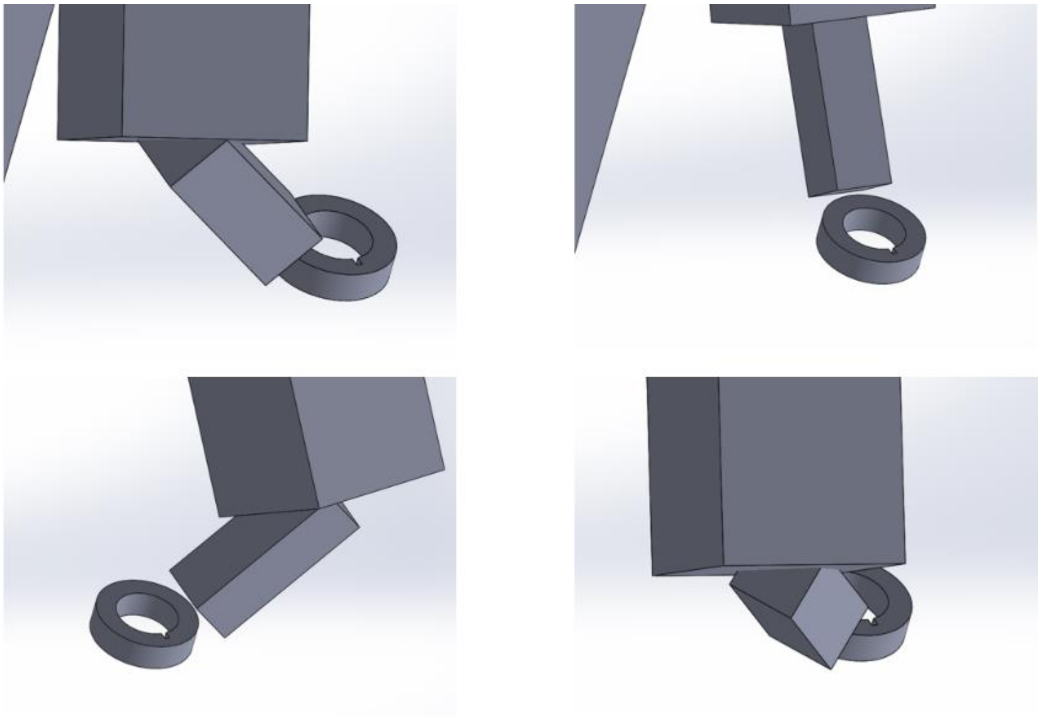


Figure 3-16: Different snap shots for an arc edge collosion detection test

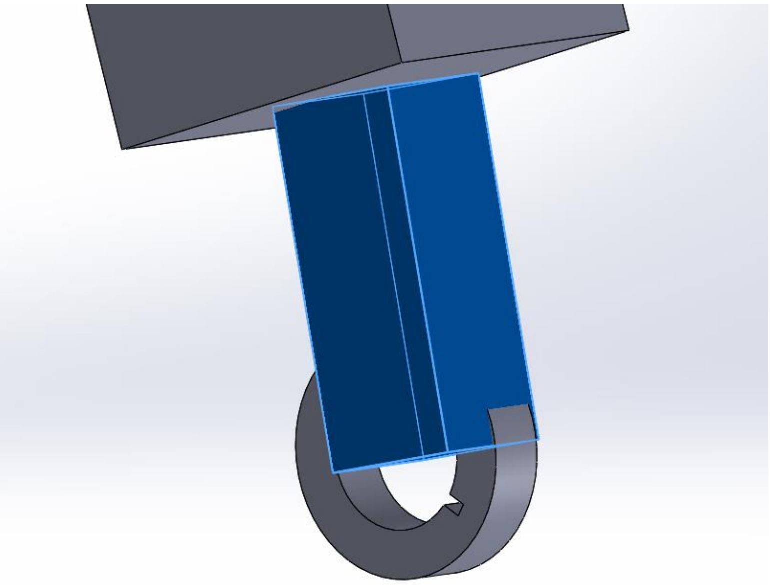


Figure 3-17: An example of a detected collision.

While checking for collisions, the program will also be monitoring the joints' angles. If an angle exceeds its physical limitation then such motion cannot be performed by the robot.

3.3 Introducing Critical Positions Concept

This part is important for the understanding of the following subsection. A critical position (CP) refers to the starting or ending position of the end-effector in a processing motion. Figure 3-18 gives a good example of some critical positions.

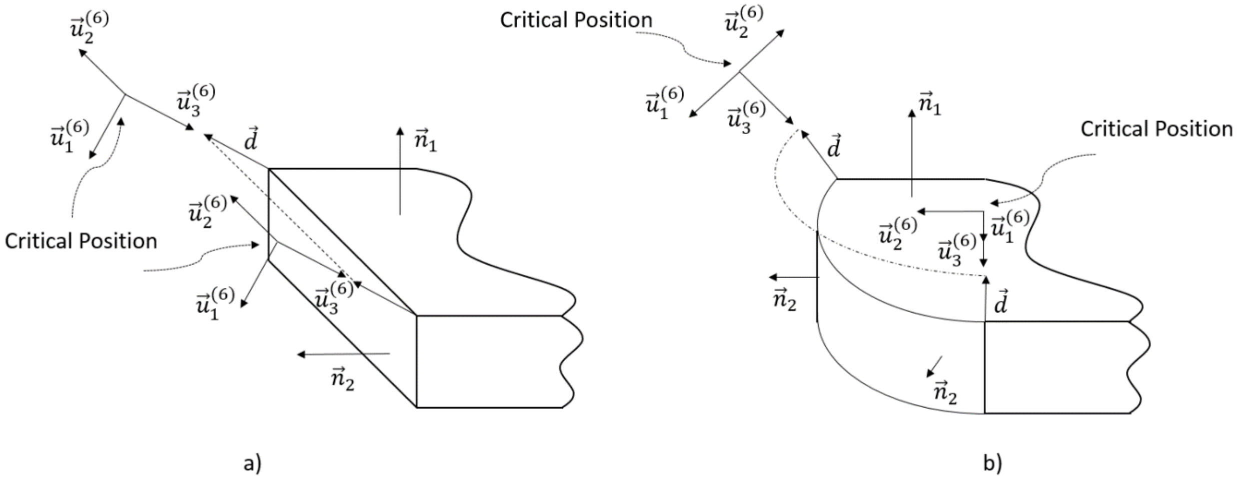


Figure 3-18: Each processing motion has two critical positions. a) shows the critical positions of a straight edge b) gives the critical positions of an arc edge

Notice that each processing motion will have two critical positions. Any non-processing motion has to connect two critical positions from two different processing motions. In this way, non-processing motions will complete the gap and create, together with processing motions, an overall continuous path to be executed by the robot.

Finally, in order to keep the consistency, the home position (initial pose of the robotic hand before start executing the motion) is also considered as a critical position.

3.4 Non-Processing Motion Generation

Recall that the overall motion of the robotic manipulator can be divided into two main motions. The processing motion “the motion needed to scan or deburr an edge” and the non-processing motion “a motion of which is needed to change the end-effector position after processing an edge to go to another one”. The latter motion does not have any actual processing and therefore it is not bounded to a specific trajectory as far as the trajectory is collision-free.

3.4.1 Calculating Non-Processing Motions

Such paths can be found using many different known algorithms in the literature [20] to yield an efficient collision-free path. Some of the famous approaches for the case of robotic manipulators in static environments are the probabilistic roadmaps planner [21], which is probabilistically complete algorithm. Khatib [22] offered a very interesting algorithm known as artificial potential field approach. However, the main drawback of this method is the possibility of sticking in local minima, that is, the algorithm is not necessarily complete. Another algorithm inspired from khatib's is the artificial force field approach. Many papers discuss slightly different models of such algorithm, [23] is an example; nonetheless, the main idea is that obstacles applies forces on the robot links in such a way that they repel the robot dynamically. Unfortunately, this method have the possibility of sticking in a local minima, similar to the artificial potential field method. On the other hand, due to the nature of the problem in hand a simple yet effective approach is going to be discussed in this section. This does not mean that any of the algorithms found in the literature related to this topic cannot be applied to this problem and implemented in the main program.

One may take an advantage from the nature of the working space in hand. Notice that if the manipulator assumes an upper configuration throughout its motion, there will not be any collision with the part at any position above a certain elevation. To find such elevation, imagine a bounding box that the piece can fit in. Figure 3-19 illustrates an example of the bounding box of an arbitrary part. There are eight vertices of such box, the ones adjacent to the top face share an elevation of value $z = z_0$. Adding half the width of the last link, it can be guaranteed that the manipulator will not collide with the part as long as the end-effector's elevation point (z-axis component of the location of the end-effector) is above $z = z_0 + 0.5w_6$, where w_6 is the maximum width of the Tool holder.

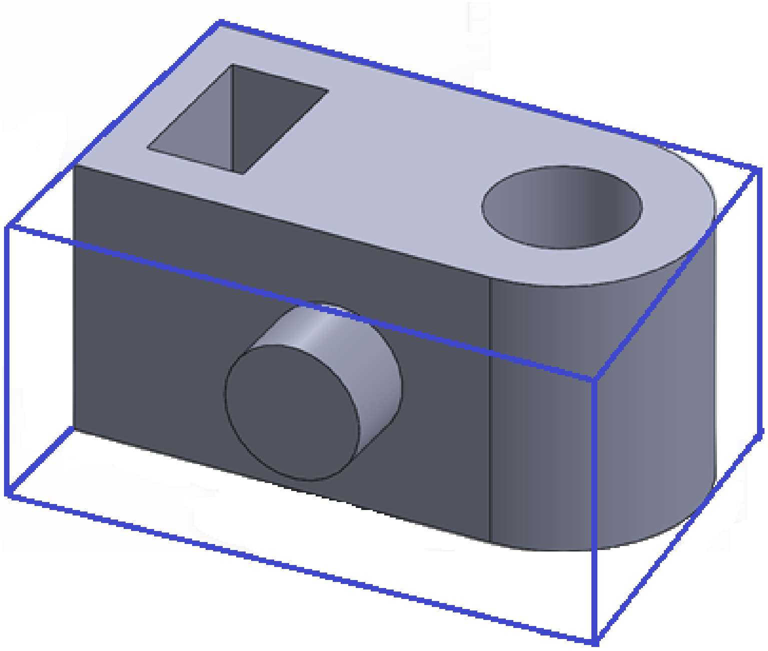


Figure 3-19: The bounding box of an arbitrary part.

Hence, if both the starting and ending positions of the non-processing motion are above the aforementioned elevation, a straight line motion (which is also the shortest) is available. However, most start and end positions are not above the safety elevation level. Therefore, a retraction motion is to be executed to transfer the end-effector from starting position to a position above the safety level. Similarly an approach motion is needed to bring the end-effector from free-space to the ending position.

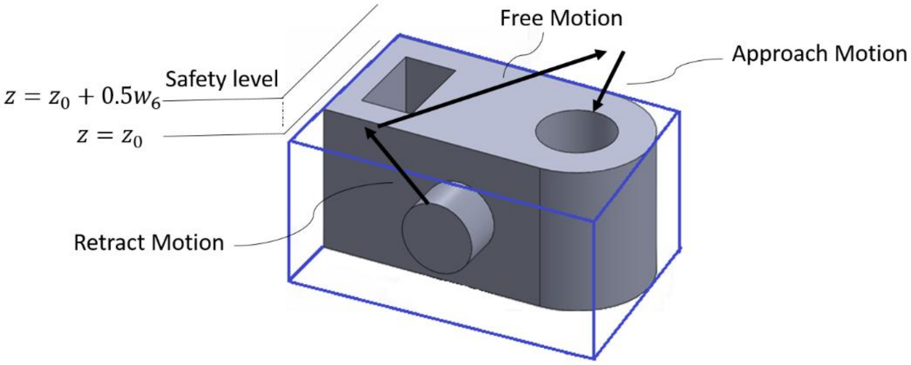


Figure 3-20: Outline representation of the non-processing motion generation algorithm.

There are two scenarios when the end-effector's location is under the safety level. First, only the last link is below $z = z_0 + 0.5w_6$. The second scenario, is to have the last link to be fully immersed and a portion of the fifth link be below $z = z_0 + 0.5w_6$.

For the first case, since the manipulator has six degrees of freedom, it is possible to retract the end-effector with a constant orientation along $\vec{u}_3^{(6)}$ from the start position to the safety level with a guarantee that this motion is collision-free. Mathematically, the orientation of the end-effector remains constant through the motion, $C_1 = C_2 = C^{(0,6)}$. While the location of the end position is $\bar{r}_2^{(0)} = \bar{r}_1^{(0)} - sC^{(0,6)}\bar{u}_3$. s is found using simple trigonometry as follows:

$$s = \frac{z_0 + 0.5w_6 - r_{1,3}}{\cos\beta} \tag{3.13}$$

Where $\cos\beta = \frac{r_{1,3} - w_{1,3}}{d_6}$ (3.14)

$r_{1,3}$ is the third component of $\bar{r}_1$ and $w_{1,3}$ is the third component of $\bar{w}_1$.

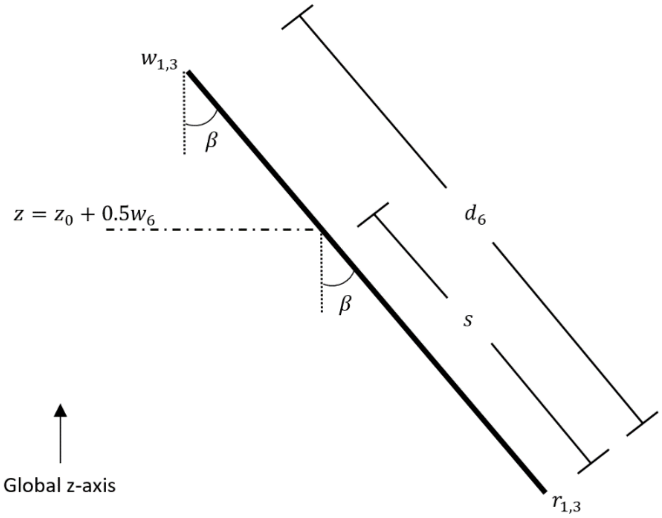


Figure 3-21: Trigonometry used to find the value of s.

For the second case, this motion can be divided into two segments. First one moves the end-effector from the start position to the wrist point with an orientation similar to the initial orientation of the fifth link. A collision test is to be performed at this case to make sure it is a safe motion. As for the second segment, the end-effector moves with a constant orientation along the new $\vec{u}_3^{(6)}$ until it reaches safety level.

Mathematically, the end position of the first segment motion can be defined by the initial orientation of the fifth link, $C_2 = C^{(0,4)}$. While the location is equal to the initial wrist point, $\bar{r}_2^{(0)} = \bar{r}_1^{(0)} - d_6 C^{(0,6)} \bar{u}_3$. Consequently for the second segment, the start position is basically the end position of the first segment. The end position, on the other hand, consists of the same orientation as with the start position, $C_3 = C_2$, and the location point $\bar{r}_3^{(0)} = \bar{r}_2^{(0)} - sC^{(0,6)} \bar{u}_3$. s is found from (3.13) and (3.14).

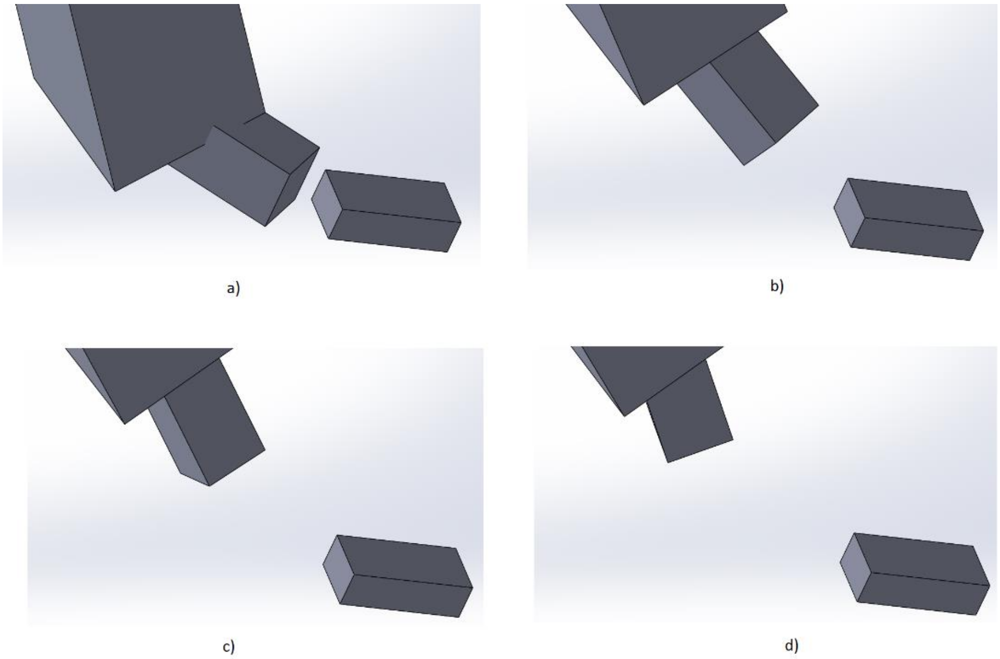


Figure 3-22: Different snap shots of the first segment of the retraction motion from a critical position where both last two links are under the safety level.

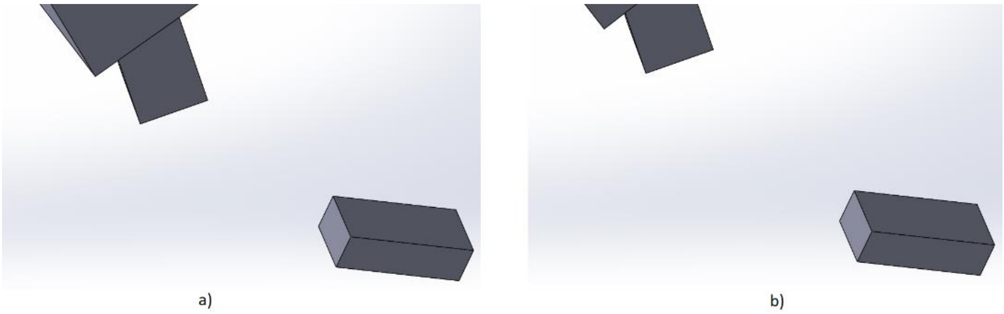


Figure 3-23: The first and last positions of the second segment of the retraction motion from a critical position where both last two links are under the safety level.

It can be realized that the approach motion to the same critical position can be generated by simply reversing its retraction motion.

Therefore, given a starting and an ending positions to generate a non-processing motion between them, all is needed is to first generate the retraction and approach motion for the corresponding positions. Then, connect these two motions with a straight line motion in free-space. Such algorithm will be referred to as the Retraction-Free-Approach Motion (RFA Motion). This approach significantly reduces the complexity that most collision-free path generation methods suffer from.

The RFA motion may not be the shortest possible collision-free path between two given positions, nonetheless, it offers a very efficient solution regarding CPU time and path length.

In some special cases the shortest collision-free path is very easy to find. First special case is when both start and end positions share the same vertex, check Figure 3-24. The motion is a direct one (straight line in the configuration space). Such motion is collision-free since the shared vertex is reachable.

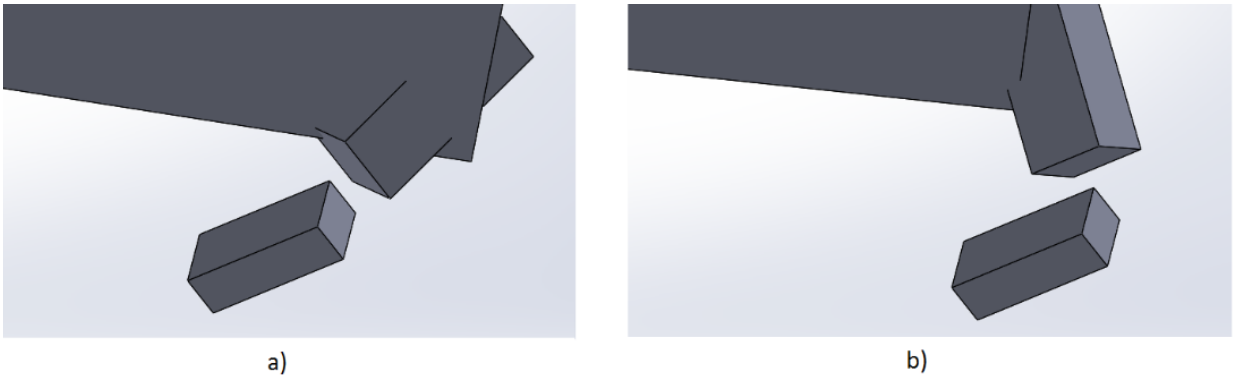


Figure 3-24: Illustration of the non-processing motion when both start and end positions point at the same vertex.

The second case is when both positions share a reachable edge. A direct motion can be made by utilizing the previously generated processing motion of the shared edge, see Figure 3-25. This motion is also collision-free since it is constructed of smaller collision-free motions.

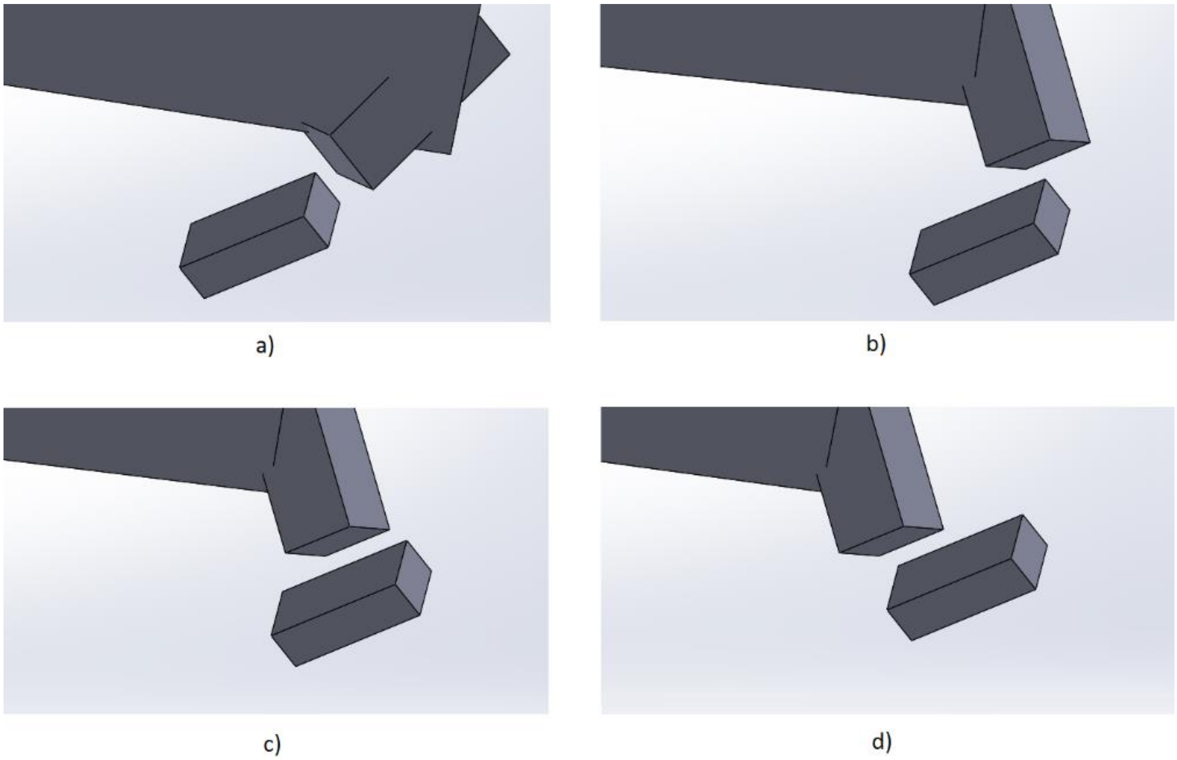


Figure 3-25: Illustration of the non-processing motion when both the start and end positions share a reachable edge.

For generating the non-processing motion between two given positions, the algorithm will first check if these two positions share a vertex or an edge. If so, the method explained above is used to generate the path of the end-effector. Otherwise, an RFA motion is generated.

This approach reduces the complexity of finding all possible non-processing motions from O(n^2) to O(n). Instead of generating a motion between each pair of critical positions, the problem is reduced to calculate one retraction motion for each critical position.

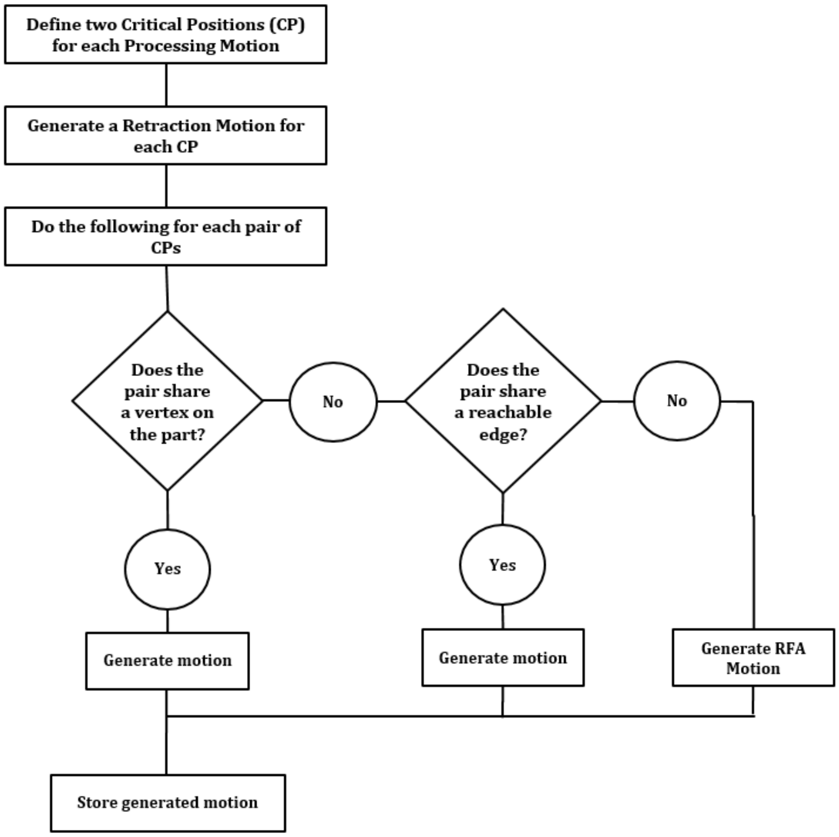

Define two Critical Positions (CP) for each Processing Motion
Generate a Retraction Motion for each CP
Do the following for each pair of CPs
Does the pair share a vertex on the part?
No
Does the pair share a reachable edge?
No
Yes
Yes
Generate motion
Generate motion
Generate RFA Motion
Store generated motion

Figure 3-26: Flowchart of generating all possible non-processing motions.

Lastly a 2D array of non-processing motions is constructed. This array includes all possible non-processing motions between each pair of critical positions. Any of these motions is represented by two indices. The first index represent the number of the critical position that the end-effector will start from. The second index is for the ending position of the end-effector.

3.4.2 Collision Detection Test

As in the case of processing motions, the non-processing motions generated between each two positions need to be collision-free. Only the retraction and approach parts of the RFA motion need to be checked. Since the portion of the RFA motion made in free space is by definition free of collisions.

It is critical to realize that both the approach and retract motions for a given critical position is in fact the same motion but reversed in the order of execution. Therefore, the collision detection test boils down to checking the retraction motion of each critical position separately.

The test nature is similar with the one performed on processing motions. That is, the motion is discretized into a number of poses that in turn covers the whole range of motion. Figure 3-22 and 3-23 show an example of such discretization of a retraction motion.

If the retraction motion from a critical position is collision-free, then its corresponding edge is reachable from any other position in free space. Otherwise, the given edge is considered unreachable.

While checking for collisions, the program will also be monitoring the joints' angles. If an angle exceeds its physical limitation then such motion cannot be performed by the robot.

3.5 Planning the Overall Path

After generating all possible non-processing motions next step is to select a subset of them that creates a connected path together with the processing motions. In fact, such problem is a TSP (Traveler Salesman Problem).

The Traveling Salesman Problem, or TSP, is an ongoing study in computer science. The TSP has a long history ranging back to the 1920's [24]. This problem became popular after it was publicized by a mathematician named Merrill Flood at the RAND corporations in 1948 [25].

The TSP problem is defined as the following. Given a complete graph G with Vertices V and edges E, where each edge $e_{ij} \in E$ has an associated cost c_{ij} incurred when traversing from vertex $i \in V$ to $j \in V$, find the optimal, or cheapest, Hamiltonian cycle of G. The vertices can be considered buildings, landmarks or other geographical locations. Thus, a Hamiltonian cycle of G is also considered a Tour. In this study, these vertices represent the critical positions of end-effector. The edges represent motions of the robot. Therefore, a Hamiltonian cycle represents the overall path of the robot.

To further understand this representation refer to Figure 3-27. Notice that each node in the graph represents a critical position of the end-effector, labeled as $\text{Pose}_{(i)}$. The connected edges represents processing motions $\text{PM}_{(i)}$ and the dashed ones is for non-processing motions. So the problem statement would be similar to that of the TSP problem (each position is to be passed once) with the addition that the connected edges are also to be passed one time each. Unfortunately, general TSP is known to be an NP-complete problem. Hence, there is not a systematic approach to reach the optimum solution in a polynomial time algorithm.

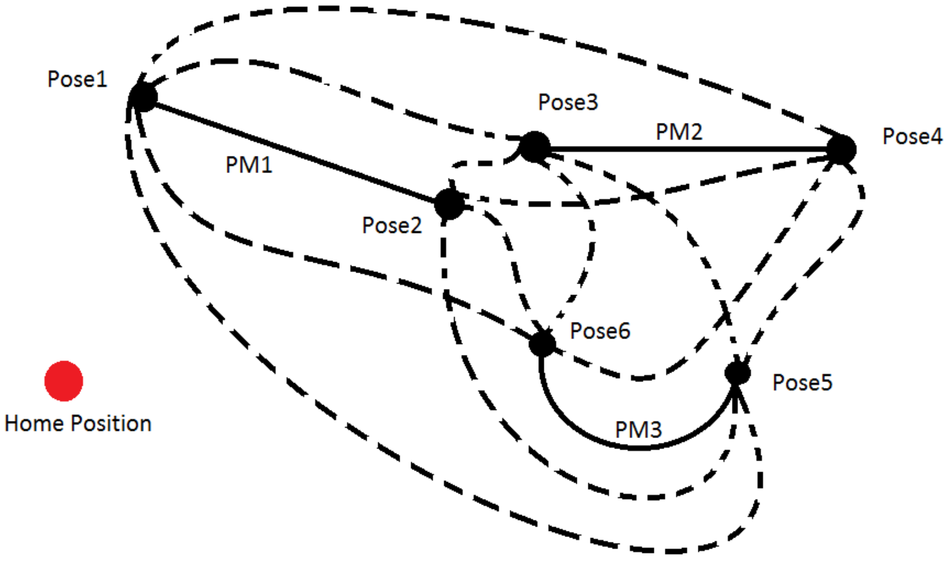


Figure 3-27: An example of a three processing motions (PM) and the representing graph for finding route solution of the TSP “The non-processing motions between the home position and other poses are not shown for simplicity”

There are different variations of the TSP. In the general case, there are no restrictions on the edge costs. Therefore, each edge may have two associated costs, $c_{ij} \in \Re$ and $c_{ji}\in \Re$, which may not necessarily be equal. Thus, G can be a directed graph and the edge costs can be negative.

Another variation is the metric TSP, the main feature of this type is that the triangle inequality holds on all the vertices, which means that for any three vertices A, B and C, if you wish to go from vertex A to vertex C, it is always cheaper to go from A directly to C rather than passing through B. This added restriction gives a possibility to find approximated solutions, which is not possible for the general case. These solutions can be found in a polynomial time scale, one of these algorithms is known as 2-approximation algorithm. As the name implies, it gives a solution that is at most twice the length of an optimal tour. 3/2-approximation algorithm [26] is another approach that guarantees the founded tour to be at most 1.5 times longer than an optimal solution.

A third variation is the Euclidean TSP. Along with the validity of the triangle inequality, all of the vertices are points in space. The plane can be 2-dimensional, or d-dimensional in general. The edge costs are the Euclidean distances between the points. Since the problem here takes place in Euclidian space, there are restrictions on the edge costs, and some assumptions can be made, which simplify the problem. Making it possible to apply even better approximated solution in polynomial time. Arora [27] presented a polynomial time approximation solution (PTAS) algorithm for the two dimensional case and shortly after two years generalized the algorithm to d-dimensions [28]. The accuracy and CPU time depends on the approximation parameter c. Where the accuracy is (1+1/c)-approximation and the processing time is $O\left(n(\log n)^{\left(O(\sqrt{d}c)\right)^{d-1}}\right)$. In the same time period Mitchell [29] independently proposed a PTAS algorithm where he was able to achieve $\left(1+\frac{2\sqrt{2}}{m}\right)$-approximation that runs in $O(n^{20m+5})$ time.

Unfortunately, the TSP representing the problem of this section is neither Euclidian nor metric in its nature. As mentioned before, general TSP does not have any polynomial time approximation algorithms. The practical approach is to apply heuristic algorithms to find a solution. There are many heuristic algorithms that can be found in the literature. The nearest neighbor (NN) algorithm is one example. NN always choose to visit the nearest vertex to the current reached vertex until all vertices has been visited. This approach will often keep its tour within 25% of the Held-Karp lower bound. Another heuristic known as Greedy algorithm gradually constructs a tour by repeatedly selecting the shortest edge and adding it to the tour as long as it doesn’t create a cycle with less than N edges, or increases the degree of any node to more than 2. This algorithm normally keeps the solution within 15-20% of the Held-Karp lower bound. Kahng and Reda [30] proposed a heuristic they called Match Twice and Stitch (MTS). Their approach offers four different variations where a trade-off between accuracy and run time is present. As they reported, the algorithm on average yield a solution that is 6-8% of the Held-Karp lower bound.

In this study, the nearest neighbor algorithm is adopted and implemented in the software. Starting from the home position, the red node shown in Figure 3-27, the shortest non-processing motion will be selected, where the distance is the length of the trajectory of each motion. Next, the processing motion connected to the reached node is taken. After that, the shortest non-processing motion is selected and so on so forth until all the nodes are reached. Lastly, the end-effector is brought back to the home position using the last available non-processing motion.

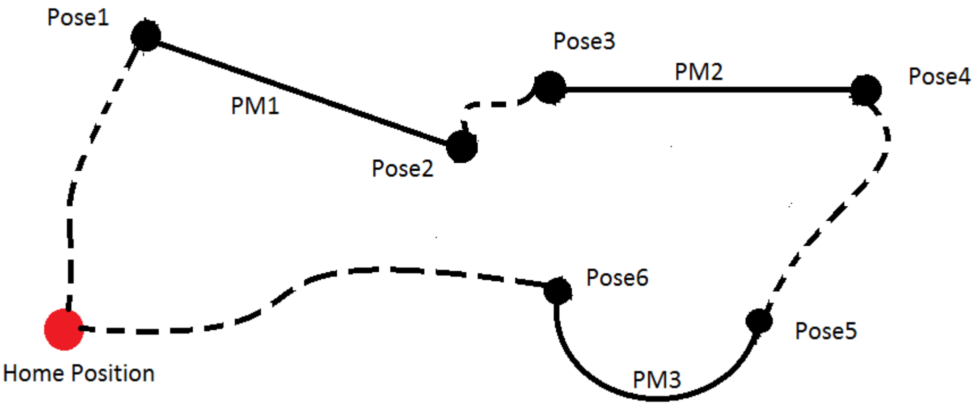


Figure 3-28: The solution of the TSP graph given in Figure 3-27 using nearest neighbor algorithm.

In conclusion, the solution of the TSP given in this subsection will correspond to a collision-free motion trajectory which represents the overall path to be executed by the robotic manipulator.

CHAPTER 4

COMPUTER PROGRAMS

.

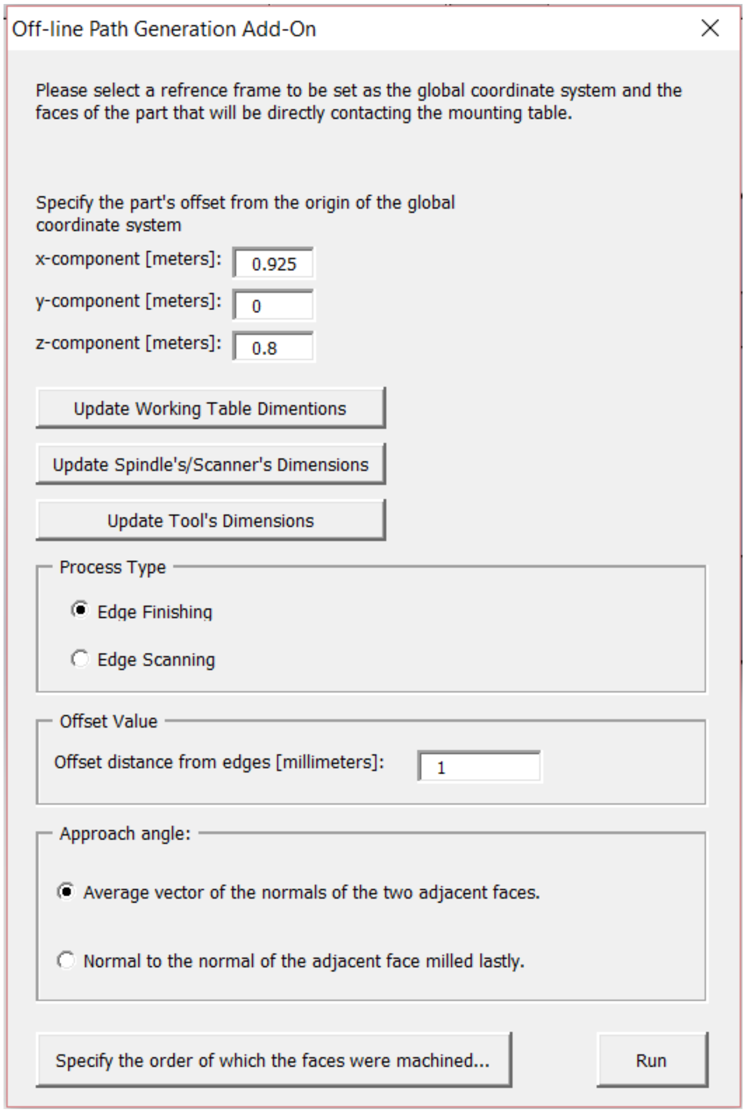
4.1 SolidWorks Add-Ons Package

Here, all the previously discussed points are developed and applied in this software. First part is a user guide to give the reader the appropriate way of using the software. Second part explains the general working procedure of the software. Refer to appendix A for more details on the structure and main methods used in the program.

4.1.1 User Guide of Software

Preparation Stage

In this stage, the parameters that the user is expected to specify before generating the path are entered to the program. Figure 4-1 shows the main window of the software.



Off-line Path Generation Add-On

Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

x-component [meters]: 0.925

y-component [meters]: 0

z-component [meters]: 0.8

Process Type

● Edge Finishing

○ Edge Scanning

Offset Value

Offset distance from edges [millimeters]: 1

Approach angle:

● Average vector of the normals of the two adjacent faces.

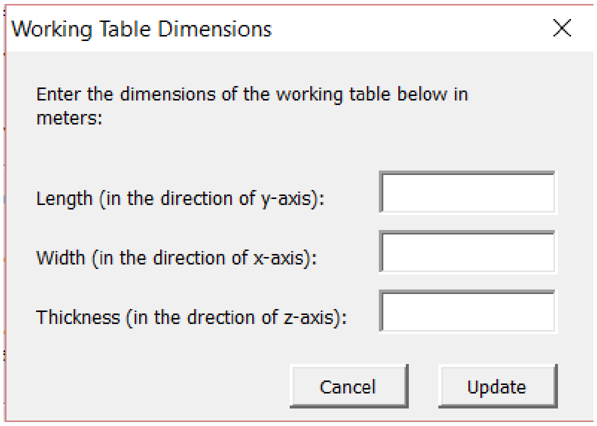
○ Normal to the normal of the adjacent face milled lastly.

Update Working Table Dimentions
Update Spindle's/Scanner's Dimensions
Update Tool's Dimensions
Specify the order of which the faces were machined...
Run

Figure 4-1: Main window of the SolidWorks Add-Ons Package.

As mentioned in earlier chapters, the user has first to define a reference frame (RF) and attach it to the workpiece or to the fixture if present, refer to section 3.1 for a detailed explanation of RF. Staring from the top, the user can specify the origin of RF by simply entering the x, y and z components of its location in respect of the global coordinate system of the ABB.

Entering the dimensions of the working table can be achieved by first clicking on the Update Working Table Dimension button. Clicking on this button, another window, shown in Figure 4-2, appears and the user then can enter all three dimensions of the table. Clicking Update will store these values and bring the user back to the main window.



Working Table Dimensions

Enter the dimensions of the working table below in meters:

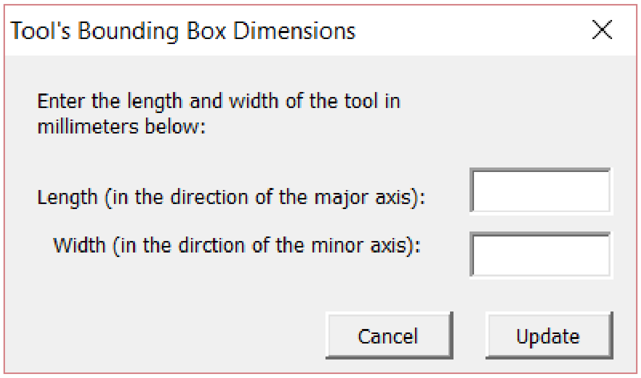
Length (in the direction of y-axis):

Width (in the direction of x-axis):

Thickness (in the drection of z-axis):

Figure 4-2: Setting the working table dimensions.

In the Process Type frame the user choose between finishing and scanning. The main reason for this is for the software to decide whether to include a finishing tool or not. In case the user selects the finishing process he/she needs to enter the dimensions of the deburring/chamfering tool by clicking on the Update Tool's Dimensions button. Once clicked a window opens as shown in Figure 4-3. Then, after entering the dimensions the user clicks on Update.


Tool's Bounding Box Dimensions

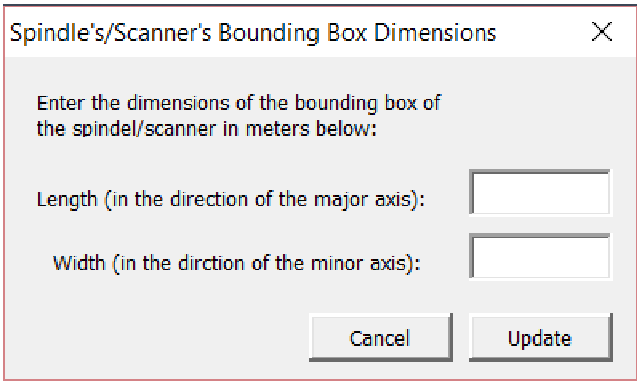
Enter the length and width of the tool in millimeters below:

Length (in the direction of the major axis):

Width (in the dirction of the minor axis):

Figure 4-3: Setting the tool's dimensions.

The “Update Spindle’s/Scanner’s Bounding Box” button allows the user to specify the cuboid that represents the bounding box of the spindle/scanner. Figure 4-4 shows the window appearing after clicking on the aforementioned button.


Spindle's/Scanner's Bounding Box Dimensions

Enter the dimensions of the bounding box of the spindel/scanner in meters below:

Length (in the direction of the major axis):

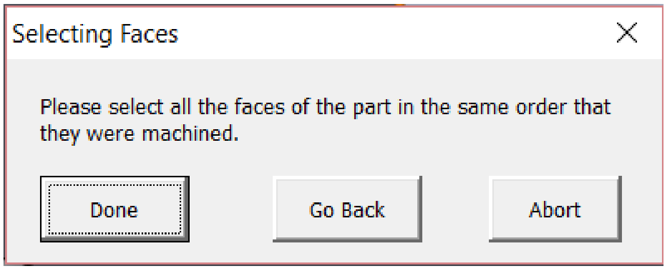
Width (in the dirction of the minor axis):

Figure 4-4: Setting the tool's body box dimensions.

Offset value frame is simply for entering the magnitude distance of the end-effector while processing the edges.

Approach angle frame is for selecting between approaching the edge with a direction equal to the average of its adjacent faces’ normals or normal to the first machined face between the adjacent faces. Refer to section 3.1 for more details.

Lastly, if the second option in the approach angle is selected, the user need to give the order in which the faces of the part where machined. This can be done by first clicking on the button next to Run button. After the user clicks on the button he/she starts selecting the faces of the part in the order they were machined. Figure 4-5 shows the mentioned window.

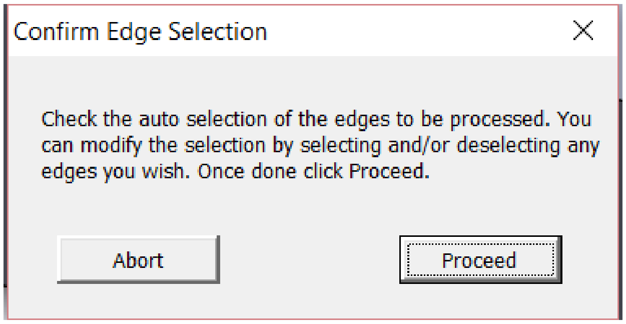

Selecting Faces

Please select all the faces of the part in the same order that they were machined.

Figure 4-5: Confirming the selection of faces.

Edge Selection

After entering all the necessary information the user clicks the Run button in the main window. The software then selects the convex edges of the part. At this point, all the default selected edges are highlighted on the workpiece and the software is paused. The user is asked to check if the selection is satisfactory and if he/she would like to edit the selection by either selecting more edges or deselecting some of the already selected ones. After that the user click on Proceed button.



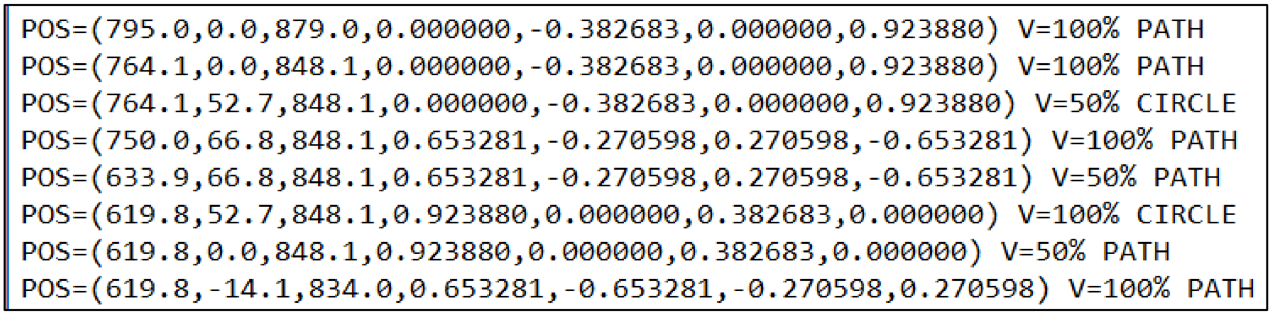
Confirm Edge Selection

Check the auto selection of the edges to be processed. You can modify the selection by selecting and/or deselecting any edges you wish. Once done click Proceed.

Figure 4-6: Window for confirming the selected edges.

Getting Results

Now, the display of the part is temporarily turned off to make the program run faster. Calculating all the necessary motions and checking for collisions, the program at the end will creates a text file in the same directory with the opened SolidWorks part. This text file will include the overall nominal path to be executed by the robot. The syntax is similar to that used by ABB IRB2000 when generating paths by the teaching method. The only difference between these formats is that in S3 controller the information of each position is implicitly given. While here the position is explicitly given after the POS keyword, e.g. POS=(x, y, z, q_1, q_2, q_3, q_4) where x, y and z are the Cartesian coordinates of the tip-point location and q_i's are the four quaternion components that describe the end-effector orientation [31]. Figure 4-7 gives an example of a program path opened from such text file. Refer to the programming manual of ABB IRB2000 for detailed explanation of the syntax [32]. Lastly, a message box pops up to inform the user that the path has successfully been generated.



```
POS=(795.0,0.0,879.0,0.000000,-0.382683,0.000000,0.923880) V=100% PATH
POS=(764.1,0.0,848.1,0.000000,-0.382683,0.000000,0.923880) V=100% PATH
POS=(764.1,52.7,848.1,0.000000,-0.382683,0.000000,0.923880) V=50% CIRCLE
POS=(750.0,66.8,848.1,0.653281,-0.270598,0.270598,-0.653281) V=100% PATH
POS=(633.9,66.8,848.1,0.653281,-0.270598,0.270598,-0.653281) V=50% PATH
POS=(619.8,52.7,848.1,0.923880,0.000000,0.382683,0.000000) V=100% CIRCLE
POS=(619.8,0.0,848.1,0.923880,0.000000,0.382683,0.000000) V=50% PATH
POS=(619.8,-14.1,834.0,0.653281,-0.653281,-0.270598,0.270598) V=100% PATH
```

Figure 4-7: A portion of a path program generated by the software and stored in a text file.

4.1.2 Code’s Working Procedure

The main function of the overall code “main()” utilizes two class types, namely, MainWindow and PathGeneratorEngine classes. MainWindow class coordinates the flow of the software as a whole. Responding to user inputs, it organizes the variables needed to be set before starting the actual path generation. It also schedule the calling of main methods in the PathGeneratorEngine. Lastly, it creates the final output of the software as a text file that is stored independently from the program on the working computer. On the other hand, PathGeneratorEngine class is responsible for all the background calculations of the different motions. It also checks each motion segment for collisions. And lastly generates the overall path and pass it to the MainWindow class.

For the preparation stage, the procedure was indirectly explained in the user guide section. This step is simple as it is, only assigning the inputs of the user into their corresponding variables that will be used by other parts of the program.

First main method that gets called is the EdgeSelection(SelectionScheme) function. Here by utilizing the PowerSelect utility, provided by SolidWorks, all the convex edges of the workpiece get selected. Then, by checking each selected edge the ones that are adjacent to a ground face are deselected.

After the user click Proceed. A PathGeneratorEngine instant is created which then takes all the selected edges and calculate a processing motion for each one of them. Next, each of the calculated process motions is checked for collision. In case a process motion found to cause a collision then the corresponding edge is discarded along with the process motion. Then, both starting and ending end-effector positions of all the collision-free process motions are stored properly, such poses are referred to as critical positons. After that, the retraction motion from each critical position is calculated and checked for collision. Any critical position that possesses a retraction motion that includes a collision is marked and their corresponding edges are taken as unreachable. Now, PathGeneratorEngine creates a non-processing motion between every two critical poses and store these motions in a 2D array. The indices indicate the starting and ending positions of each motion, respectively. What is left is a search algorithm to connect all the processing motions by finding a suitable non-processing motions from the 2D array previously created resulting in an overall connected path. Finally, this path is then translated into motion commands and stored in a text file to be later uploaded to the robot.

4.2 ABB User Interface Software

This software was developed for the general purpose of communicating with the ABB IRB2000 robot. A more specific function of this program is uploading the previously generated path from the SolidWorks Add-Ons package. The first sub-section serves as a user guide. Second sub-section describes briefly the procedure adopted by the software to function properly. Refer to appendix A for more detailed explanation of the structure and main methods of the software.

4.2.1 User Guide of Software

Serial Port Connection

The very first thing need to be done when launching the software is to open the serial port between the computer and ABB robot. This step has been simplified from the user point of view. That is, all of the serial communication's configuration is set automatically by the software according to ADLP10 communication protocol. The user only has to select the COM number that the ABB is connected to. The software will also search for the available COM ports and show them to the user.

Such step can be easily done as shown in the figures below:

- First, select Open Serial Port from File.

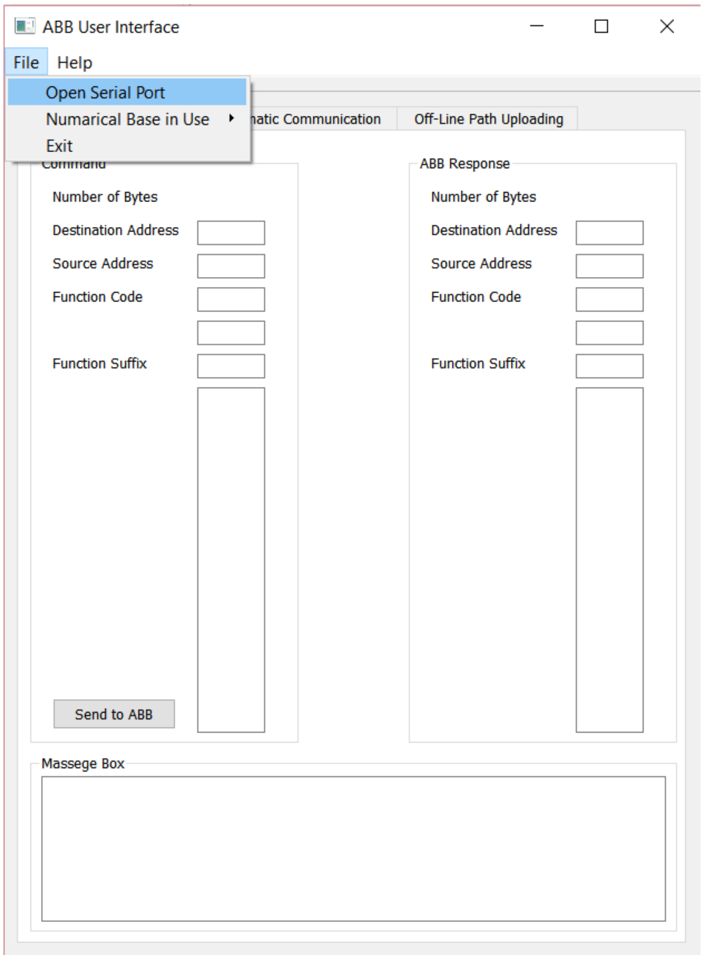
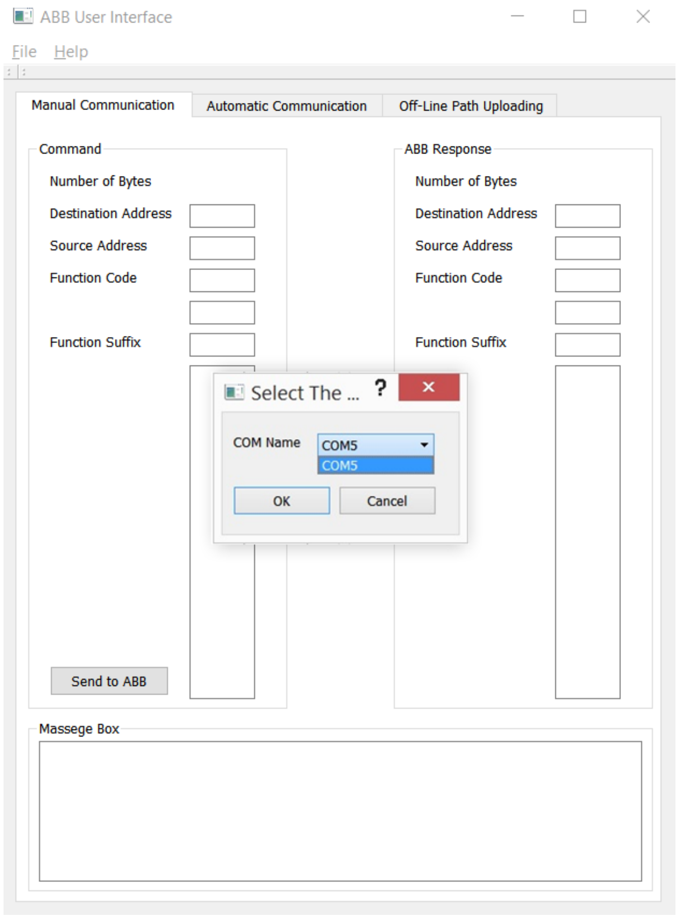


Figure 4-8: Selecting open serial port from File tab.

- Second, select the port connected to ABB robot.



COM Name COM5

Figure 4-9: Selecting the corresponding COM port connecting the ABB robot.

- Finally, in case the user desire to close the serial port. He/she can click on Close Serial Port in File tab.

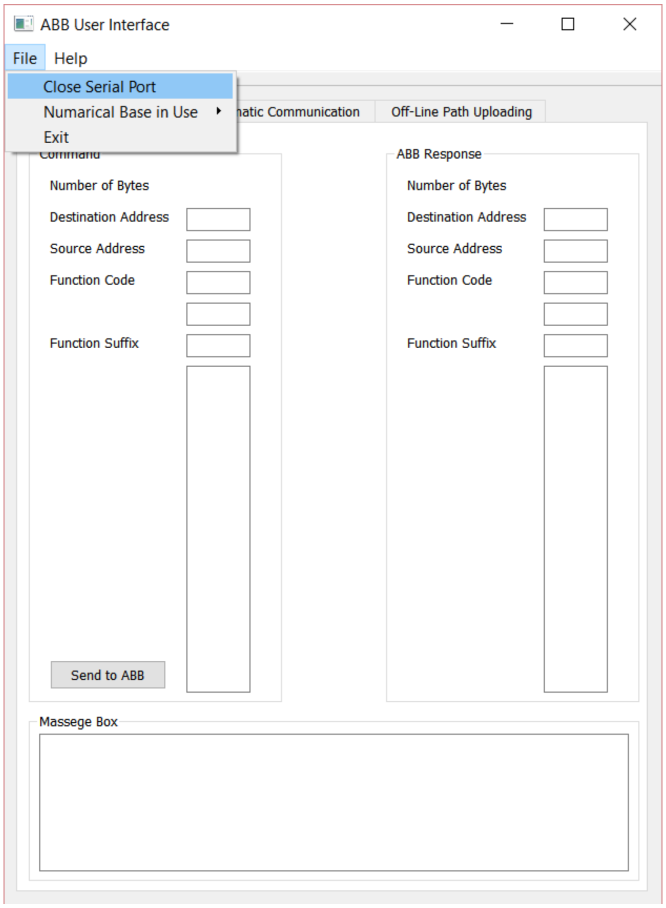


Figure 4-10: Closing the serial port from File tab.

Numerical Bases Available

In the manual communication mode, the user can choose between hexadecimal and decimal bases for writing and receiving telegrams. He/she can also switch between the two bases while writing the telegram in case some bytes were calculated in different base. The selection of the numerical base can be found in File/ Numerical Base in Use, as shown in Figure 4-11.

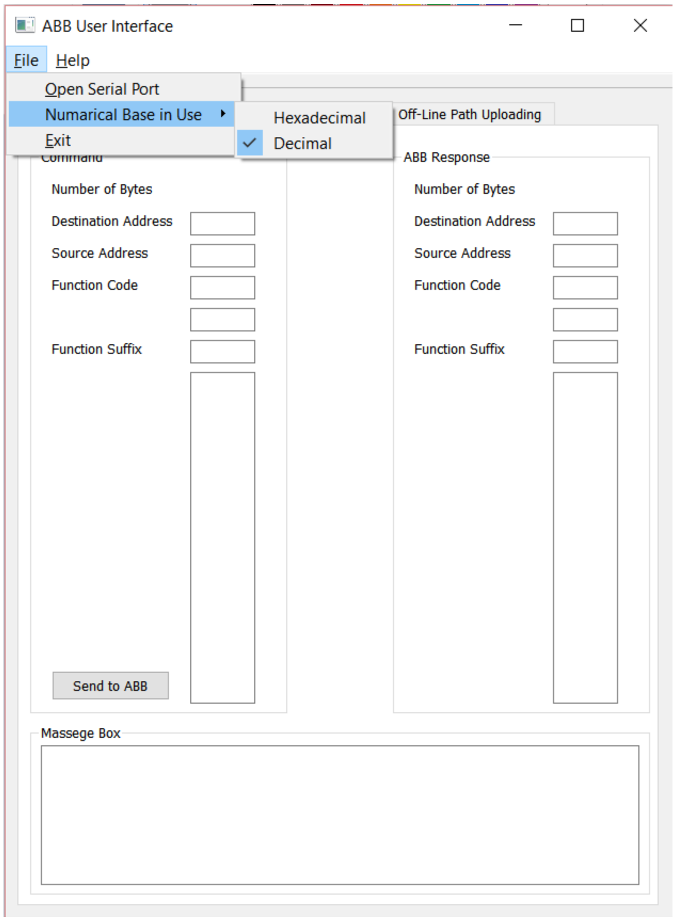
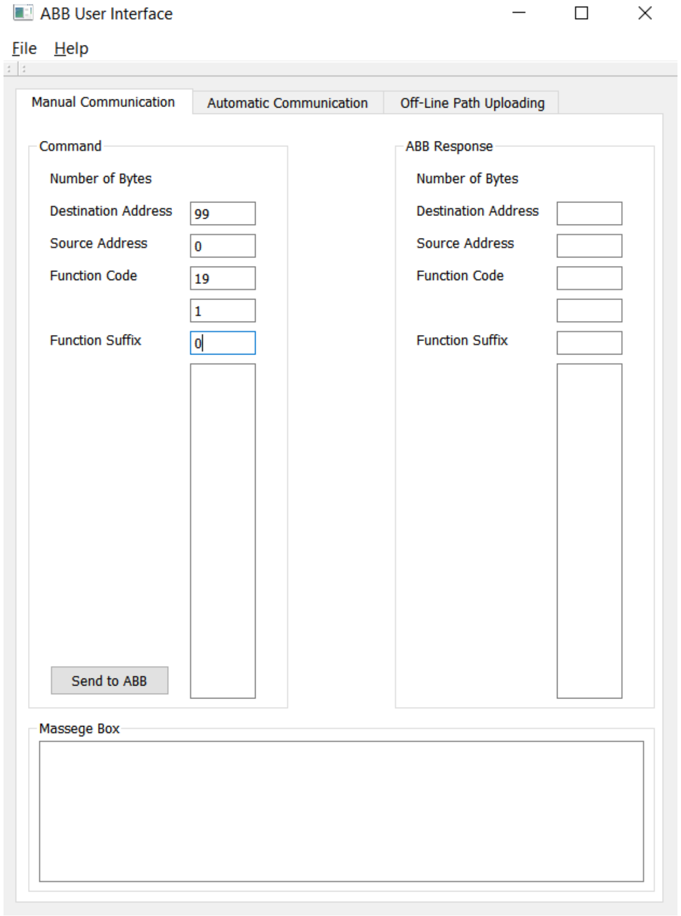


Figure 4-11: Choosing Numerical Base for Manual Communication.

Manual Communication

As the first mode of communication offered by the software, such mode requires the user to be familiar with ARAP Protocol [33]. The manual communication tab is divided into three regions, namely, Command, ABB Respond and Massage Box. As for the user, the command to be send is filled in the command region according to ARAP Protocol.

As an example consider a user desire to know the status of the ABB robot. Hence, the corresponding function for such task is function 19 in ARAP Protocol. Assuming the computer address is 0 and ABB's is 99.



Command

Number of Bytes

Destination Address 99

Source Address 0

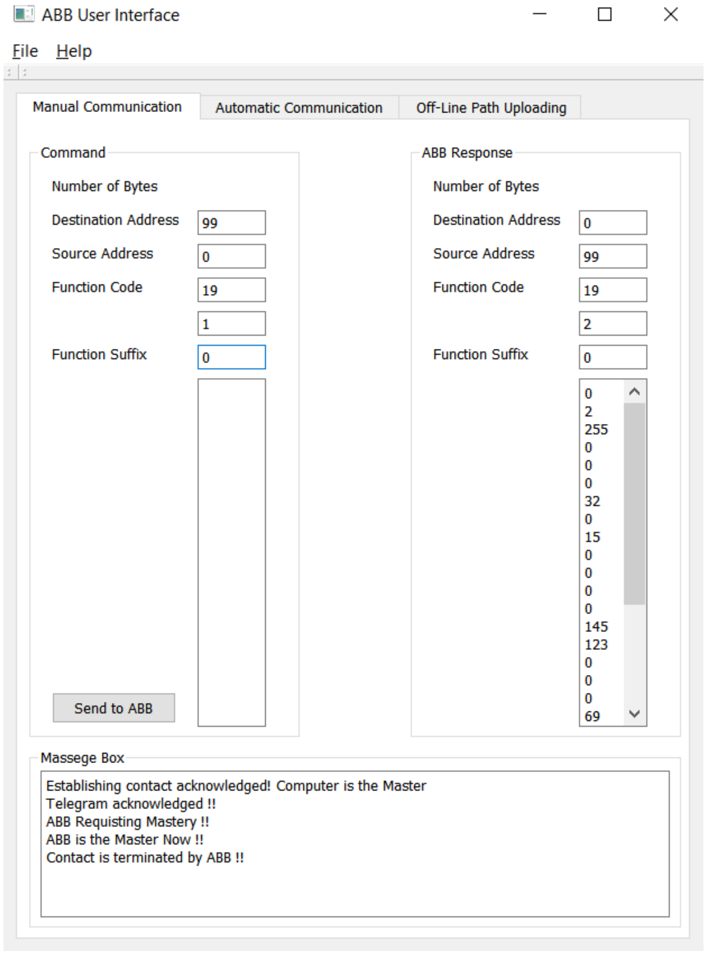
Function Code 19

1

Function Suffix 0

Figure 4-12: Function 19 in ARAP Protocol filled in the command region.

After filling the command the user click Send to ABB button. ABB will respond by a telegram that will be shown in ABB Response region. Note that the Massage Box gives feedback to inform the user about the status of the communication continuously.



Command

Number of Bytes

Destination Address 99

Source Address 0

Function Code 19

1

Function Suffix 0

ABB Response

Number of Bytes

Destination Address 0

Source Address 99

Function Code 19

2

Function Suffix 0

```
0
2
255
0
0
0
32
0
15
0
0
0
0
145
123
0
0
0
69
```

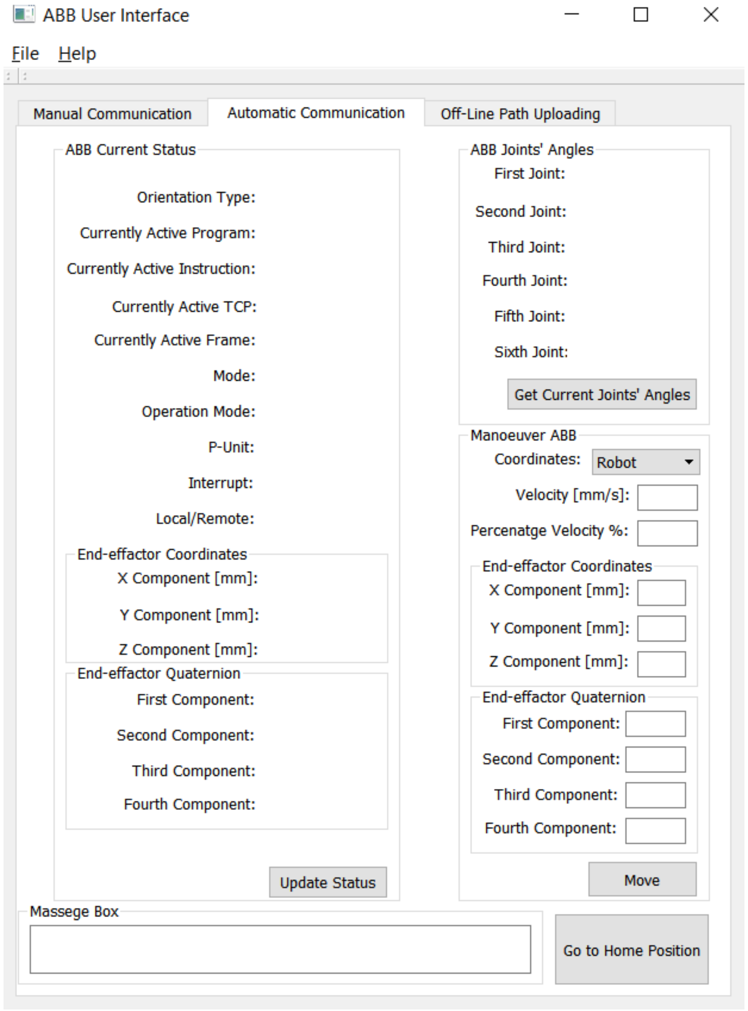
Massege Box

```
Establishing contact acknowledged! Computer is the Master
Telegram acknowledged !!
ABB Requisting Mastery !!
ABB is the Master Now !!
Contact is terminated by ABB !!
```

Figure 4-13: Response of function 19 in ARAP Protocol filled in the command region.

Automatic Communication

The second mode of communication is called Automatic Communication. The reason is called this way is because instead of manually typing the telegram of the function to be executed by the ABB, the user only fill some parameters to be sent to ABB. While the software itself generates the telegram according to ARAP protocol. However, only the most common functions are considered in this mode, namely, status request (function 19), requesting the six joint angles (part of function 44), maneuvering of ABB (function 24) and lastly running a program inside the ABB registers to let it go to the home position (using function 2).



Coordinates: Robot

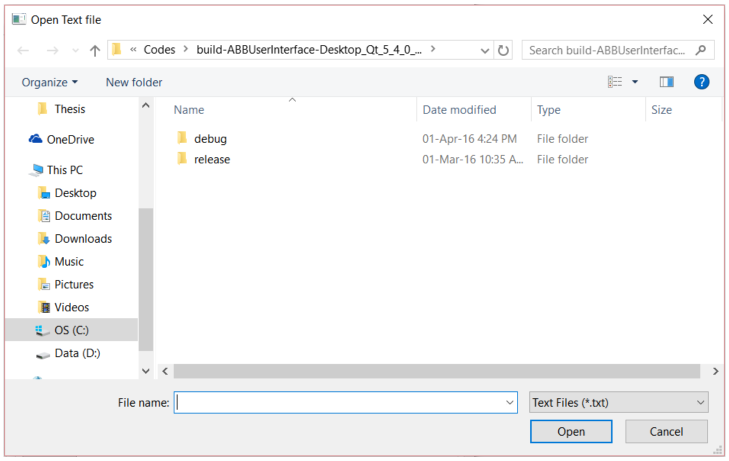
Figure 4-14: The GUI of the Automatic Communication Mode.

In order for the user to get the status of ABB all he/she has to do is click on the Update Status button, the software will automatically fill the parameters in the ABB Current Status box. Same goes for the joint angles. On the other hand, for maneuvering of ABB, the user has to first specify if Robot or Rectangular coordinates is to be used and also fill the velocity of the end-effector and its desired position and orientation with respect to the base coordinates of ABB. Lastly, Go To Home Position button lets the user command the robot to change its configuration to the home position posture. It is also worth mentioning that the massage box will be informing the user useful information especially in case some errors arise while communicating with ABB.

Off-Line Path Uploading

The third, and most important part for this thesis, is the uploading of the path program from the computer to ABB IRB2000. There are two types of “syntax” a user can use in defining the path program. First one is used only in cases where the user is very experienced with S3 controller assembly language. The user has to set each byte separately to be sent by the software to the robot. The main goal of this option is to make the software more flexible for potential programs that other software may generate.

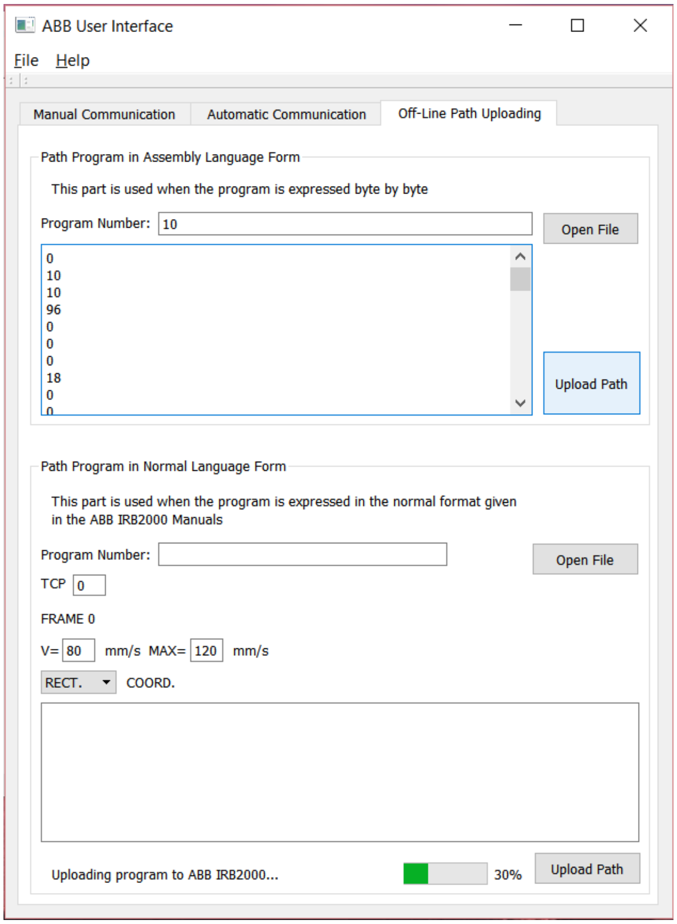
After selecting the Off-Line Path Uploading tab in the main window of the software, select the open file button in the upper part of the window, which is dedicated for this type of syntax. A window will show up as shown below, select the text file that includes the path program in it. By doing so, the software will display the contents of the text file in the text box so that the user can check the program and perform modifications if desired.



Name	Date modified	Type	Size
debug	01-Apr-16 4:24 PM	File folder	
release	01-Mar-16 10:35 A...	File folder	

Figure 4-15: Window for selecting the text file that includes the path program.

Then, the user specifies the number of the program to be assigned in the robot memory. The software will display the progress of the upload procedure to the user, at the end of the window a small notification and a progress bar will pop up directly after the user hit the upload button. In case an error occurs the software will give feedback to the user so he/she could either recheck the program syntax or resend the program.



Path Program in Assembly Language Form

This part is used when the program is expressed byte by byte

Program Number: 10

```
0
10
10
96
0
0
0
18
0
0
```

Path Program in Normal Language Form

This part is used when the program is expressed in the normal format given in the ABB IRB2000 Manuals

Program Number:

TCP 0

FRAME 0

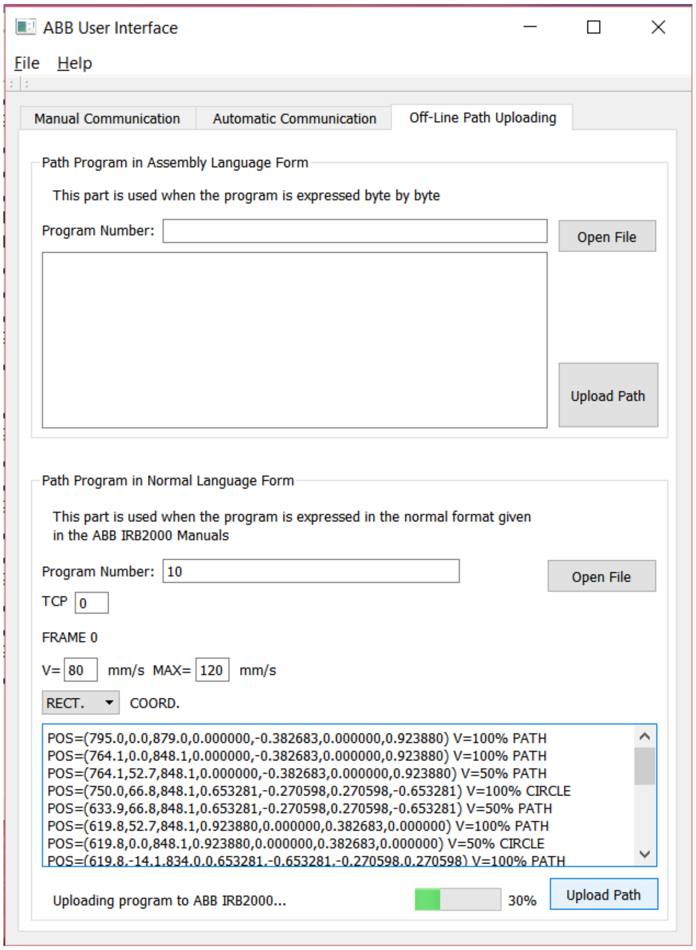
V= 80 mm/s MAX= 120 mm/s

RECT. COORD.

Uploading program to ABB IRB2000... 30%

Figure 4-16: Displaying the path program of the previously selected text file.

The second syntax to express the path program is more user friendly and understandable. This format is similar to that of the S3 controller way of expressing the taught paths. Figure 4-17 gives an example of a program path opened from a text file in a similar manner as in the previous case. Again the software will give the user continues feedback of the upload process.



Path Program in Assembly Language Form

This part is used when the program is expressed byte by byte

Program Number:

Path Program in Normal Language Form

This part is used when the program is expressed in the normal format given
in the ABB IRB2000 Manuals

Program Number: 10

TCP 0

FRAME 0

V= 80 mm/s MAX= 120 mm/s

RECT. COORD.

```
POS=(795.0,0.0,879.0,0.000000,-0.382683,0.000000,0.923880) V=100% PATH
POS=(764.1,0.0,848.1,0.000000,-0.382683,0.000000,0.923880) V=100% PATH
POS=(764.1,52.7,848.1,0.000000,-0.382683,0.000000,0.923880) V=50% PATH
POS=(750.0,66.8,848.1,0.653281,-0.270598,0.270598,-0.653281) V=100% CIRCLE
POS=(633.9,66.8,848.1,0.653281,-0.270598,0.270598,-0.653281) V=50% PATH
POS=(619.8,52.7,848.1,0.923880,0.000000,0.382683,0.000000) V=100% PATH
POS=(619.8,0.0,848.1,0.923880,0.000000,0.382683,0.000000) V=50% CIRCLE
POS=(619.8,-14.1,834.0,0.653281,-0.653281,-0.270598,0.270598) V=100% PATH
```

Uploading program to ABB IRB2000... 30%

Figure 4-17: Displaying the path program of the previously selected text file.

Unlike the case of using the assembly syntax, here the path information taken from any text file should consists of lines in the form of POS=(795, 0, 879, 0, -0.383, 0, 0.924) V=100% PATH. While the initialization of the program can be done by filling the text lines shown in Figure 4-17.

4.2.2 Code’s Working Procedure

The main function of the overall code “main()” utilizes two class types, namely, ADLP10 and ARAP class. The former class mimics ABB data link protocol, which is the protocol created by ABB Company to look over the connection and communication between the computer and ABB robots connected to it. Such protocol take charge of establishing the connection and maintain it as long as needed. On the other hand, the latter protocol is only responsible for generating telegrams for each specific task or massage that is to be exchanged. These two protocols are fully explained by ABB user manual in the Computer Link part.

In the case of the manual communication, there is no need for the use of ARAP class. Since the user gives the telegram already. Once the user hit send, an instant of ADLP10 class is created, and the method EstablishContactAndSendTelegram(SerialPort, Telegram) is called where the first argument is the serial port to be used for the connection and the second one is the telegram to be sent. Next, this method will arrange the telegram and prepare it to be sent according to ADLP10 protocol standards. Each byte is sent separately and after sending all the bytes in the telegram a checksum is created and sent to the corresponding ABB robot. Then, the class awaits for ABB feedback. In case the telegram was successfully sent, the class either prepare for receiving a telegram from the robot or terminates the communication, according to the nature of the initial command or message sent from the computer. On the other hand if the telegram was corrupted while sending it, the class will act according to the protocol and either send the telegram again or terminates the connection.

For the case of automatic communication, an instance of ARAP class is created and for each button in the automatic communication tab there is a corresponding method to convert that specific order to a telegram. After that the same procedure with the manual communication takes place to send the desired command.

Moving on to off-line path uploading, when the user open a text file expressed in the assembly level syntax and click on the upload button, the following series of actions happen in the background. First, the program is stored as an array where each element represents a byte. Then the ConstructTelegramsOfPathProgram(AssemblyProgram, ProgNumber) method in ARAP class takes both the number of the program and the program itself and divides the overall program into a number of telegrams to be sent in ordered manner. The number of telegrams depends solely on the size of the program (since the program can exceeds 128 bytes, which is the maximum length of one telegram). Then, ADLP10 class take charge in sending these telegram one at a time according to the rules put by ABB. If in the process of uploading the program something goes wrong in the communication, ADLP10 class will inform the user and terminates the process.

Consequently, if the user prefers to use the second syntax type describes earlier, the software will first convert the program into its assembly language format using ConvertToAssembly(Program) method. Then, the same procedure described in the paragraph above takes place.

CHAPTER 5

SAMPLE RUNS AND EXPERIMENTAL RESULTS

5.1 Introduction

In this chapter three sample workpieces are presented to test the practical functionality of the study. First part will show the result of generating a path for edge scanning. The second one is presented to show the capability of the software to generate a path for a relatively crowded part, in terms of edge numbers and shapes, on one plane. Lastly, the third part will provide an example of an overall collision-free path that can process different edges in different planes and elevations.

The test setup consists of the ABB IRB2000, working table and the workpiece to be processed. In addition, for the two last parts, a spindle holder and a flex attachment spindle (Dremel 225) are used, see Figure 5-1. As for the computer connected to ABB IRB200 it has an Intel® Core™ i7-4700HQ CPU processor working at 2.4 GHz.

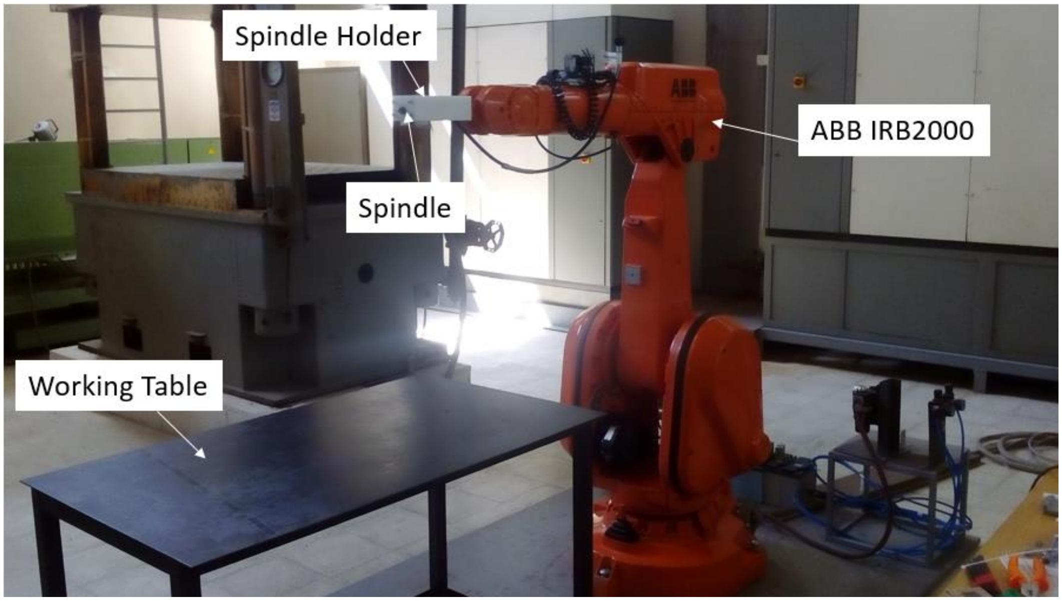


Figure 5-1: Test Setup Illustration.

The spindle holder consists of two pieces that are attached to each other by four M6 screw rods. Check Appendix D for the technical drawing of the spindle holder.

5.2 Sample Part 1

This part is an arbitrary part that was manufactured for the sole purpose of testing and confirming the overall algorithm of the study, see Figure 5-2. The bounding box of the part is 125mmx95mmx70mm. For the full technical drawing refer to appendix D.

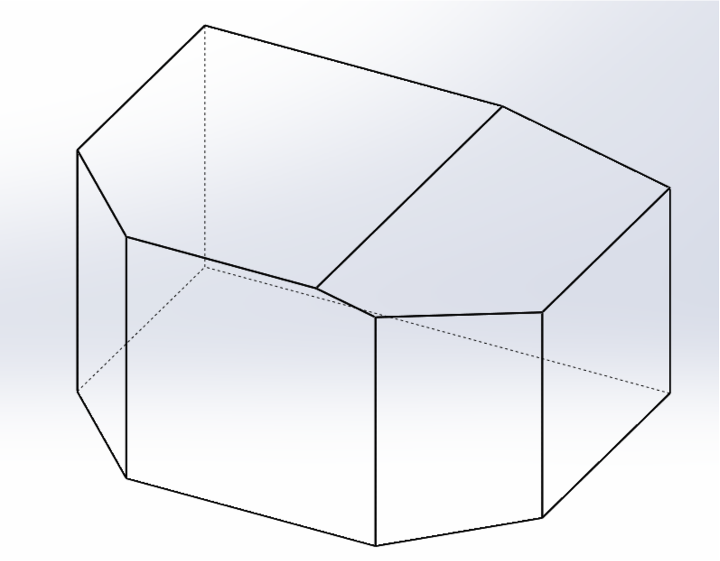


Figure 5-2: Trimetric view of part 1.

The objective is to generate a path for the end-effector in order to scan the edges of the given part. The part is not directly mounted on the working table. Instead, a smaller piece acts like a base, is mounted on the working table and then the part is placed on top of it as shown in Appendix F. This is done just to give the robot extra freedom to reach more edges.

First step is defining the position of the part with respect to the global coordinates system. This is done by defining a reference frame and attach it to the part. Figure 5-3 shows such reference frame.

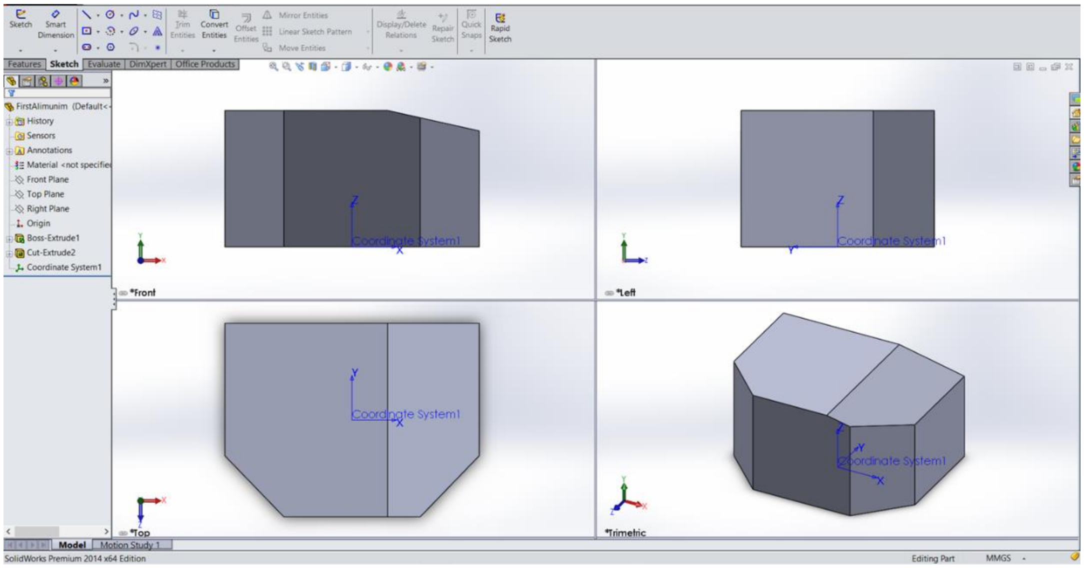


Figure 5-3: The defined reference frame is shown as Coordinate System1.

Next, the ground face is selected, as seen in Figure 5-4.

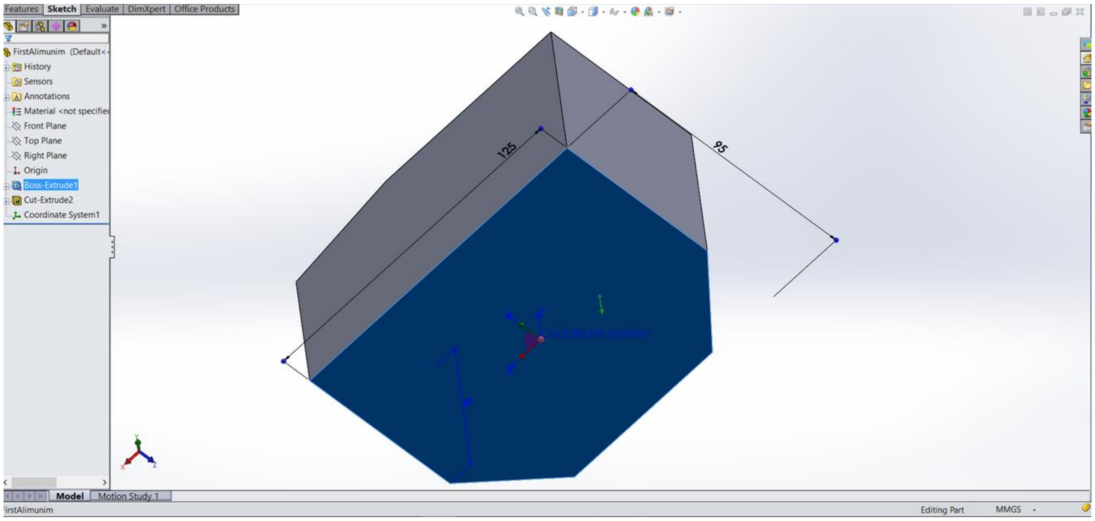
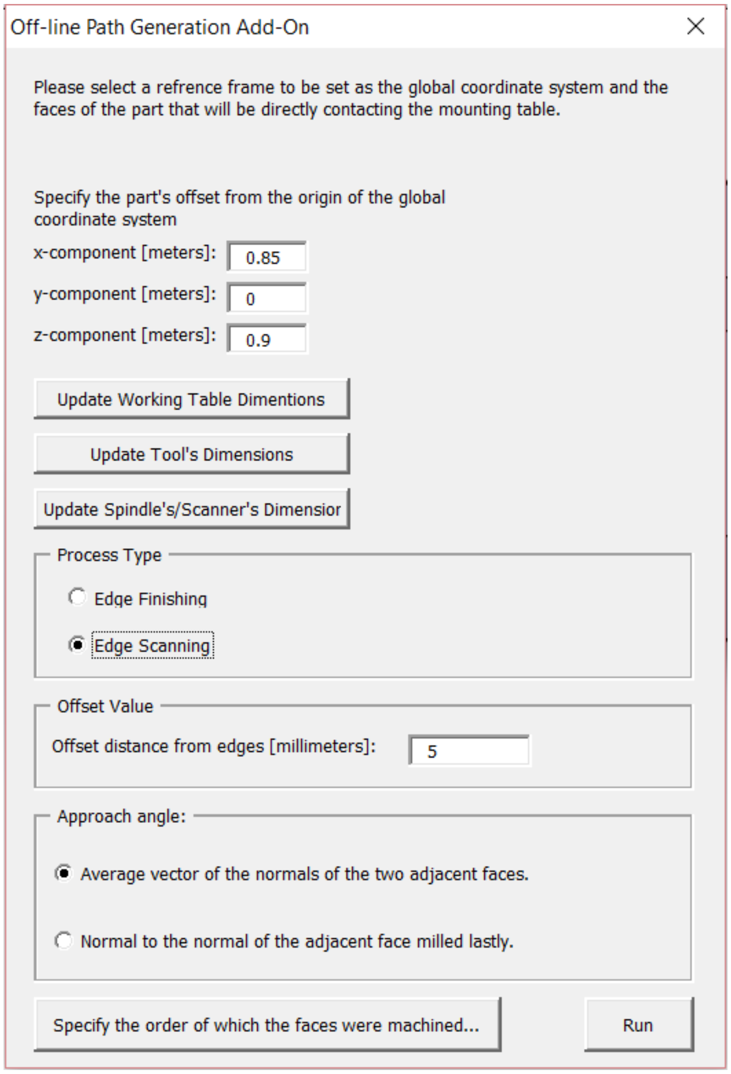


Figure 5-4: Selecting the ground face.

Then, following the steps explained in chapter four, the user input were selected as follows, see Figure 5-5:

- Process Type: Scanning

- offset value: 5mm
- offset direction: equal to the average of the normals of the two adjacent faces of each edge



Off-line Path Generation Add-On

Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

x-component [meters]: 0.85

y-component [meters]: 0

z-component [meters]: 0.9

Process Type

○ Edge Finishing

● Edge Scanning

Offset Value

Offset distance from edges [millimeters]: 5

Approach angle:

● Average vector of the normals of the two adjacent faces.

○ Normal to the normal of the adjacent face milled lastly.

Update Working Table Dimentions
Update Tool's Dimensions
Update Spindle's/Scanner's Dimensior
Specify the order of which the faces were machined...
Run

Figure 5-5: User input for scanning Part one.

The part has a total of 21 edges, all convex. Excluding the six ground edges, the software tries to generate a path to scan the remaining 15 edges. However, two of the edges turns out to be unreachable in the given orientation, shown in red in Figure 5-6. Therefore, the overall generated path goes over 13 edges of the part. The order at which these edges are processed is given in Figure 5-6. The software took 19 seconds to generate the output motion.

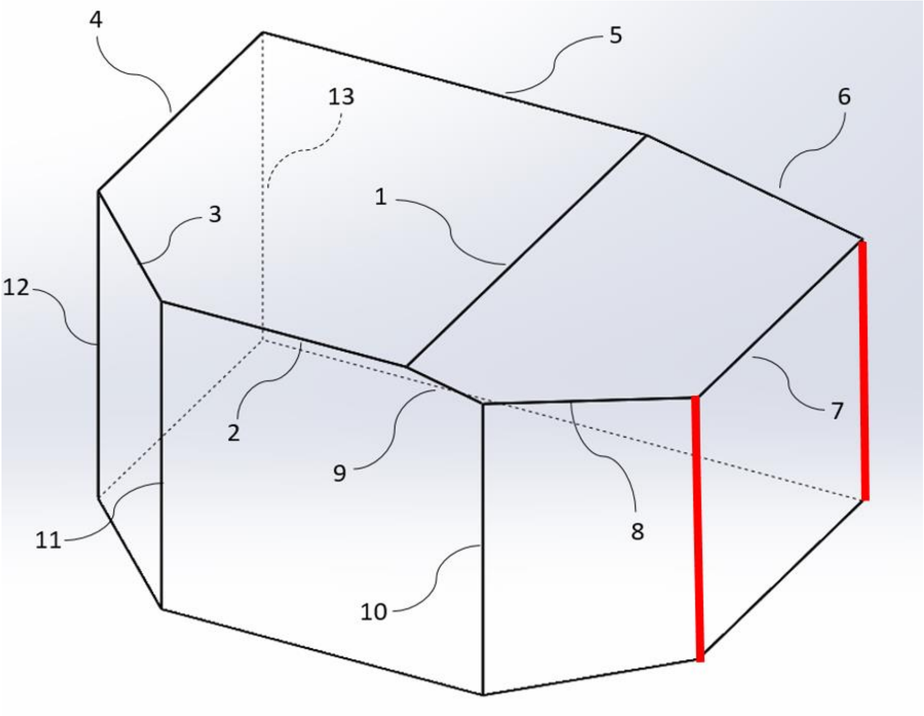


Figure 5-6: The order of which the edges of part one are scanned.

The complete generated path for scanning the mentioned 13 edges can be found in Appendix E.

5.3 Sample Part 2

This part represents the general characteristics of a casing part, i.e, pump casing, engine casing etc., see Figure 5-7. The bounding box of the part is 250mmx200mmx23mm. For the full technical drawing refer to appendix D.

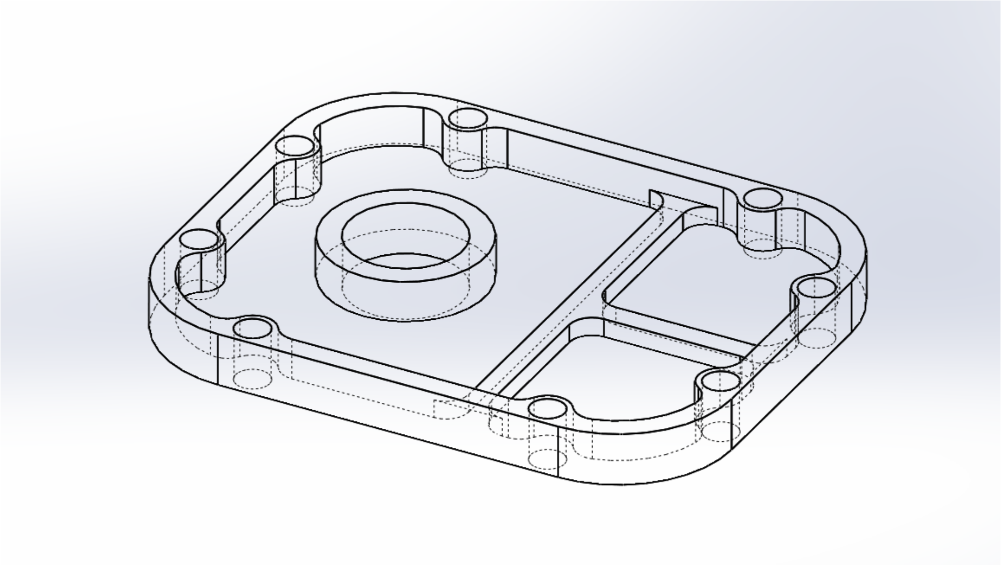


Figure 5-7: Trimetric view of part 2.

The objective is to generate a path for the end-effector in order to deburr the edges of the given part. Since no actual deburring is taking place the part is directly mounted in the working table without fixing it to a fixture.

First step is defining the position of the part with respect to the global coordinates system. This is done by defining a reference frame and attach it to the part. Figure 5-8 shows such reference frame.

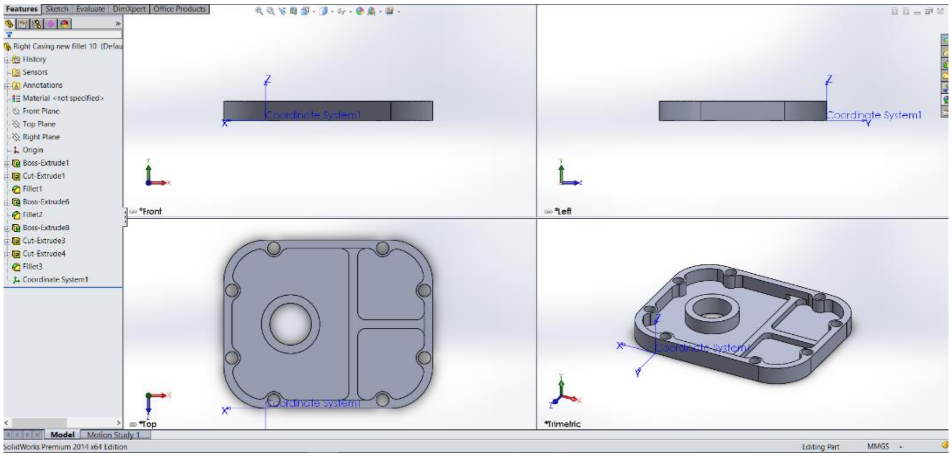


Figure 5-8: The defined reference frame is shown as Coordinate System1.

Next, the ground face is selected, as seen in Figure 5-9.

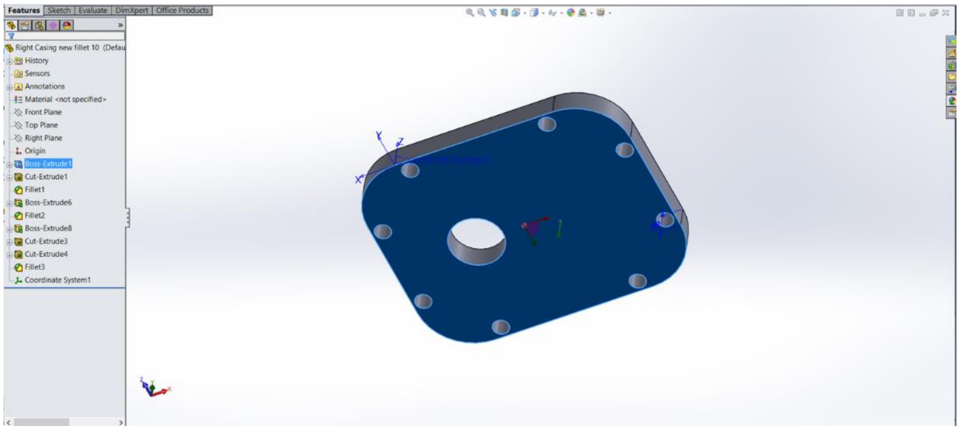
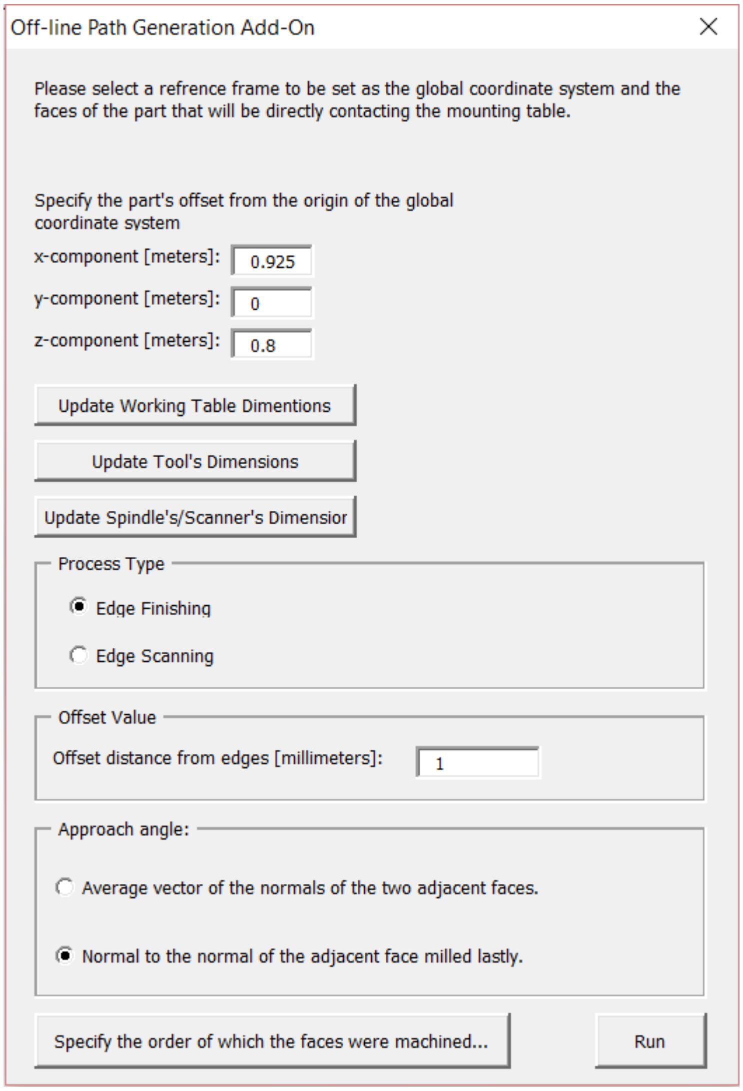


Figure 5-9: Selecting the ground face.

Then, following the steps explained in chapter four, the user input were selected as follows, see Figure 5-10:

- Process Type: Finishing
- offset value: 1mm
- offset direction: Normal to the formation of burrs



Off-line Path Generation Add-On

Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

x-component [meters]: 0.925

y-component [meters]: 0

z-component [meters]: 0.8

Update Working Table Dimentions
Update Tool's Dimensions
Update Spindle's/Scanner's Dimensior

Process Type

● Edge Finishing

○ Edge Scanning

Offset Value

Offset distance from edges [millimeters]: 1

Approach angle:

○ Average vector of the normals of the two adjacent faces.

● Normal to the normal of the adjacent face milled lastly.

Figure 5-10: User input for scanning Part two.

The part has a total of 71 convex edges. Excluding the eight ground edges, the software tries to generate a path to process the remaining 63 edges. All these edges are actually reachable. Therefore, the overall generated path goes over all of the 63 edges. The order at which these edges are processed is given in Figure 5-11.Moreover, the software took 45 seconds to generate the overall path.

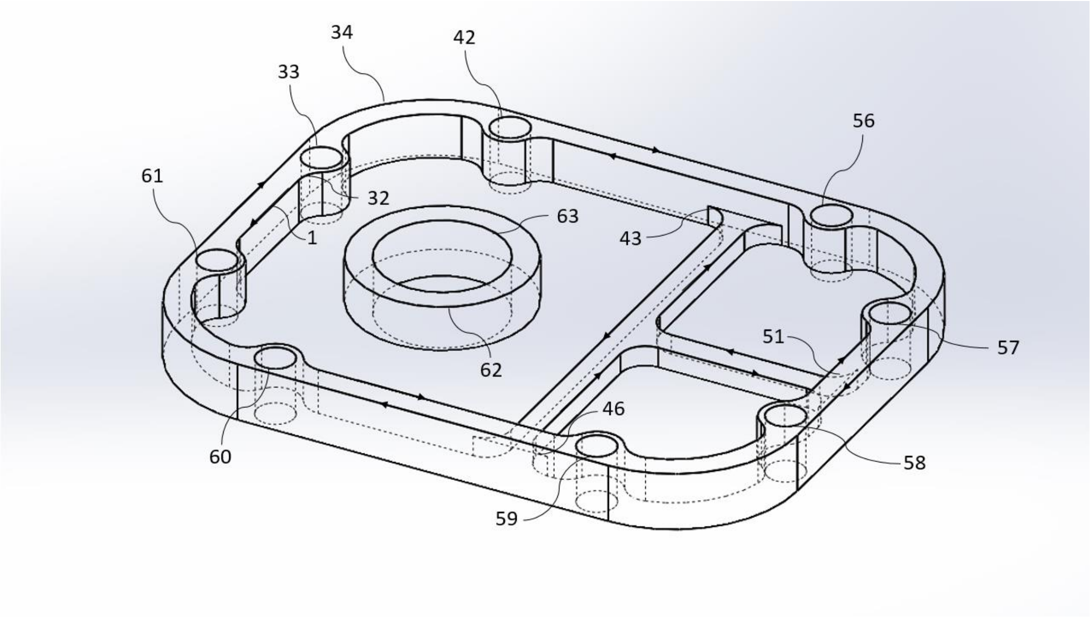


Figure 5-11: The order of which the edges of part two are deburred.

The complete generated path for deburring the mentioned 63 edges can be found in Appendix E.

5.4 Sample Part 3

This part represents the general characteristics of a V6 engine block, see Figure 5-12. The bounding box of the part is 200mmx150mmx100mm. For the full technical drawing refer to appendix D.

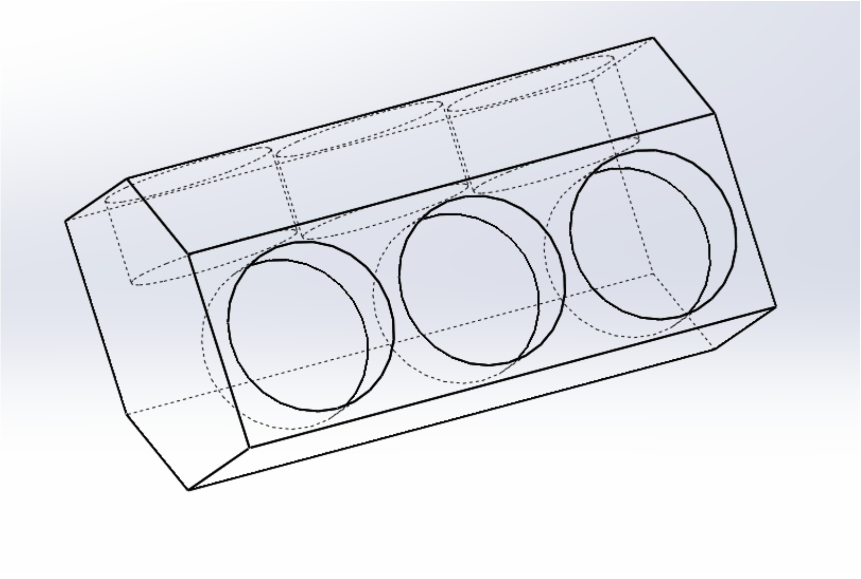


Figure 5-12: Trimetric view of part 3.

The objective is to generate a path for the end-effector in order to deburr the convex edges of the given part. Since no actual deburring is taking place the part is directly mounted in the working table without fixing it to a fixture.

First step is defining the position of the part with respect to the global coordinates system. As discussed earlier, this is done by defining a reference frame and attach it to the part. Figure 5-13 shows such reference frame.

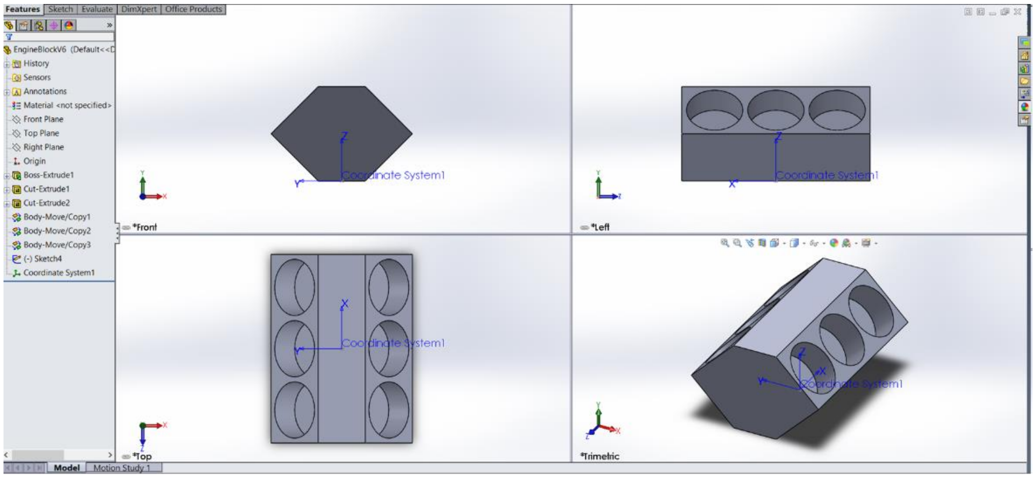


Figure 5-13: The defined reference frame is shown as Coordinate System1.

Next, the ground face is selected, as seen in Figure 5-14.

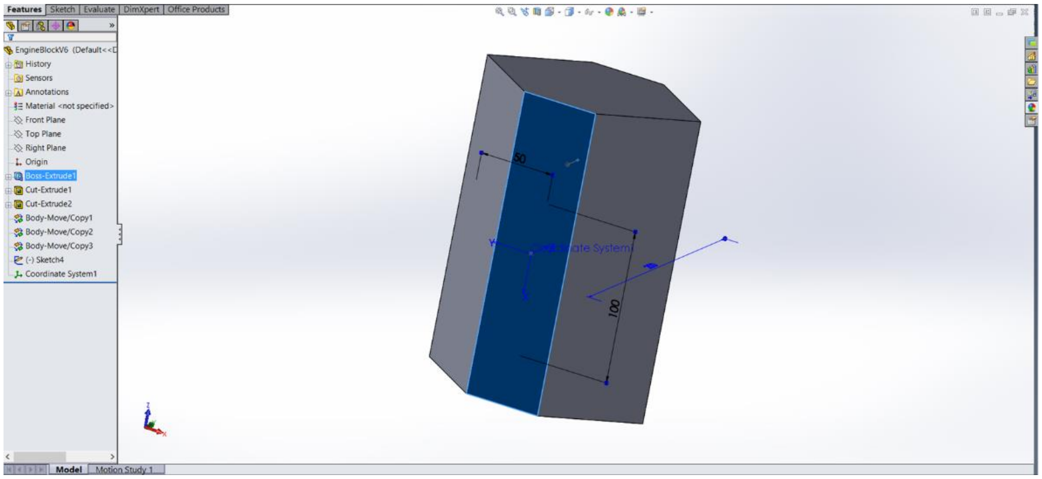
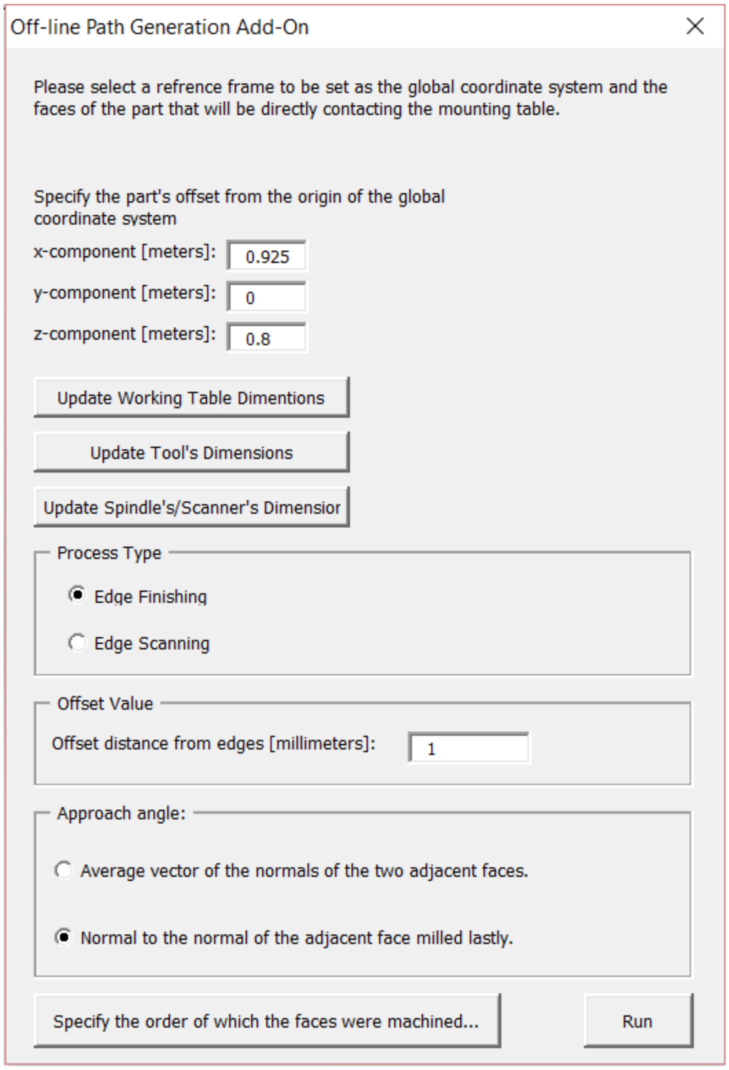


Figure 5-14: Selecting the ground face.

Then, following the steps explained in chapter four, the user input were selected as follows, see Figure 5-15:

- Process Type: Finishing
- offset value: 1mm
- offset direction: Normal to the formation of burrs


Off-line Path Generation Add-On

Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

x-component [meters]: 0.925

y-component [meters]: 0

z-component [meters]: 0.8

Update Working Table Dimentions
Update Tool's Dimensions
Update Spindle's/Scanner's Dimensior

Process Type

● Edge Finishing

○ Edge Scanning

Offset Value

Offset distance from edges [millimeters]: 1

Approach angle:

○ Average vector of the normals of the two adjacent faces.

● Normal to the normal of the adjacent face milled lastly.

Figure 5-15: User input for scanning Part three.

The part has a total of 24 convex edges. Excluding the four ground edges, the software tries to generate a path to process the remaining 20 edges. Four of these edges turn to be unreachable in the given orientation by the end-effector, shown in red in Figure 5-16. Therefore, the overall generated path goes over 16 edges. The order at which these edges are processed is given in Figure 5-16. Where the time needed by the software to complete the path generation was 26 seconds.

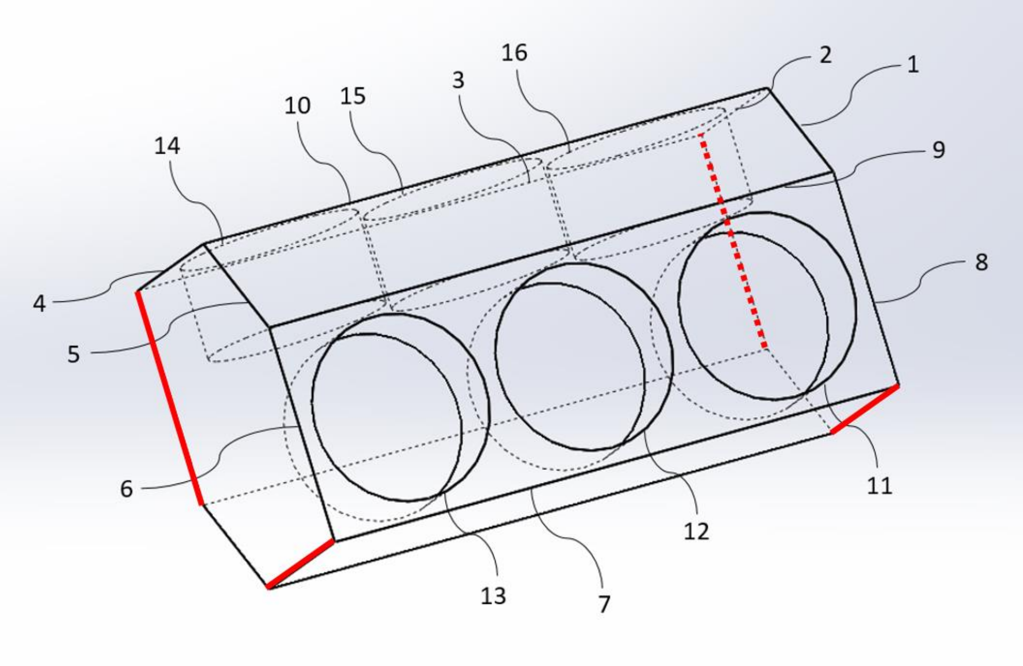


Figure 5-16: The order of which the edges of part three are deburred.

The complete generated path for deburring the mentioned 16 edges can be found in Appendix E.

CHAPTER 6

CONCLUSION

6.1 Discussion and Summary

The main goal of this study is to automatically generate a nominal collision-free path for 6-DoF robotic arm for performing edge finishing and scanning processes. This objective was achieved by developing a software in the SolidWorks API using Visual Basic programming language. ABB IRB2000 model was considered as the robotic manipulator throughout the study. The performance of the software was verified by running tests on several workpieces.

The system has five main logical steps for reaching the main objective, as described in Chapters 2, 3, and 4. In the first step, all the processing motion segments are generated and the ones that are collision-free are stored.

In the second step, all the starting and ending positions of each processing motions are recognized and stored, referred to as critical positions, mainly for the generation of non-processing motions, since each non-processing motion connects two of these critical positions.

In the third step, all possible non-processing motion segments are generated and again the ones with no collision are stored in memory. This step is considered as one of the main contributions of the study. The RFA motion approach reduces the complexity of both calculating a motion between two positions and the number of calculations needed for constructing all possible non-processing motions.

In the fourth step, a greedy search algorithm is utilized to find a suitable subset of non-processing motions from the list of all possible non-processing motions. Such subset when put together with the collision-free processing motions results in a connected overall path that the robot is meant to execute.

In the fifth step, the overall path is transformed into motion commands and sent to the ABB IRB2000 as a motion program that get stored in the memory of the robot and can be executed at any time.

The study was verified by performing three sample runs on different workpieces. In all three runs the algorithm was able to generate a path in less than a minute for each workpiece, making it a fast off-line path generator algorithm for edge finishing and scanning. Compared with the classical point-to-point teaching method, which takes several hours to a day to perform, depending on the skill of the worker and the complexity of the piece. While other off-line path generation approaches using the CAD model of the considered workpiece, such as [2], need a processing time around three hours on average.

6.2 Future Work

This study can be further extended to include any arbitrarily shaped edges. Different approaches may be applied to achieve this objective. Discretizing the 3D curve of the edge into small straight lines can be a solution. The sampling period may be of constant value or a variable one that depends on the curvature of the curve at the sampled point.

More algorithms can be applied to generate the non-processing motions between critical poses. RFA motion was developed and used in this study; however, many other different algorithms can be implemented and a comparison may be made. Examples of such algorithms can be artificial potential field approach [22], probabilistic road maps [21] and artificial force field method [23]. Moreover, RFA motion algorithm can be modified by utilizing one of the previously discussed algorithms to generate the retract/approach motions, making RFA more efficient.

Different search algorithms can be utilized for solving the TSP problem. Although the nearest neighbor algorithm gives a solution that on average is in the 25% of best solutions. Some other heuristic algorithms, like Match Twice and Stitch method (MTS) [30], discussed earlier, can be implemented.

REFERENCES

[1] T. Jawanjal and B. S., "An Advanced Chamfering System," *International Journal of Emerging Technology and Advanced Engineering,* vol. 3, pp. 598-601, 2013.

[2] F. Leali, M. Pellicciari and F. Pini, "An Offline Programming Method for the Robotic Deburring of Aerospace Components," *Robotics in Smart Manufacturing, Communications in Computer and Information Science,* vol. 371, pp. 1-13, 2013.

[3] O. Valente, "A New Approach for Tool Path Control in Robotic Deburring Operations," in *17th international Congress of Mechanical Engineering,* Sao Paulo, 2003.

[4] N. P. Murphy, "CAD Directed Robotic Deburring," in *Second International Symposium on Robotics and Manufacturing Research, Education and Applications*, 1988.

[5] S. Sugita, T. Itaya and T. Y. , "Development of robot teaching support devices to," *International Journal of Advanced,* vol. 23, p. 183–189, 2004.

[6] M. Elbestawi, G. Bone and P. Tam, "An Automated Planning, Control, and Inspection System for Robotic Deburring," *CIRP Annals - Manufacturing Technology,* vol. 41, pp. 397-401, 1992.

[7] N. Asakawa, K. Toda and Y. Takeuchi, "Automation of chamfering by an industrial robot; for the case of hole on free-curved surface," *Robotics and Computer Integrated Manufacturing,* vol. 18, p. 379–385, 2002.

[8] H. Zhang, H. Chen, N. Xi, G. Zhang and J. He, "On-Line Path Generation for Robotic Deburring of Cast Aluminum Wheels," in *Proceedings of the 2006*

IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, 2006.

[9] S. Lee, C. Li, D. Kim, J. Kyung and H. ChangSoo, "THE DIRECT TEACHING AND PLAYBACK METHOD FOR ROBOTIC DEBURRING SYSTEM USING THE ADAPTIVEFORCE-FORCE-," in *2009 IEEE International Symposium on Assembly and Manufacturing*, Suwon, 2009.

[10] H. Song, K. Byeong-Sang and S. Jae-Bok, "Tool Path Generation based on Matching between Teaching Points and CAD Model for Robotic Deburring," in *The 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Kaohsiung, 2012.

[11] G. Ziliani, A. Visioli and G. Legnani, "A mechatronic approach for robotic deburring," *Mechatronics,* vol. 17, p. 431–441, 2007.

[12] L. Princely and S. T, "Vision Assisted Robotic Deburring of Edge Burrs in Cast Parts," *Procedia Engineering,* vol. 97, pp. 1906-1914, 2014.

[13] C. M. Rejc, "Dimensional measurements of a gray-iron object using a robot and a laser displacement sensor," *Robotics and Computer-Integrated Manufacturing,* pp. 155-167, 2009.

[14] W. Jayaweera, "Measurement Assisted Automated Robotic Edge Deburring of Complex Components," in *9th WSEAS international conference on Signal processing, robotics and automation*, 2010.

[15] S. Xi, "CAD-based path planning for 3-D line laser scanning," *Computer-Aided Design,* p. 473–479, 1999.

[16] K. L. Son, "Path planning of multi-patched freeform surfaces for laser scanning," *The International Journal of Advanced Manufacturing Technology,* pp. 424-435, 2003.

[17] M. P. Morozov, "Computer-Aided Tool Path Generation for Robotic Non-Destructive Inspection," in *52nd Annual Conference of the British Institute for Non-Destructive Testing*, Telford, 2015.

[18] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans ASME J. Appl. Mech,* vol. 23, p. 215–221, 1955.

[19] H. B. J. Kazerooni and B. Kramer, "An Approach to Automated Deburring by Robot Manipulators," *Journal of Dynamic Systems, Measurement, and Control* , vol. 108, pp. 353-359, 1986.

[20] S. Lavalle, Planning Algorithms, Cambridge: Cambridge University Press, 2006.

[21] L. E. Kavraki, P. Svestka, J.-C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation,* vol. 12, p. 566–580, 1996.

[22] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research,* vol. 5, p. 90–98, 1986.

[23] P. Chotiprayanakul, D. K. Liu, D. Wang and D. G., "A 3-Dimensional Force Field Method for Robot Collision Avoidance in Complex Environments," in *24th Internationa Symposium on Automation & Robotics in Construction*, Madras, 2007.

[24] D. AppleGate, R. Bixby, V. Chvatal and W. Cook, "On the Solution of the Traveling Salesman Problems," *Documenta Mathematica – Extra Volume ICM, chapter 3,* pp. 645-656, 1998.

[25] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* John Wiley & Sons, 1985.

[26] N. Christofides, *Worst-case analysis of a new heuristic for the travelling salesman problem*, Report 388, Graduate School of Industrial Administration, CMU, 1976.

[27] S. Arora, *Polynomial Time Approximation Schemes for Euclidean Traveling Saleman and Other Geometric Problems,* Princeton University, 1996.

[28] S. Arora, *Nearly Linear Time Approximation Schemes for Euclidean TSP and other Geometric Problems,* Princeton University, 1997.

[29] J. S. B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions_Part II - A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems,* State University of New York, Stony Brook, 1996.

[30] A. B. Kahng and S. Reda, "heuristic, Match twice and stitch: a new TSP tour construction," *Operations Research Letters,* vol. 32, p. 499–509, 2004.

[31] *Asea Robotics,* Quaternions or how to twist a robot, October, 1987.

[32] ABB-IRB2000, *Product Manual,* 1992.

[33] *ABB Robotics Products,* Computer Link / ADALP 10 / ARAP S3, 3HAA 3911-102, January, 1992.

APPENDIX A

METHODS DEVELOPED AND USED IN THE SOFTWARE

A.1 SolidWorks Add-Ons Package

This software was developed in SolidWorks API using Visual Basic programming language. Therefore, an object oriented approach is adopted. All the methods belong to one of the main classes of the code, namely, MainWindow and PathGeneratorEngine. The general purpose of each class is as follows:

- MainWindow: Responsible for the graphical interface with user, both for inputs and outputs.
- PathGeneratorEngine: Calculates both the processing and non-processing motions of the end-effector and checks if any of these motions have collisions. It also generates the overall path by selecting a subset from the non-processing motions in order to yield a connected motion.

A.1.1 MainWindow Methods

SelectEdges(): Performs the default edge selection.

OrderFaces(ByRef swSelectMgr As SldWorks.SelectionMgr): This methods assigns the order of which the faces were machined. This is used when the approach angle is chosen to be normal to the first machined face between the two adjacent faces of the given edge.

CreateTxtFile(Collection OverallPath): Creates a text file of the overall path in the syntax given by ABB mentioned earlier.

A.1.2 PathGeneratorEngine Methods

CalculateProcessMotions(NumberOfLines As Integer, l() As Line): Calculates the processing motion of each selected edge.

InverseKinematics(Rot As Variant, r As Variant): this method performs the positional inverse kinematics of ABB IRB2000 for a given position.

BuildLink(ByVal Link As LinkPose, ByVal LinkNumber As Integer): This method is called six times, each time for creating the cuboid representing one of the six links of the robot arm.

CalculateLinksPose(theta As Variant): This is the positional forward kinematics analysis. Where the methods decides on the position of each link for a given joint angles.

GetProcessMotionsOfReachableEdges(NumberOfLines As Integer, l() As Line): Check each processing motion for collisions and takes the ones that are collision-free.

GetCriticalPoses(Collection ProcessMotions): Creates a critical pose instant for each starting or ending position of a processing motion.

GetRetractionMotionFromPoseAndCheckCollision(Pose As CriticalPose): Calculates the retraction motion of the given critical pose and check it for collisions.

GenerateNonProcessMotionBetween(StartPose As CriticalPose, EndPose As CriticalPose): This method generates the non-processing motion between the given start and end positions.

GetOverallPath(ProcessMotion As Collection, AllNonProcessMotions As Motion()): This is for the search algorithm applied for selecting a suitable subset from the list of all non-processing motions to yield a final overall connected path.

A.2 ABB User Interface Software

As mentioned earlier this software was developed in C++ language. Therefore, an object oriented approach is adopted. All the methods belong to one of the main classes of the code, namely, MainWindow, ADLP10Protocol and ARAPProtocol. The general purpose of each class is as follows:

- MainWindow: Responsible for the graphical interface with user, both for inputs and outputs.
- ADLP10Protocol: Establishes the connection and maintains it as long as needed. Makes sure each telegram received by any party is fully understood and if not act accordingly. Finally it ends the connection once all the data is exchanged.
- ARAPProtocol: Transform the command given by the user into telegrams, which is later sent by ADLP10Protocol to the robot.

A.2.1 MainWindow Methods

OpenCloseSerialPort(QString COMname): Toggles the state of the serial port specified by COMname.

UpdateMsgBoxes(): This method checks which tab is activated an update all the widgets in that tab according to the data received by ABB.

A.2.2 ADLP10Protocol Methods

EstablishContactAndSendTelegram(QSerialPort *serial, QList<QByteArray> Telegram): Establish the connection between the computer and ABB robot and then send the list of telegrams in ordered manner.

ConcludeTelegram(QSerialPort *serial): Concludes the telegram by calculating the Bit Check Sum (BCS) and sending it to the robot followed by ETX.

DataRecieved(QSerialPort *serial): This method takes the received data and interprets it according to the state of the communication and decides if the received data is actual information or communication data and acts accordingly.

SendAcknowledgment(QSerialPort *serial): Sending ACK byte to ABB to acknowledge a previously sent telegram or order from the ABB.

TerminateContact(QSerialPort *serial): This method is used at the very end of the communication stage, it basically terminates the connection.

A.2.3 ARAPProtocol Methods

UpdateStatusCommand(): Constructs a telegram for function 19 of ARAP, which requires the robot to give its status.

GetAnglesCommand (): Constructs a telegram for function 44 of ARAP, which requires the robot to give an extended status message that includes the angles of its joints.

MovementCommand(): Constructs a telegram for function 24 of ARAP, which commands the robot to move to the given position. All the parameters of the motion is retrieved from the text boxes filled by the user.

GoHomeCommand(): Constructs a telegram for function 2 of ARAP, which commands the robot to execute a program in its memory. In this case it is 2202 program.

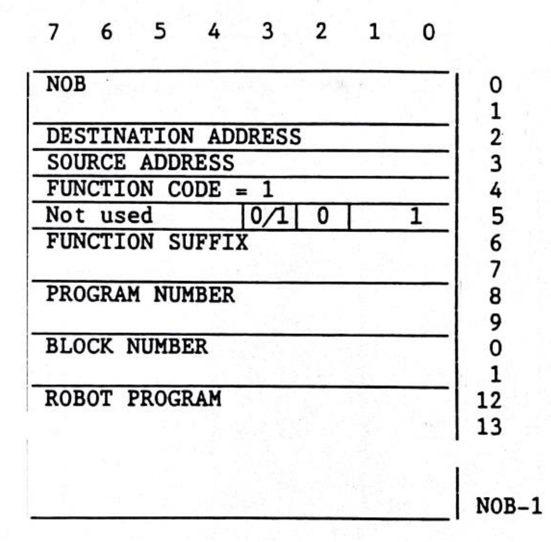
ConstructTelegramsOfPathProgram(QByteArray AssemblyProgram, QByteArray ProgNumber): Translate the normal syntax of the path program to assembly format.

APPENDIX B

USED ARAP PROTOCOL’S FUNCTIONS

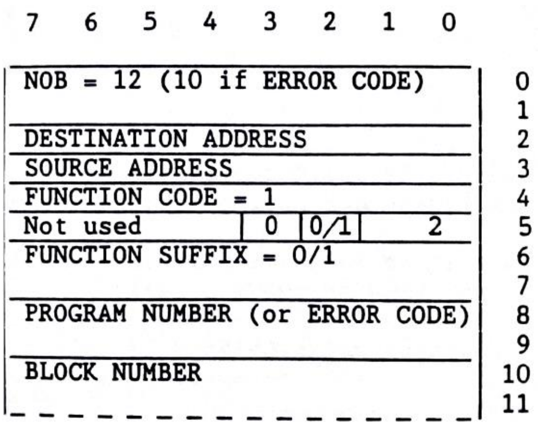
This appendix shows the syntax or structure of the functions, which is used in this study, to communicate and command the ABB IRB2000 robot. Each line is one byte of information. These telegrams are sent through a USB port.

Function 1: Transfer of program/ block of programs from the computer to robot. Used to transfer the path program.



7 6 5 4	3	2	1 0	
NOB				0 1
DESTINATION ADDRESS				2
SOURCE ADDRESS				3
FUNCTION CODE = 1				4
Not used	0/1	0	1	5
FUNCTION SUFFIX				6 7
PROGRAM NUMBER				8 9
BLOCK NUMBER				0 1
ROBOT PROGRAM				12 13
				NOB-1

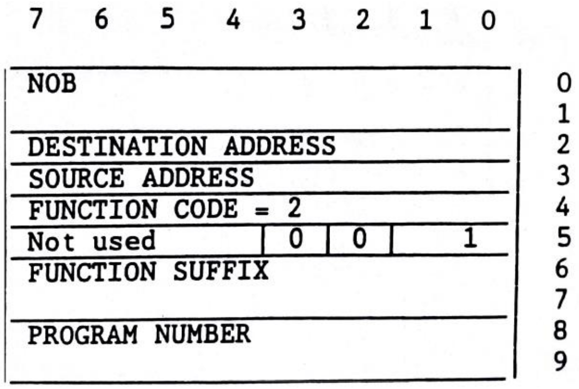
Figure B-1: Command telegram of function 1.



7	6	5	4	3	2	1	0	
NOB = 12 (10 if ERROR CODE)								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 1								4
Not used				0	0/1	2		5
FUNCTION SUFFIX = 0/1								6
								7
PROGRAM NUMBER (or ERROR CODE)								8
								9
BLOCK NUMBER								10
								11

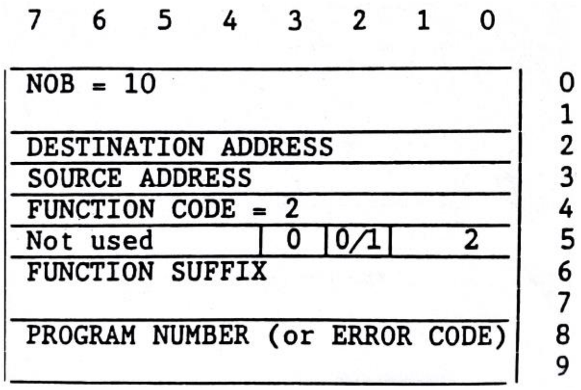
Figure B-2: Response telegram of function 1.

Function 2: Start of robot Program. Used to start the generated path and to command the robot to go to home position.



7	6	5	4	3	2	1	0	
NOB								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 2								4
Not used				0	0	1		5
FUNCTION SUFFIX								6
								7
PROGRAM NUMBER								8
								9

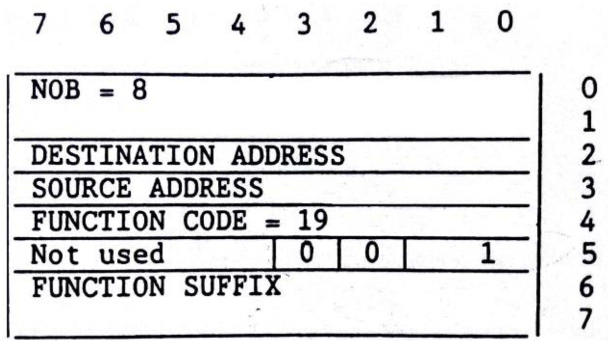
Figure B-3: Command telegram of function 2.



7	6	5	4	3	2	1	0	
NOB = 10								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 2								4
Not used				0	0/1	2		5
FUNCTION SUFFIX								6
								7
PROGRAM NUMBER (or ERROR CODE)								8
								9

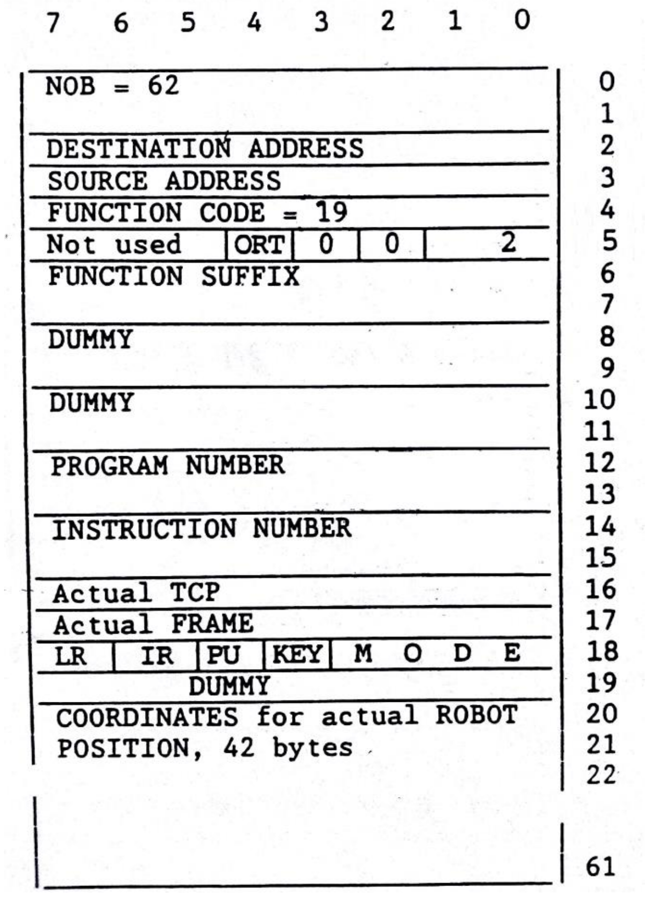
Figure B-4: Response telegram of function 2.

Function 19: Requests the status of the robot. Used to fill the information in the automatic communication tab in the ABB user interface software.



7	6	5	4	3	2	1	0	
NOB = 8								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 19								4
Not used				0	0	1		5
FUNCTION SUFFIX								6
								7

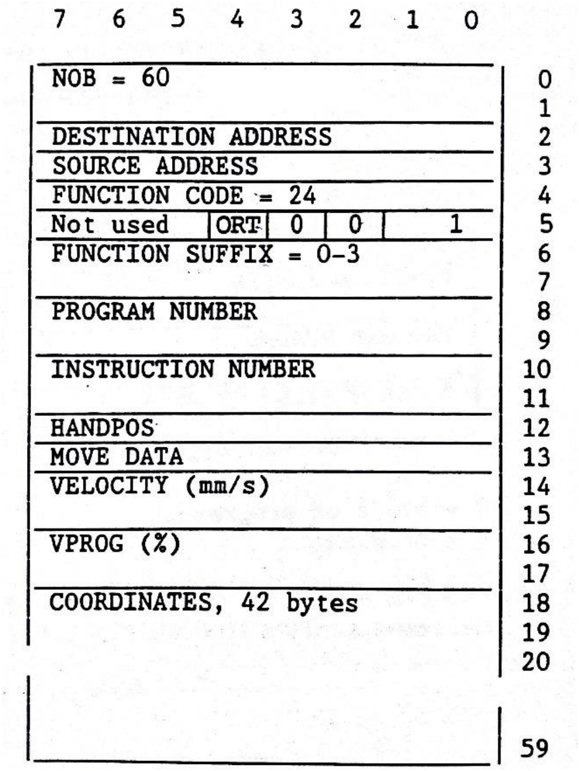
Figure B-5: Command telegram of function 19.



7	6	5	4	3	2	1	0	
NOB = 62								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 19								4
Not used			ORT	0	0	2		5
FUNCTION SUFFIX								6
								7
DUMMY								8
								9
DUMMY								10
								11
PROGRAM NUMBER								12
								13
INSTRUCTION NUMBER								14
								15
Actual TCP								16
Actual FRAME								17
LR	IR	PU	KEY	M	O	D	E	18
DUMMY								19
COORDINATES for actual ROBOT								20
POSITION, 42 bytes								21
								22
								61

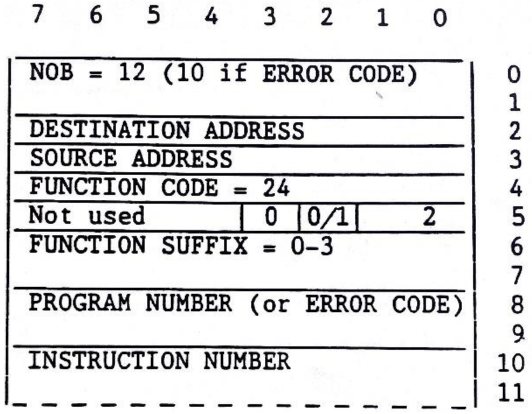
Figure B-6: Response telegram of function 19.

Function 24: Maneuvering of robot from the user computer. Used in the automatic communication tab in the ABB user interface software to command to the robot to go the desired position from its current position.



7	6	5	4	3	2	1	0	
NOB = 60								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 24								4
Not used			ORT	0	0	1		5
FUNCTION SUFFIX = 0-3								6
								7
PROGRAM NUMBER								8
								9
INSTRUCTION NUMBER								10
								11
HANDPOS								12
MOVE DATA								13
VELOCITY (mm/s)								14
								15
VPROG (%)								16
								17
COORDINATES, 42 bytes								18
								19
								20
								59

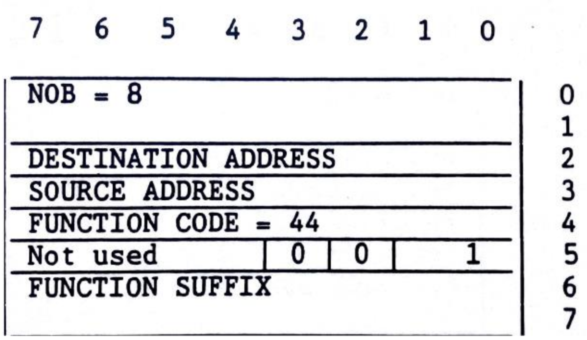
Figure B-7: Command telegram of function 24.



7	6	5	4	3	2	1	0	
NOB = 12 (10 if ERROR CODE)								0
								1
DESTINATION ADDRESS								2
SOURCE ADDRESS								3
FUNCTION CODE = 24								4
Not used				0	0/1	2		5
FUNCTION SUFFIX = 0-3								6
								7
PROGRAM NUMBER (or ERROR CODE)								8
								9
INSTRUCTION NUMBER								10
								11

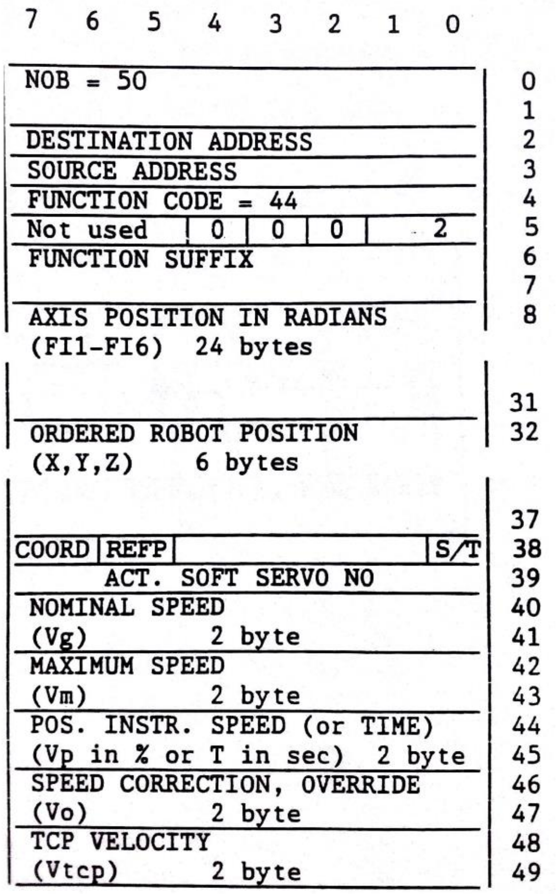
Figure B-8: Response telegram of function 24.

Function 44: Requests an extended status of the robot. Used to get the current joint angles values in the automatic communication tab in the ABB user interface software.



7 6 5 4 3 2 1 0	
NOB = 8	0
	1
DESTINATION ADDRESS	2
SOURCE ADDRESS	3
FUNCTION CODE = 44	4
Not used | 0 | 0 | 1	5
FUNCTION SUFFIX	6
	7

Figure B-9: Command telegram of function 44.

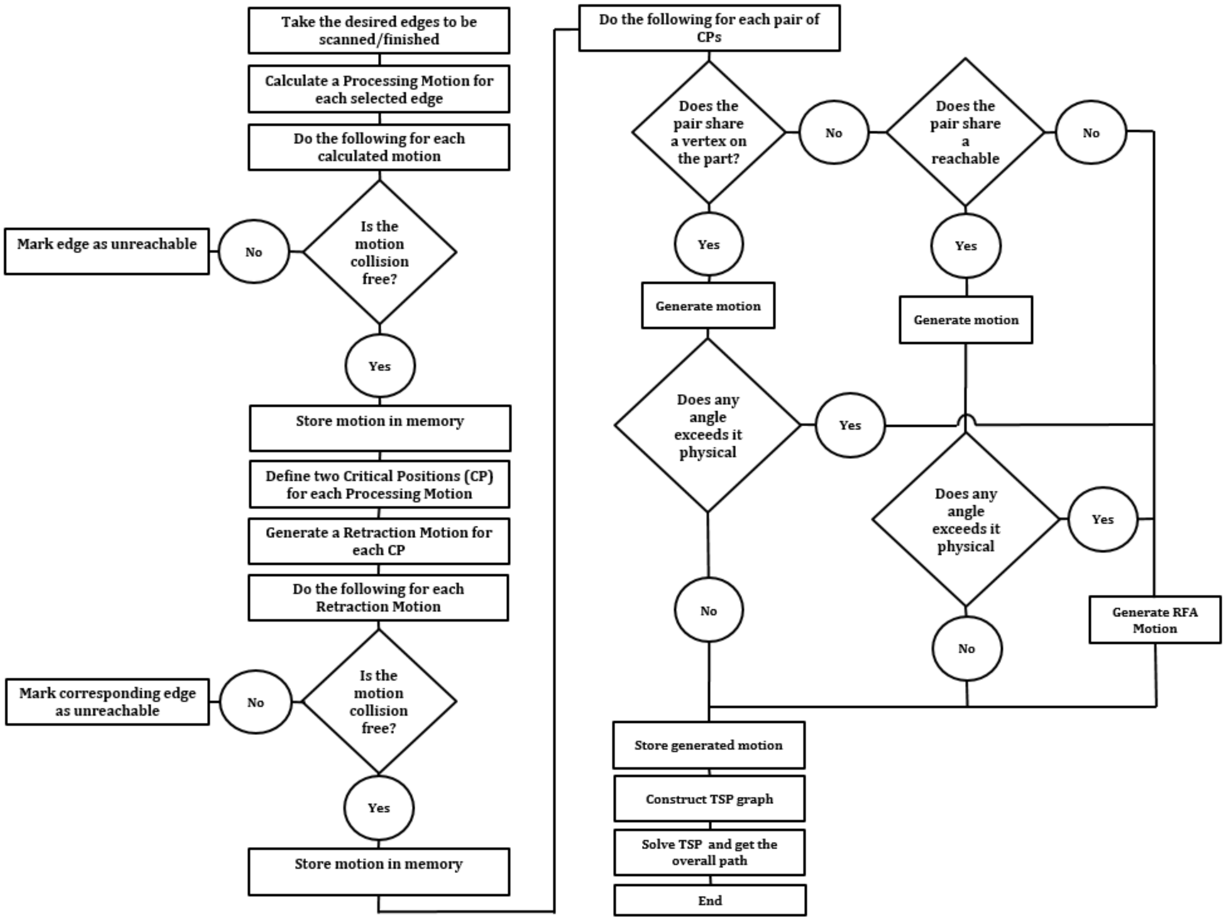


7 6 5 4 3 2 1 0	
NOB = 50	0
	1
DESTINATION ADDRESS	2
SOURCE ADDRESS	3
FUNCTION CODE = 44	4
Not used | 0 | 0 | 0 | 2	5
FUNCTION SUFFIX	6
	7
AXIS POSITION IN RADIANS	8
(FI1-FI6) 24 bytes	
	31
ORDERED ROBOT POSITION	32
(X,Y,Z) 6 bytes	
	37
COORD | REFP | | S/T	38
ACT. SOFT SERVO NO	39
NOMINAL SPEED	40
(Vg) 2 byte	41
MAXIMUM SPEED	42
(Vm) 2 byte	43
POS. INSTR. SPEED (or TIME)	44
(Vp in % or T in sec) 2 byte	45
SPEED CORRECTION, OVERRIDE	46
(Vo) 2 byte	47
TCP VELOCITY	48
(Vtcp) 2 byte	49

Figure B-10: Response telegram of function 44.

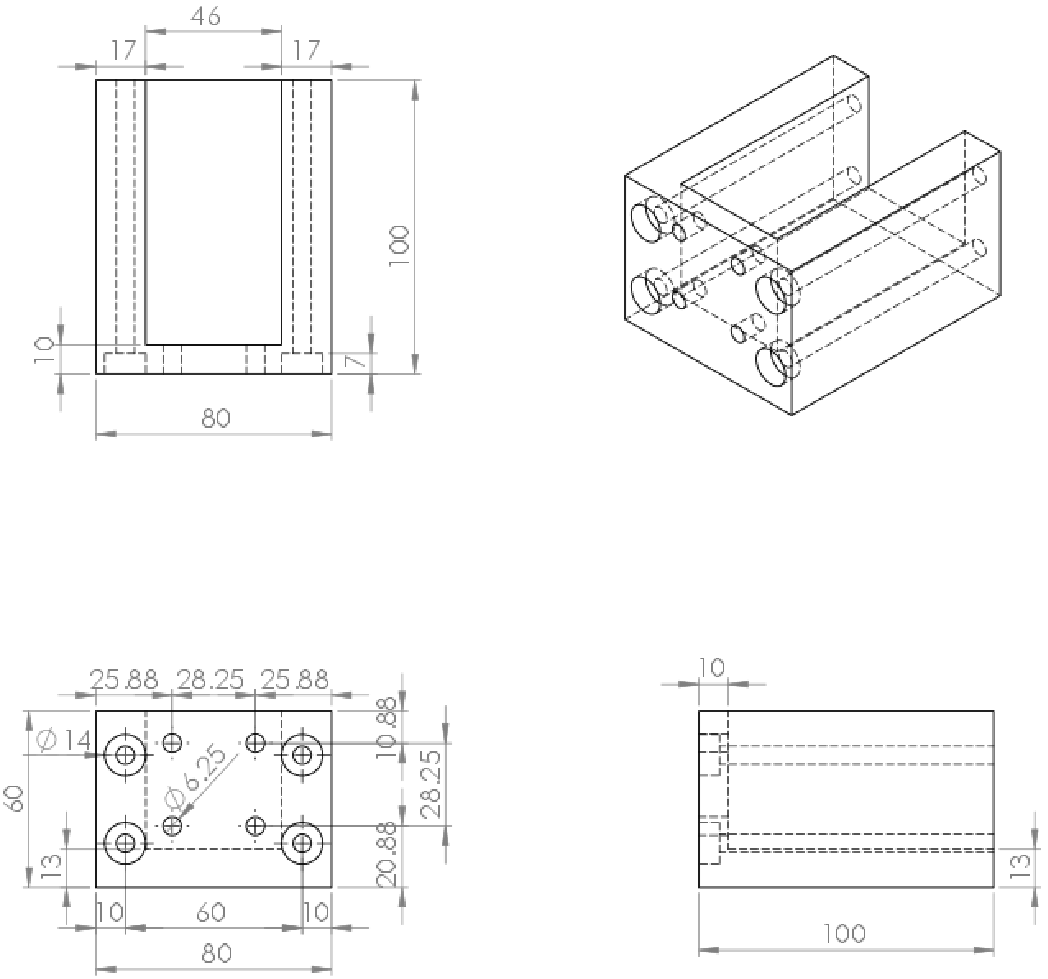
APPENDIX C

FLOWCHART OF PATH GENERATION ALGORITHM


Take the desired edges to be scanned/finished
Calculate a Processing Motion for each selected edge
Do the following for each calculated motion
Is the motion collision free?
No
Mark edge as unreachable
Yes
Store motion in memory
Define two Critical Positions (CP) for each Processing Motion
Generate a Retraction Motion for each CP
Do the following for each Retraction Motion
Is the motion collision free?
No
Mark corresponding edge as unreachable
Yes
Store motion in memory
Do the following for each pair of CPs
Does the pair share a vertex on the part?
No
Does the pair share a reachable
No
Yes
Generate motion
Yes
Generate motion
Does any angle exceeds it physical
Yes
Does any angle exceeds it physical
Yes
No
Generate RFA Motion
No
Store generated motion
Construct TSP graph
Solve TSP and get the overall path
End

APPENDIX D

TECHNICAL DRAWING



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBUR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Mahmoud Nemer		1/4/2016
CHK'D	Dr. Kanukseven		4/4/2016
APPV'D	Dr. Kanukseven		4/4/2016
MFG			
Q.A			

TITLE:

Spindle Holder, Part 2

MATERIAL:
Z-PCABS

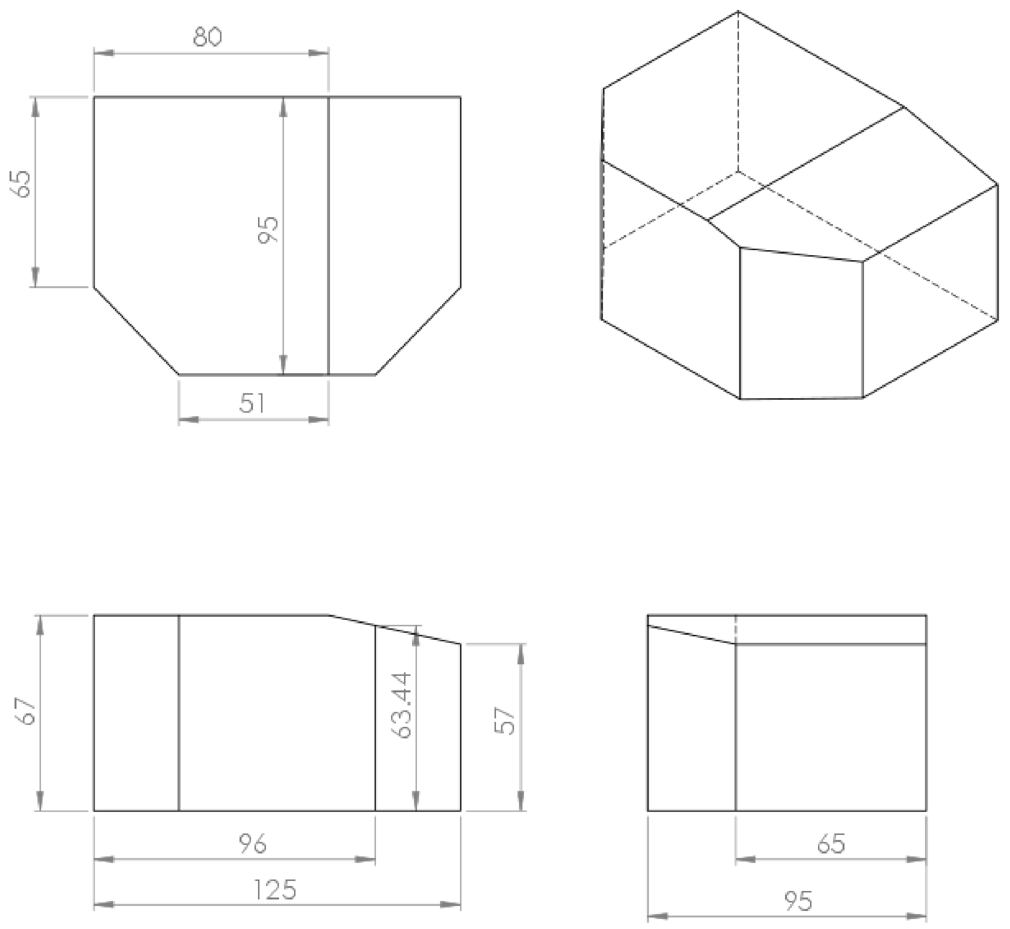
DWG NO.
1

A4

WEIGHT:

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBUR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Mahmoud Nemer		1/4/2016
CHK'D	Dr. Konukseven		4/4/2016
APPV'D	Dr. Konukseven		4/4/2016
MFG			
Q.A			

TITLE:

Part One

MATERIAL:
Aluminium

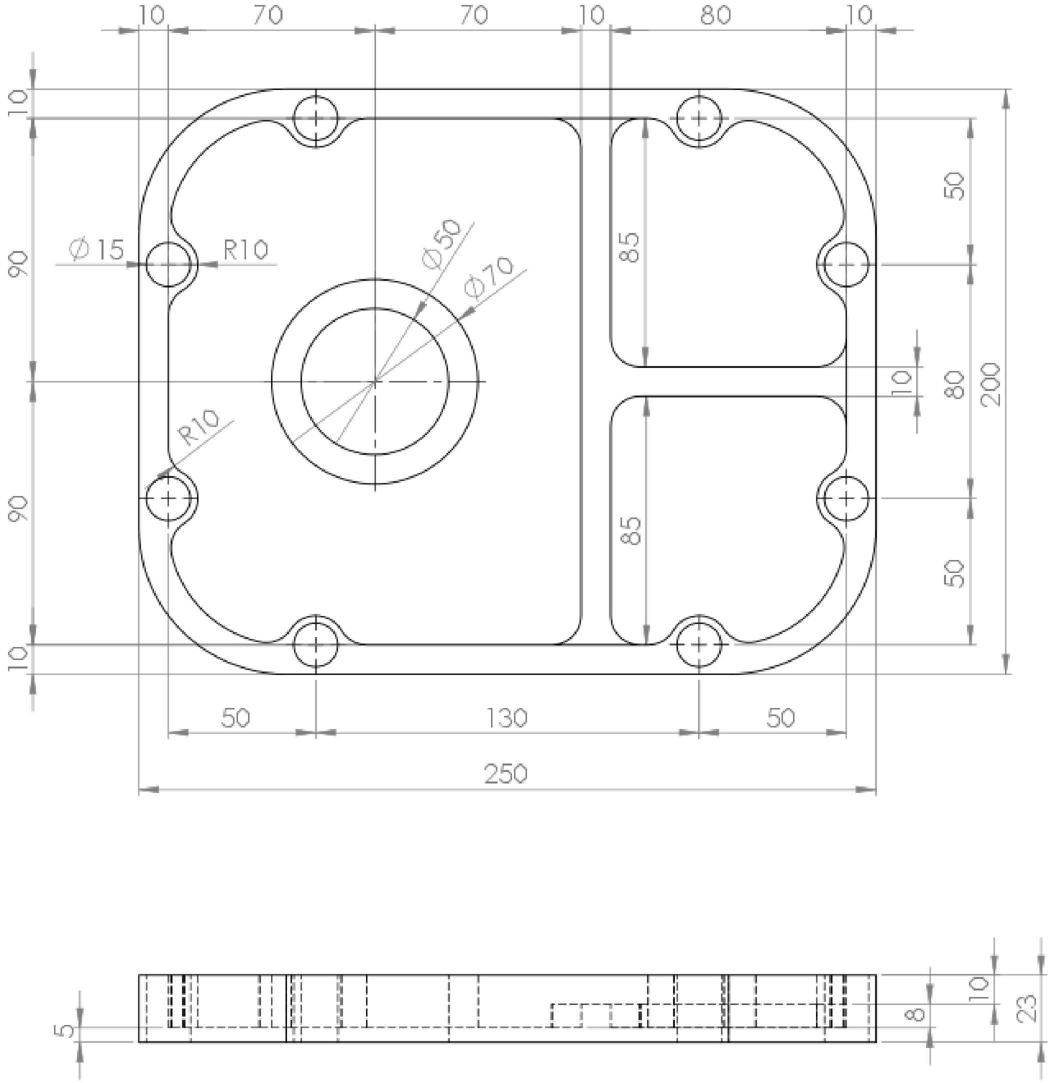
WEIGHT:

DWG NO.
1

A4

SCALE:1:2

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBUR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Mahmoud Nemer		1/4/2016
CHK'D	Dr. Konukseven		4/4/2016
APPV'D	Dr. Konukseven		4/4/2016
MFG			
Q.A			

TITLE:

Part Two

MATERIAL:
Aluminium

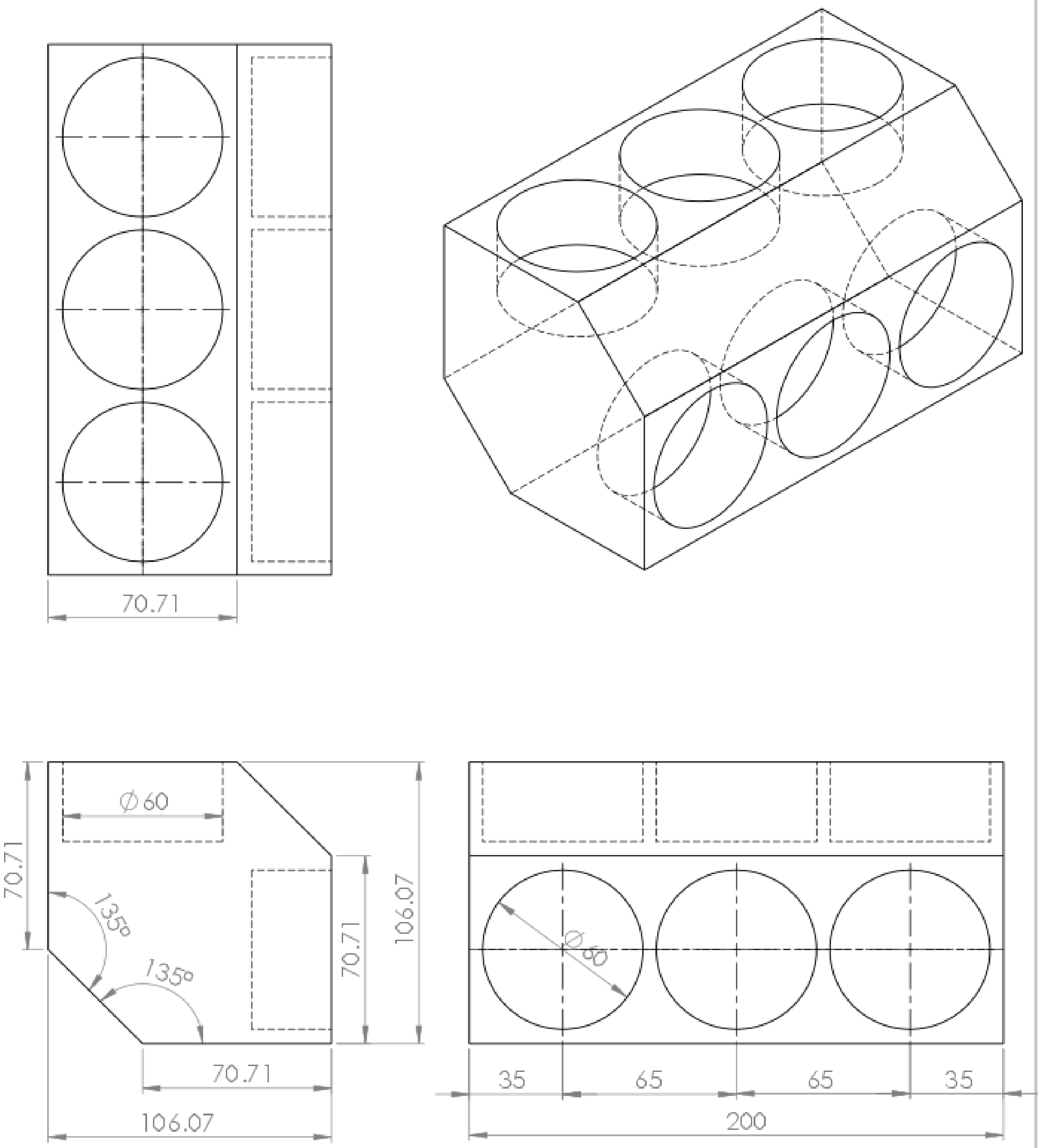
DWG NO.
1

A4

WEIGHT:

SCALE:1:1

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:

FINISH:

DEBUR AND
BREAK SHARP
EDGES

DO NOT SCALE DRAWING

REVISION

	NAME	SIGNATURE	DATE
DRAWN	Mahmoud Namer		1/4/2016
CHK'D	Dr.Konukseven		4/4/2016
APPV'D	Dr. Konukseven		4/4/2016
MFG			
Q.A			

TITLE:

Part Three

MATERIAL:
Aluminium

DWG NO.
1

A4

WEIGHT:

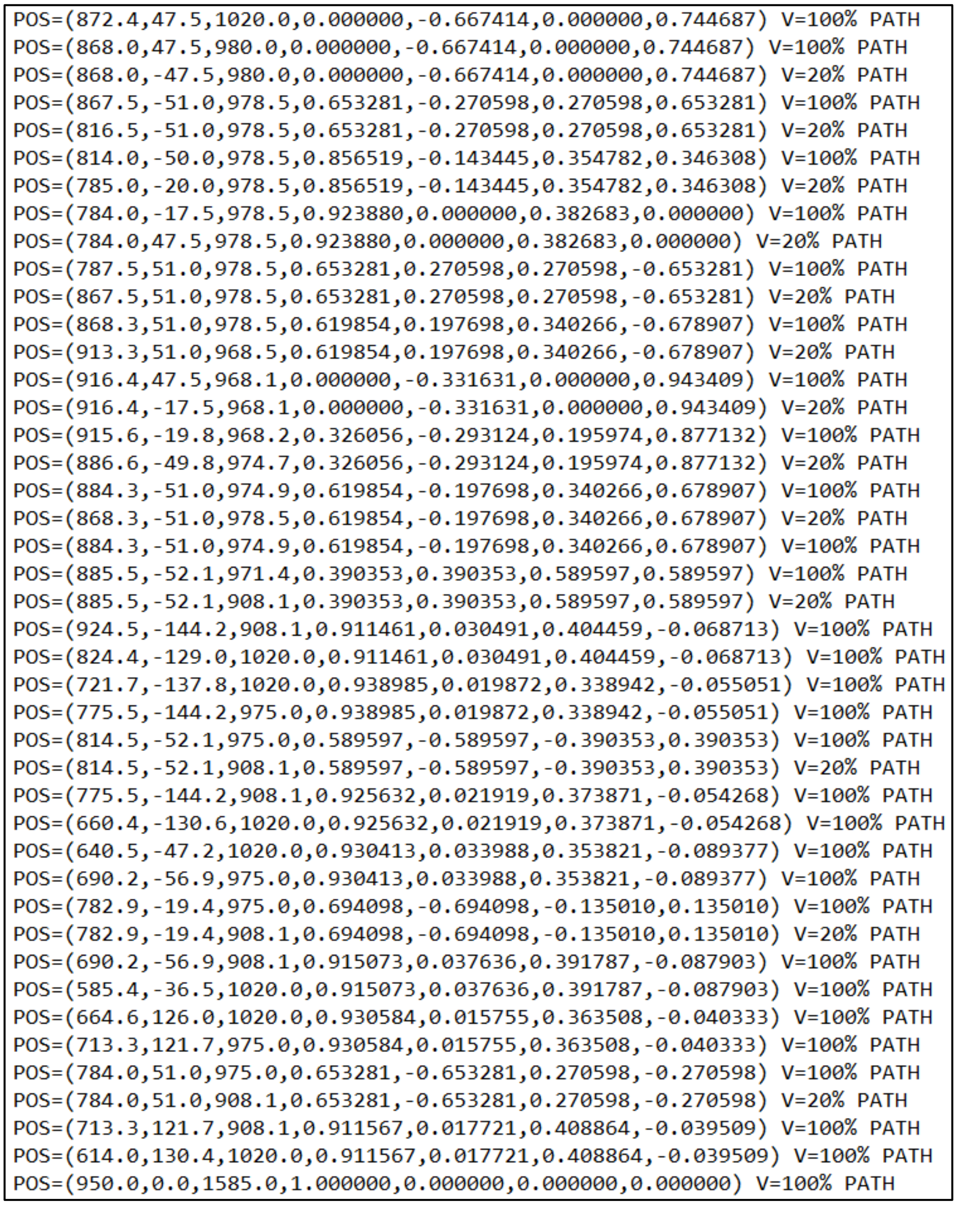
SCALE:1:1

SHEET 1 OF 1

APPENDIX E

GENERATED PATHS IN CHAPTER 5

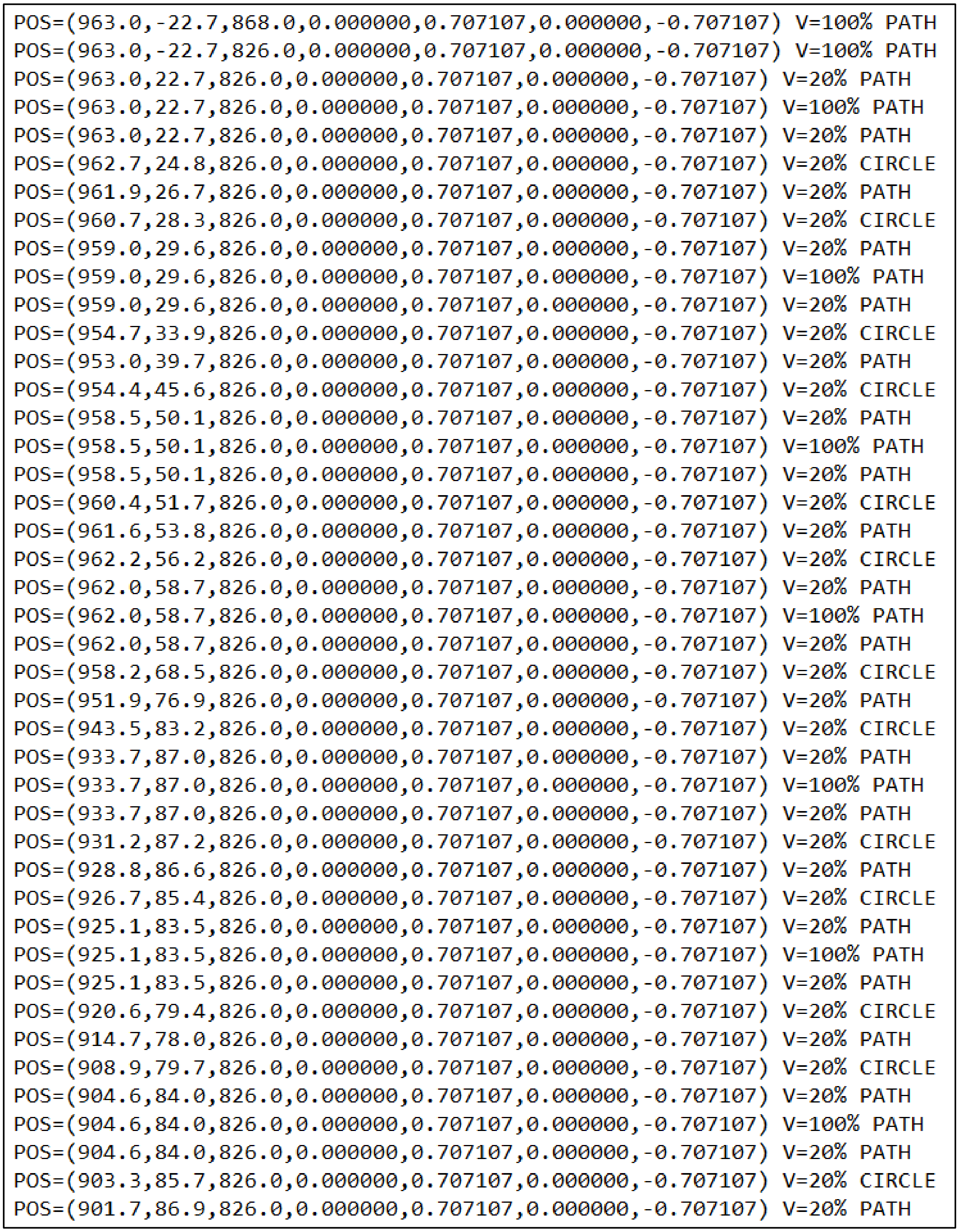
E.1 Part 1



```
POS=(872.4,47.5,1020.0,0.000000,-0.667414,0.000000,0.744687) V=100% PATH
POS=(868.0,47.5,980.0,0.000000,-0.667414,0.000000,0.744687) V=100% PATH
POS=(868.0,-47.5,980.0,0.000000,-0.667414,0.000000,0.744687) V=20% PATH
POS=(867.5,-51.0,978.5,0.653281,-0.270598,0.270598,0.653281) V=100% PATH
POS=(816.5,-51.0,978.5,0.653281,-0.270598,0.270598,0.653281) V=20% PATH
POS=(814.0,-50.0,978.5,0.856519,-0.143445,0.354782,0.346308) V=100% PATH
POS=(785.0,-20.0,978.5,0.856519,-0.143445,0.354782,0.346308) V=20% PATH
POS=(784.0,-17.5,978.5,0.923880,0.000000,0.382683,0.000000) V=100% PATH
POS=(784.0,47.5,978.5,0.923880,0.000000,0.382683,0.000000) V=20% PATH
POS=(787.5,51.0,978.5,0.653281,0.270598,0.270598,-0.653281) V=100% PATH
POS=(867.5,51.0,978.5,0.653281,0.270598,0.270598,-0.653281) V=20% PATH
POS=(868.3,51.0,978.5,0.619854,0.197698,0.340266,-0.678907) V=100% PATH
POS=(913.3,51.0,968.5,0.619854,0.197698,0.340266,-0.678907) V=20% PATH
POS=(916.4,47.5,968.1,0.000000,-0.331631,0.000000,0.943409) V=100% PATH
POS=(916.4,-17.5,968.1,0.000000,-0.331631,0.000000,0.943409) V=20% PATH
POS=(915.6,-19.8,968.2,0.326056,-0.293124,0.195974,0.877132) V=100% PATH
POS=(886.6,-49.8,974.7,0.326056,-0.293124,0.195974,0.877132) V=20% PATH
POS=(884.3,-51.0,974.9,0.619854,-0.197698,0.340266,0.678907) V=100% PATH
POS=(868.3,-51.0,978.5,0.619854,-0.197698,0.340266,0.678907) V=20% PATH
POS=(884.3,-51.0,974.9,0.619854,-0.197698,0.340266,0.678907) V=100% PATH
POS=(885.5,-52.1,971.4,0.390353,0.390353,0.589597,0.589597) V=100% PATH
POS=(885.5,-52.1,908.1,0.390353,0.390353,0.589597,0.589597) V=20% PATH
POS=(924.5,-144.2,908.1,0.911461,0.030491,0.404459,-0.068713) V=100% PATH
POS=(824.4,-129.0,1020.0,0.911461,0.030491,0.404459,-0.068713) V=100% PATH
POS=(721.7,-137.8,1020.0,0.938985,0.019872,0.338942,-0.055051) V=100% PATH
POS=(775.5,-144.2,975.0,0.938985,0.019872,0.338942,-0.055051) V=100% PATH
POS=(814.5,-52.1,975.0,0.589597,-0.589597,-0.390353,0.390353) V=100% PATH
POS=(814.5,-52.1,908.1,0.589597,-0.589597,-0.390353,0.390353) V=20% PATH
POS=(775.5,-144.2,908.1,0.925632,0.021919,0.373871,-0.054268) V=100% PATH
POS=(660.4,-130.6,1020.0,0.925632,0.021919,0.373871,-0.054268) V=100% PATH
POS=(640.5,-47.2,1020.0,0.930413,0.033988,0.353821,-0.089377) V=100% PATH
POS=(690.2,-56.9,975.0,0.930413,0.033988,0.353821,-0.089377) V=100% PATH
POS=(782.9,-19.4,975.0,0.694098,-0.694098,-0.135010,0.135010) V=100% PATH
POS=(782.9,-19.4,908.1,0.694098,-0.694098,-0.135010,0.135010) V=20% PATH
POS=(690.2,-56.9,908.1,0.915073,0.037636,0.391787,-0.087903) V=100% PATH
POS=(585.4,-36.5,1020.0,0.915073,0.037636,0.391787,-0.087903) V=100% PATH
POS=(664.6,126.0,1020.0,0.930584,0.015755,0.363508,-0.040333) V=100% PATH
POS=(713.3,121.7,975.0,0.930584,0.015755,0.363508,-0.040333) V=100% PATH
POS=(784.0,51.0,975.0,0.653281,-0.653281,0.270598,-0.270598) V=100% PATH
POS=(784.0,51.0,908.1,0.653281,-0.653281,0.270598,-0.270598) V=20% PATH
POS=(713.3,121.7,908.1,0.911567,0.017721,0.408864,-0.039509) V=100% PATH
POS=(614.0,130.4,1020.0,0.911567,0.017721,0.408864,-0.039509) V=100% PATH
POS=(950.0,0.0,1585.0,1.000000,0.000000,0.000000,0.000000) V=100% PATH
```

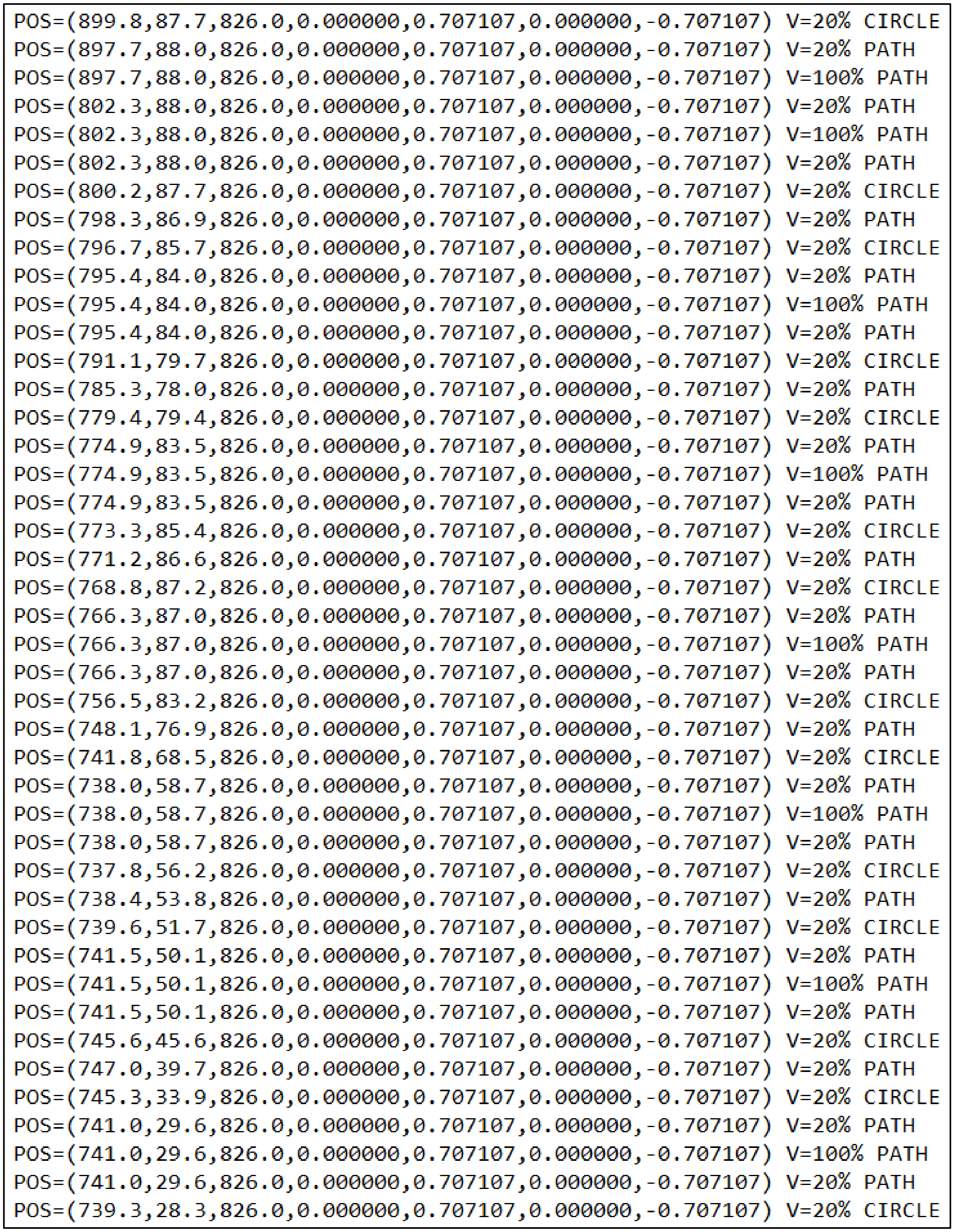
Figure E-1: The generated path to scan the mentioned 13 edges of part one.

E.2 Part 2



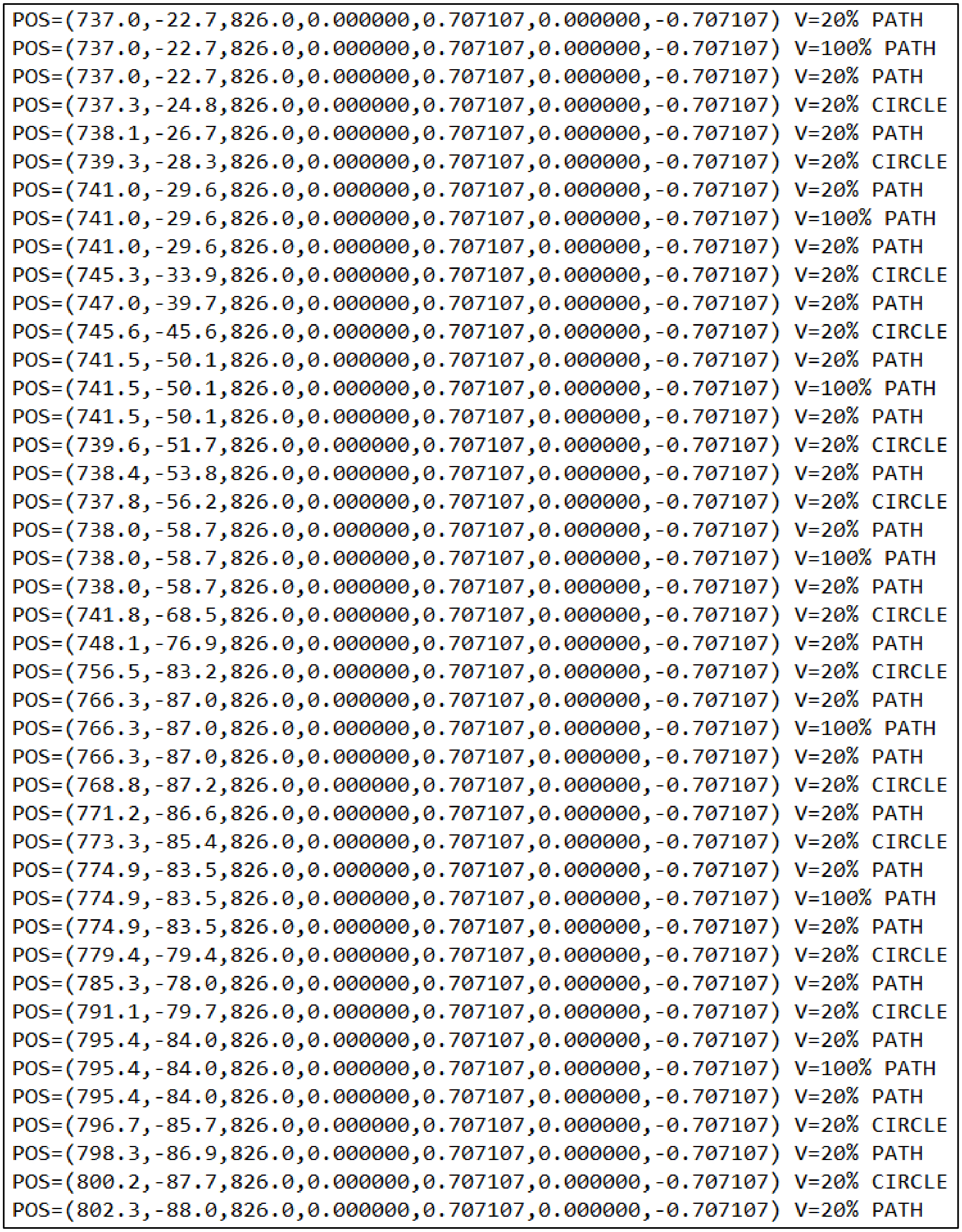
```
POS=(963.0,-22.7,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(963.0,-22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(963.0,22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(963.0,22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(963.0,22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.7,24.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(961.9,26.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(960.7,28.3,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(959.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(959.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(959.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(954.7,33.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(953.0,39.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(954.4,45.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(958.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(958.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(958.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(960.4,51.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(961.6,53.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.2,56.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(962.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(962.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(958.2,68.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(951.9,76.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(943.5,83.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(933.7,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(933.7,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(933.7,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(931.2,87.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(928.8,86.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(926.7,85.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(925.1,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(925.1,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(925.1,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(920.6,79.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(914.7,78.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(908.9,79.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(904.6,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(904.6,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(904.6,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(903.3,85.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(901.7,86.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
```

Figure E-2: First part of the generated path to deburr the mentioned 63 edges of part two.



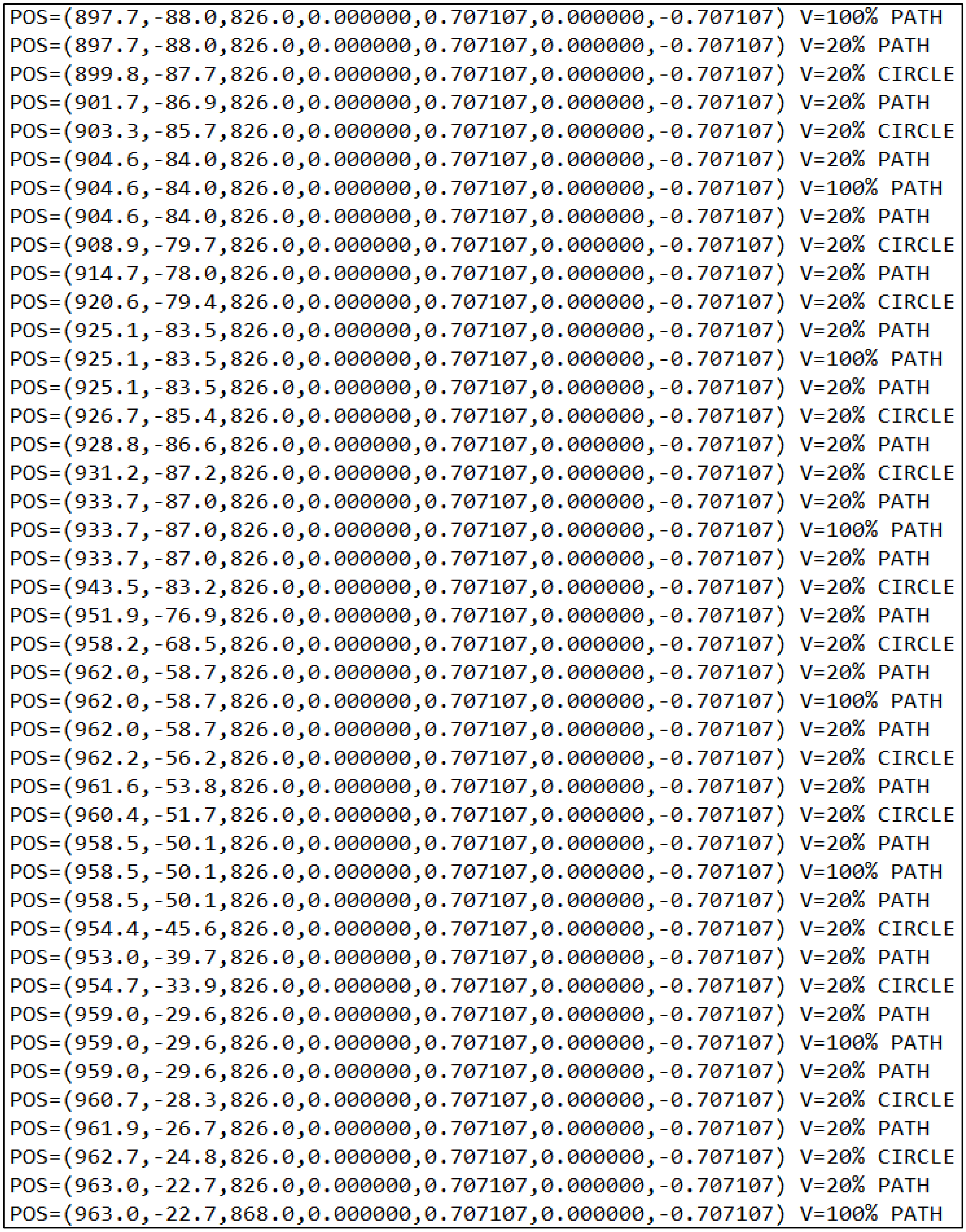
```
POS=(899.8,87.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(897.7,88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(897.7,88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(802.3,88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(802.3,88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(802.3,88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(800.2,87.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(798.3,86.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(796.7,85.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(795.4,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(795.4,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(795.4,84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(791.1,79.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(785.3,78.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(779.4,79.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(774.9,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(774.9,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(774.9,83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(773.3,85.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(771.2,86.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(768.8,87.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(766.3,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(766.3,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(766.3,87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(756.5,83.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(748.1,76.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.8,68.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(738.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(738.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(738.0,58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.8,56.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(738.4,53.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(739.6,51.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(741.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(741.5,50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.6,45.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(747.0,39.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.3,33.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(741.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(741.0,29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(739.3,28.3,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
```

Figure E-3: Second part of the generated path to deburr the mentioned 63 edges of part two.



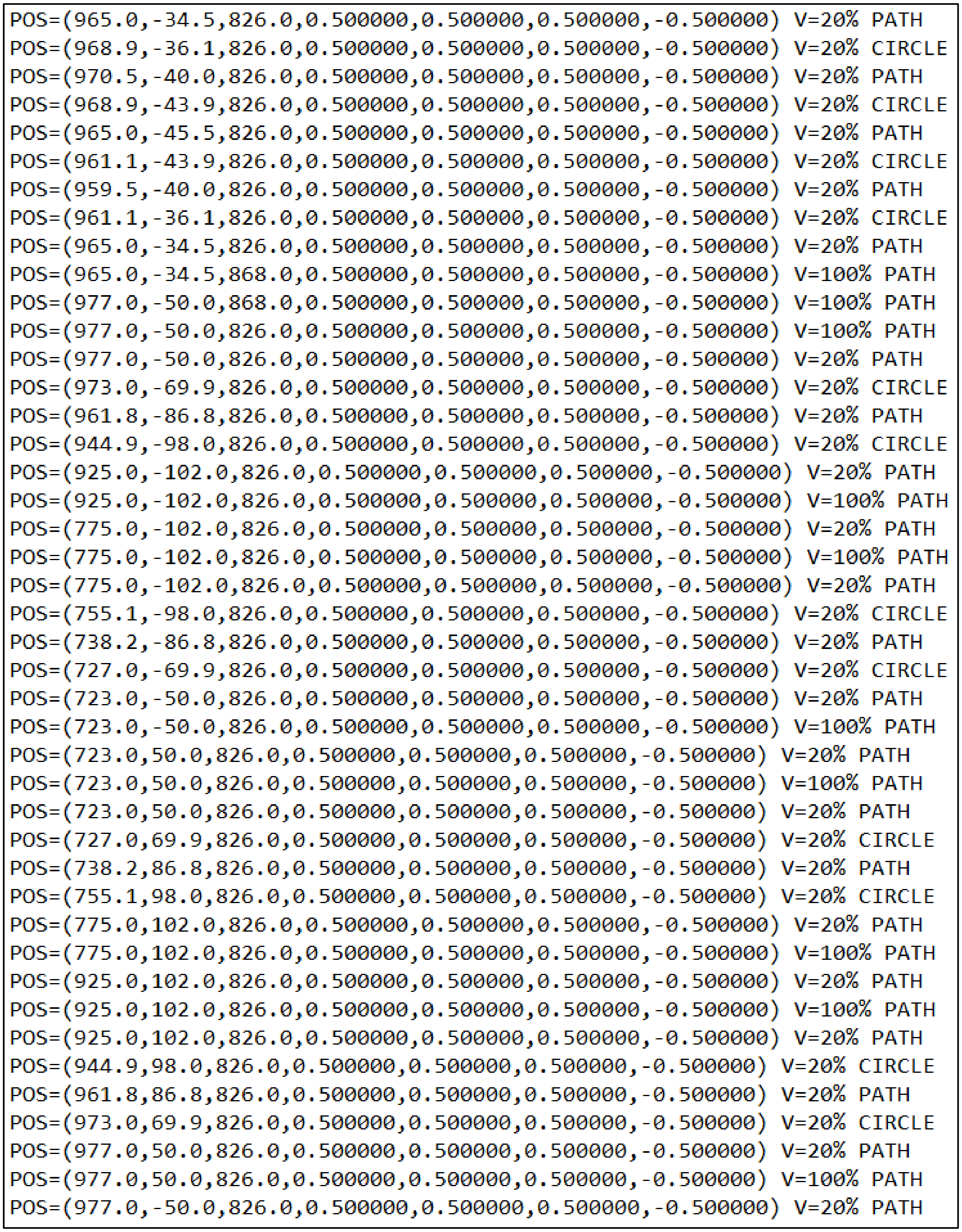
```
POS=(737.0,-22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.0,-22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(737.0,-22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.3,-24.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(738.1,-26.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(739.3,-28.3,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(741.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(741.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.3,-33.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(747.0,-39.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.6,-45.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(741.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(741.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(739.6,-51.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(738.4,-53.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.8,-56.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(738.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(738.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(738.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.8,-68.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(748.1,-76.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(756.5,-83.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(766.3,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(766.3,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(766.3,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(768.8,-87.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(771.2,-86.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(773.3,-85.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(774.9,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(774.9,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(774.9,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(779.4,-79.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(785.3,-78.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(791.1,-79.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(795.4,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(795.4,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(795.4,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(796.7,-85.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(798.3,-86.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(800.2,-87.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(802.3,-88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
```

Figure E-4: Third part of the generated path to deburr the mentioned 63 edges of part two.



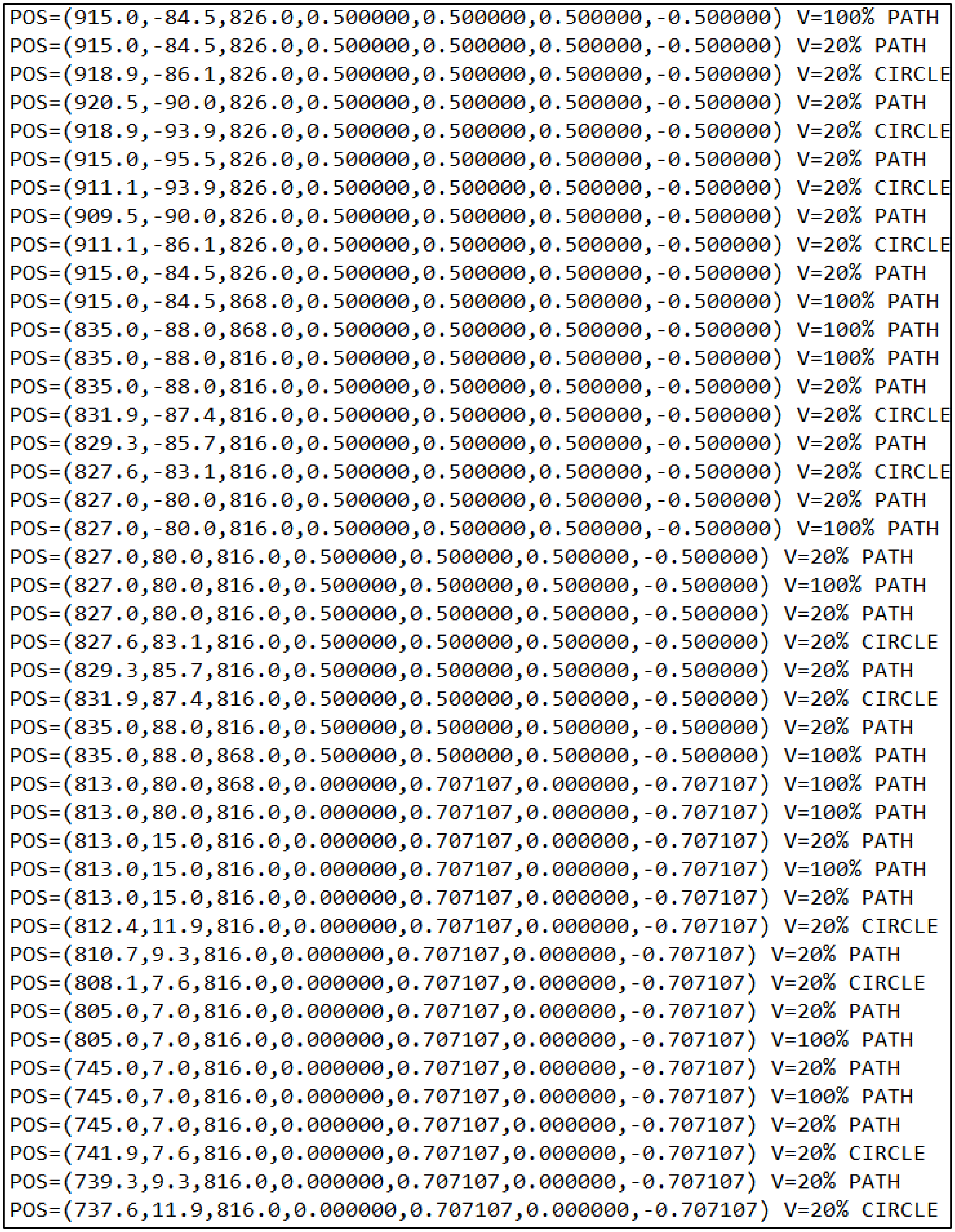
```
POS=(897.7,-88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(897.7,-88.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(899.8,-87.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(901.7,-86.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(903.3,-85.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(904.6,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(904.6,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(904.6,-84.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(908.9,-79.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(914.7,-78.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(920.6,-79.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(925.1,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(925.1,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(925.1,-83.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(926.7,-85.4,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(928.8,-86.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(931.2,-87.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(933.7,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(933.7,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(933.7,-87.0,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(943.5,-83.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(951.9,-76.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(958.2,-68.5,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(962.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(962.0,-58.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.2,-56.2,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(961.6,-53.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(960.4,-51.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(958.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(958.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(958.5,-50.1,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(954.4,-45.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(953.0,-39.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(954.7,-33.9,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(959.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(959.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(959.0,-29.6,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(960.7,-28.3,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(961.9,-26.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(962.7,-24.8,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(963.0,-22.7,826.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(963.0,-22.7,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
```

Figure E-5: Fourth part of the generated path to deburr the mentioned 63 edges of part two.



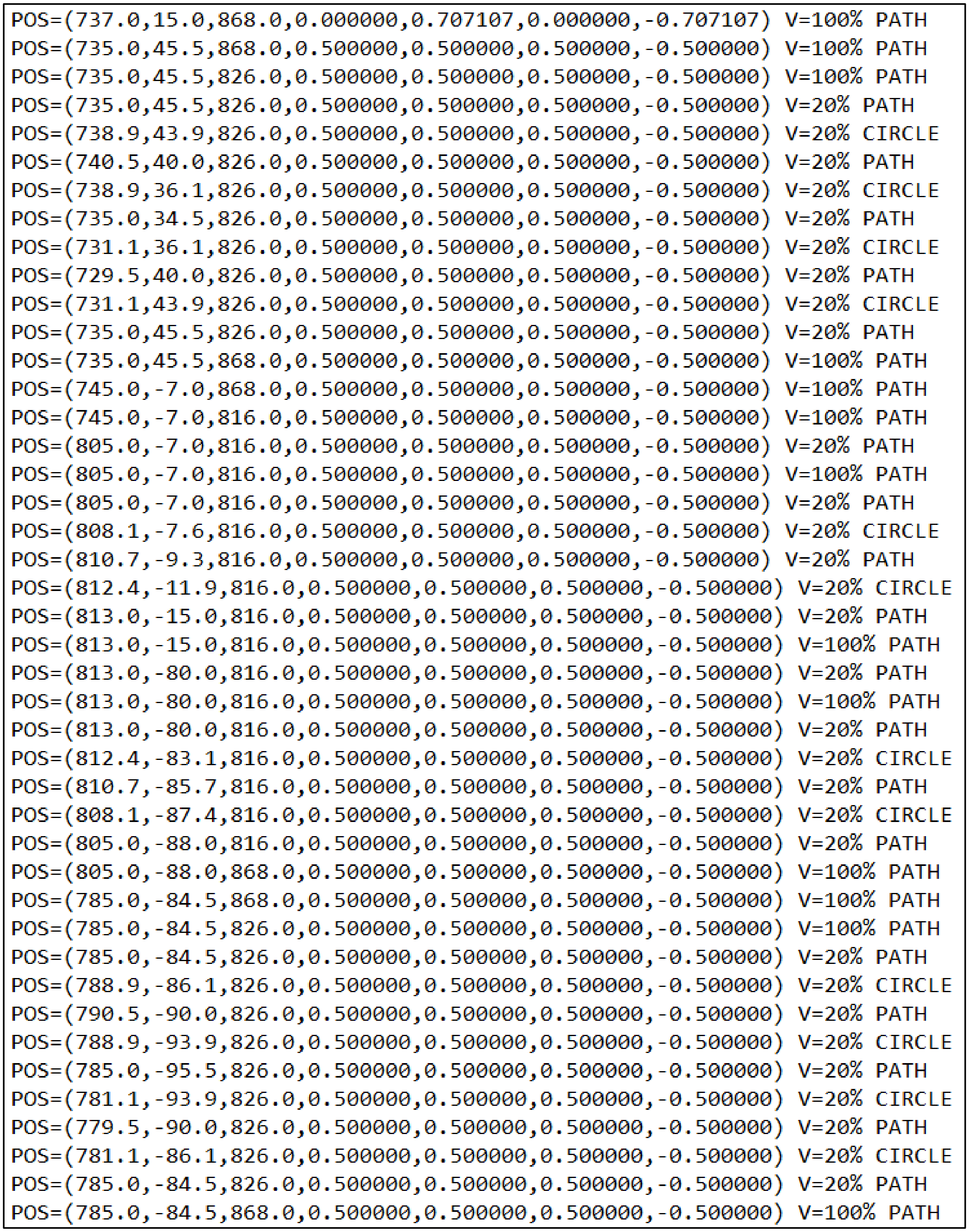
```
POS=(965.0,-34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(968.9,-36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(970.5,-40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(968.9,-43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(965.0,-45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(961.1,-43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(959.5,-40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(961.1,-36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(965.0,-34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(965.0,-34.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(977.0,-50.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(977.0,-50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(977.0,-50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(973.0,-69.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(961.8,-86.8,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(944.9,-98.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(925.0,-102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(925.0,-102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(775.0,-102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(775.0,-102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(775.0,-102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(755.1,-98.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(738.2,-86.8,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(727.0,-69.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(723.0,-50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(723.0,-50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(723.0,50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(723.0,50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(723.0,50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(727.0,69.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(738.2,86.8,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(755.1,98.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(775.0,102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(775.0,102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(925.0,102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(925.0,102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(925.0,102.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(944.9,98.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(961.8,86.8,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(973.0,69.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(977.0,50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(977.0,50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(977.0,-50.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
```

Figure E-6: Fifth part of the generated path to deburr the mentioned 63 edges of part two.



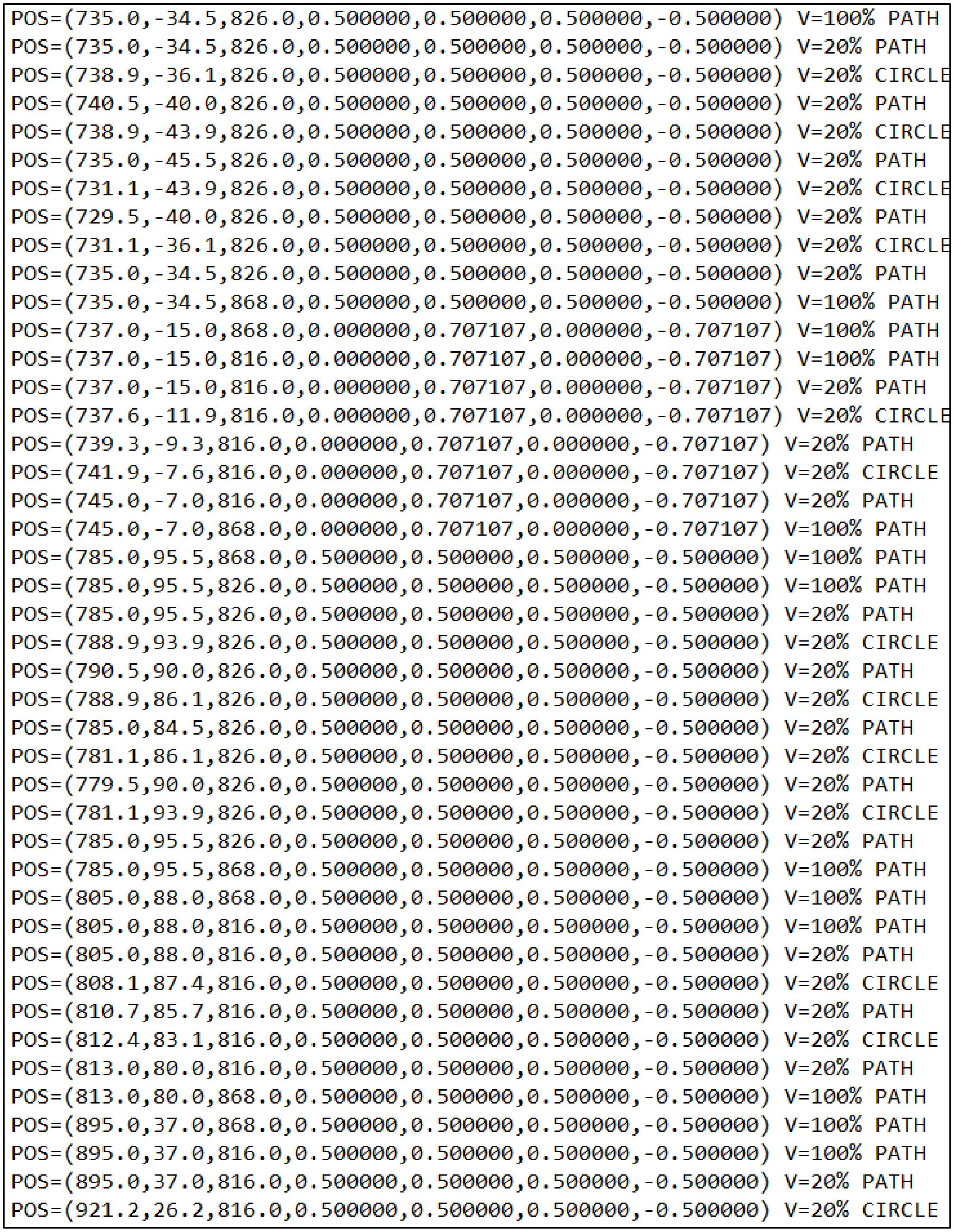
```
POS=(915.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(915.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(918.9,-86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(920.5,-90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(918.9,-93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(915.0,-95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.1,-93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(909.5,-90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.1,-86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(915.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(915.0,-84.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(835.0,-88.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(835.0,-88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(835.0,-88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(831.9,-87.4,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(829.3,-85.7,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(827.6,-83.1,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(827.0,-80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(827.0,-80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(827.0,80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(827.0,80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(827.0,80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(827.6,83.1,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(829.3,85.7,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(831.9,87.4,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(835.0,88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(835.0,88.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(813.0,80.0,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(813.0,80.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(813.0,15.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(813.0,15.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(813.0,15.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(812.4,11.9,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(810.7,9.3,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(808.1,7.6,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(805.0,7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(805.0,7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(745.0,7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.0,7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(745.0,7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.9,7.6,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(739.3,9.3,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.6,11.9,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
```

Figure E-7: Sixth part of the generated path to deburr the mentioned 63 edges of part two.



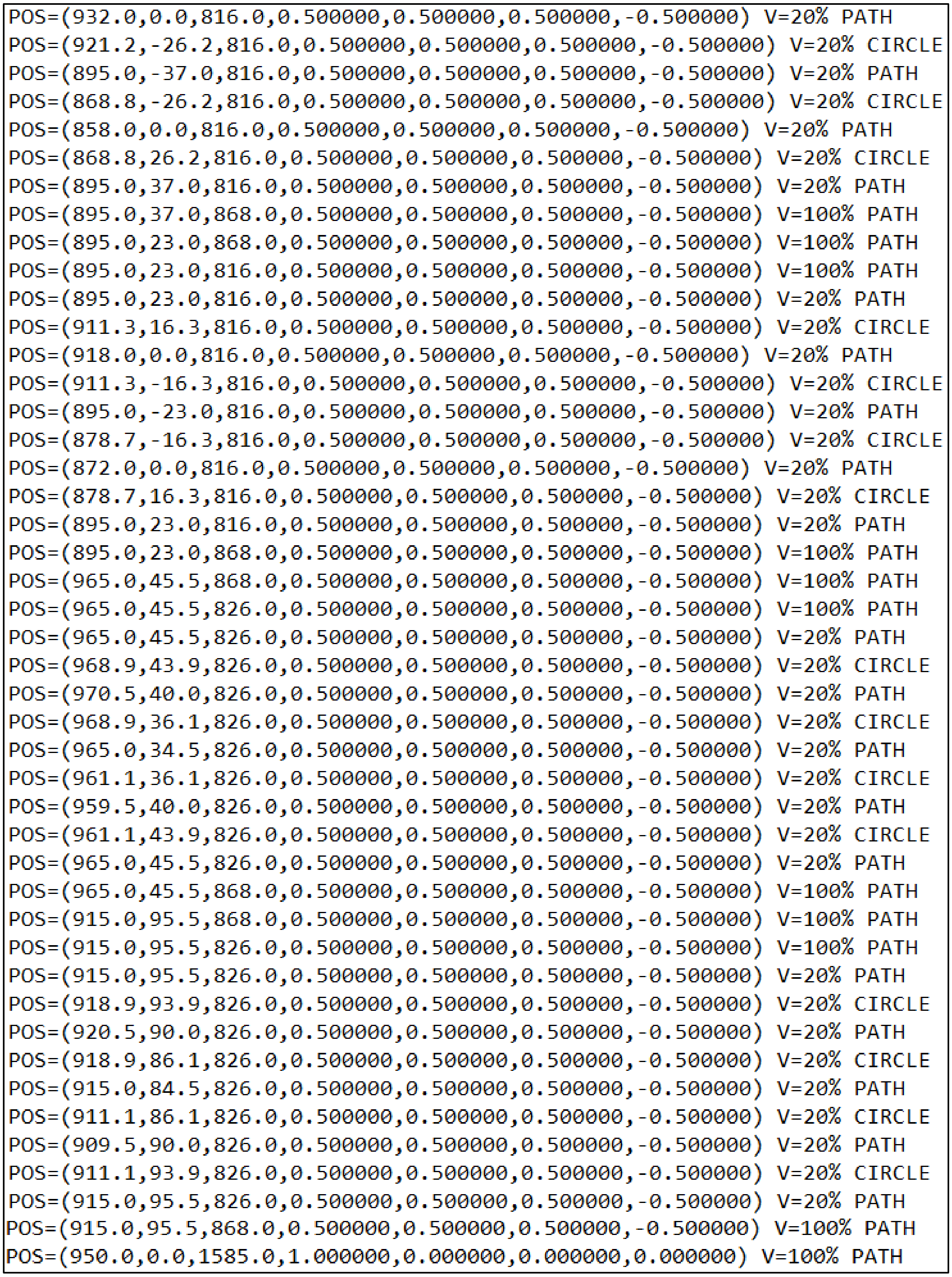
```
POS=(737.0,15.0,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(735.0,45.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(735.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(735.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(738.9,43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(740.5,40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(738.9,36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(735.0,34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(731.1,36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(729.5,40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(731.1,43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(735.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(735.0,45.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(745.0,-7.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(745.0,-7.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(805.0,-7.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(805.0,-7.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(805.0,-7.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(808.1,-7.6,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(810.7,-9.3,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(812.4,-11.9,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(813.0,-15.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(813.0,-15.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(813.0,-80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(813.0,-80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(813.0,-80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(812.4,-83.1,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(810.7,-85.7,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(808.1,-87.4,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(805.0,-88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(805.0,-88.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(785.0,-84.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(785.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(785.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(788.9,-86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(790.5,-90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(788.9,-93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(785.0,-95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(781.1,-93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(779.5,-90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(781.1,-86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(785.0,-84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(785.0,-84.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
```

Figure E-8: Seventh part of the generated path to deburr the mentioned 63 edges of part two.



```
POS=(735.0,-34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(735.0,-34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(738.9,-36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(740.5,-40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(738.9,-43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(735.0,-45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(731.1,-43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(729.5,-40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(731.1,-36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(735.0,-34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(735.0,-34.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(737.0,-15.0,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(737.0,-15.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(737.0,-15.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(737.6,-11.9,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(739.3,-9.3,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(741.9,-7.6,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% CIRCLE
POS=(745.0,-7.0,816.0,0.000000,0.707107,0.000000,-0.707107) V=20% PATH
POS=(745.0,-7.0,868.0,0.000000,0.707107,0.000000,-0.707107) V=100% PATH
POS=(785.0,95.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(785.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(785.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(788.9,93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(790.5,90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(788.9,86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(785.0,84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(781.1,86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(779.5,90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(781.1,93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(785.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(785.0,95.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(805.0,88.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(805.0,88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(805.0,88.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(808.1,87.4,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(810.7,85.7,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(812.4,83.1,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(813.0,80.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(813.0,80.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,37.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,37.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,37.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(921.2,26.2,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
```

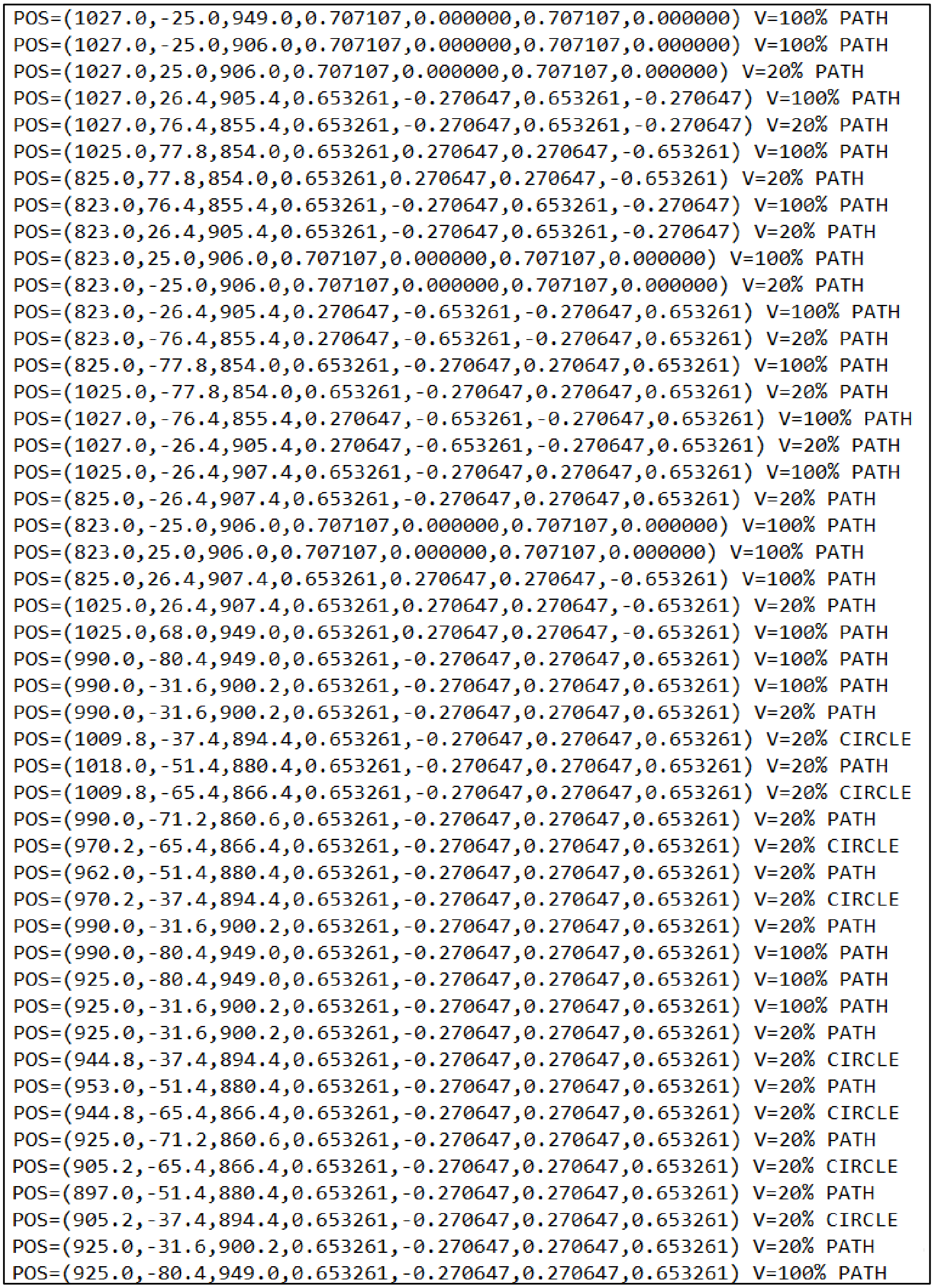
Figure E-9: Eighth part of the generated path to deburr the mentioned 63 edges of part two.



```
POS=(932.0,0.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(921.2,-26.2,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(895.0,-37.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(868.8,-26.2,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(858.0,0.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(868.8,26.2,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(895.0,37.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(895.0,37.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,23.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,23.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(895.0,23.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.3,16.3,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(918.0,0.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.3,-16.3,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(895.0,-23.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(878.7,-16.3,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(872.0,0.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(878.7,16.3,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(895.0,23.0,816.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(895.0,23.0,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(965.0,45.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(965.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(965.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(968.9,43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(970.5,40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(968.9,36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(965.0,34.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(961.1,36.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(959.5,40.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(961.1,43.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(965.0,45.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(965.0,45.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(915.0,95.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(915.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(915.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(918.9,93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(920.5,90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(918.9,86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(915.0,84.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.1,86.1,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(909.5,90.0,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(911.1,93.9,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% CIRCLE
POS=(915.0,95.5,826.0,0.500000,0.500000,0.500000,-0.500000) V=20% PATH
POS=(915.0,95.5,868.0,0.500000,0.500000,0.500000,-0.500000) V=100% PATH
POS=(950.0,0.0,1585.0,1.000000,0.000000,0.000000,0.000000) V=100% PATH
```

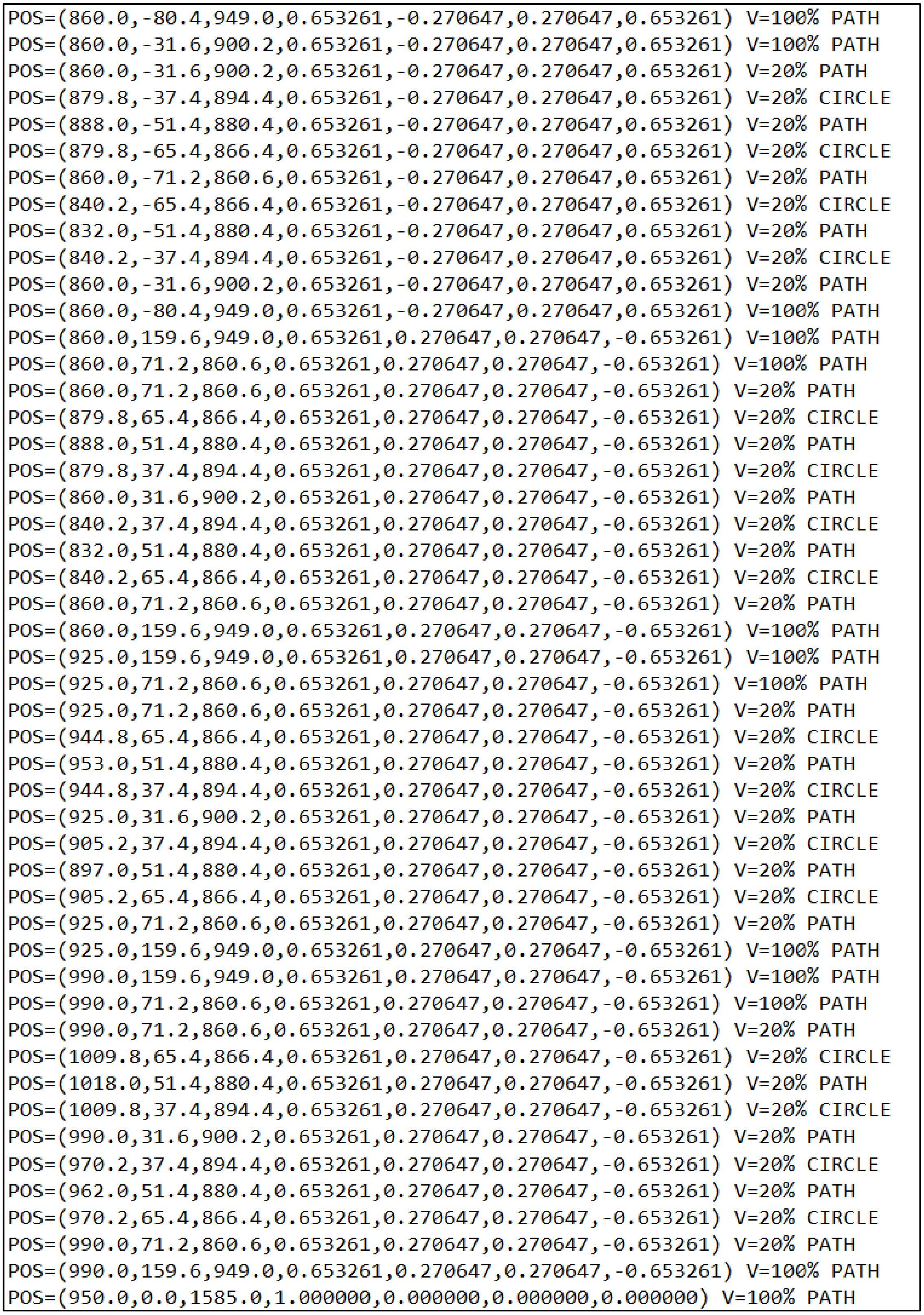
Figure E-10: Ninth part of the generated path to deburr the mentioned 63 edges of part two.

E.3 Part 3



```
POS=(1027.0,-25.0,949.0,0.707107,0.000000,0.707107,0.000000) V=100% PATH
POS=(1027.0,-25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=100% PATH
POS=(1027.0,25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=20% PATH
POS=(1027.0,26.4,905.4,0.653261,-0.270647,0.653261,-0.270647) V=100% PATH
POS=(1027.0,76.4,855.4,0.653261,-0.270647,0.653261,-0.270647) V=20% PATH
POS=(1025.0,77.8,854.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(825.0,77.8,854.0,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(823.0,76.4,855.4,0.653261,-0.270647,0.653261,-0.270647) V=100% PATH
POS=(823.0,26.4,905.4,0.653261,-0.270647,0.653261,-0.270647) V=20% PATH
POS=(823.0,25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=100% PATH
POS=(823.0,-25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=20% PATH
POS=(823.0,-26.4,905.4,0.270647,-0.653261,-0.270647,0.653261) V=100% PATH
POS=(823.0,-76.4,855.4,0.270647,-0.653261,-0.270647,0.653261) V=20% PATH
POS=(825.0,-77.8,854.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(1025.0,-77.8,854.0,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(1027.0,-76.4,855.4,0.270647,-0.653261,-0.270647,0.653261) V=100% PATH
POS=(1027.0,-26.4,905.4,0.270647,-0.653261,-0.270647,0.653261) V=20% PATH
POS=(1025.0,-26.4,907.4,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(825.0,-26.4,907.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(823.0,-25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=100% PATH
POS=(823.0,25.0,906.0,0.707107,0.000000,0.707107,0.000000) V=100% PATH
POS=(825.0,26.4,907.4,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(1025.0,26.4,907.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(1025.0,68.0,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(990.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(990.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(990.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(1009.8,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(1018.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(1009.8,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(990.0,-71.2,860.6,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(970.2,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(962.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(970.2,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(990.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(990.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(925.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(925.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(925.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(944.8,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(953.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(944.8,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(925.0,-71.2,860.6,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(905.2,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(897.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(905.2,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(925.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(925.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
```

Figure E-11: First part of the generated path to deburr the mentioned 63 edges of part three.



```
POS=(860.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(860.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(860.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(879.8,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(888.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(879.8,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(860.0,-71.2,860.6,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(840.2,-65.4,866.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(832.0,-51.4,880.4,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(840.2,-37.4,894.4,0.653261,-0.270647,0.270647,0.653261) V=20% CIRCLE
POS=(860.0,-31.6,900.2,0.653261,-0.270647,0.270647,0.653261) V=20% PATH
POS=(860.0,-80.4,949.0,0.653261,-0.270647,0.270647,0.653261) V=100% PATH
POS=(860.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(860.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(860.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(879.8,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(888.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(879.8,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(860.0,31.6,900.2,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(840.2,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(832.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(840.2,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(860.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(860.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(925.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(925.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(925.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(944.8,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(953.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(944.8,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(925.0,31.6,900.2,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(905.2,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(897.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(905.2,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(925.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(925.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(990.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(990.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(990.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(1009.8,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(1018.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(1009.8,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(990.0,31.6,900.2,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(970.2,37.4,894.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(962.0,51.4,880.4,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(970.2,65.4,866.4,0.653261,0.270647,0.270647,-0.653261) V=20% CIRCLE
POS=(990.0,71.2,860.6,0.653261,0.270647,0.270647,-0.653261) V=20% PATH
POS=(990.0,159.6,949.0,0.653261,0.270647,0.270647,-0.653261) V=100% PATH
POS=(950.0,0.0,1585.0,1.000000,0.000000,0.000000,0.000000) V=100% PATH
```

Figure E-12: Second part of the generated path to deburr the mentioned 63 edges of part three.

APPENDIX F

SNAPSHOTS OF THE PROCESSES IN CHAPTER 5

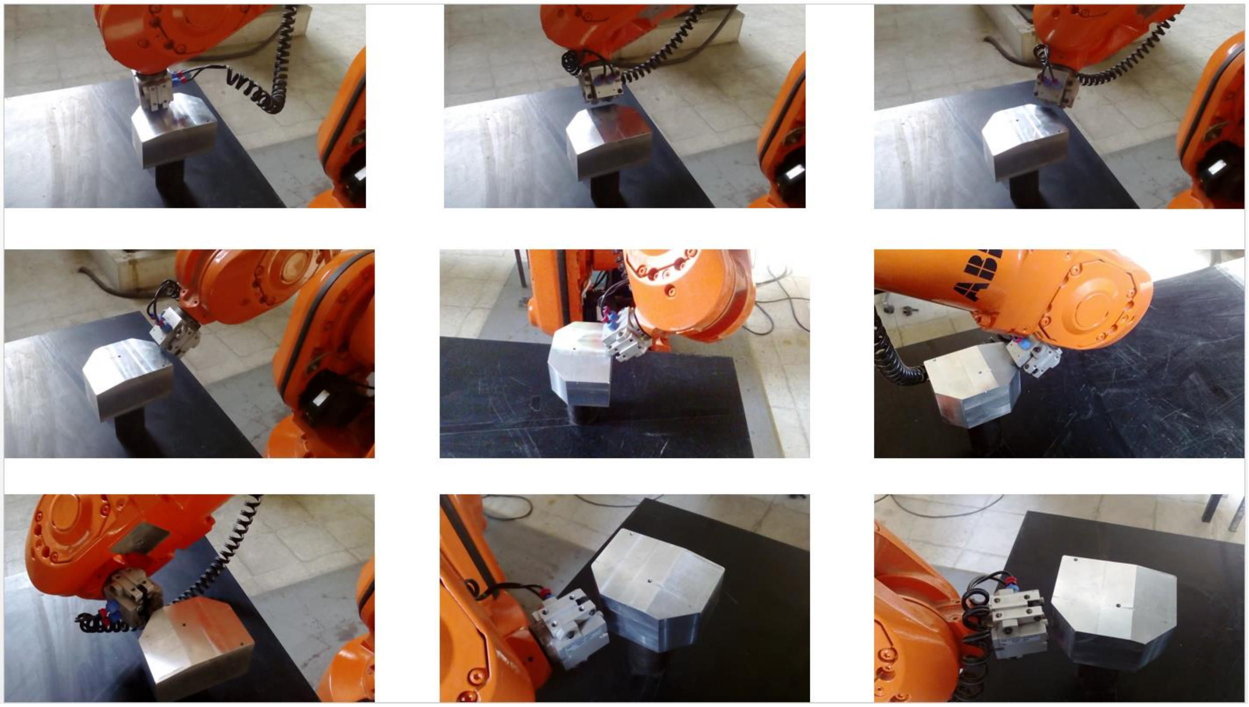


Figure F-1: Snapshots from the processing of part one in chapter 5.

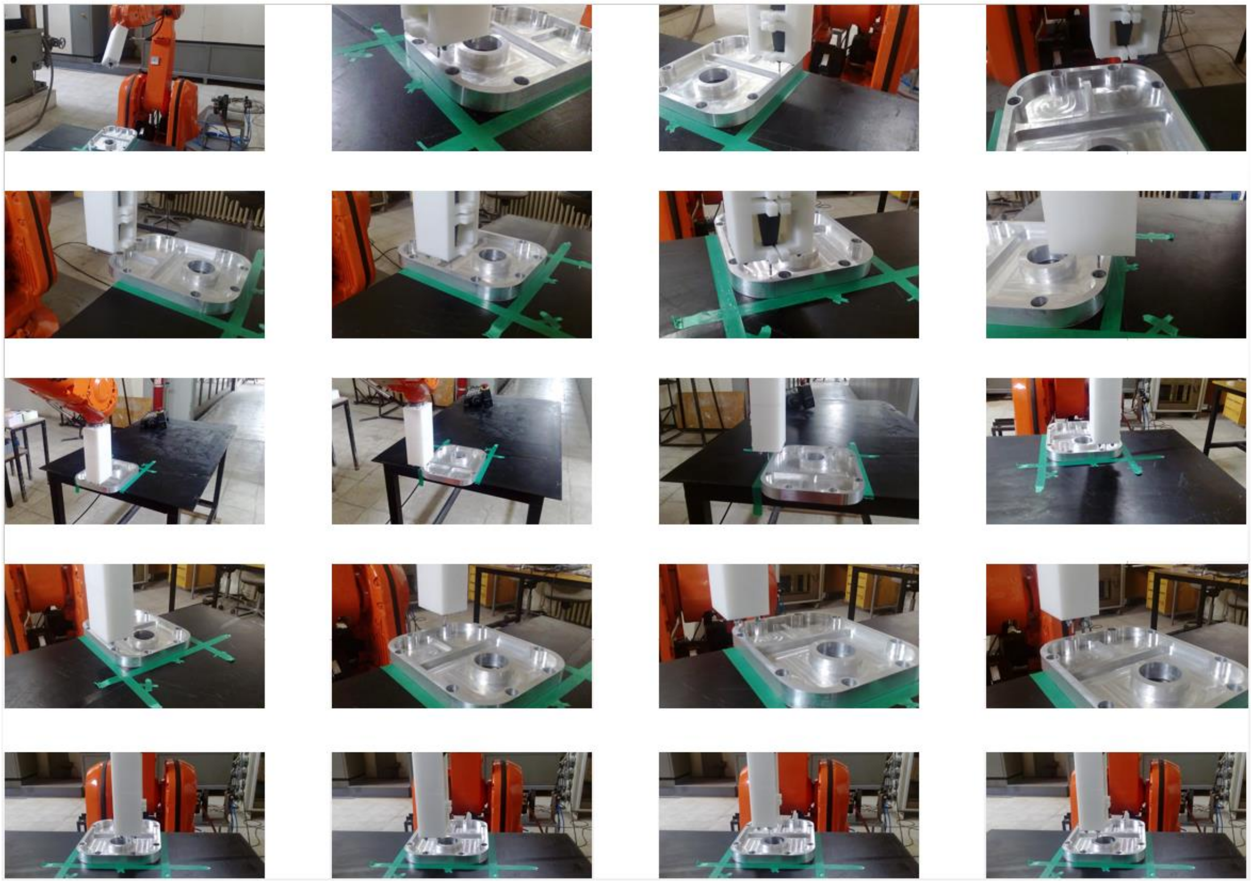


Figure F-2: Snapshots from the processing of part two in chapter 5.

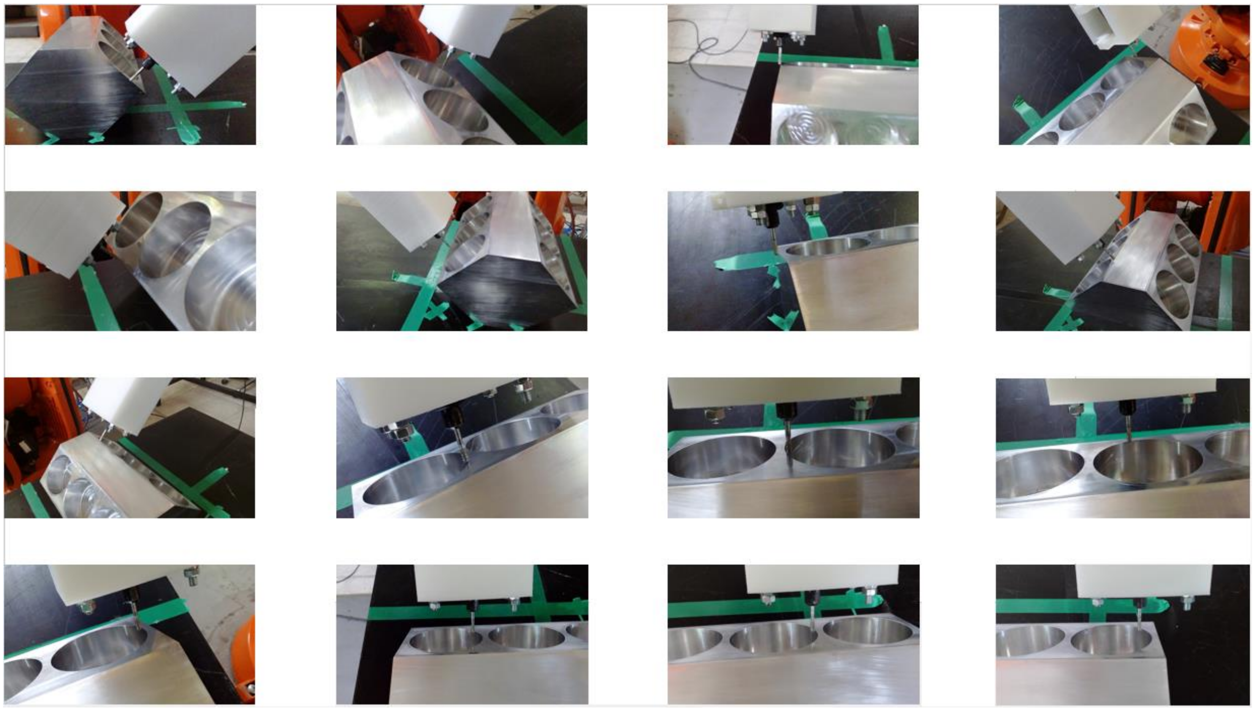


Figure F-3: Snapshots from the processing of part three in chapter 5.

Appendix G

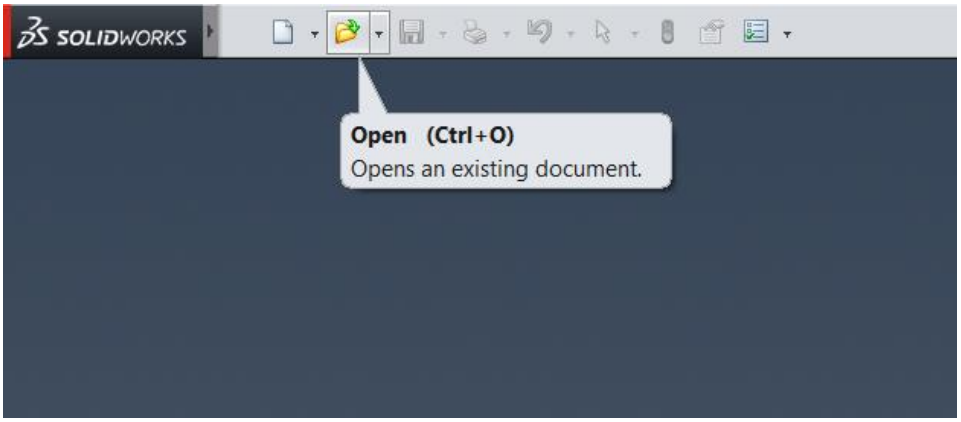
A STEP BY STEP EXAMPLE FOR USING THE SOFTWARE

This appendix gives a fully detailed example for using both the SolidWorks package and the ABB user interface software. Part 3 from chapter 5 is used to illustrate the correct way of using both programs.

G.1. Path Generation

SolidWorks Package is the software responsible for generating the nominal path to perform edge finishing or scanning on a workpiece. The following steps give the full detailed explanation to generate a path using this package.

1. Launch SolidWorks
2. Open the workpiece you would like to generate a path for.



Open (Ctrl+O)
Opens an existing document.

Figure G-1: Opening the considered Workpiece.

3. Define a reference frame and attach it to the workpiece. This can be done from “Features/Reference Geometry/Reference Coordinate System”. This frame shows the desired orientation of the workpiece with respect to the global coordinates of the robot.

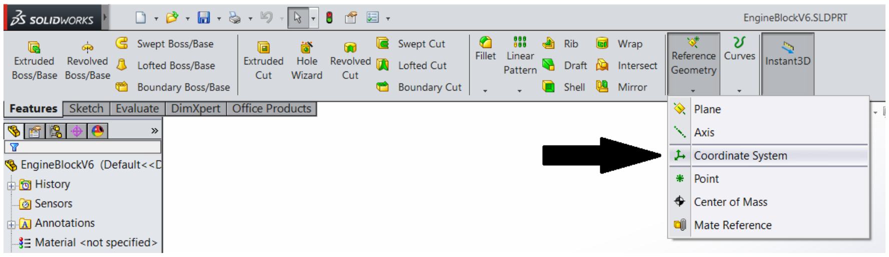


Figure G-2: Defining a new reference frame.

4. Launch the path generator package. This is done by selecting Tools/Macros/.

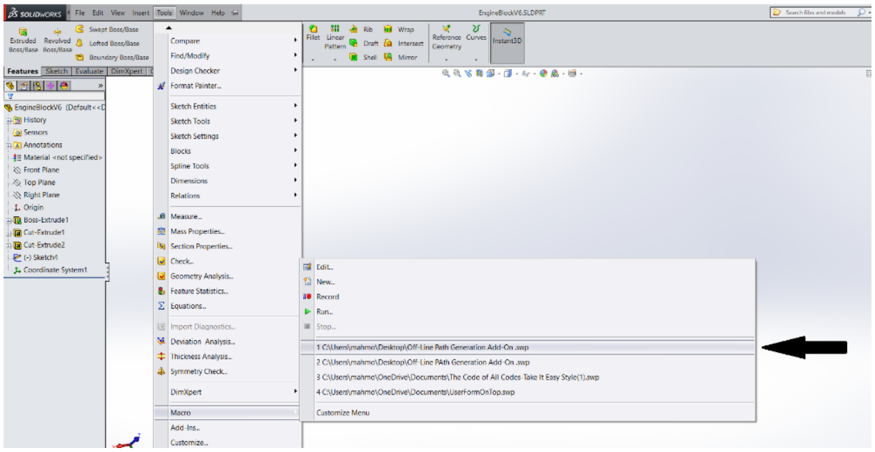
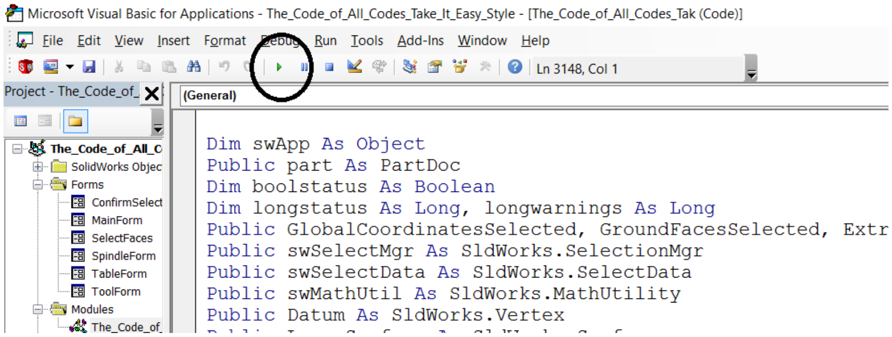


Figure G-3: Launching Off-Line Path Generation Add-On.

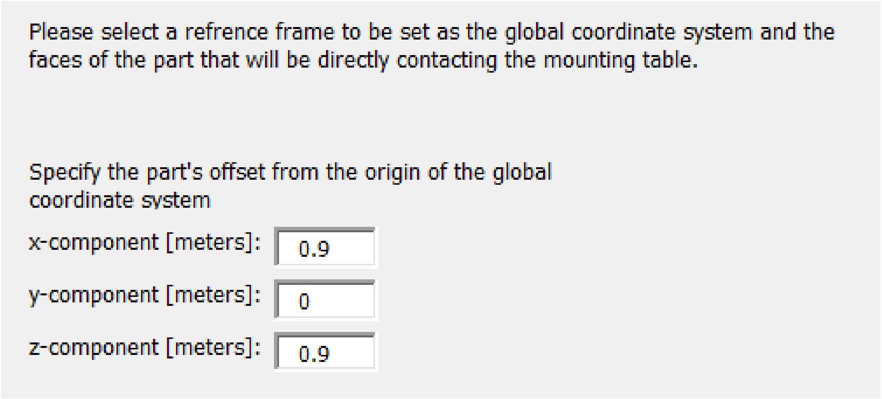
5. Press play to start the macro.



```
Dim swApp As Object
Public part As PartDoc
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Public GlobalCoordinatesSelected, GroundFacesSelected, Extr
Public swSelectMgr As SldWorks.SelectionMgr
Public swSelectData As SldWorks.SelectData
Public swMathUtil As SldWorks.MathUtility
Public Datum As SldWorks.Vertex
```

Figure G-4 Starting the Off-Line Path Generation Add-On.

6. Enter the location of the previously defined reference frame. This location represent the offset value from the global coordinates' origin. In this example, x=0.9, y=0 and z=0.9



Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

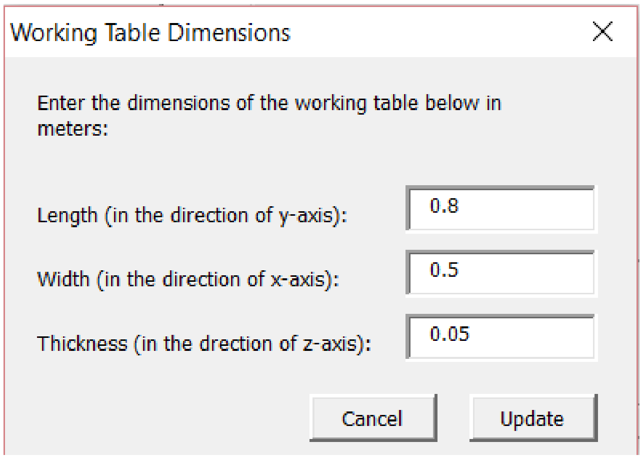
x-component [meters]: 0.9

y-component [meters]: 0

z-component [meters]: 0.9

Figure G-5: Entering the offset value of the defined reference frame with respect to the global coordinates of the robot.

7. Update the dimensions of the working table. Clicking on the (Update Dimensions of Working Table) button a window pops up and the user can enter the width, length and thickness of the working table. If you are using the black metal table in the lab, then there is no need to do this step.



Working Table Dimensions

Enter the dimensions of the working table below in meters:

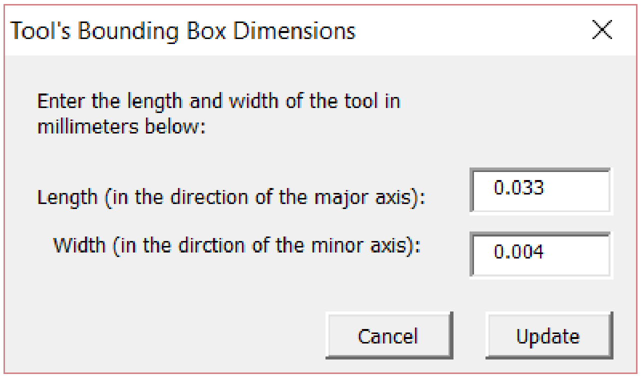
Length (in the direction of y-axis): 0.8

Width (in the direction of x-axis): 0.5

Thickness (in the drection of z-axis): 0.05

Figure G-6: Entering the working table dimensions.

8. Update the tool’s dimensions. Clicking on the (Update Tool’s Bounding Box) button a window pops up and the user can enter the length and width of the tool. For this example, enter 0.033 for length and 0.002 for width. In case you are to generate a path for scanning, you can skip this step.



Tool's Bounding Box Dimensions

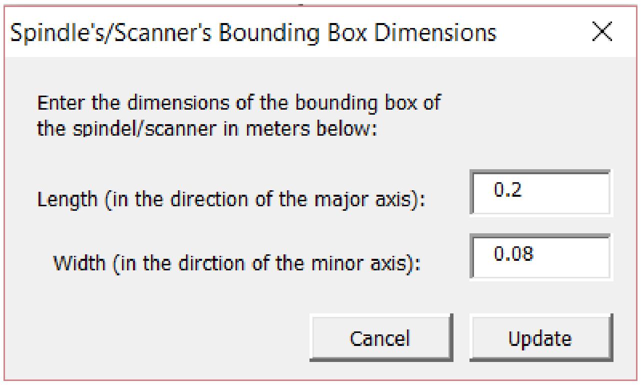
Enter the length and width of the tool in millimeters below:

Length (in the direction of the major axis): 0.033

Width (in the dirction of the minor axis): 0.004

Figure G-7: Entering the tool's bounding box dimensions.

9. Update the Spindle’s/Scanner’s dimensions. Clicking on the (Update Spindle’s/Scanner’s Bounding Box) button a window pops up and the user can enter the length and width of the spindle or scanner. If you are using the white spindle holder that was 3D printed by Mahmoud Nemer, enter 0.2 for length and 0.04 for width.



Spindle's/Scanner's Bounding Box Dimensions

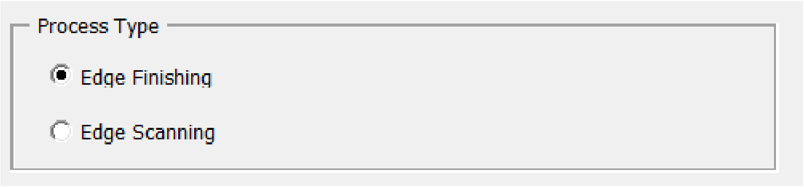
Enter the dimensions of the bounding box of the spindel/scanner in meters below:

Length (in the direction of the major axis): 0.2

Width (in the dirction of the minor axis): 0.08

Figure G-8: Entering the spindle's/scanner's bounding box dimensions.

10. Select the process type. Select between edge finishing or scanning, depending in your goal of the path. In this example, Edge Finishing is selected.



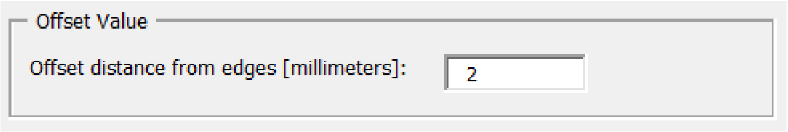
Process Type

● Edge Finishing

○ Edge Scanning

Figure G-9: Selecting the type of process to be done.

11. Select the offset value from the edge. Zero offset value means directly touch the edge by either the tool or scanner. In this example, it is taken as 2 mm.

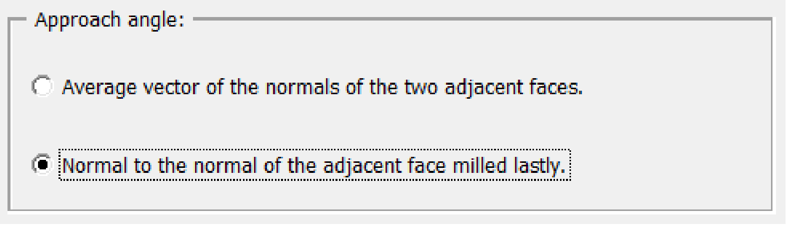


Offset Value

Offset distance from edges [millimeters]: 2

Figure G-10: Entering the offset value between the each edge and the tip-point of the robot.

12. Select the approach angle option. The user get to select from two options. Either approach each edge with an angle that is the average of the adjacent faces' normal, or to approach the edge in the direction normal to the formed burrs. In this example the second option is selected.



Approach angle:

○ Average vector of the normals of the two adjacent faces.

● Normal to the normal of the adjacent face milled lastly.

Figure G-11: Choosing the approach angle of the end-effector.

13. If the second option of step 12 is selected then the user needs to specify the order of which the faces were machined. This is necessary so that the program can calculate the direction of the formed burrs. For this example the faces were selected as shown in the next 20 figures. Note the direction of burrs is along the face that was machined last.

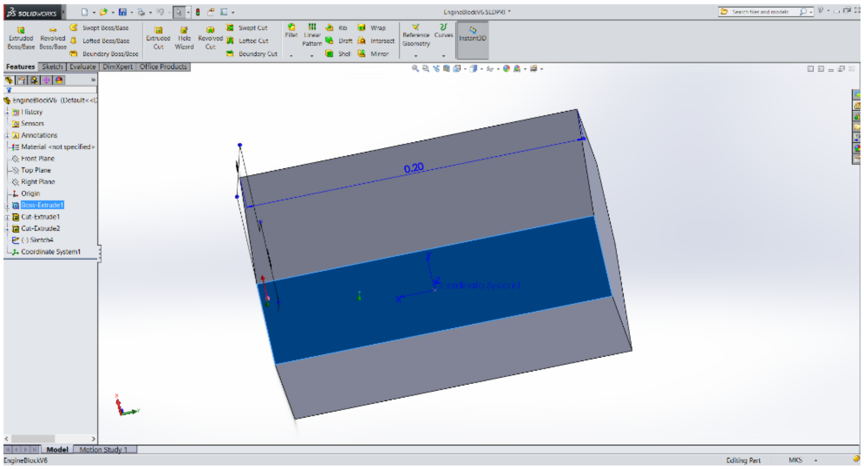


Figure G-12: Selecting the first face of the given workpiece example.

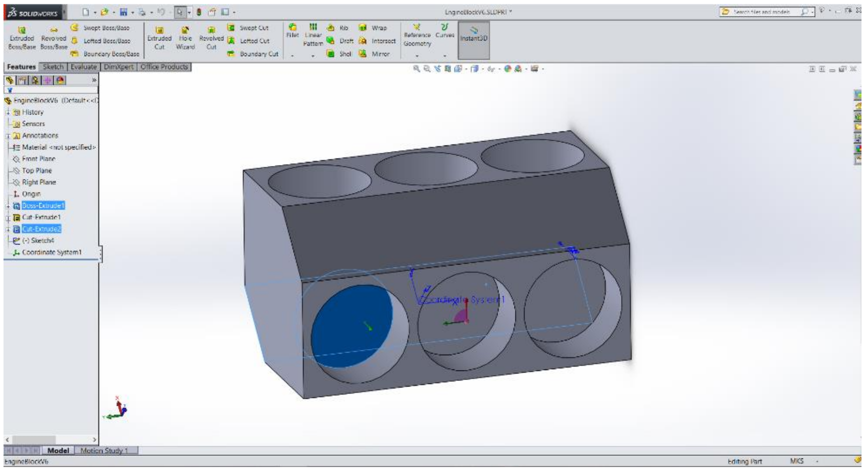


Figure G-13: Selecting the second face of the given workpiece example.

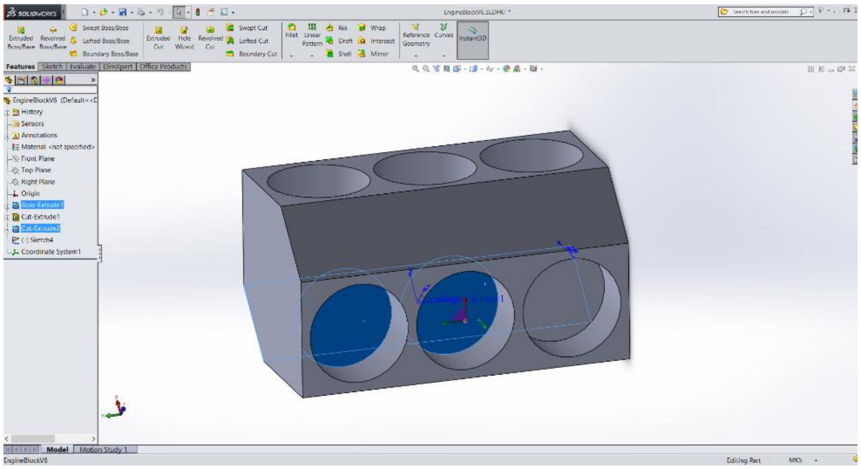


Figure G-14: Selecting the third face of the given workpiece example.

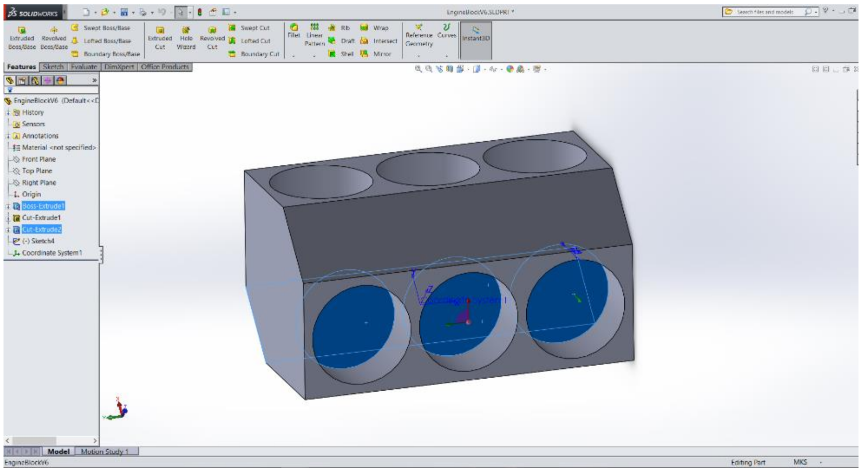


Figure G-15: Selecting the fourth face of the given workpiece example.

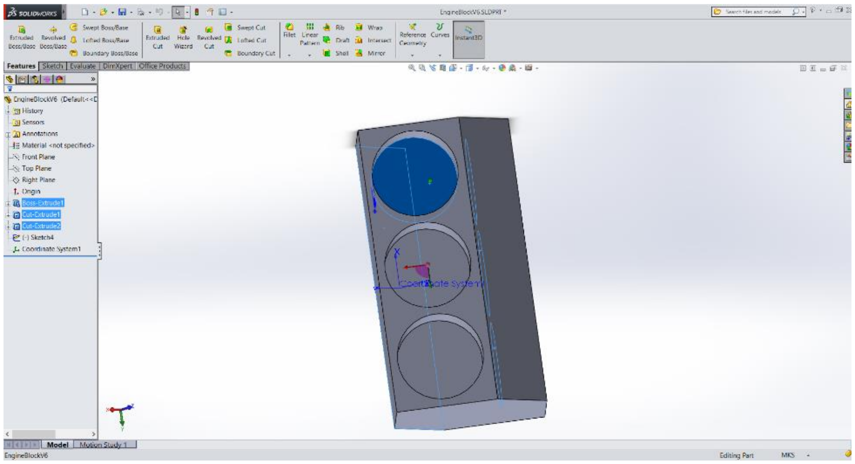


Figure G-16: Selecting the fifth face of the given workpiece example.

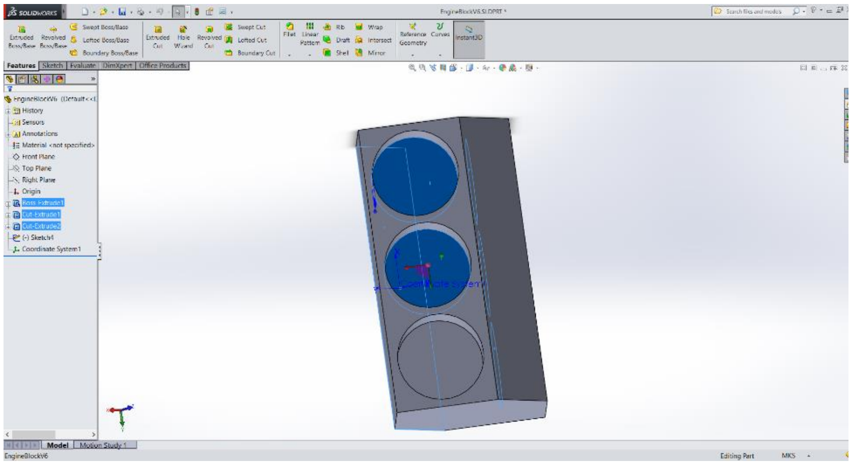


Figure G-17: Selecting the sixth face of the given workpiece example.

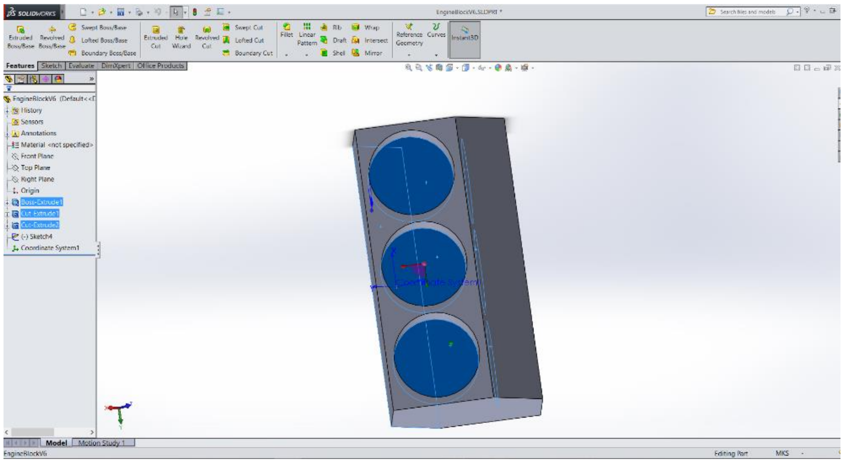


Figure G-18: Selecting the seventh face of the given workpiece example.

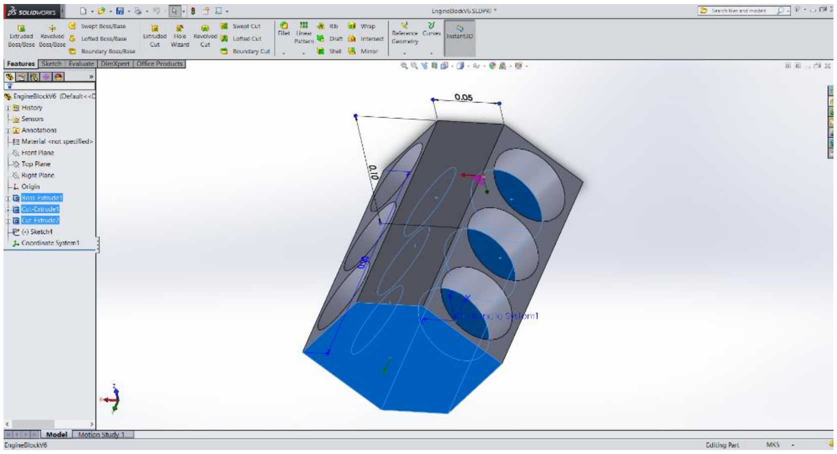


Figure G-19: Selecting the eighth face of the given workpiece example.

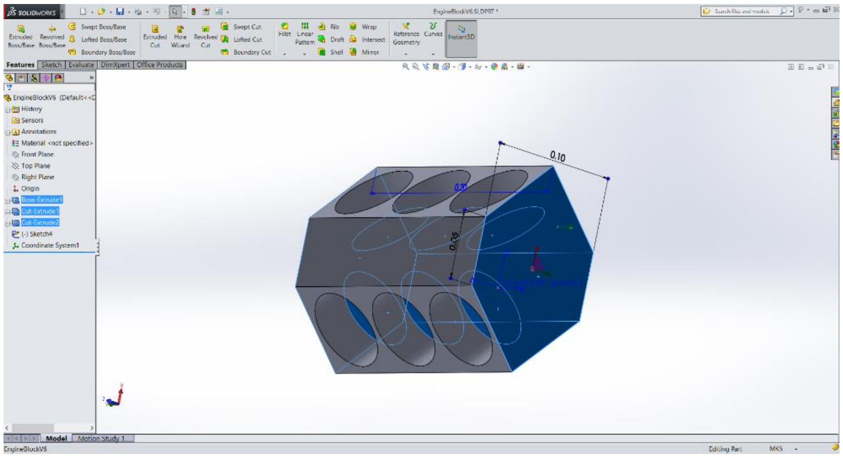


Figure G-20: Selecting the ninth face of the given workpiece example.

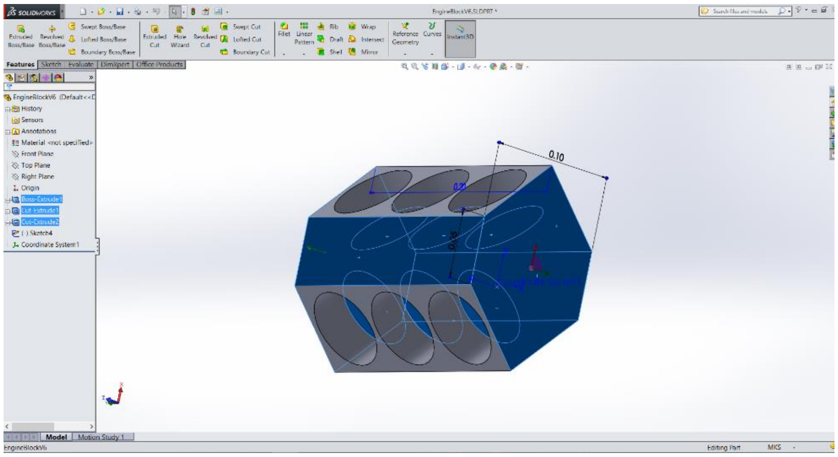


Figure G-21: Selecting the tenth face of the given workpiece example.

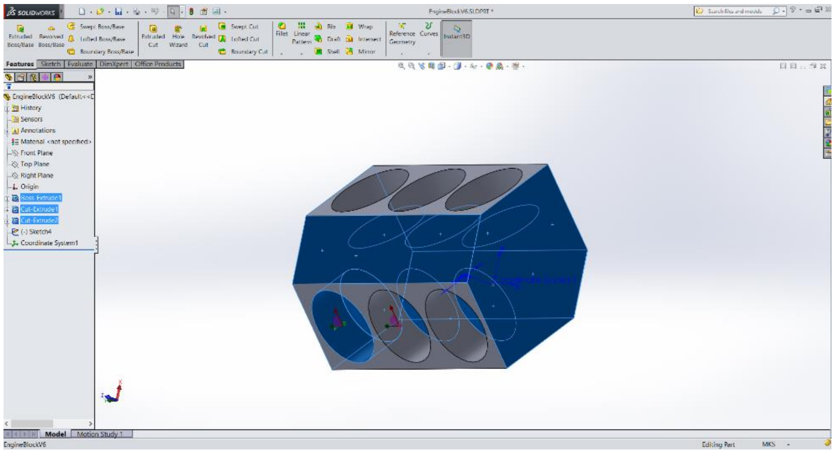


Figure G-22: Selecting the eleventh face of the given workpiece example.

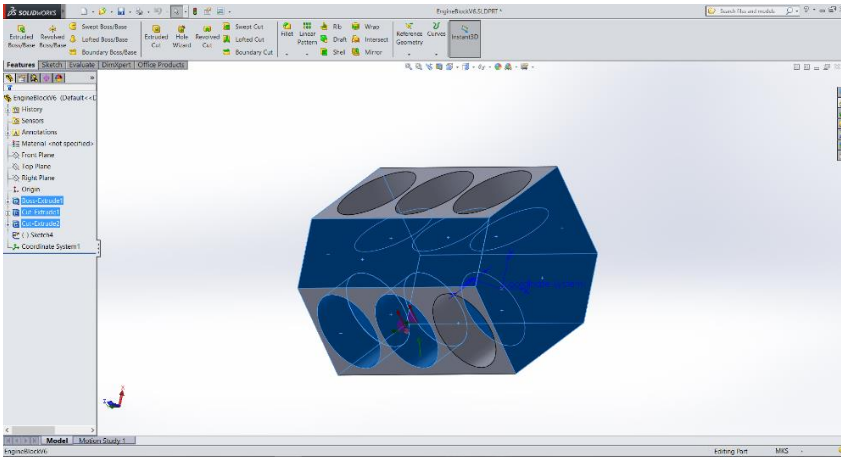


Figure G-23: Selecting the twelfth face of the given workpiece example.

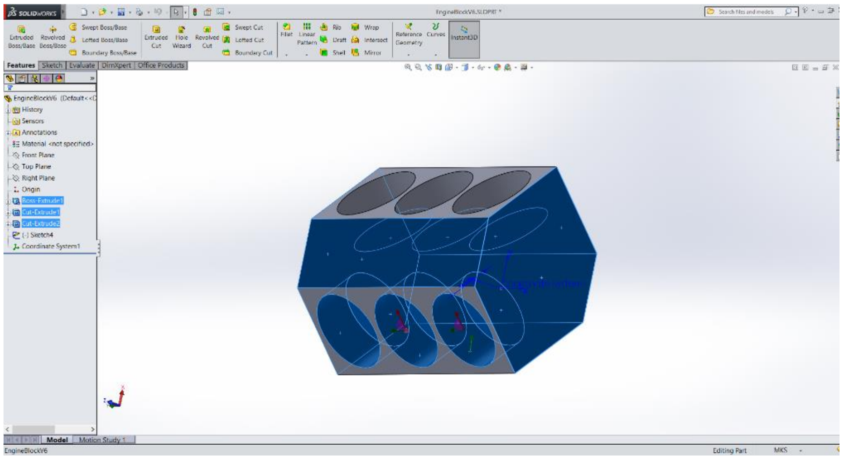


Figure G-24: Selecting the thirteenth face of the given workpiece example.

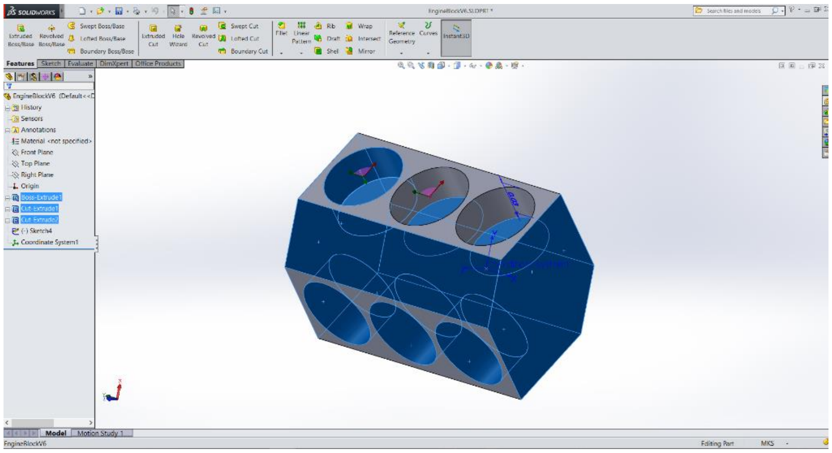


Figure G-25: Selecting the fourteenth face of the given workpiece example.

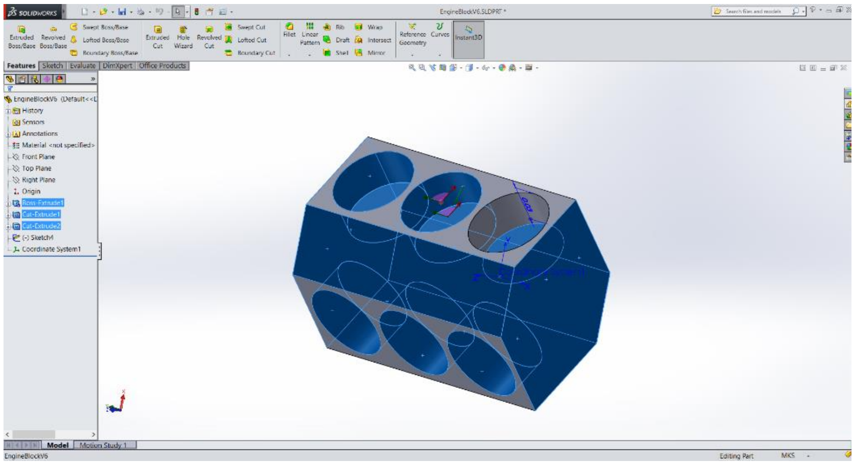


Figure G-26: Selecting the fifteenth face of the given workpiece example.

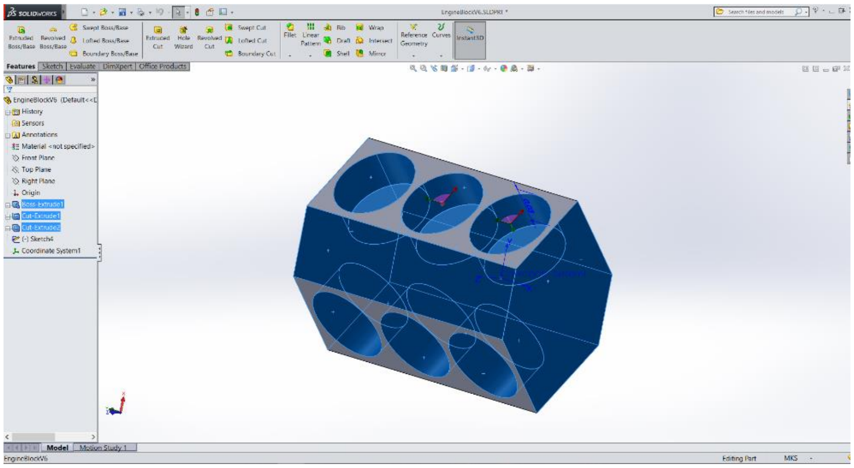


Figure G-27: Selecting the sixteenth face of the given workpiece example.

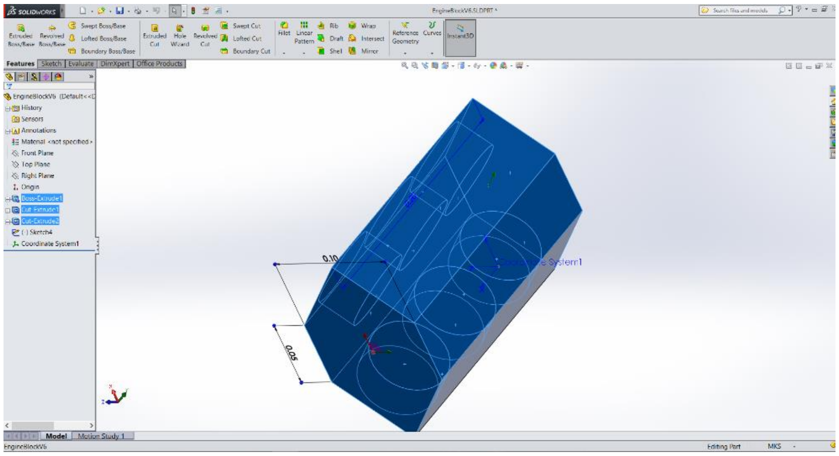


Figure G-28: Selecting the seventeenth face of the given workpiece example.

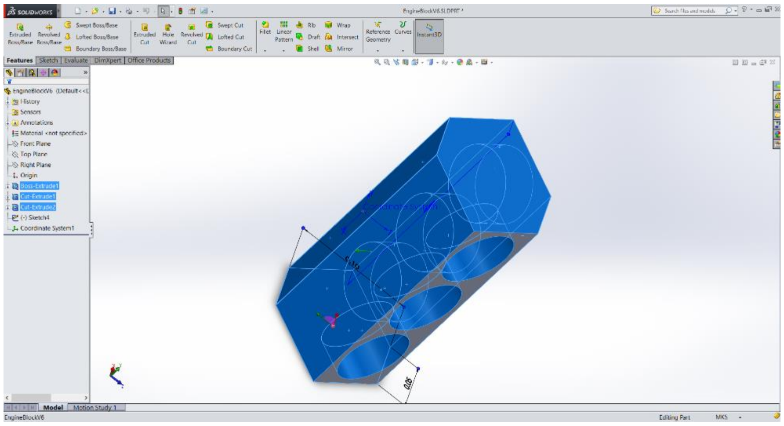


Figure G-29: Selecting the eighteenth face of the given workpiece example.

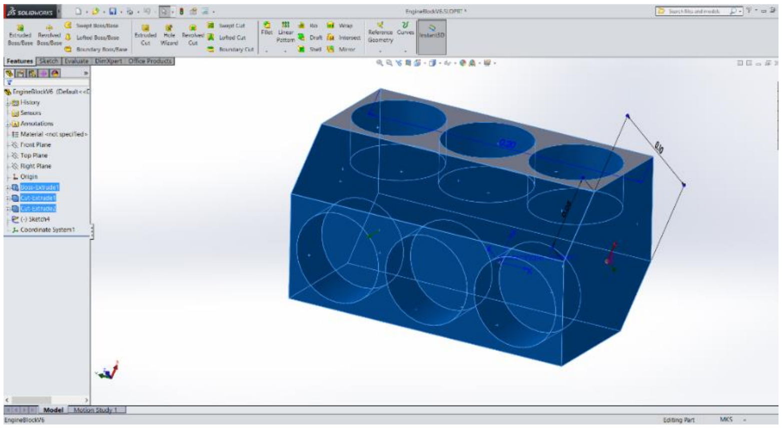


Figure G-30: Selecting the nineteenth face of the given workpiece example.

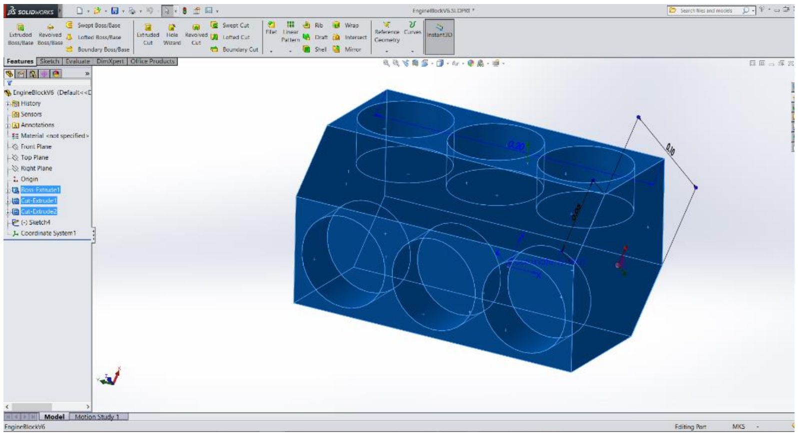
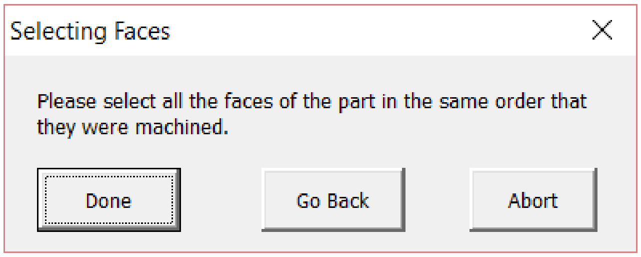


Figure G-31: Selecting the twentieth face of the given workpiece example.

14. After selecting all faces, in order. Click on Done. This will save the order and now you can remove the face selection.

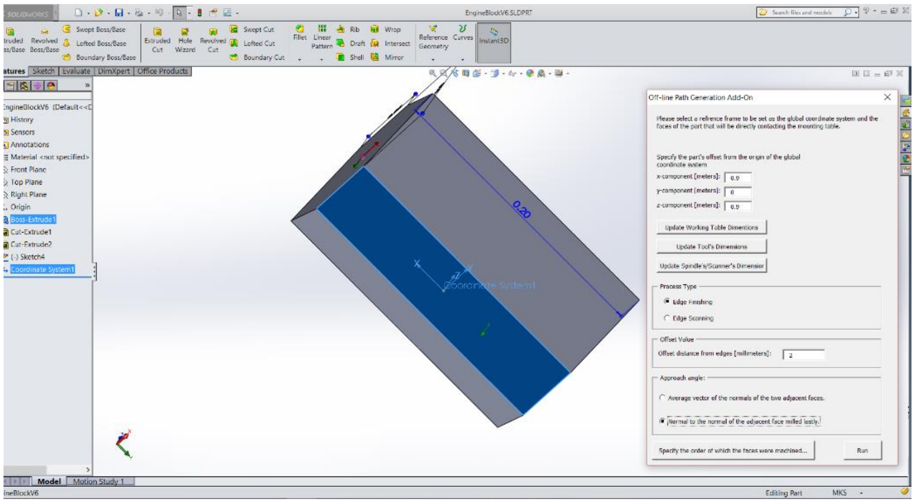


Selecting Faces

Please select all the faces of the part in the same order that they were machined.

Figure G-32: Confirming the face selection procedure.

15. Remove the face selection by clicking anywhere in empty space.
16. Select the reference frame defined earlier and any faces that are in direct contact with the working table or fixture. In this example only one face is in direct contact with the table, as shown below.



Off-line Path Generation Add-On

Please select a refrence frame to be set as the global coordinate system and the faces of the part that will be directly contacting the mounting table.

Specify the part's offset from the origin of the global coordinate system

x-component [meters]: 0.9

y-component [meters]: 0

z-component [meters]: 0.9

Update Working Table Dimentions
Update Tool's Dimensions
Update Spindle's/Scanner's Dimensior

Process Type

● Edge Finishing

○ Edge Scanning

Offset Value

Offset distance from edges [millimeters]: 2

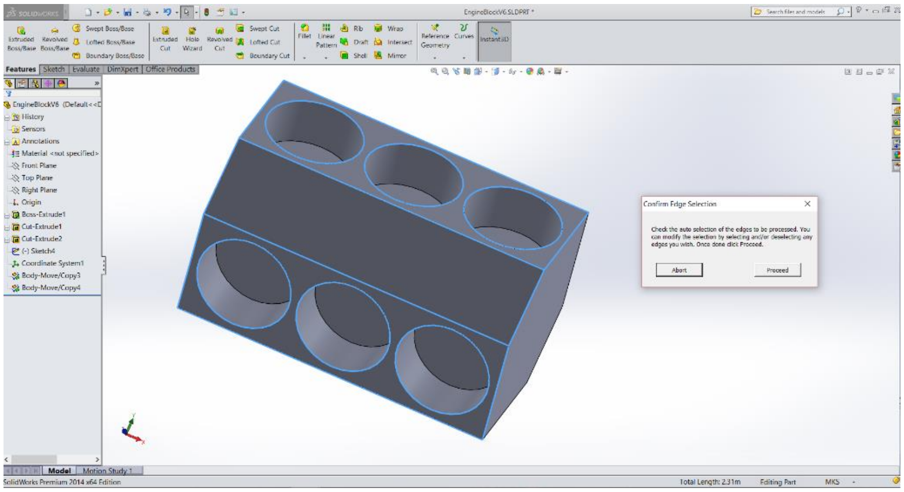
Approach angle:

○ Average vector of the normals of the two adjacent faces.

● Normal to the normal of the adjacent face milled lastly.

Figure G-33: Selecting the defined reference frame and ground face.

17. Click Run.
18. Check the default edge selection. The software predicts the edges to be processed and select them automatically for the user. The user can still modify this selection by selecting or deselecting any other edges at well. In this example four edges were deselected from the default selection. After finishing with modifying edge selection click on Proceed.

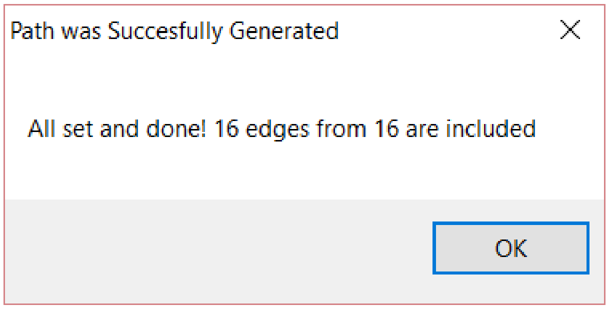


Confirm Edge Selection

Check the auto selection of the edges to be processed. You can modify the selection by selecting and/or deselecting any edges you wish. Once done click Proceed.

Figure G-34: Modifying the default edge selection and confirming it.

19. Wait for the program to generate the path.
20. When path is generated a message box shows up to inform the user. Click OK.



Path was Succesfully Generated

All set and done! 16 edges from 16 are included

Figure G-35: Feedback message to the user at the end of path generation process.

21. Go to the directory of the CAD model part. The generated path is going to be stored in the same directory with the name NominalPath.
22. Congratulations now you generated a path for your task.

G.2. Path Execution

The ABB User Interface is the software responsible for transferring the nominal path from the computer to ABB IRB2000's memory, in order to be able to execute the path. The following steps give the full detailed explanation to generate a path using this package.

1. Launch ABB User Interface software.
2. Plug the USB cable of the ABB to your computer.
3. Set up the connection. Simply go to "File/Open Serial Port" then choose the correct COM number.

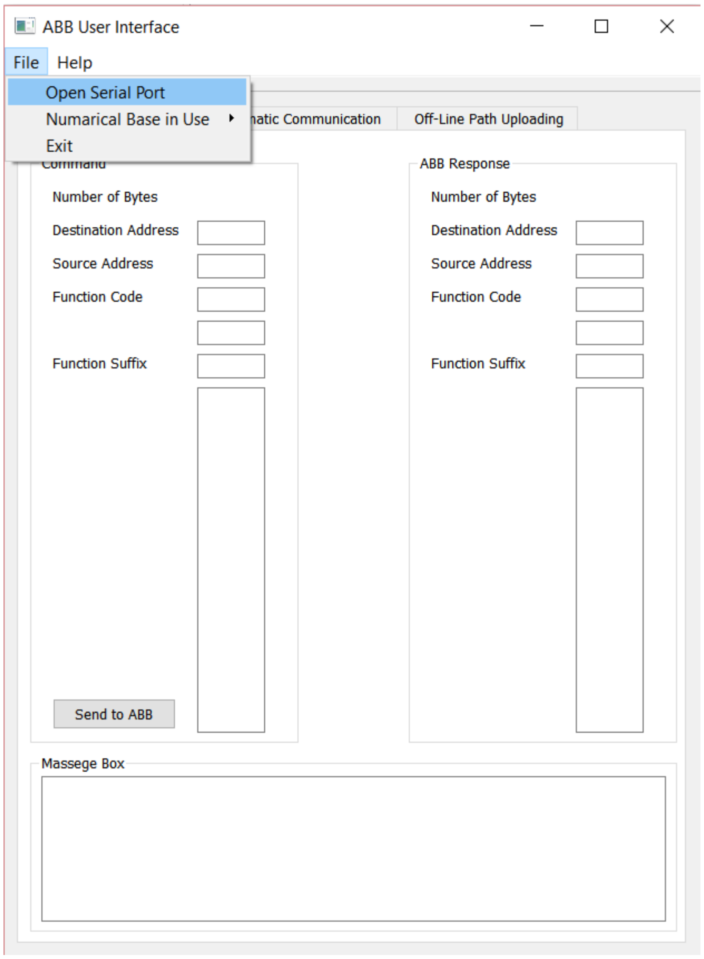
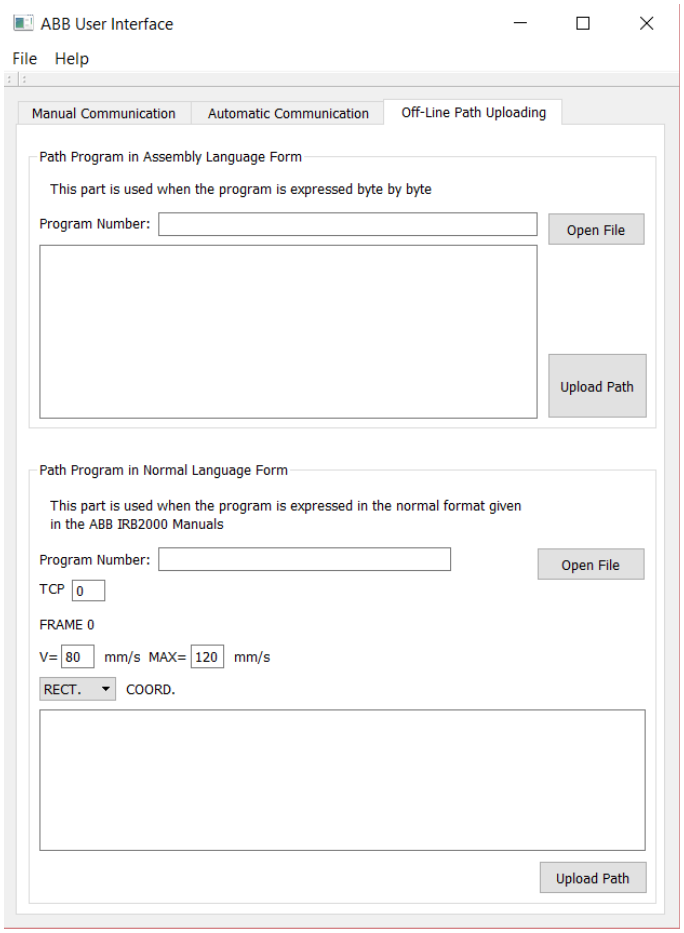


Figure G-36: Opening the serial port with ABB.

4. Open “Off-Line Path Uploading” tab.



Path Program in Assembly Language Form

This part is used when the program is expressed byte by byte

Program Number:

Path Program in Normal Language Form

This part is used when the program is expressed in the normal format given in the ABB IRB2000 Manuals

Program Number:

TCP 0

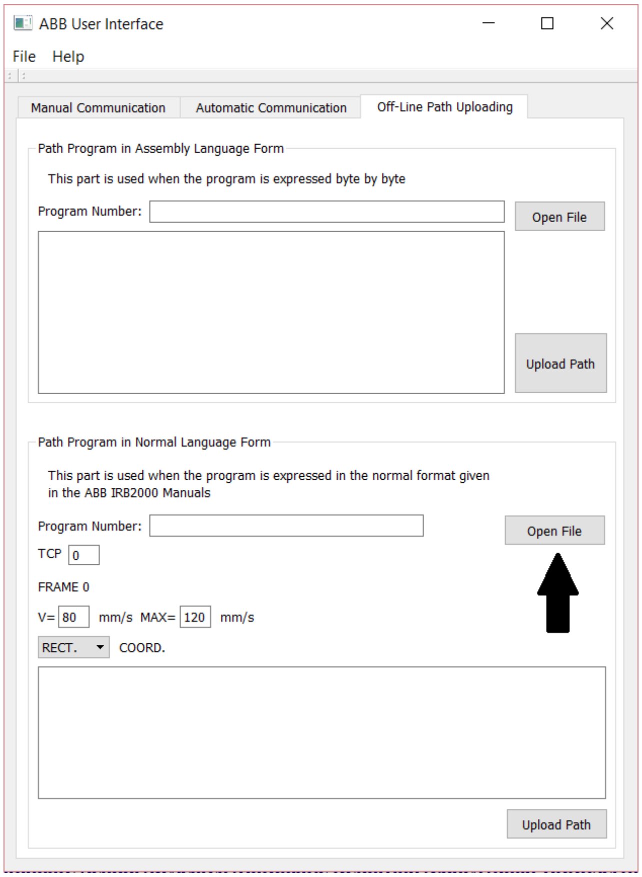
FRAME 0

V= 80 mm/s MAX= 120 mm/s

RECT. COORD.

Figure G-37: Opening "Off-Line Path Uploading" tab.

5. Click on “Open File” button in “Path Program in Normal Language Form” frame.



This part is used when the program is expressed byte by byte

This part is used when the program is expressed in the normal format given in the ABB IRB2000 Manuals

TCP 0

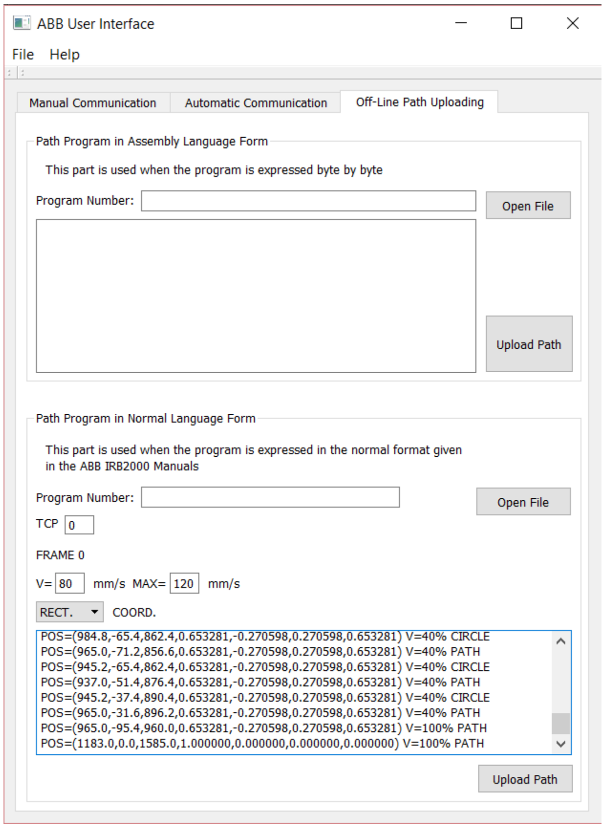
FRAME 0

V= 80 mm/s MAX= 120 mm/s

RECT. COORD.

Figure G-38: Opening the previously generated path text file.

6. Select the previously generated path.
7. Check if the text is uploaded on the software. Make sure that there is not any empty lines in the text. Especially at the very end of the text.



Path Program in Assembly Language Form

This part is used when the program is expressed byte by byte

Program Number:

Path Program in Normal Language Form

This part is used when the program is expressed in the normal format given in the ABB IRB2000 Manuals

Program Number:

TCP 0

FRAME 0

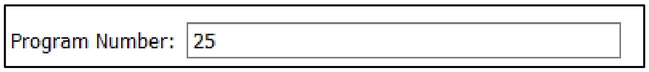
V= 80 mm/s MAX= 120 mm/s

RECT. COORD.

```
POS=(984.8,-65.4,862.4,0.653281,-0.270598,0.270598,0.653281) V=40% CIRCLE
POS=(965.0,-71.2,856.6,0.653281,-0.270598,0.270598,0.653281) V=40% PATH
POS=(945.2,-65.4,862.4,0.653281,-0.270598,0.270598,0.653281) V=40% CIRCLE
POS=(937.0,-51.4,876.4,0.653281,-0.270598,0.270598,0.653281) V=40% PATH
POS=(945.2,-37.4,890.4,0.653281,-0.270598,0.270598,0.653281) V=40% CIRCLE
POS=(965.0,-31.6,896.2,0.653281,-0.270598,0.270598,0.653281) V=40% PATH
POS=(965.0,-95.4,960.0,0.653281,-0.270598,0.270598,0.653281) V=100% PATH
POS=(1183.0,0.0,1585.0,1.000000,0.000000,0.000000,0.000000) V=100% PATH
```

Figure G-39: Result of uploading the generated path example.

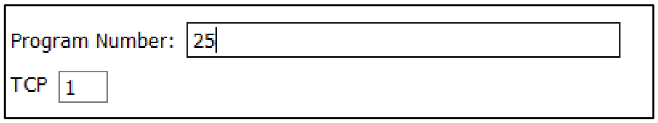
8. Enter the program number for the path. This is the name of the path when it will be stored in the robot’s memory. Choose any number from 20-2000. In this example 25 is entered.



Program Number: 25

Figure G-40: Setting the program number in ABB memory.

9. Define the TCP number for the path. This is for the kinematics for the end-effector, read programming manual for full details. In this example, TCP 1 is taken. This is the TCP that correspond to the white spindle holder.

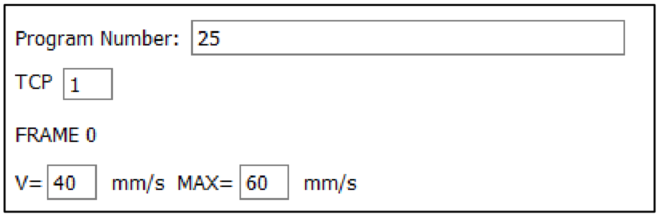


Program Number: 25

TCP 1

Figure G-41: Setting the TCP number to be used in the path.

10. Set the velocity of the end-effector while executing the path. For processing motions, the end-effector’s speed is 40% of the set speed. In this example 40 mm/s is set as the speed. Therefore when processing, the end-effector moves at 16mm/s.
11. Set the MAX speed. Recommended value is 1.5V. Hence, for this example MAX=1.5*40=60mm/s.



Program Number: 25

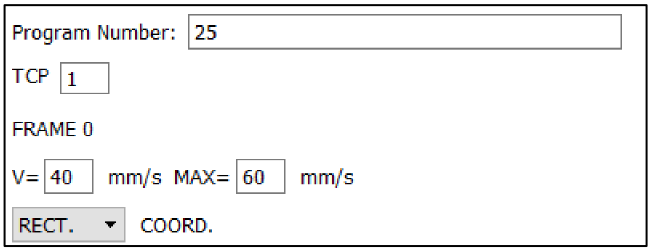
TCP 1

FRAME 0

V= 40 mm/s MAX= 60 mm/s

Figure G-42: Setting the end-effector velocities.

12. Always let the Coordinates be (RECT.). Use ROBOT COORD. Only if you are very familiar with the robot!



Program Number: 25

TCP 1

FRAME 0

V= 40 mm/s MAX= 60 mm/s

RECT. COORD.

Figure G-43: Selecting RECT. COORD.

13. Click “Upload Path”.
14. Feedback message next to the “Upload Path” is going to show up to inform the user if the upload was successful or not. Depending on the length of the path text. As an average time, wait for one minute or less for the path to be fully transferred to ABB.