TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Jamal Muhammad

Introduction to an Efficient Process for Automatic Offline Program Generation for a Robotic Spot Welding Assembly Line.

MASTER OF SCIENCE THESIS

Examiner: Professor ReijoTuokko
Examiner and topic approved in the
Council of the Faculty of Engineering
Sciences on 13.08.2014

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Program in Automation
Muhammad, Jamal:
Master of Science Thesis, 70 pages, Appendices 19 pages
September 2014
Major: Mechatronics
Minor: Robotics and Automation
Examiner: Prof. ReijoTuokko
Keywords: Robotics, Spot Welding, Offline Programming, CAD, Simulation.

One of the most important applications of industrial robots is spot welding which is used in high production applications mostly in automotive industries where mass production is required. The speed, precision, efficiency and the resulting cost reduction due to mass production are well accepted and well documented advantages of automation of spot welding process using robots. In order to meet the new challenges of increased global competition, manufacturers are forced to seek new technologies for improved production and cost reduction. Such cost cutting efforts can only be achieved by improving the offline programming method.

Offline programming is one of the most crucial parts of modern automotive manufacturing process. In this Master's thesis a process was developed for faster and efficient offline programming of industrial manipulators in spot welding application. The thesis work has been conducted in Visual Components Oy, Espoo, Finland. In traditional practice there are lots of manual steps involved in the robotic spot welding area. The whole process design of the robotic spot welding is not simple and includes CAD design of the part, shape and complexity of the parts which needs to be spot weld, design of the robot work cell, design and selection of spot weld gun, required production rate, offline programming tool, robot calibration, work cell calibration, work piece positioner design etc. In this report an approach to implement the offline programming of robot based on simulation software with the process knowledge of car-body in white was proposed and partially developed. Some common problems such as motion simulation, collision detection and calibration can be partly solved by this approach.

The thesis consisted of a theoretical section to investigate the current state of art of offline programming tools and methods and a practical section to develop working prototype for demonstration. The implementation of the prototype used the application programmer's interface (API) available with the simulation software. A prototype was developed to propose an efficient process for putting the whole spot welding process starting for CAD design, work cell setup, offline programming and calibration in a closed loop.

PREFACE

This Master of Science Thesis has been undertaken at Tampere University of Technology (TUT) at the Department of Mechanical Engineering and Industrial Systems.

First of all I would like to thank Professor Reijo Tuokko and Jyrki Latokartano for the opportunity to work in the department and also for leading me into this challenging but still fascinating world of industrial robotics. I would also like to express my gratitude to them for their guidance and support during this thesis. I always remember my working with department colleagues as a nice, well-equipped and friendly multinational place.

Once again I would like to thank my teacher and mentor Jyrki Latokartano for preparing and giving such a high level education and motivation in the field of robotics and process automation and for giving me such wonderful moments of learning and progressing during my study and work in the university.

I am also honestly grateful to Dr.Scott Walter, M.Sc Samuli Ahonen and M.Sc Jarno Hapasaari of Visual Components Oy for their guidance and valuable comments related to this thesis.

A world of thanks to my parents for their unconditional love, constant support and encourange during my whole life.Thanks to my younger sister Rabeya who always persuaded me to aim for the highest in my life and to never surrender to the problems.

ACRONYMS

API	Application Programming Interface
AWS	American Welding Society
BIW	Body in White
CAD	Computer Aided Design
CAGR	Compound Annual Growth Rate
COM	Component Object Model
CSV	Comma Seperated Value
DCOM	Distributed Component Object Model
DPE	Digital Process Engineering
DPM	Digital Process Manufacturing
EOA	End of Arm
EOAT	End of Arm Tooling
FEM	Finite Element Method
IFR	International Federation of Robotics
MTBF	Mean Time Between Failures
OLE	Object Linking and Embedding
OLP	Offline Programming
PCB	Printed Circuit Board
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PTP	Point To Point
RAC	Radical Alignment Constraint
RMS	Root Mean Square
ROS	Robot Operating Software
RRS	Realistic Robot Simulation
RRS-II	Realistic Robot Simulation II
RSL	Robotic Sequence Language
SCARA	Selectively Compliant Assembly Robot Arm
SME	Small and Medium Sized Enterprized
TCP	Tool Centre Point
TSP	Travelling Salesman Problem
VRLM	Virtual Reality Modelling Language

LIST OF FIGURES

LIST OF TABLES

Table of Contents

1. INTRODUCTION

1.1. Research Statement and Motivation

In the current work the research is focused on one of the sub-assembly processes involved in the spot welding process of automotive industry. This sub-assembly process is offline programming (OLP) of spot welding robots. This thesis tries to promote the integration of CAD Software, 3D Robotic Simulation Software and Robotic Spot Welding task in automotive industry so that different CAD software, 3D Robotic Simulation Software and different Robots can exchange information with each other for designing the spot welding assembly line in automotive industry. The research problem can thus be stated as improving the OLP method of spot welding process in automotive industry.

1.2. Objective of the Thesis

According to statistics about half of the industrial robots around the world are engaged with various forms of welding job. This has reduced the high risk of radiation and hazard environment due to manual welding by skilled welders and increased the working condition and welding quality and efficiency. Robot technology is a high-tech integrated multidisciplinary of computer, cybernetic, mechanism, information and sensor technology, artificial intelligence and bionics. There are two ways of programming industrial robots. One is *Online programming and another is *Offline programming (OLP). Online programming consists of Teach pendant, Playback or Manual programming. Offline programming consists of Textual programming, CAD programming, Macro programming or Acoustic programming. More than half of the total industrial robots applied in factories over the world are in automotive industries and among them more than half of the robot's are in spot welding operation. The objective of this thesis work was to find a faster and more versatile OLP method for the automotive industry. Today offline programming and simulation software are the decisive and crucial tools of production planning and process in automotive industry. Automotive manufacturers are facing fierce competition from their competitors and always on the search for better and more efficient solutions for the production process. [1]

Industrial robots use proprietary software and control systems. In future more and more engineers are bridging up the barriers of process simulation and OLP by advance digital manufacturing solutions enabling multi vendor systems. Today's industrial robotic system in general can be termed open and closed. An open system refers to the robotic system which is flexible and easy to develop further solutions on its platform, as a closed system refers to the proprietary solutions where only limited tasks can be done and development of the system is not possible without the manufacturers consent. ROS (Robot Operating System) or RRS (Realistic Robot Simulation) are examples of the effort towards more flexible and open frameworks.

However the area of this thesis work is focused on automotive spot welding application of industrial robots. Long before a car is being manufactured the assemblies are designed in CAD software and a virtual copy of the robot and its environment in created in the simulation software where the robots programs are made, edited, simulated and verified before they are transferred to the physical robot. Robotic workcell simulation is a modelling based problem solving approach and the methodology consists of 6 steps, as shown below.

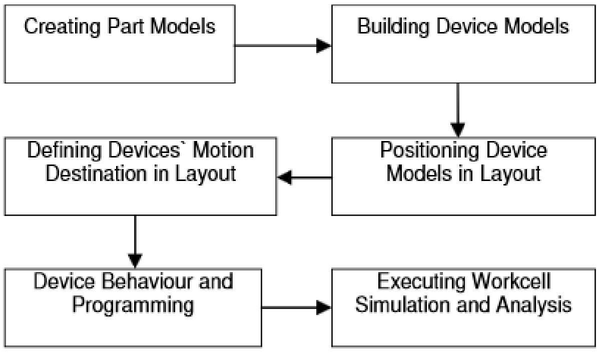

Creating Part Models
Building Device Models
Positioning Device Models in Layout
Defining Devices` Motion Destination in Layout
Device Behaviour and Programming
Executing Workcell Simulation and Analysis

Figure 1-1 A methodology for robotic workcell modelling [2]

For the thesis work CAD based OLP method was selected as the target is to propose a better method for the Virtual Manufacturing Process in the robotic spot welding application.The thesis work was done on the Factory Simulation and OLP software from Visual Components. The benefits of CAD based methods are verification of the following facts –

- Components attitude in space.
- Movement of components and tools.
- Are working points reachable?
- Time needed for movement.
- Collision with environment.
- Checking for alternatives.
- OLP with generation of robot program as well as PLC programs

2. INDUSTRIAL ROBOTICS

In this chapter the most important features of current industrial manipulators are going to be discussed. In modern flexible manufacturing systems robots are essential elements. Traditional manufacturing systems are now vastly replaced by flexible manufacturing systems (FMS). The behaviour of an assembly system is never completely predictable and can be hardly anticipated, because it has many sources of uncertainity [2] [3]. Those are –

- material and parts variation
- fixturing errors
- positioning errors
- auxiliary manufacturing process equipment error

Flexible manufacturing is the current approach to minimize production costs. In a modern flexible manufacturing system industrial robots are seen as the key element because of their high efficiency in flexibility and programmability [3]. Automotive spot welding has always been one of the major applications of robots. In old days large, heavy car body parts have been held by clamping jigs, tacked together by operators using multiple welding guns and then spot welded manually. The industrial revolution changes the pace and competition of the automotive industry and it all started back in 1966 when first steps were taken to use a robot to guide the welding guns and combine the control of the robot with the control of the gun. In 1969 General Motors in USA installed 26 Unimate robots in a car body spot welding line. Then in 1970 Daimler/Benz in Europe used Unimate robots for body side spot welding. Currently there are big robot companies like ABB, Motoman, Kuka, Comau, Nachi, Fanuc etc. which provide the automotive industry with specialized robotic spot welding systems. [4]

2.1. Types of Industrial Robots

Articulated – This robot design features rotary joints and can range from simple two joint structures to ten or more joints. The arm is connected to the base with a twisting joint and the links are connected via rotary joints.

Cartesian – These are also called rectilinear or gantry robots. Cartesian robots have three linear joints that use the cartesian coordinate system (X, Y & Z). They also may have an attached wrist to allow rotational movement.

Cylindrical – The robot has at least one rotary joint at the base and at least one prismatic joint to connect the links. They work within a cylindrical work envelope.

Polar – The arm is connected to the base with a twisting joint and a combination of two rotary joints and one linear joint.

SCARA – Commonly used in assembly applications, this Selective Compliant Assembly Robot Arm for robotic assembly features two parallel joints.

Delta – Jointed parallelogram connected to a common base. Delta robots are heavily used in food, pharmaceutical and electronic industry and capable for fast and delicate movements.

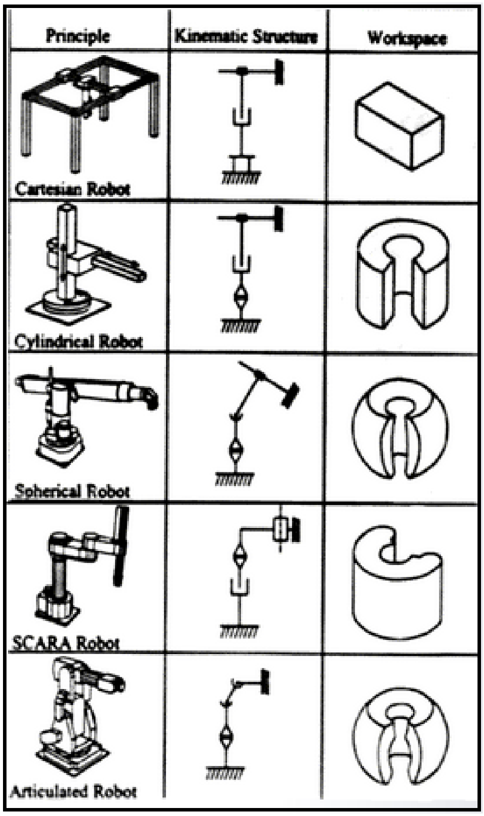
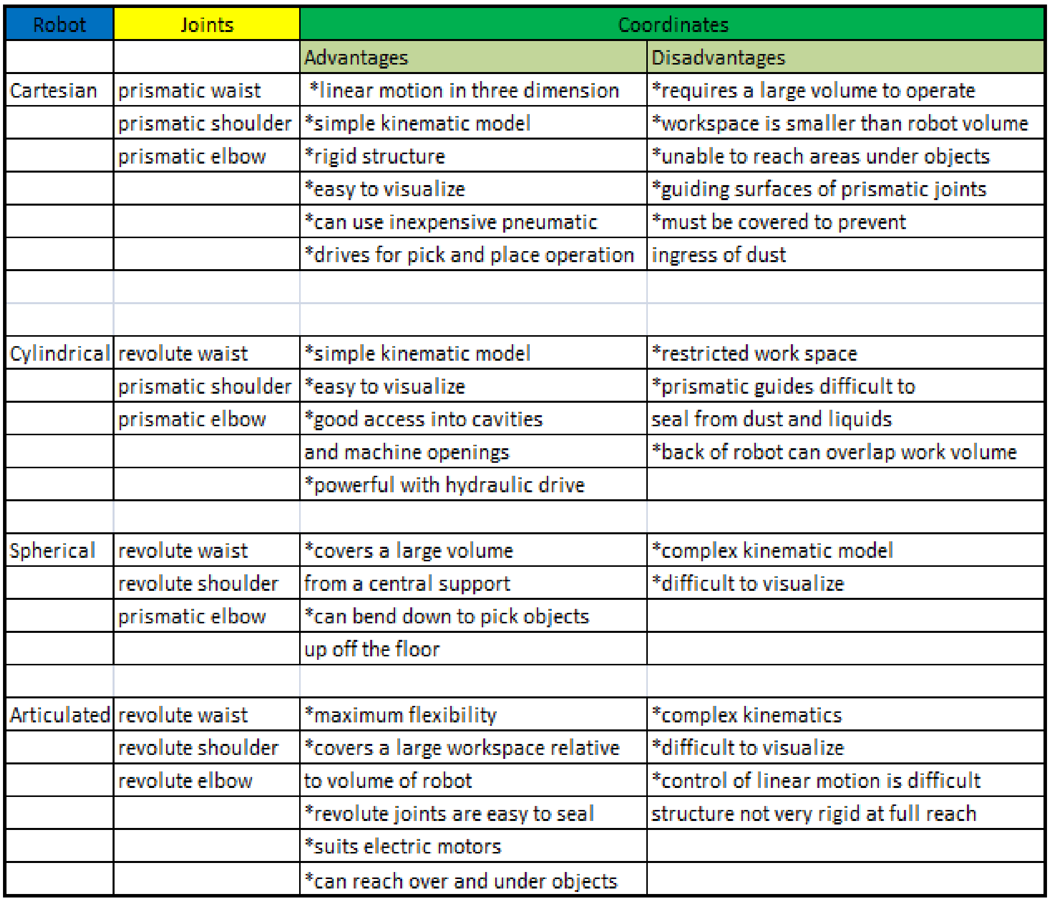


Figure 2-1 Types of Industrial Manipulator Arms



Robot	Joints	Coordinates	
		Advantages	Disadvantages
Cartesian	prismatic waist	*linear motion in three dimension	*requires a large volume to operate
	prismatic shoulder	*simple kinematic model	*workspace is smaller than robot volume
	prismatic elbow	*rigid structure	*unable to reach areas under objects
		*easy to visualize	*guiding surfaces of prismatic joints
		*can use inexpensive pneumatic	*must be covered to prevent
		*drives for pick and place operation	ingress of dust
Cylindrical	revolute waist	*simple kinematic model	*restricted work space
	prismatic shoulder	*easy to visualize	*prismatic guides difficult to
	prismatic elbow	*good access into cavities	seal from dust and liquids
		and machine openings	*back of robot can overlap work volume
		*powerful with hydraulic drive	
Spherical	revolute waist	*covers a large volume	*complex kinematic model
	revolute shoulder	from a central support	*difficult to visualize
	prismatic elbow	*can bend down to pick objects	
		up off the floor	
Articulated	revolute waist	*maximum flexibility	*complex kinematics
	revolute shoulder	*covers a large workspace relative	*difficult to visualize
	revolute elbow	to volume of robot	*control of linear motion is difficult
		*revolute joints are easy to seal	structure not very rigid at full reach
		*suits electric motors	
		*can reach over and under objects	

Table 2-1 Comparison of different robotic arms

Programmable Industrial Robots

In today's industry programmable automation is the key focus and it is adaptable to manufacture a wide variety of products in variable lot sizes with mass production costs. Today's modern factories are designed with industrial robots, numerical control of machine-tools, computer aided design and manufacturing and production information and control [5]. The major applications of programmable robots are with sensors to material handling, inspection and assembly operations e.g. spot welding, arc welding, gluing etc.

Commercial Figure

Today industrial robots are the key element of Factory Automation. More that 1.1 million industrial robots are operating in factories all over the world [6]. In the year 2012 more than 1, 60,000 industrial robots were sold around the world [7]. The first industrial robot was developed in USA at General Motors plant to work with heated die-casting machines. It was named Unimate and was developed by George Devoland Joseph Engelberger. After than industrial robotics has gone through tremendous innovation and development and in 2013 any European manufacturing plant can’t be even considered without industrial robots.

In 2011 automotive industry was the biggest purchaser of industrial robots; it contributed to 36% of total annual supply. After that electrical&electronic industry accounted for 23%, rubber&plastic industry bought 10,500 units of industrial robots, food&beverage industry accounted for 3% and metal&machinery industry accounted for 9% of total annual supply of Industrial robots [6].

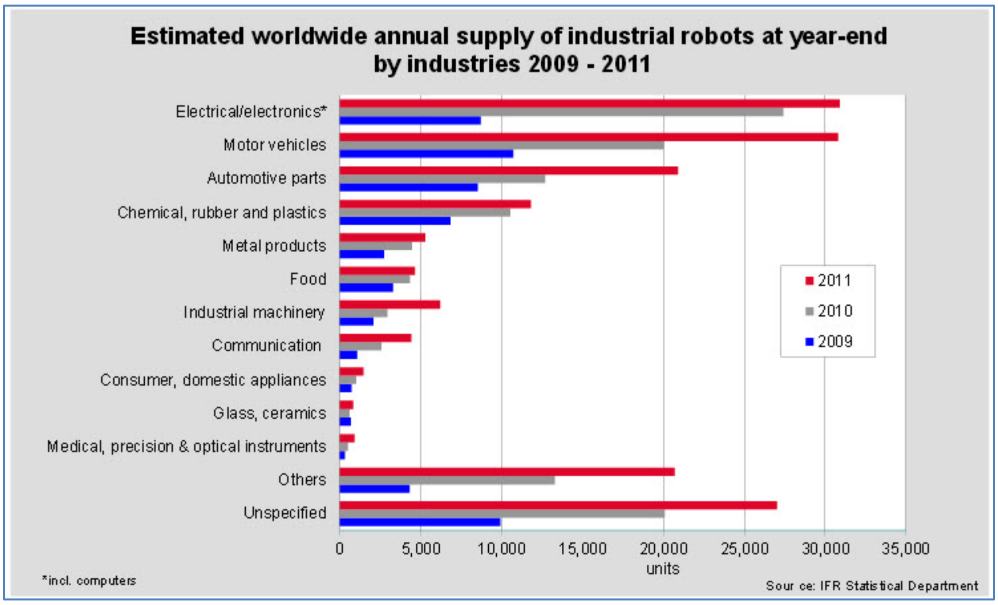
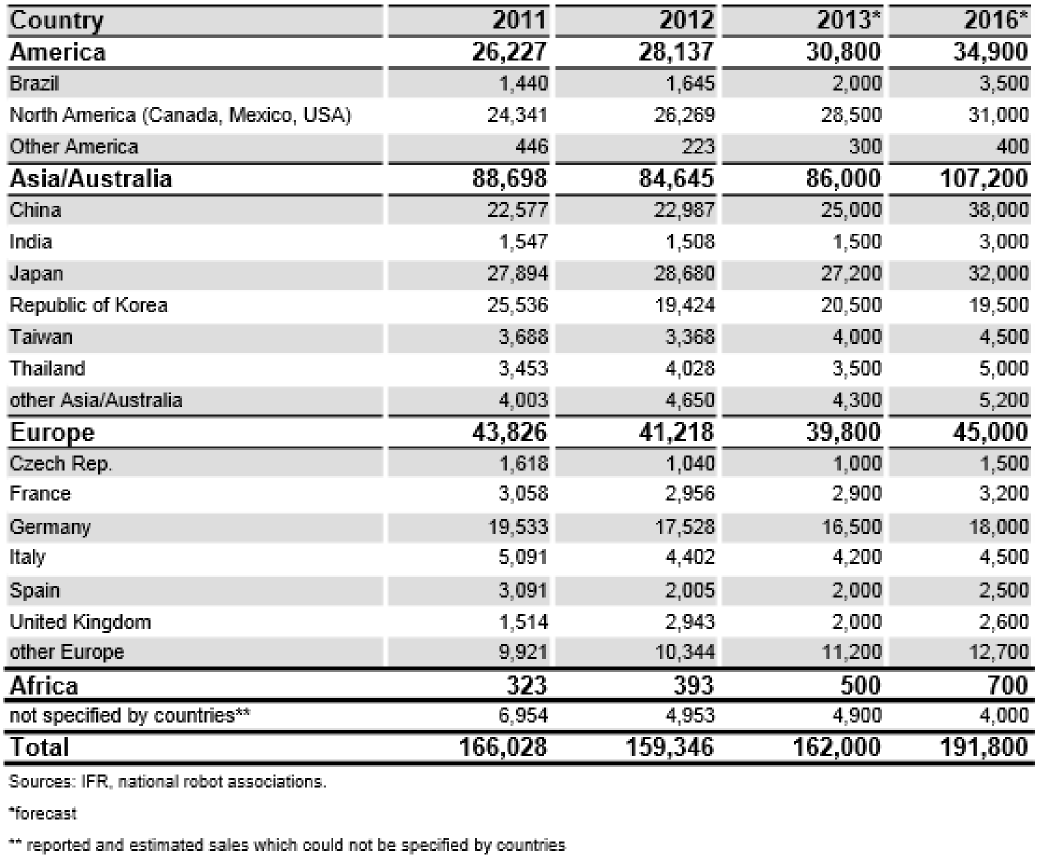


Figure 2-2 Industrial Robot supply during 2009-2011

According to the estimates of IFR(International Federation of Robotics) the total worldwide stock of operational industrial robots at the end of 2011 was in the range of 1,153,000 and 1,400,000 units.

Robot density is a parameter to represent the usage of industrial robots in more precise way, rather than installed number of units. It gives the number of multipurpose industrial robots per 10,000 person employed in manufacturing industry. In 2011 Republic of Korea achieved the highest figure with robot density of 347, Japan in second place with 339 and Germany the third with a robot density of 261. Between 2013 and 2015 worldwide robot sales will increase by 5% per year, 5% in Americas, about 6% in Australia/Asia and about 2% in Europe. There are new markets that are growing like Turkey and Middle East. From 2014 to 2016 installation of industrial robots are estimated to increase by 6% on average per year (CAGR), about 8% in Asia/Australia and about 4% in the Americas and Europe.



Country	2011	2012	2013*	2016*
America	**26,227**	**28,137**	**30,800**	**34,900**
Brazil	1,440	1,645	2,000	3,500
North America (Canada, Mexico, USA)	24,341	26,269	28,500	31,000
Other America	446	223	300	400
Asia/Australia	**88,698**	**84,645**	**86,000**	**107,200**
China	22,577	22,987	25,000	38,000
India	1,547	1,508	1,500	3,000
Japan	27,894	28,680	27,200	32,000
Republic of Korea	25,536	19,424	20,500	19,500
Taiwan	3,688	3,368	4,000	4,500
Thailand	3,453	4,028	3,500	5,000
other Asia/Australia	4,003	4,650	4,300	5,200
Europe	**43,826**	**41,218**	**39,800**	**45,000**
Czech Rep.	1,618	1,040	1,000	1,500
France	3,058	2,956	2,900	3,200
Germany	19,533	17,528	16,500	18,000
Italy	5,091	4,402	4,200	4,500
Spain	3,091	2,005	2,000	2,500
United Kingdom	1,514	2,943	2,000	2,600
other Europe	9,921	10,344	11,200	12,700
Africa	**323**	**393**	**500**	**700**
not specified by countries**	6,954	4,953	4,900	4,000
Total	**166,028**	**159,346**	**162,000**	**191,800**

Sources: IFR, national robot associations.

*forecast

** reported and estimated sales which could not be specified by countries

Table 2-2 Estimated operational stock of multipurpose industrial robots. [8]

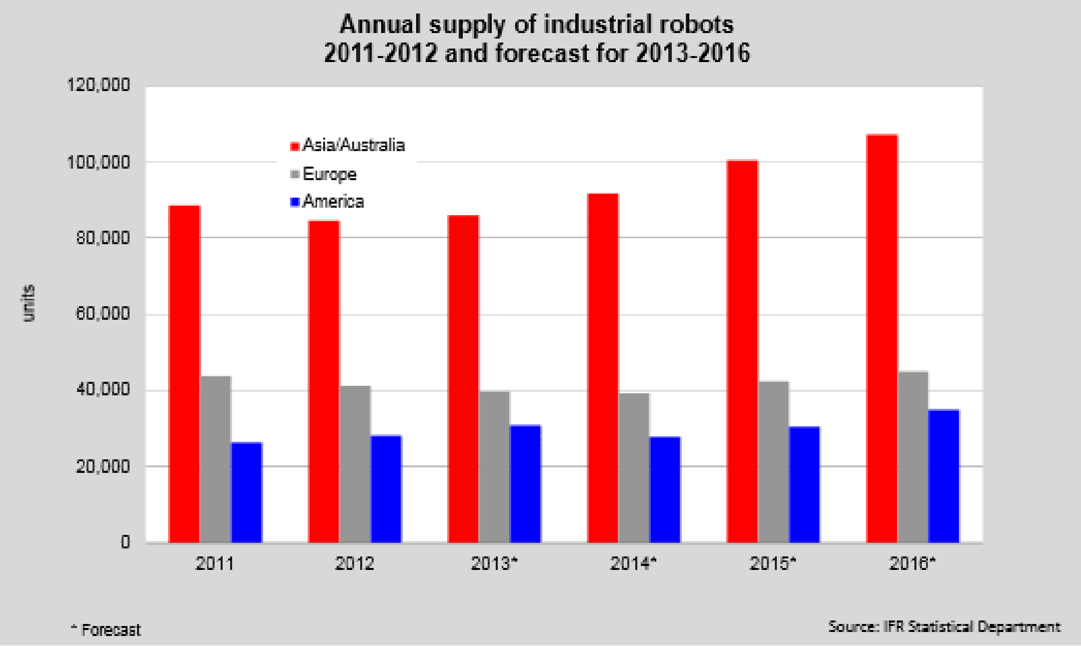


Figure 2-3 Annual supply of Industrial Robots [8]

2.2. Industrial Robot Programming

Most of the applications of industrial robots are in the field of handling, welding and assembly. This section wil discuss one of the most important topics regarding the efficiency of industrial robots i.e. motion planning.

One of the most challenging topics of robot motion planning has been advanced path planning or trajectory planning, intelligent collision avoidance and there have been numbrous researches to develop this kind of motion planning. A short introduction is given below on some of the current researches done on this topic.

2.2.1. Collision Free Trajectory Planning

There are algorithms for collision free path generation which are computed in complex space. In robotic spot welding application almost all the robots are 6-axis articulated, while 7-axis articulated robots are also getting ground, and spot welding paths are often complex and many obstacles on a sequence of spot welds. With traditional tools available in the OLP software the spot welding initial motion planning can be done quite easily and quickly. Major time is required for fine tuning the path movements and making them collision free. In [8] a collision free path planning algorithm is introduced for four cartesian robotic arms in a PCB manufacturing appplication. Although the problem in spot welding application is quite different it’s worth studying the procedure. A test bed was made which included a flying probe system, a path planning algorithm was introduced which is based on previous well established algorithm’s like NN Algorithm, genetic algorithm, nearest neighbour algorithm, band division algorithm etc. For the test a mechanical structure with PCB holder and four cartesian robotic arms were installed. A probe was mounted at the end of each arm. A computer vision system with two different micro-CCD cameras for inspection and an industrial PC for controlling the cartesian robots were installed. The below figure shows the proposed algorithm for the calculation of the trajectory that each probe must follow to travel to the next test point in the PCB avoiding collision with the PCB as well as with other cartesian arm.

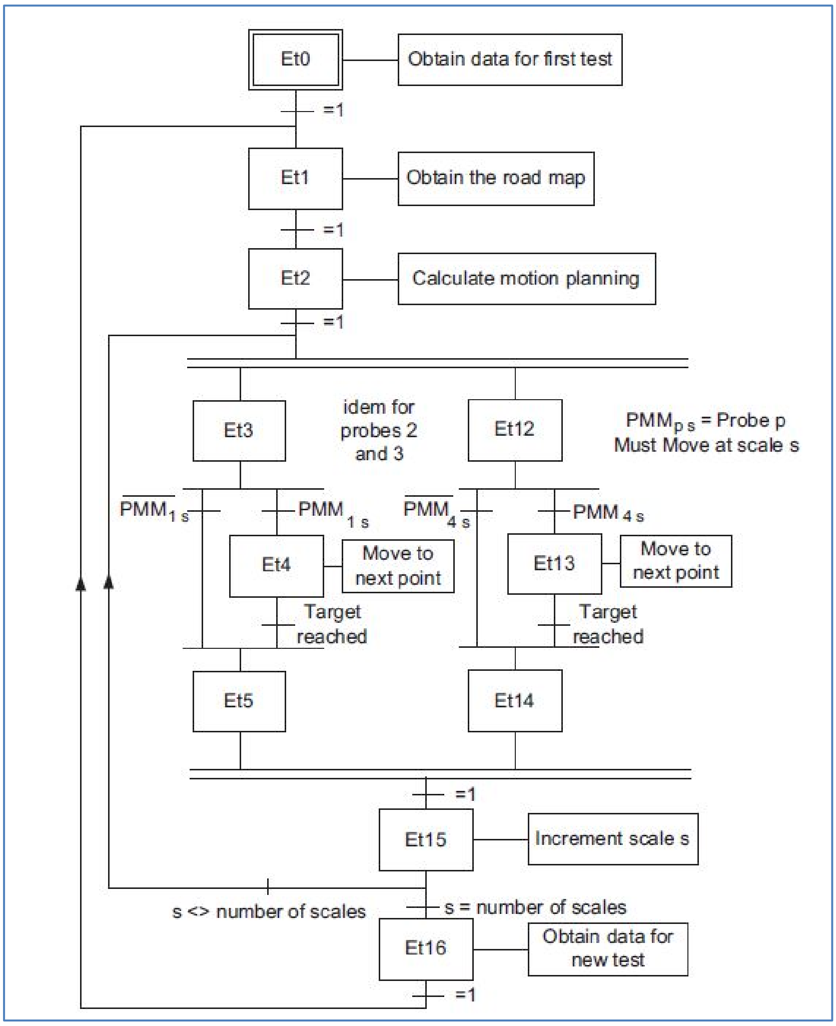


Figure 2-4 Collision free path planning algorithm proposed in [9]

2.2.2. Fuzzy Path Planning of Industrial Manipulators

A technique has been developed by (Zavlangas & Tzafestas,2006)for fast online collision-free trajectory generation which is based on separate fuzzy logic based obstacle avoidance units, each controlling one individual link lj, j=1...n and each unit having two principal inputs:

1. dj, the distance between the link and the nearest obstacle
2. Ɵj-Ɵj, target the difference between the current link configuration and the target configuration.

The output variable of each unit is the motor command τ_j which is the actuation signal to the link motor and fed to the manipulator in each iteration cycle.

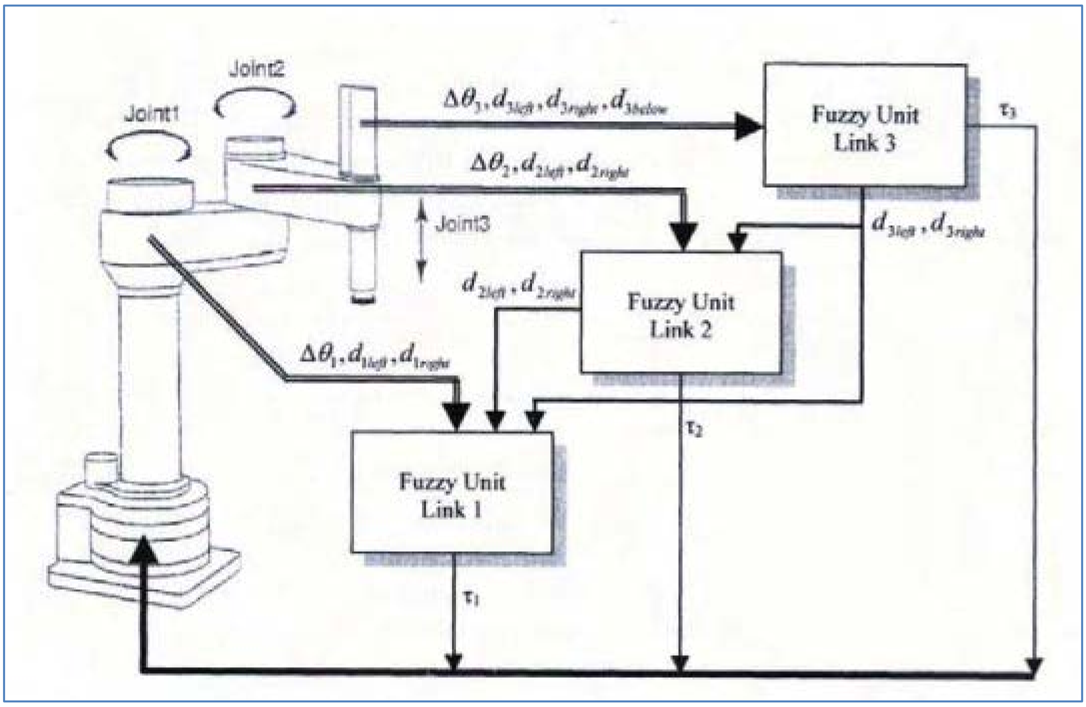


Figure 2-5 Test bench for Adept1 manipulator connected to fuzzy units. [10]

The fuzzy unit receives three inputs. First one is the difference between target and actual configuration. Second one the distance between the corresponding link and the nearest obstacle on the left and on the right of this link. First two inputs are ultrasonic sensor inputs. Third input can be a camera or a set of cameras which oversee the entire workspace. All the three inputs form an appropriate motor command from the given inputs and can avoid obstacles while reaching its target. [9]

2.2.3. Advanced Robot Controllers

Industrial robot performance is specified in terms of functional operations and cycle time. For assembly robots it’s the number of pick and place cycles per minute, welding robots are specified with weld pattern, weave speed as well as the fast repositioning speed. For painting robots the deposition/coverage rate and the spray pattern speed are important. The performance of a robot is determined by several factors (Table 2-3):

Speed	Accuracy
Acceleration	Component Life
Repeatability	Duty Cycle
Resolution	Collision Detection

Table 2-3 Robot performance factors

Major reasons for failure in electrically powered robots are actuators, transmissions and power/signal cable. Mean Time Beτtween Failures (MTBF) should be a minimum of 2000h online and ideally at least 5000 operating hours should pass between major components preventive maintenance replacements schedules [10].

Collision detection in industrial robotics has been developed from overload (slip) clutches, elastic members and padded surfaces to vision system and proximity sensors. In modern automotive plants where in a single station at least 4-6 robots work in synchronization is very common where sometimes the arms of robots works within centimetre proximity of each other. Advanced motion planning and collision prevention algorithm has made this possible.

Robot manufacturers have been constantly working to develop their robot's controller, to allow them for coordinated movement of several robots and simultaneous working on the same work piece. Robotics is widely used in the automotive industry for automated assembly, welding, painting fabrication, and production lines. Consequently, automotive manufacturers are constantly searching for ways to increase manufacturing productivity for faster return on assets.

Some exemplary milestones in the development:

1994: The Motoman MRC control system was introduced with the ability to control up to 21 axes. It could also synchronize the motions of two robots.

1998: The introduction of the XRC controller allowed the control of up to 27 axes and the synchronized control of three to four robots.

2004: Motoman, Japan, introduced the improved robot control system (NX100) which provided the synchronized control of four robots, up to 38 axes.

2012: In the production vicinity of Toyota plant in Japan, there are some assembly stations (performing spot welding job) the advance robot controller has made it possible for 20-32 robots working synchronously in a single station in a very compact environment [11].

2014: Motoman, Japan, introduced patented multiple robot control technology – DX100. This controller can easily handle multiple robots upto 8, has built-in collision detection, 25% more energy efficient and quicker I/O response.

2.3. History of Factory Simulation Tools

Today Simulation is the most multifaceted topics that an industrial, mechanical or automation engineer faces in the workplace. Today simulation software are extensively used to increase production capacity, increasing competitiveness and profits of the company.

First tools for factory simulation can be traced back in the 70's. These tools were developed to study discrete event simulation. These tools took numbers from timing studies from the production line, analyzed them and generated data which aided the industrial engineer to run experiments for manufacturing process improvement. Later 2D charts and graphs were included to the simulation tools. [12] [13]

Simulation tools with 3D models came to the markets on 80’s. They were mainly based on robotics application and had physics functionality within. These software were able to simulate the motion of the robots and check for collisions. This was the revolutionary step of factory simulation from a 2D world to the 3D world. [13]

In 90’s the first intelligent simulation software came to the market which were able to represent 3D graphics capabilities and discrete event simulation capabilities together and this revolutionized the manufacturing industry. Later physics was included with the development of advance 3D engine platforms. Today the simulation of manufacturing processes like welding, machining, gluing, sorting, palletizing, assembly, packing, handling and many more are planned, designed and tested in the 3D world with the factory simulation software almost as precisely as the real world. The machines and robots programmed in the virtual world and then those programmes transferred and executed in the real world with least effort.

Two common fears of simulation in early 80’s were:

1. Simulation is extremely complicated, so only experts can use it.
2. Simulation takes forever because of programming and debugging.

Today, CAD and simulation software are advanced to such a state that the software enables the user to model, execute and animate any manufacturing system in any level of detail. A complex 2000-foot conveyor, articulated robot, automotive BIW, machine tools, process station with several machines and robots, factory floor with discrete production system can me modeled in minutes to hours. [14]

One of the major challenges of Automatic Path Planning has been researched in various approaches. One of them is a reconstructed surface model. In [15] a virtual object was proposed which represents the real object to be welded, and is constructed from geometric polygonal mesh model or point cloud. From the surface model tangent vector and tangent plane of the sampling points are calculated. From this calculated tangent vector and normal plane the position and pose of torch of each sampling point is determined and the final path planning of arc welding robot in joint space is obtained by inverse kinematics.

2.4. Offline Programming (OLP)

OLP (Offline Programming) is the way to teach a robot its tasks without any connection to the robot or stopping its current task. It allows to develop new design, faster and task creation, creating complicated tasks in a very easy way, improve cycle time without taking the real robot out of production [16].

The simulation software provides a virtual environment of the real robot workcell. In this virtual world the user can import external CAD files, model parts and fixtures. Then the user can create the tasks for welding, palletizing, material handling or painting with the robot. The reachability and accessibility of the end effector of the robot can be verified in different positions and the best one can be assigned. This same job with the real robot would be enormous time consuming task. Then the simulation is run and checked for reachability, collision detection or any other errors which can be then fixed. Motion path optimization and cycle time is analysed from the simulation. Finally using a translator the robot program is generated and downloaded directly to the robot controller [16]. The following sections some reference to the research works done on advanced motion planning of industrial robots.

Current State Model

Nowadays robots are used in vast range of industrial application, incl – pick and place, palletizing, welding, assembly, handling, packing, gluing, painting etc. Robots are connected by various peripheral devices depending on application and installed in a production environment which is called a robot workcell.

In industrial robotics, the configuration and reconfiguration of a robot worcell is a major fact for making production more efficient [17].

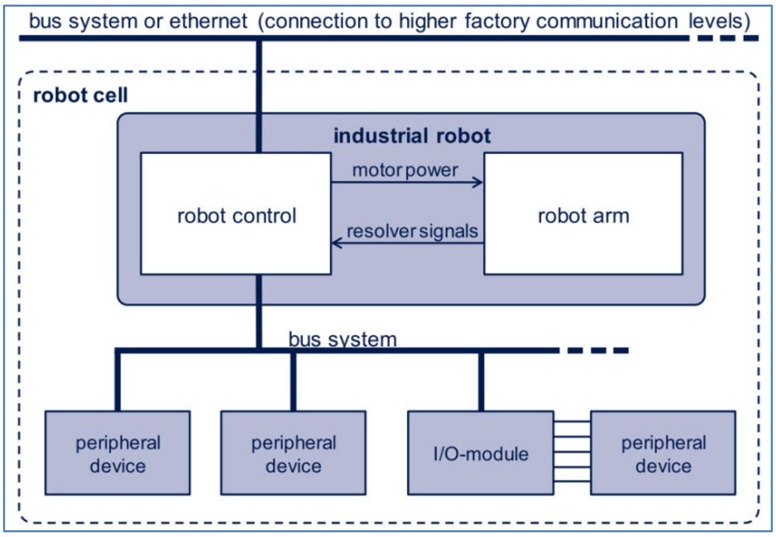


Figure 2-6 Typical control setup of a robot cell .[11][19]

According to [10] the stages to configure or integrate a robot system consist of five following stages (Figure 2-7).

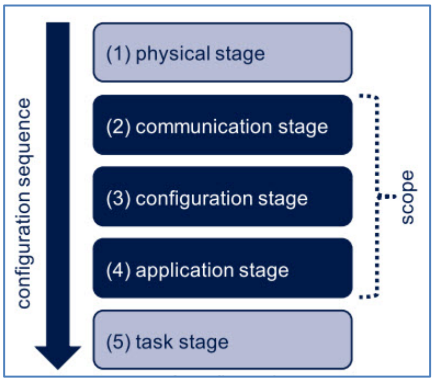


Figure 2-7 stages of configuration

In the task stage the robot program is created. Different robot manufacturer have their proprietary programming languages. In industry teach-in method for online programming and CAD based method for offline programming are popular.

Heterogeneous peripheral devices, different programming environments and the need for expert knowledge is still a big challenge for automatic reconfiguration of robots involved in industrial process.

In [18] two use cases i.e. (1) Handling/Assembly & (2) Welding were studied with configuration and reconfiguration. From there an information flow was generated and then from the created data pool showed high diversity in the information content, data complexity, size, change frequency, real time requirements, safety relevance and physical communication path.

From the results of the analysis of the data received from the two use cases improvements for a better automatic reconfigurable process of robot system were proposed. These included –

a) All necessary information for configuration needs to be provided in machine readable form.

b) The system must be able to automatically establish connection to the attached devices.

c) System integrator must consider the automatic configuration aspects.

A new approach for easy automatic reconfiguration of robot systems and OLP environments was presented by using a state-model.

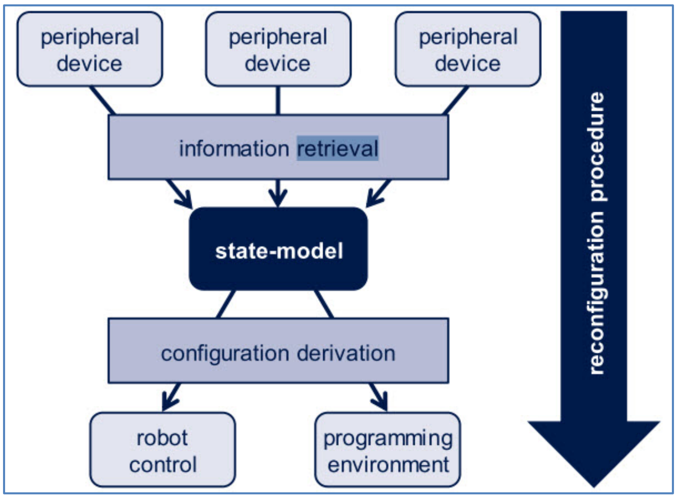


Figure 2-8 State-model based reconfiguration approach.[19]

This state-model contains all the necessary information of the current setup of the robot workcell. In this proposed model the system integrator will be able to derive robot configuration files (e.g controller) and programming environment (e.g geometry) files from the state-model and create OLP in a more efficient way. [18]

Robot Calibration

There always exists a deviation between the offline program and real world execution of the OLP on a real robot. The reasons for the deviation can be classified into geometric parameters and non-geometric parameter [19]. Geometric parameters are backlash, deflection due to unloaded robot weight and load applied to each joint. Non-geometric errors include transmission system errors between gears and thermal strain.

A typical welding line with 30 robots and 40 welding spots per robot takes about 300 hours of robot teaching. The most difficult part is to determine accurately a desired location in the workspace. Robot pose errors are attributed to several sources a) constant errors in parameters (link length and joint offset) b) position errors (compliance, gear transmission) and c) random errors (due to the finite resolution of joint encoders). Robot calibration can be done by model or modeless methods. Researchers have used specific kinematic models that depend on a particular robot geometry and calibration method. Model identifiability has already been addressed [20] [21], and experimental and simulation results using a rational technique was shown by Motta and McMaster [22] and Motta, Carvalho & McMaster [23] to find an optimal model for a specific joint configuration, requiring a few number of measurement points (for a 6 DOF robot only 15 measurement points) for a model with only geometric parameters (30), in opposition to hundreds of measurement points claimed by other authors Drouet et al. [24] and Park, Xu & Mills [25] . [26]

In 1993 Toyota Motor Corporation introduced an algorithm for correction of the robot mechanical errors which algorithm generated a formula for a robot with mechanical errors programmed to position $(X_0, Y_{0,} Z_0)^T$

$$\begin{bmatrix} \Phi_X + J_{JX} * (d_{iX}, d_{iY}, d_{iZ}, \delta_{iX}, \delta_{iY}, \delta_{iZ})^T \\ \Phi_Y + J_{JY} * (d_{iX}, d_{iY}, d_{iZ}, \delta_{iX}, \delta_{iY}, \delta_{iZ})^T \\ \Phi_Z + J_{JZ} * (d_{iX}, d_{iY}, d_{iZ}, \delta_{iX}, \delta_{iY}, \delta_{iZ})^T \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \quad \begin{matrix} i=0\sim6 \\ i=0\sim6 \\ i=0\sim6 \end{matrix}$$

This method suggested 15 robot orientations at a single point for a 6-axis robot, which is quite high. Later Dassault Systems introduced a 6 point calibration method in Delmia [27]. This calibration method can be used when the product has three mutually orthogonal planar surfaces, and does have at least three point-like features that can be easily identified and measured.

Vision based robot calibration system is popular nowadays because of easy processing, higher precision and low cost. Vision based robot calibration are of two types: 1) Setting camera in the robot surrounding & 2) Hand mounted camera robot calibration. The aim of robot calibration by camera is to develop a mathematical model of the transformation between world points and observe image points resulting from the image formation process. The parameters which affect this mapping are categorized into three types – a) extrinsic parameters b) intrinsic parameters and c) distortion parameters. The algorithm developed to obtain the above mentioned camera parameters is a two step method based n the Radical Alignment Constraint (RAC) model which is recognized as a good compromize between accuracy and simplicity. [26]

2.4.1. Steps of OLP

The first step to OLP is to model the whole robot workcell in 3D. This is essential to test the a) robot movement b) reachability c) collision detection and d) process related information for generating an error free offline program in the robot. With OLP the robot programs can be developed earlier in the design/production cycle without hampering the current production process. Moreover generation of OLP is a lot easier and efficient compared to jog and teach method. As simulation is incorporated with almost all OLP methods so a) robot movement b) robot collision c) productivity and d) safety are pre-checked minimizing the chance of error. [28]

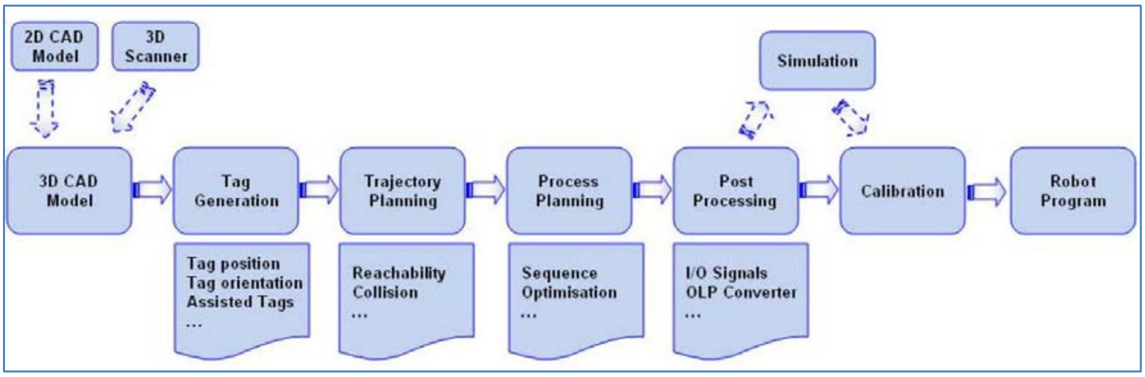


Figure 2-9 Key steps of OLP.[29]

Generation of 3D CAD model

Usually 3D models are generated by using different CAD software. In case of changing dimensions there are several methods to generate the required 3D model. One of them is using a 3D scanner to capture the workpiece geometry [29]. The 3D scanner generates a point cloud and it is converted to the surface model of the workpiece. When only 2D CAD is available the 3D model of the workpiece can be obtained in three ways a) multiple view of 2D drawing b) additional sensors or c) simply programming the robot in 2D [30] [31]. There are different types of CAD files and current OLP software can import almost all well known CAD files to their own compatible format. There are software companies whose business is to develop common platform (CAD kernel) related to read/write different CAD file types e.g. Coretech (Germany), Hoops (USA) etc.

Tag Creation

OLP software have built-in functions to generate tags with a specific tool centre point (TCP) from features such as corners and edges, frames or small geometry features. In this thesis work the developed AddOn [SpotWelding] tag points have been created from a *.csv file and geometry features in the CAD model. There are also assistant tag points e.g. home points, approach points and retreat points. The position and orientation information of the TCP to reach each tag point is automatically created by the OLP software. Research is going on to automatically extract robot motion information from the CAD data [32]

Trajectory planning

Inverse kinematics of industrial articulated robots usually has multiple solutions in the cartesian space. Trajectory planning includes a) motion type- (linear/joint) b) joint configuration c) joint speed d) joint acceleration/deceleration e) reachability and f) collision detection. In this thesis work the developed AddOn [SpotWelding] the offline programmer can assign retract distance to each spot welding tag point, also assign approach and retract via points for the start and end of a spot weld sequence. Also the orientation of a group of spot welding tags can be rotated along X axis, Y axis or Z axis to any required degree for the best configuration.

Process planning

The requirements of a specific process needs to be studied carefully first. It is very essential in the sense that the success of automation of the process depends widely on careful study of the requirements of the process development. Process planning includes a) optimization of resource b) robot selection for the process c) calculate the number of robots needed d) productivity e) robot workcell design f) material handling process design g) calculation of cycle time etc. For processes which involves large number of spot welds in limited cycle time can be treated as the “travelling salesman problem”(TSP) a solution which is based on a genetic algorithm. [33] [34]

Simulation

In this step the simulation of the process is run in the simulation software. The robot runs and does its works as programmed by the offline programmer. This gives a very clear visualization of the process, what is going to be implemented in the real world. From the running simulation the offline programmer can see and calculate the flaws in the process, and correct them as necessary. He can calculate the process deadlocks, cycle time, productivity, resource utilization and get all these data in a graphical (graph/chart) format or numerical format (excel or *.csv). The greatest benefit of simulation is that all the process can be verified beforehand without the need of any physical robot or other machine’s and removes the risk of any wrong/inefficient implementation of robot’s or machines in the final stage.

Post Processing

The robot program in the simulation software is after satisfactory verification implemented to the real robot in the physical world. A convertion in the middle of OLP software and the real controller is utilized to convert the robot program in the OLP software to the program language of the specific robot target. Post processing of robot programs is a special issue for all the generic OLP software as the robot program needs to be made compatible for different robot manufacturers. It is still an issue that straightforward robot programs with just I/O signals and linear or point-to-point movement in the OLP software can be converted for the real robot specific language without much problem but it’s tough to implement the intelligent programs (conditional loops, multitasking etc.) in the software to the robot specific program. For that the generic OLP software needs to have the virtual controller from the specific robot manufacturer. E.g. Visual components have the access to virtual controller of Kuka, Staubli and Motoman.

Calibration

Unfortunately in the real world the dimensions, position and orientation of the CAD models are not the same as in the virtual world or simulation world. So the workcell needs to be calibrated to make the robot program perform its task without collision or error (positioning error or mishandling etc.). Calibration is divided into two sections i.e. – Robot Calibration and Workcell Calibration; a) **Robot Calibration** is the process of determining the actual values of kinematic (position, orientation of links and joints) and dynamic parameters (joint mass, joint friction etc.) of an industrial robot (IR). An industrial robot needs to be calibrated after certain hours of operation. A calibrated robot has a higher absolute positioning accuracy than an un-calibrated one. b) **Workcell Calibration** means the calibration of its tools and the workpieces/fixtures in the workcell. It minimizes occurring inaccuracies and improves process security.

2.5. Existing Robotics OLP software

Today’s robot simulation software provides the robot programmer with the functions of creating virtual robot points and programming virtual robot motions in an interactive and virtual 3D design environment [35]. There are many robot simulation software packages, among them the most advance one’s are – Delmia (Dassault Systems), Robcad (Technomatix), Robotmaster (Robmaster), Ciros Studio, 3D Automate (Visual Components). These simulation software provide the users comprehensive and generic simulation functions, industrial robot models, CAD data translator and robot program translators [35] [36] [37].

	Software/Ref	Company/Feature
Generic Robotics Software	Delmia	Dassault Systems;VR
	3D Automate	Visual Components Oy
	RobCAD	Technomatix; VR
	Robotmaster	Robotmaster
	Robsim	Camelot
	Workspace 5	Wat Solutions
	Cosimir	Festo
Robotics Software from Robot Manufacturers	RobotStudio	ABB
	MotoSim	Motoman
	KUKA-Sim, CAMrob	KUKA
	Robotguide	Fanuc
	Wincaps III	Denso
	3D STUDIO	Staubli
	MELFA WORKS	Mitsubishi
	Pc-ROSET	Kawasaki
	AX on Desk	Nachi
Academic/Open Source robotics software	[38]	Various MATLAB based software
	[39]	Aristotle's University of Thessaliniki, Greece, Based on SolidWorks
	[40]	Orebro University; Sweden ; Based on standard CAD
	[39] [41]	Autolisp, Based on Auto CAD
	PIN	European Centre for Mechatronics, OpenGL based macro programming
	ROBOMO	OpenGL
	[41]	Daegu University, Korea; VRML, Tribon
	RoBott	University of Minto Portugal

Table 2-4 OLP Software in Industry and Education

OLP software from robot manufacturers

It can be seen from table 2, that almost every robot manufacturer has its own OLP software. Since the OLP software is more compatible to the robot hardware, secondary development of the OLP system is relatively easier. The cost of this type of OLP package is generally lower than one using genetic OLP software as the hardware and software are packaged together. This explains why ABB RobotStudio is by far the most widely used OLP software.

Generic OLP software

This category includes two most powerful OLP software, Delmia (formally IGRIP, ENVISION with third party add-ons from Kineo, CENIT) from Dassault Systems and RobCAD (Em-Workplace) from Technomatix. The advantage of generic packages is that they are more flexible for hardware from different manufacturers and include normally link to product lifecycle management (PLM) packages to provide production line optimisation. Major automobile and airplane manufacturers use these packages to integrate the robotic systems into their general automated production lines. Also, both software packages have the feature of Virtual Reality which allows the user to be fully immerged into the simulation environment. Today, OLP systems are able to do more than just simulate robot trajectories and perform assembly simulation. Simulation technologies are also able to model the interaction of several manufacturing processes, manufacturing resources, and product maintenance issues.

Open Source or Academic OLP software

Due to the high cost and limited accessibility of commercial OLP software, a number of researchers have developed alternative OLP software. While some researchers [27-29] [16] have developed OLP packages based on the existing CAD software, such as AutoCAD and Solidworks, others [20] [30-31] have started from scratch using OpenGL, VRML and Java technology.

3. ROBOTIC SPOT WELDING

The high demand of spot welding has forced the automobile manufacturers to keep looking for new technologies to make faster and more accurate spot welding. There have been constant upgrades on the spot welding robot's and spot welding guns. Advancement in both AC and DC weld controllers, electric servo powered gun, modular design of spot weld guns, using lightweight aluminium metals in the weld guns are the forefront of the new spot welding technology. Another development is to use multiple robots with a single controller which enabled for multiple robots to work in a closely confined space without the risk of collision. Also especially for a dense robot population, in the robots there were modification of different castings and designing shorter arms but utilizing the same motor and sensors of the parent bigger version robot. Specialized robots have been designed for spot welding application and they have utilities (air, water and power) routed in cable harnesses through the arm and out to the robot wrist, this greatly reduced downtime associated with external cables. Other improvement in automated spot welding was use of a pedestal or a stationary weld gun. The robot moves with the part to the stationary pedestal which holds the weld gun. A spot weld gun can weigh a couple of hundred kg while a panel can be 20-30 kg. In this approach the robot dressing is minimized, part changeover is easy and capital equipment like robot is highly utilized all the time. To improve the weld cell uptime EOA systems an "Intelliflow Water Saver" was applied which monitors the temperature and coolant flow rate to the entire cell including weldgun, transformer, shunts, cable etc. [42]

3.1. Spot Welding Process

Resistance spot welding is an efficient process to join vehicle body parts, by the means of strong interaction between electrical, thermal, metallurgical and mechanical phenomena [43]. Spot welds are normally used to join vehicle parts upto 3 mm in thickness. There is a thickness ratio which should be also maintained, usually which should not be less than 3:1. Spot weld diameter ranges from 3mm upto 12.5 mm [44].

Principle

Spot welding is one form of resistance welding where two metal sheets are joined together without using any filler material by applying pressure and heat to the area to be welded. The process uses copper alloy electrodes for applying pressure and electricity, the material between the electrodes yields, squeezed together, it forms 'nugget' of molten materials which when solidifies forms the welded joint.

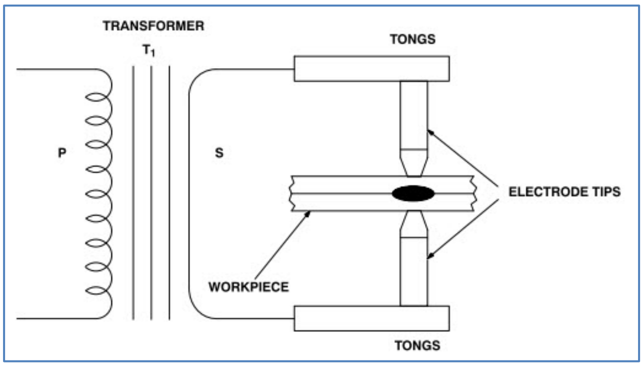


Figure 3-1 Principal of resistance spot welding.

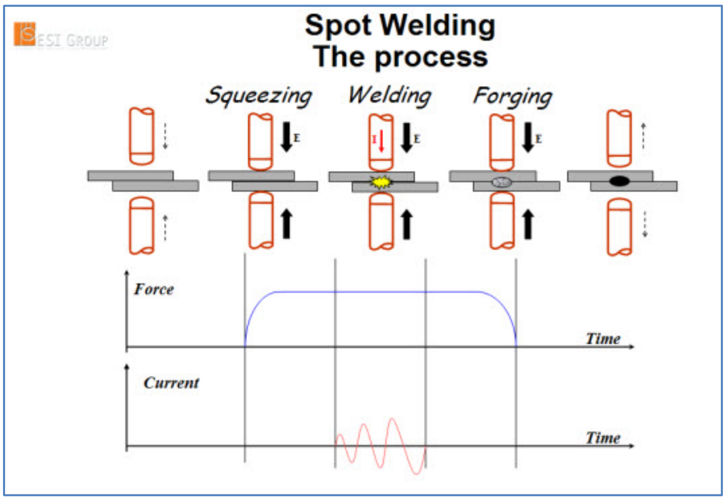


Figure 3-2 Force and Current analysis of spot welding process

Heat Generation and Time Cycle

The heat generated while spot welding process is directly proportional to the resistance at any point in the circuit. Formula for calculating generated heat as following [45]:

$$H = I^2RTK$$

H = Heat

I^2= Current Squad

R = Resistance

T = Time

K = Heat Loss

Control of time is an important factor as too long or too short time will lead to gas porosity, expulsion of molten metal or smaller weld nugget. Sometimes time is the only parameter that can be controlled in a spot welding operation when the current type is a single impulse [45]. The time cycle of resistance spot welding is shown below -

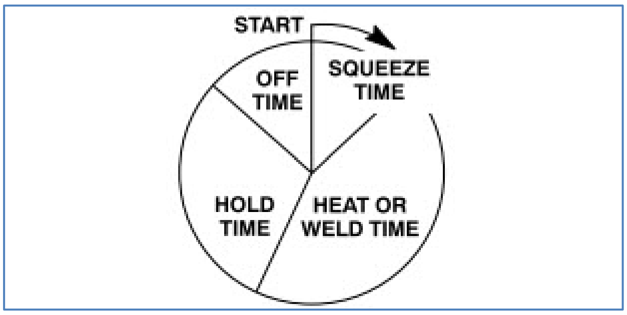


Figure 3-3 Heat genearation while spot welding

Electrode Tip Size

Electrode tip size is an important factor for spot welding. It depends on the material type of workpiece. A general formula has been developed for low carbon steel [45].

$$Electrode\ tip\ diameter = 0.011 + 2t$$

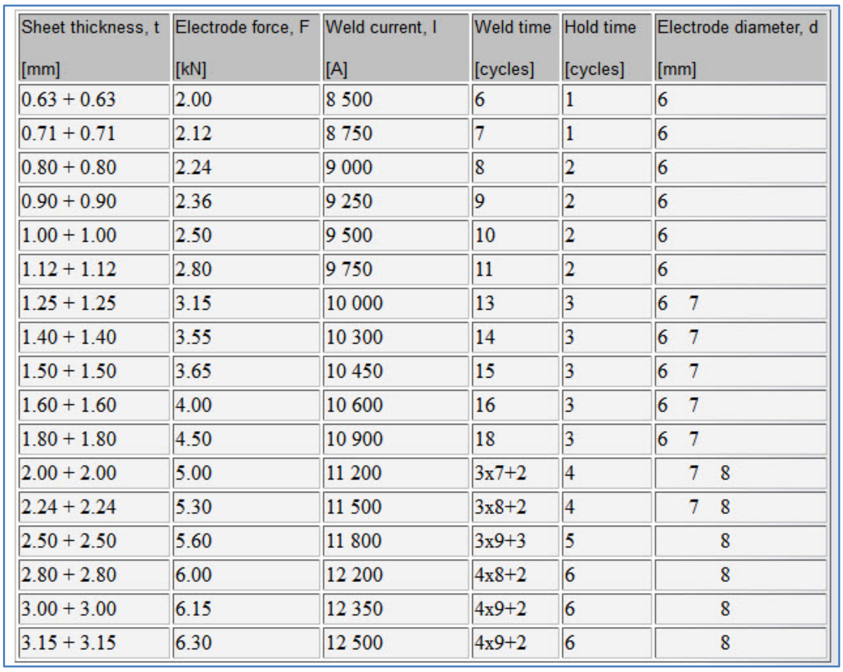
, where t is the thickness in inch, and the formula is applied to each thickness individually. But this formula has been originally developed for low carbon steel; it needs changes for other materials.

Welding Force and Heat Balance

Appropriate amount of applied pressure is also necessary for spot welding. The greater the pressure, the greater the welding current passes through the joint. Setting the right pressure for a spot weld job is trial-error method which is usually practiced in the industry, but it can latest be determined by using FEM software and numerical analysis software before hand, when both the material to be welded is of equal material then the heat balance is not a problem. Heat balance is the amounts of heat generated while welding in which the fusion zones of the pieces to be joined together are subjected to equal heat and pressure. As an example welding of two parts of different material, such as steel and copper, steel has higher electrical resistance and low thermal transfer characteristic than copper and therefore there will be greater amount of localized heating in the steel than in copper. This results to poor weld quality. In this case the electrode tip area is customized which results in adjusted current density to equalize the heat in the weldment.

Spot Welding Parameter

Parameter setting is very important in spot welding process, which depends on the material of the sheets to be welded, sheet thickness, thickness ratio, electrode dia etc. As spot welding application in automotive industry is incredibly huge, the American Welding Society (AWS) has established a standard which is published in the book – “Specification for Automotive Resistance Spot Welding Electrodes” [46]. The table below gives a target value for welding parameters although one of the important parameters (metal type/material) is not mentioned.



Sheet thickness, t	Electrode force, F	Weld current, I	Weld time	Hold time	Electrode diameter, d
[mm]	[kN]	[A]	[cycles]	[cycles]	[mm]
0.63 + 0.63	2.00	8 500	6	1	6
0.71 + 0.71	2.12	8 750	7	1	6
0.80 + 0.80	2.24	9 000	8	2	6
0.90 + 0.90	2.36	9 250	9	2	6
1.00 + 1.00	2.50	9 500	10	2	6
1.12 + 1.12	2.80	9 750	11	2	6
1.25 + 1.25	3.15	10 000	13	3	6 7
1.40 + 1.40	3.55	10 300	14	3	6 7
1.50 + 1.50	3.65	10 450	15	3	6 7
1.60 + 1.60	4.00	10 600	16	3	6 7
1.80 + 1.80	4.50	10 900	18	3	6 7
2.00 + 2.00	5.00	11 200	3x7+2	4	7 8
2.24 + 2.24	5.30	11 500	3x8+2	4	7 8
2.50 + 2.50	5.60	11 800	3x9+3	5	8
2.80 + 2.80	6.00	12 200	4x8+2	6	8
3.00 + 3.00	6.15	12 350	4x9+2	6	8
3.15 + 3.15	6.30	12 500	4x9+2	6	8

Figure 3-4 Welding parameter [45]

Planning Robotic Spot Welding Lines

In order to plan a robotic assembly spot welding line which meets a the manufacturer's specific production needs the engineers have to design a detailed and optimized solution. Design data from the product to be manufactured can be brought together to give an estimate of the machinery and equipment (robot, conveyor, fixture etc.). These include the following [4]:

- The parts to be assembled
- The geometrical conformation of these parts and the corresponding number of stations required.
- The distribution of spot welds and the number of robots required to weld them.
- The production rate and the number of lines required to meet the production needs.
- The desired degree of flexibility.

To complete the design additional information is still needed:

- The basic principles relating to the transfer and positioning of the assembly.
- The final selection of the robot, its equipment and its installation.
- The environment and the available space.

3.2. Simulation Planning

Advanced versions of today's industrial robotic simulation software already support following features:

- Uniquely structured environment lets the user to quickly enter the geometry and production requirements of a model.
- Expert system technology generates details automatically while windows and pop-up menus guide the user through the modelling process.

- Parts can be modelled completely parametric with functions and formulas. Changes can be made quickly and easily with far less chances of errors.
- Built in material handling templates make the user more productive and faster programming.
- The user can verify and test designs, explore more alternatives and catch system glitches and 3-D animation before implementation.
- 3-D graphics are automatically created as the user enters data.
- Results can be communicated in real time animation. Robots can be operated/ monitored from the 3D simulation with the help of virtual controller. OLP software have virtual controllers according to their partnership deals with the robot manufacturers.

All the simulation software in the market for process-planning and offline programming have various tools and algorithms for making process simulation of robotic spot welding and arc welding. In this chapter the leading software Delmia V5 (from Dassault Systems) and Robcad (from Siemens) are discussed.

Delmia V5

Simulation planning consists of weld gun selection, station layout, line balance, robot reachability analysis and motion path optimization. For example the virtual simulation software Delmia can be mentioned. It includes Digital Process Engineering (DPE) to manage the process, product and resources and Digital Process Manufacturing (DPM) to simulate the assembly and welding process it has. Process planning of welding line design in Delmia is shown in following figure (Figure 3-5).

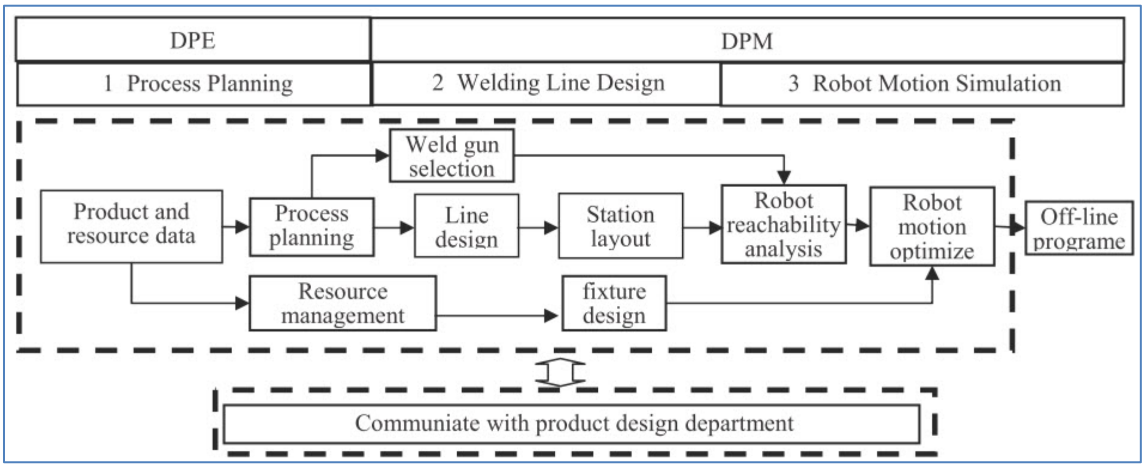


Figure 3-5 Process planning in Delmia

The most difficult and the very first task in spot welding is selection of the right spot weld gun. There are different types of spot weld guns e.g. X-type, C-type etc. The automatic weld gun selection function in Delmia looks for reachability, collision and interference of the spot weld points with different guns and helps the offline programmer to easily select the right gun for a spot welding task.

The next tasks are **Robot Selection**- which is mainly based on robot load & work range; and **Analysis of Robot Location**- to ensure that all the spot welds are reachable conveniently. Figure 3-6 clarifies the process planning sequence for robot location in Delmia.

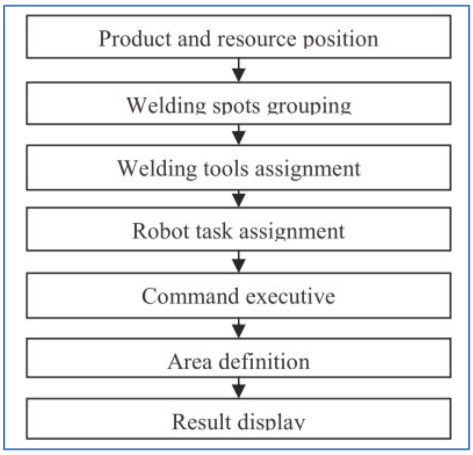


Figure 3-6 Process sequence in Delmia

Robot Motion Optimization: Depending on the work condition the spot points are assigned to the robot in four different motions – joint, linear, circle and slew. While selecting the motion type via points are assigned and joint configurations are modified to avoid crash and collision.

Offline Programming: After successfully completing the simulation, finalizing the robot station layout, workcell calibration needs to be done to make virtual simulation nearly the same as the real world. After that using the robot translator in Delmia the simulation parameters are compiled to generate the corresponding robot programs (KUKA, ABB, MOTOMAN, DENSO etc)which can be directly read by the robot controller [47].

Robcad

The next pioneer in virtual robotic simulation software in the market is Robcad by Siemens. Robcad is built on the RRS-(Realistic Robot Simulation) protocol. It is capable to work with different kinds of CAD data and robot controller. The OLP process in Robcad can be realized as presented in figure 3-7.

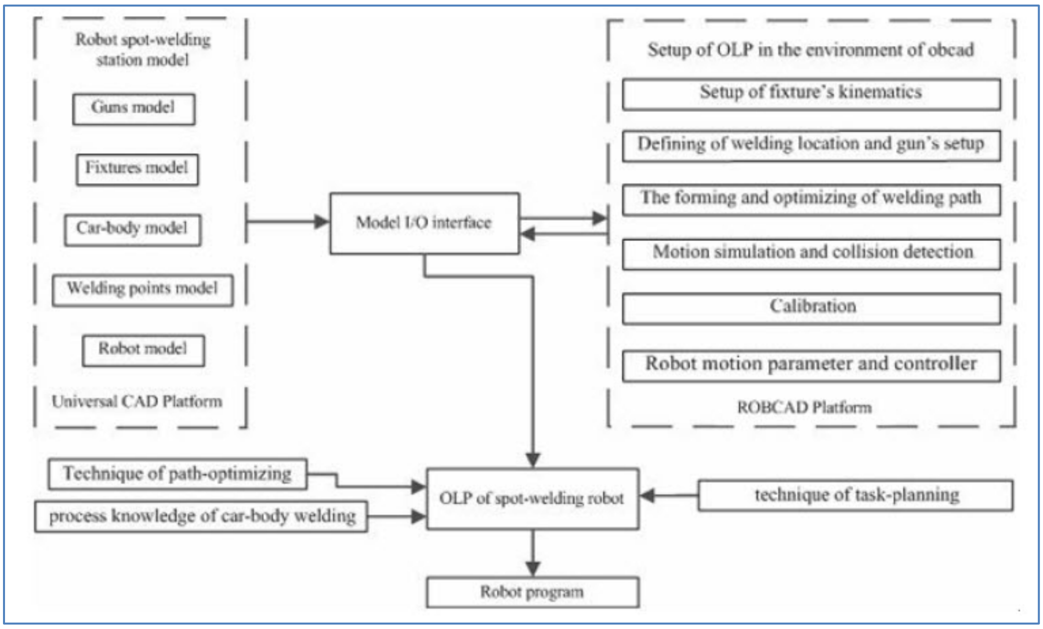

Robot spot-welding station model
Guns model
Fixtures model
Car-body model
Welding points model
Robot model
Universal CAD Platform
Model I/O interface
Setup of OLP in the environment of obcad
Setup of fixture's kinematics
Defining of welding location and gun's setup
The forming and optimizing of welding path
Motion simulation and collision detection
Calibration
Robot motion parameter and controller
ROBCAD Platform
Technique of path-optimizing
process knowledge of car-body welding
OLP of spot-welding robot
technique of task-planning
Robot program

Figure 3-7 OLP process planning in Robcad.[49]

The step by step process planning for OLP of spot welding application can be realized in the following way.

a) **Robot spot-welding station model:** CAD reader is utilized to import different format of CAD data (.dwg, .stp, .wrl, .cgr, .CATProduct, .iges etc.) to bring in the components of workpiece, fixture, end effectors, conveyors, Robots, tables & stations etc. to the virtual world. Thus 3D model of spot weld station is created in less time. Different format of components brought in Robcad can be converted to .CO format which is the default CAD format for Robcad. This way improves the processing speed of the software.

b) **Fixture Kinematic setup:** The kinematics, motion of the spot weld gun is first modelled by using the gear unit, bar linkage mechanism in Robcad.

c) **Defining Welding location and path optimization:** Points for spot welding are projected on a workpiece to create welding location which includes welding orientation. The Z axis of the TCP of spot weld gun usually coincides the normal surface of the workpiece and X axis is the approaching direction of welding. Path optimization means to minimize the welding time as much as possible without collision. The problem is realized as travelling salesman problem and genetic algorithm is used to solve this problem [41] [47].

d) **Collision Detection:** Then the spot welding simulation is run in Robcad and any collision occurring between geometries creates a warning message for the offline programmer. The points where the collisions are occuring the programmer have to modify the orientation of the spot weld points and also via points.

e) **Calibration:** To adjust the difference in between the real world and virtual world the workcell calibration is done and in Robcad it calculates the RMS (root mean square) value of spot welding locations. There are several algorithms to get the minimum RMS value and one of them is the Steep Descent & Hessian method [48].

3.3. Steps in OLP of Robotic Spot Welding

The focus of this thesis work is related to spot welding line in Automotive Industry. Spot welding is one of the most mature applications in robotics. The high demand for popular models of cars, to keep in pace with competitors in recent years big car makers have invested heavily in developing intensive automated spot welding techniques with short set-up time, modular & lightweight systems with increased cycle times and improved end of arm tooling(EOAT). Robotic welding plays the key roll enabling car companies to keep pace with demand for new, advance and higher quality product. Automakers are always on the search for robots with greater repeatability and weld requirements down to ±2 mm. [42]

In current practice the first step of OLP with spot welding is to model the car in companywide CAD system like AutoCAD, SolidWorks, Creo, CATIA etc. Then fixtures are designed which is going to present the parts to the welding robots. These fixtures are made by specialist and certified tooling companies. The points of fixture which are going to clamp the car part are measured very precisely using a 3D co-ordinate measuring device. These measurements are recorded using manual theodolites and standard theodolite triangulation. Recently also the photogrammetric method is used for measurement.

Usually spot weld guns are bought from the weld gun manufacturing companies from an array already designed products. But with special requirements often the car manufacturers ask for customized spot weld guns. The first fixture is then installed in the production line, using this position data the position of the robots in a station are defined. The position of the second fixture is defined based on the first fixture position and so on, all the way down the production line.

Next step is the whole workcell building in the simulation software by mostly imported CAD files. The robots are imported from the ready robot library or received from vendor. The co-ordinate frames representing the position and orientation of spot welds, via points, weld schedules, motion type can be defined in the CAD software e.g. CATIA and then exported to the simulation software in *.csv file format or all the information within the CAD model itself. A COM AddOn was developed as part of this thesis work to export the spot weld information to a *.csv file.

The robot program is generated utilizing the spot weld information. The wrist joint limits of the robot often limit the orientations that are achievable and the simulation user must be creative to define the joint configuration, robot base location and robot tool.

Then the simulation is run to check collisions and the designer plans a collision free path, but he also has to be careful that it doesn't add excessively to the overall cycle time. There is a difference between the cycle time in the simulation software and the real robot. The robot companies consider motion algorithms in their robot controllers of such commercial value that they always refuse to release this to their simulation vendors. But fortunately every simulation software company is able to establish strong business relationship with some robot vendors and they can integrate the "Black Box" or the unreadable part of the robot controller software. This makes it possible to get accurate cycle time from the simulation software as well reliable information of the collision and near miss information provided by the simulation.

Next step is the calibration of the workcell and robot as well. Robots are designed to be extremely repeatable but not all that accurate (repeatabilities of 0.1mm and accuracies of 10mm or worse are not unusual). Once the robot is calibrated then the mountplate is attached with a pointer to measure the position of the fixture datum point. This is called "3 point touch up" and is used for positioning the model of the fixture correctly relative to the robot model within the simulation world. This makes the offline programming truly meaningful. Finally the correct tool offset is determined in the robot in the simulation world. If the tool dimensions are unknown or changed due to collision or modification then a tool calibration is carried out.

After all the above procedures are carried out, the robot program in the simulation is really ready to be downloaded to the real world. Badly implementing or ignoring any of the above steps can result in bad consequence which can include damage of robot to the damage of workcell equipment as well as human injury. [49]

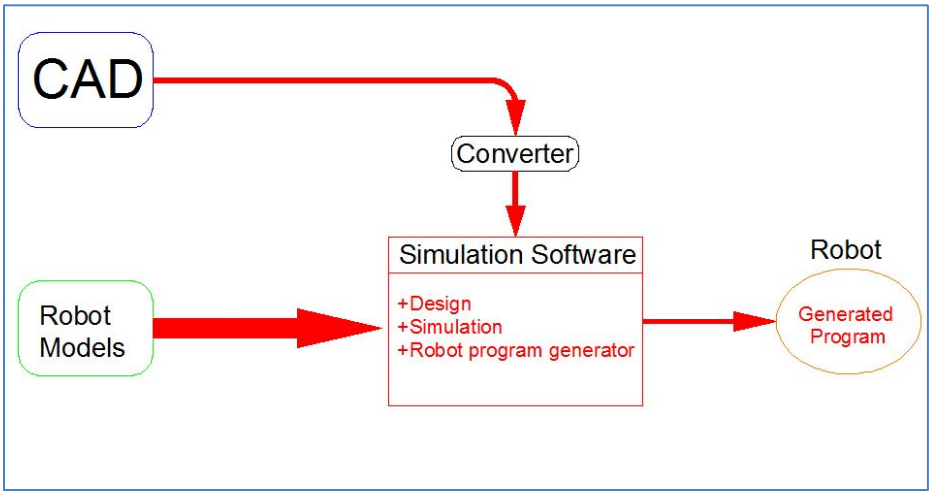


Figure 3-8 Steps in OLP of robotic spot welding

4. IMPLEMENTATION

4.1. API

An API (Application Programming Interface) is like a coding contract i.e. it specifies the ways a program can interact with an application. In this case the Visual Components higher end products 3D Create and 3D Automate have been integrated with two API's.
1) COM API and 2) Python API

COM API

COM (Component Object Model) was developed by Microsoft. COM is used by developers to create re-usable software components, link components together to build applications and enable software components to communicate. COM objects can be created by a variety of object oriented programming languages, such as C, C++, C#, Visual Basic etc. The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX® controls. Microsoft provides COM interfaces for many Windows application programming interfaces; also COM is used in developing applications such as Microsoft Office Family Products. COM Automation allows users to build scripts in their applications to perform repetitive tasks or control one application from another. [50]

COM provides a framework for integrating software objects, which are collection of related functions. COM supports interoperability and reusability of distributed objects by allowing program developers to boils systems by assembling reusable components from different vendors that communicates safely via COM. [50]

COM has the architecture presented in figure 4-1 –

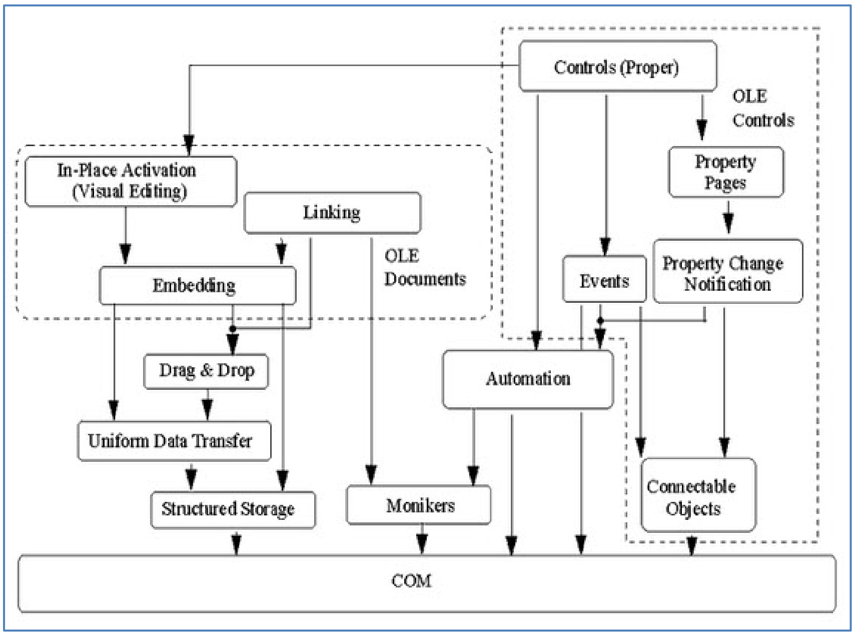


Figure 4-1 COM as a foundation for OLE technology.[53]

The figure 4-1 shows the architecture of COM which provides the foundation for the higher level software services like OLE (Object Linking and Embedding). OLE is a compound document standard developed by Microsoft Corporation. It enables to create objects with one application and then embed them to a second application. COM fundamentals are as following [51]–

- A binary standard for function calling between components
- A provision for strongly typed groupings of functions into interfaces
- A base interface providing a way for components to dynamically discover the interfaces implemented by other components.
- A base interface providing reference counting to allow components to track their own lifetime and delete themselves when appropriate.
- A mechanism to identify components and their interfaces uniquely, worldwide.
- A “computer loader” to set-up and manage component interactions.

COM Objects and Interfaces

Data associated with an *object* can be manipulated by COM through an *interface* on the object. COM *interface* means a binary-compliant interface that is associated with an *object*. The meaning of interface in COM is different that meaning of interface in Visual C++ programming. A C++ refers to all of the functions that a class supports and that clients of an object can call to interact with it. A COM interface refers to a predefined group of related functions that a COM class implements, but a specific interface does not necessarily represent all the functions that the class supports. An object in COM uses code that implements each method of the interface and provides pointers to those functions to the COM library.

A COM *object* will never have direct accesses to another COM object in its entirety, instead a COM object always access another COM object through interface pointers. This is the primary architectural feature of COM.

A COM *interface* is strongly typed contract between software components to provide a small but useful set of semantically related methods. An interface is the definition of an expected behaviour and expected responsibilities.

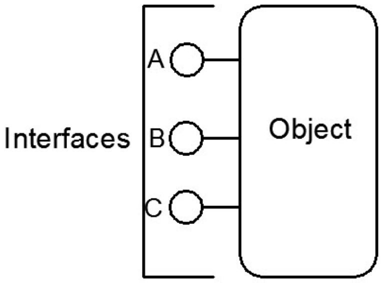


Figure 4-2 COM component that supports three interfaces A,B and C

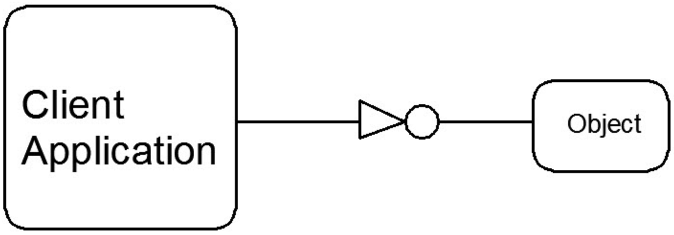


Figure 4-3 Interfaces extend towards the clients connected to them

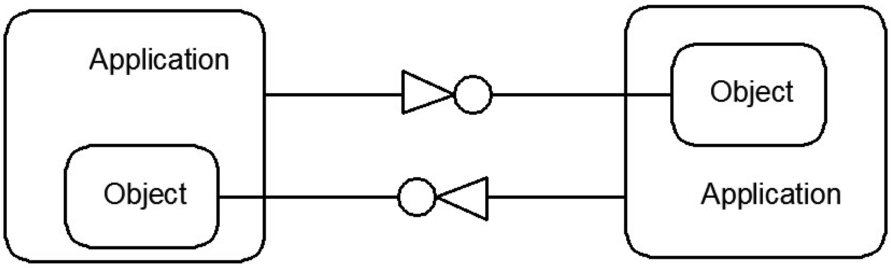


Figure 4-4 Two applications connect to each other by extending their interfaces.

Python API

Visual Components supports Python scripting language in the 3D simulation framework. In this section Python scripting language is introduced and the reasons for implementation of python as Visual Components internal scripting API tool are defined.

Guido van Rossum is mentioned as the father of Python, who first introduced this scripting language in the beginning of 1990's at Stichting Mathematical Centrum in the Netherlands. Rossum's main target was to develop a user friendly scripting language. Guido still remains the principle author of this programming language, although it includes many contributions from others to become one of the most popular scripting language of 21st century. In 2001 the Python Software Foundation (PSF) was formed, a non-profit organization to own Python related intellectual property. All Python releases are open source, although Python implementation is copyright but freely usable and distributable, even for commercial use. The latest and current production version of Python is Python 2.7.5 which was released on May 15th, 2013. The current development/testing version of Python is Python 3.4.0. [52]

Python with its clear and simple rules is so straightforward that it has been called "Programming at the speed of thought". Python is an object oriented programming (OOP) language, Languages like C#, Java and Python are all object oriented, but Python has big advantage compared to all of them. In C# and Java, OOP is not optional. This makes short programs unnecessarily complex, and it requires a bunch of explanation before a new programmer can do anything significant. In this case Python has an advantage, OOP here is optional. With python users can make short effective working programs without OOP approach or a large project with several programmers that utilize the OOP approach. Python is a glue language that can be integrated with other languages like C, C++ and Java. This means the programmer can take advantage of work already done in another language while using python. Also a great advantage of Python is its platform independence, which means regardless of operating system the user can create python program and send it to another user who uses Linux or Mac operating system, and the program will work. [53]

Features	Benefits
No Compile or link steps	Rapid development cycle turnaround
No type declaration	Simple, shorter and more flexible programs
Automatic memory management	Garbage collection avoids bookkeeping code
High level data types and operations	Fast development using built-in object types
Object oriented programming	Code reuse, C++, Java and COM integration
Embedding and extending in C	Optimization, customization, system "glue"
Classes modules, exceptions	Simplified extensions, smaller binary files
A simple clear syntax and design	Fewer restrictions and special use case
Dynamic loading of C modules	Handles unforeseen needs, end-user coding
Dynamic reloading of Python modules	Incremental development and testing
Universal "first-class" object model	Metaprogramming, introspective objects
Runtime program construction	Cross-platform programming without ports
Interactive, Dynamic nature	Tkinter scripts run on X, windows and Macs
Access to interpreter portability	Easy access to email, FTP, HTTP, CGI etc.
Compilation to bytecode	Platform-neutral of system scripting
Standard portable GUI framework	Vast collection of precode software components
Standard Internet protocol	Can be freely embedded and shipped
Standard portable systems calls	
Built-in third party libraries	
True open source software	

Table 4-1 Python language features.[56]

4.2. RSL and Path Planning

In the Simulation of spot welding in this thesis work an Add-on was developed by using python API to create the RSL (Robotic Sequence Language) statements in the Robot Controllers. RSL is a powerful tool to program the robot movement in a simple way. The RSL language consists of three levels: program, sequence and statements. It is limited in the sense that it does not have any conditional loops (if-else, for etc.). For the same reason it is very easy to program. There are multiple RSL statement types; they control different areas of RSL program execution (driving motions, waiting inputs, setting outputs, delaying etc.)

Motion planning is the task that performs the calculation for the movement and control of robot manipulators. Motion planning involves the following tasks –

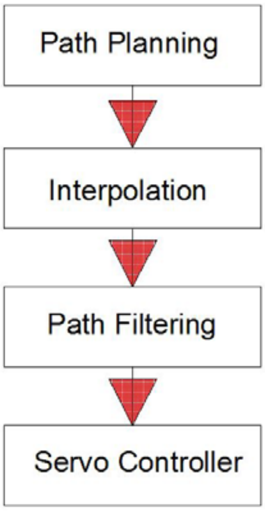


Figure 4-5 Basic tasks involved in motion planning.[57]

The path planner's basic task is to prepare the robot's path and feed the relevant data to the path interpolator. Moving a robot includes specifying an original/initial (position+orientation_T$_i$) and a final (position+orientation_T$_f$) of the TCP (Tool Centre Point) of the robot. The path interpolator computes the intermediate points using the specified velocity and acceleration. The output of the interpolator is the input to the servo controller. Prior to passing the data to the servo controller it is filtered through the path filter in order to provide smoother acceleration/deceleration and keep the motor torques in the range of the servo-motor. The final motion plan or path plan includes velocity, acceleration, deceleration and via points. [54]

The path planner must be a continuous first derivative and more preferably a second derivative [55]. The path generator can be implemented by a 5th order polynomial. The use of high order polynomial is necessary to specify the position, velocity and acceleration/deceleration at the beginning and end of each path segment.

A 5th order polynomial can be considered in the form –

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

With the following constraints –

$$\theta_0 = a_0$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5$$
$$\dot{\theta}_0 = a_1$$
$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4$$
$$\ddot{\theta}_0 = 2a_2$$
$$\ddot{\theta}_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3$$

Results in a linear system of six equations with six unknown whose solutions are:

$$a_1 = \theta_0$$

$$a_1 = \dot{\theta}_0$$

$$a_2 = \frac{\ddot{\theta}_0}{2}$$

$$a_3 = \frac{20\theta_f - 20\theta_0 - \left(8\dot{\theta}_f + 12\dot{\theta}_0\right)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^3}$$

$$a_4 = \frac{30\theta_0 - 30\theta_f + \left(14\dot{\theta}_f + 16\dot{\theta}_0\right)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^4}$$

$$a_5 = \frac{12\theta_f - 12\theta_0 - \left(6\dot{\theta}_f + 6\dot{\theta}_0\right)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^5}$$

In different robotics books there are several methods for motion planning. The function presented above gives a good indication and can be used for the objective of motion planning or trajectory planning between the target points. [54]

4.3. Spot Weld Gun & Fixture Modeling

The algorithm for selecting a spot welding gun is presented in figure 4-6.

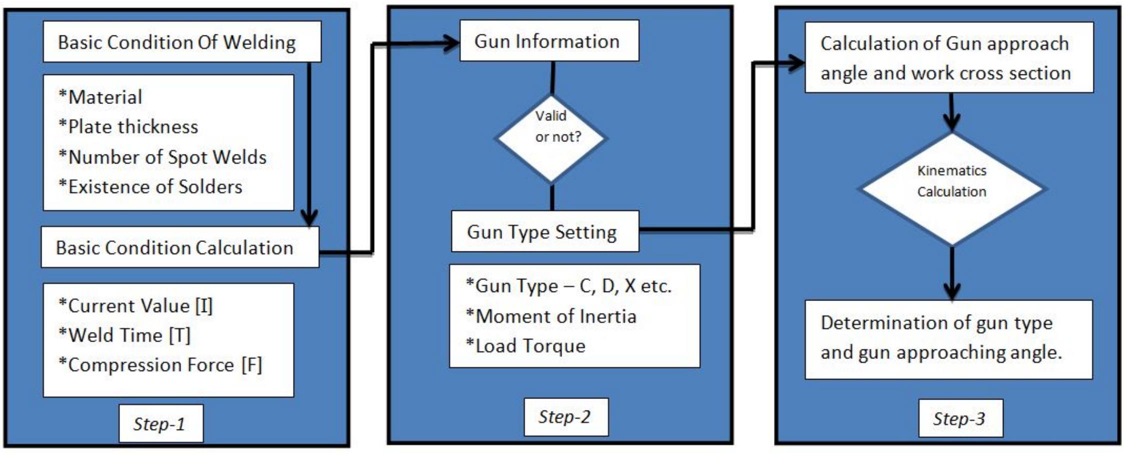


Figure 4-6 Algorithm for selecting the spot welding gun

In spot welding industry the job starts with designing the right gun and right jig for the parts which are going to be spot welded. The jig designer designs the jigs and the tool designer designs the robotic spot welding tools.

Spot welding guns are designed based on application. The guns can be categorized as following:

a) C-type (operating cylinder is connected directly to the moving electrode)
b) X-type (most common type)
c) D-type (mostly used for vertical welds)

Also every gun can be categorized in two sections -

a) Fixed jaw
b) Moving jaw

There is a fixed jaw in most of the spot weld guns. While offline programming at the tip of the fixed jaw, a frame is defined and this is the tool TCP for that spot weld gun.

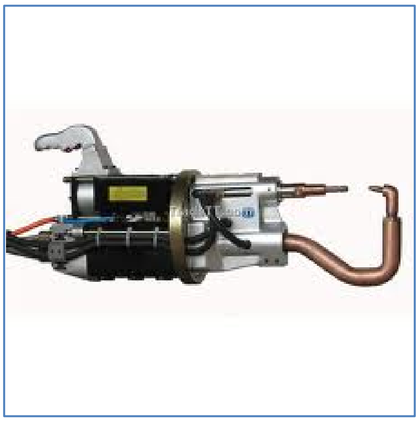


Figure 4-7 C type spot weld gun

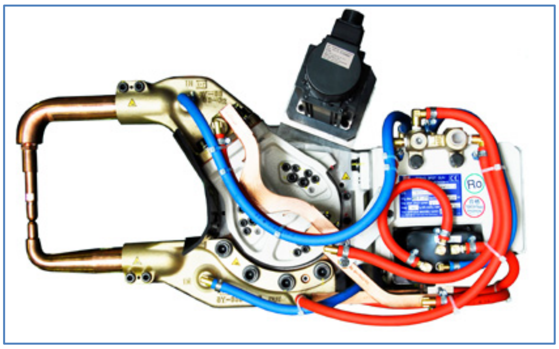


Figure 4-8 X type spot weld gun

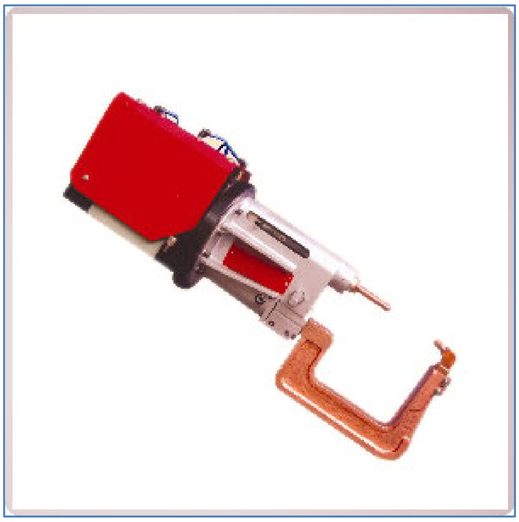


Figure 4-9 D type spot weld gun

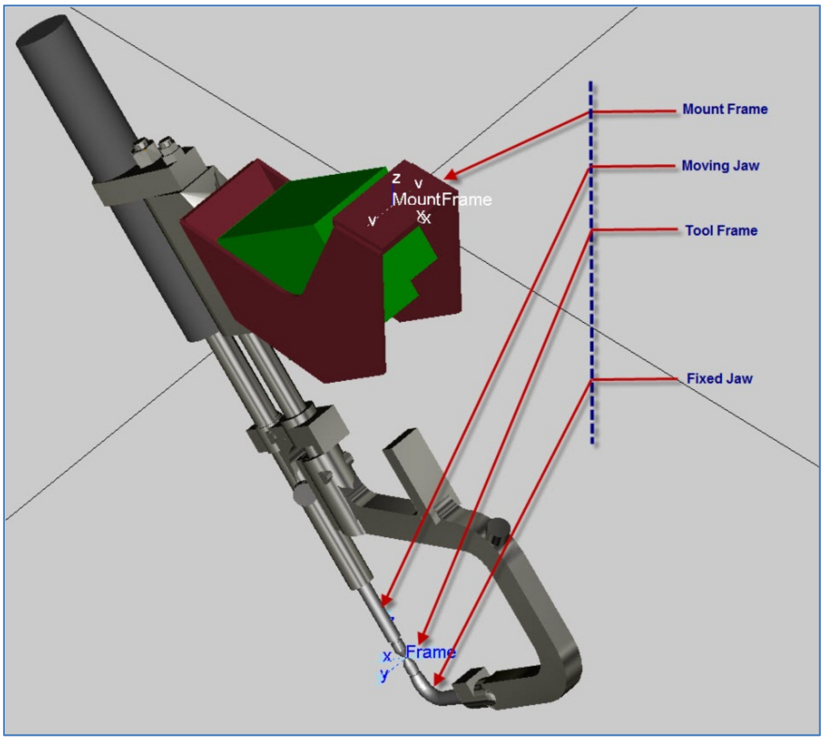


Figure 4-10 How spot weld gun is defined in the simulation software

Most of the spot weld guns have at least 3 home positions:

a. fully closed
b. semi opened
c. fully opened

Normally the spot weld guns are not servo driven but pneumatic driven. In offline programming there are several terms to be considered for the spot weld gun. There are two different types of speeds while spot welding, i.e. move speed and welding speed. Welding speed is the speed by which the spot weld gun is performing the spot weld operation (open/close) and moving speed is the speed is the speed of the robot in between spot points.

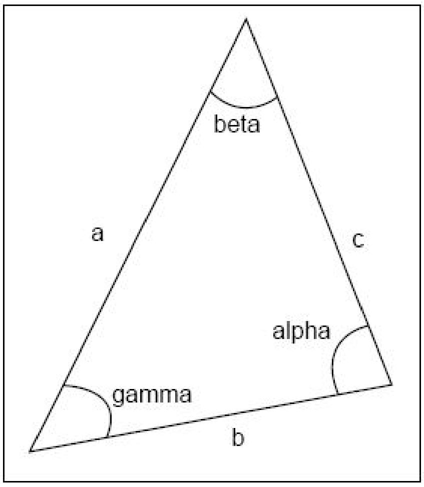
So basically what happens in industry is that each robot station has its own spot weld tool. The stations are separated based on the type of operation going to be performed in each station. Also there are provisions for tool change (spot weld gun) in same station. In offline programming of robotic spot weld with the 3D Create software each spot point contains the information of the tool definition that means each spot point is assigned with the tool name.

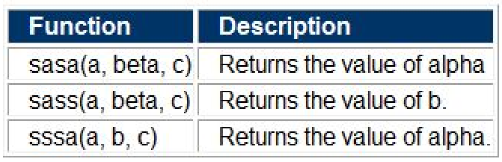
In consideration to the hardware properties of spot weld guns, among pneumatic-hydraulic-servo the pneumatic type mostly used for a lighter and uniform electrode. Hydraulic force is used in case of limited space and high force requirement. Also the spot weld guns can be conventional design, custom design and new generation. The conventional designs are the ones mentioned at the beginning of this chapter, custom ones also basically similar to the conventional one’s, the difference is in dimensions and to develop some new kinematics for the newer type. New generation spot weld guns mostly based on a modular concept to reduce cost and increase efficiency. An example is the 3G spot weld guns from ARO welding technologies. They have made 3G spot weld guns which are in 9 shared modules, 7 specific modules and 2 plug and play arms [56].

Kinematics of Spot Weld Guns

The kinematics of the spot weld guns needs to be generated from a template most of the time. Different types of guns have different arm lengths and stroke. But in general the basic kinematic structure doesn’t differ too much. C-type, D-type, X-type or servo guns have quite straight forward kinematics, unless the gun has some additional links like a 4-link closed loop mechanism. Then it needs some additional kinematics calculation. After the kinematics has been made properly then the spot weld gun is ready to be plugged with the robot. In this thesis work couple of different types of spot weld guns have been developed with kinematics and used in the final simulation layout.

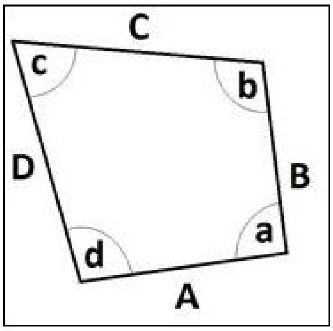
For developing a kinematics of a component the functions mostly used in 3D Create are sasad, sassd etc. The details of these functions can be found from the help file of the software, which makes developent of kinematics much easier and handy.

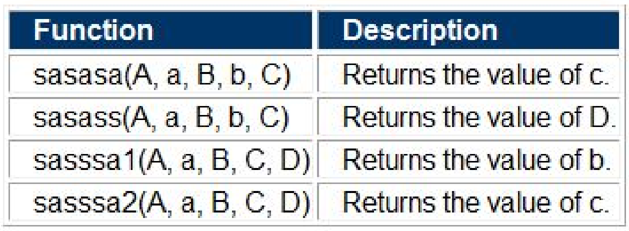




Function	Description
sasa(a, beta, c)	Returns the value of alpha
sass(a, beta, c)	Returns the value of b.
sssa(a, b, c)	Returns the value of alpha.

Figure 4-11 Helper function for solving kinematics- Triangle





Function	Description
sasasa(A, a, B, b, C)	Returns the value of c.
sasass(A, a, B, b, C)	Returns the value of D.
sasssa1(A, a, B, C, D)	Returns the value of b.
sasssa2(A, a, B, C, D)	Returns the value of c.

Figure 4-12 Helper function for solving kinematics - Qudrilateral

Parameters associated with spot welding

There are several parameters that are very important in the robotic spot welding process, incl:

- Open position: Defines the stroke length/angle of the moving arm when the gun is considered fully open.
- Close position: Defines the stroke length/angle of the moving arm when the gun is considered fully closed.
- Semi open: Defines the stroke length/angle of the moving arm when the gun is moving in between close proximity spot point.
- Weld Delay: The time the arms will be in closed position while spot welding.
- Tip Pressure: The amount of pressure the tip of the spot weld gun will generate while welding.
- Voltage: To set and alter the spot weld gun’s voltage value as necessary.

There are no universal standard for parameters for the defining a spot weld. Different spot weld companies have their own parameters and the list can be pretty lengthy. But the points mentioned above are the most essential and basic ones.

4.4. Spot Welding Line Setup

4.4.1. Macros for reading/writing spot welding data

As per the target of thesis work an Add-on was developed by using Python API of 3D Automate to implement robotic spot welding. There were several challenges. To start from the basic modelling of parts in various CAD modelling software (Auto CAD, Solidworks, CATIA etc.) and then to understand the way how spot welding data is assigned to those parts. After that to make the spot weld information readable to 3D Automate an example platform was developed. This is basically a COM add-on coded with C# to read the spot weld data from a part in 3D Automate and write them to CSV file in an order. This was done to demonstrate a use case. Practically the target is to develop a macro for each widely used external CAD software and write the spot weld data's to a CSV file in a standard format which can be read by using the developed Add-on [SpotWelding] and assign spot welding points in the same CAD file in the *.vcm file which is the CAD file extension name for 3D Automate. In Appendix 1 there is an example of an excel file which contains the spot weld data's in a CATIA part. This excel file has been created by using own developed macro by one of the OEM's of Visual Components Oy.

There are many companies who have developed their own macro on top of software like AutoCad, SolidWorks or CATIA to export spot weld data's from the CAD model to Excel or CSV file.

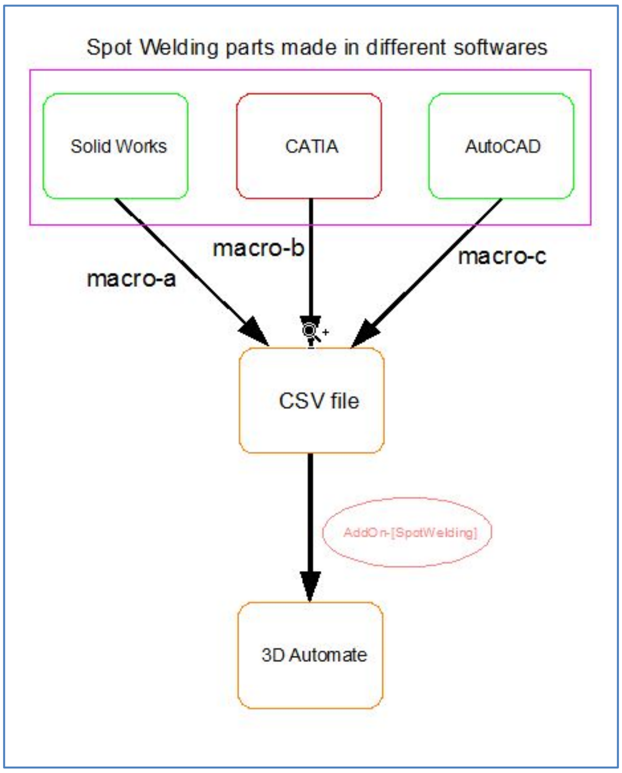
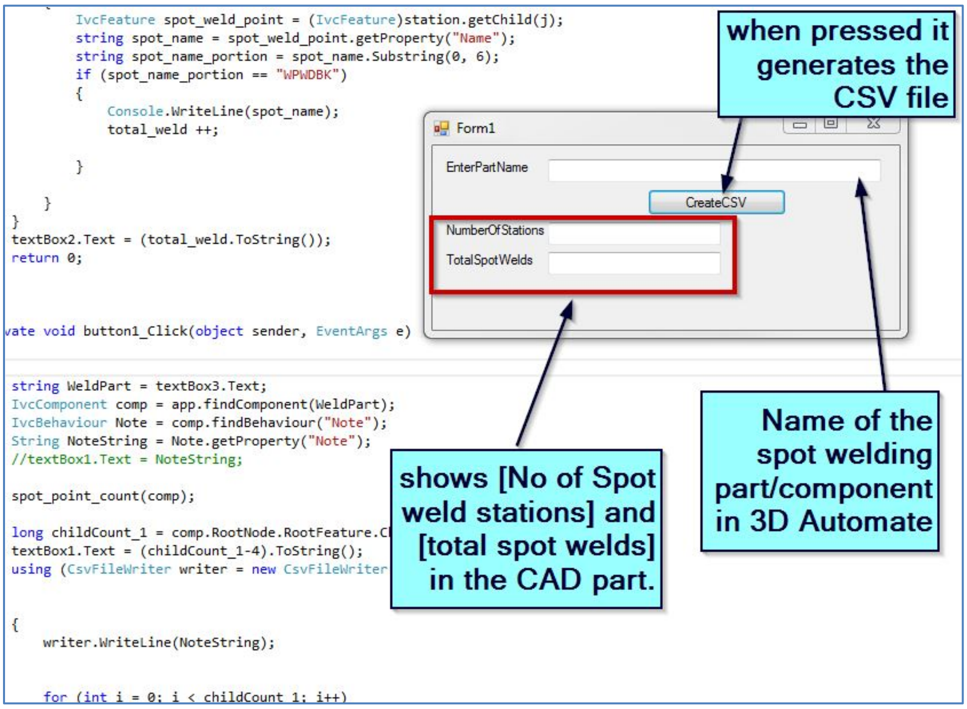


Figure 4-13 Proposal for Macro for spot weld info communication

As mentioned earlier a COM Add-on was developed by C# which reads the spot weld points from 3D Automate and creates the CSV file in a certain order where the spot weld data's are stored in the following order:

1	2	3	4	5	6	7	8	9
StationID	SpotID	x	y	z	i	j	k	AssociatedParts

Figure 4-14 presents screenshot of the COM add-on user interface:



```
        IvcFeature spot_weld_point = (IvcFeature)station.getChild(j);
        string spot_name = spot_weld_point.getProperty("Name");
        string spot_name_portion = spot_name.Substring(0, 6);
        if (spot_name_portion == "WPWDBK")
        {
            Console.WriteLine(spot_name);
            total_weld ++;

        }

    }
}
textBox2.Text = (total_weld.ToString());
return 0;


vate void button1_Click(object sender, EventArgs e)

string WeldPart = textBox3.Text;
IvcComponent comp = app.findComponent(WeldPart);
IvcBehaviour Note = comp.findBehaviour("Note");
String NoteString = Note.getProperty("Note");
//textBox1.Text = NoteString;

spot_point_count(comp);

long childCount_1 = comp.RootNode.RootFeature.C
textBox1.Text = (childCount_1-4).ToString();
using (CsvFileWriter writer = new CsvFileWriter


{
    writer.WriteLine(NoteString);


    for (int i = 0; i < childCount_1; i++)
```

when pressed it generates the CSV file

shows [No of Spot weld stations] and [total spot welds] in the CAD part.

Name of the spot welding part/component in 3D Automate

Figure 4-14 Developed COM AddOn user interface

The main idea for having parameters like [StationID, SpotID] is that a certain number of spot welds will be separated in different groups. So it will be easier to assign those spot welds to different robots in different spot weld stations. It was done this way because if a part has like 300 spot welds then it's not wise to assign them locally to all robots and then delete/remove some spot weld points from some certain and distributing them manually. It will be a very time consuming and inefficient process. Instead the user can just select the appropriate group of spot welds for specific robot in specific station and create those spot weld RSL statements to that robot executor. The facts relating to StationID and SpotID are going to be explained more elaborately in the following section.

Commercially many engineering and manufacturing design and consulting companies develop their own customized macros which extract Spot welding data from the CAD software like CATIA, Auto CAD etc. and then generate excel or CSV file containing all the necessary spot weld information. One of the example is Technodat, a Czech Republic based company who provides Macro to its customers for spot weld application. This macro can rename spot weld's in the CATIA part, save the parameters to each spot weld and export all this data to a *.csv file [57].

Another example is Cosma Engineering (one operating unit of Magna International) who has a proprietary macro to export weld data from CATIA.

4.4.2. Setup of Spot Welds in CAD model

The Add-On [SpotWelding] has been made in such a way that the spot welding can be implemented in the CAD model in 2 different approaches. The Add-On can read *.csv files and to create frames in the component. Then these frame positions and orientations are used to generate the RSL motion statements in the spot welding robots. The second approach is that the spot weld points are already available in the CAD model itself as Geo feature. So the add-on can read the position matrix of those spot weld Geo features and then generate the RSL motion statements.

In case of reading spot weld data from the *.csv file and creating frames in the spot welding workpiece, first the position matrix of the root node of the component is grabbed. Then from the root node of the component the position matrix of the frame for spot welding is calculated and assigned. There are always several parts for spot welding; they are represented as separate features in the same component. It makes a lot easier to follow the parts and just focus on the spot welding job. The parts (GEO_FEATURE's) which will be spot welded in different stations can just be made visible by signal communication, so the offline programmer doesn't have to worry about handling different components or visualizing the incoming flow of different spot welding parts in the simulation. In the testbench the car BIW is set on a carrier which is fed through different spot welding stations via conveyors.

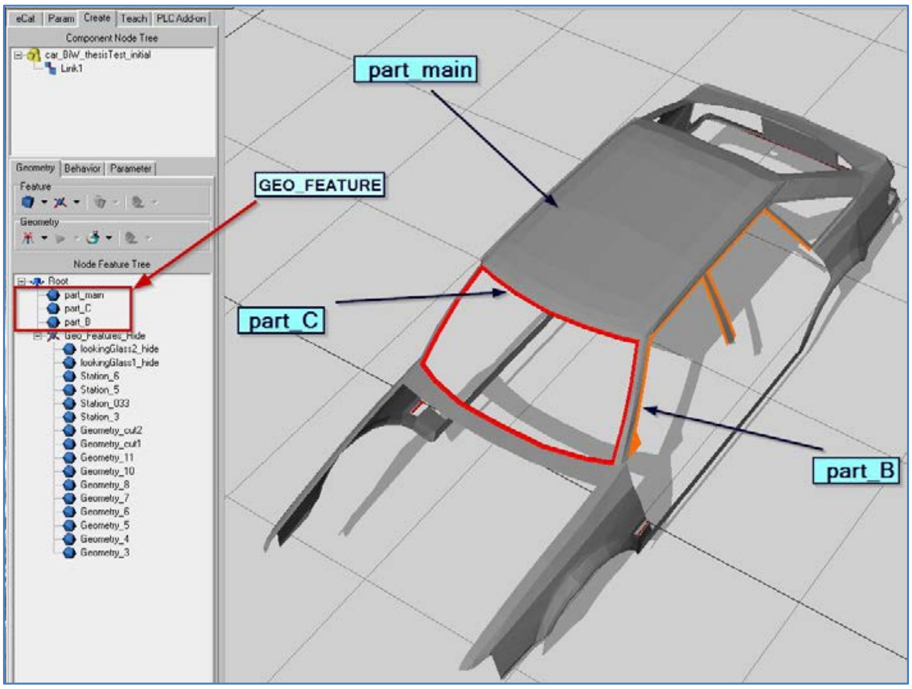


Figure 4-15 Car BIW CAD model for spot welding

A number of spot welds were grouped under certain group and these groups were named so that it will be easier to recognize the correct robot for a certain group of spot welds. If the car body is cross sectioned to four areas from top view then the first group of welds in the car is the bottom left section of the car, then the user has to think the consecutive group of welds clockwise. In this approach the 2nd, 3rd and 4th group of welds are presented in figure 4-16:

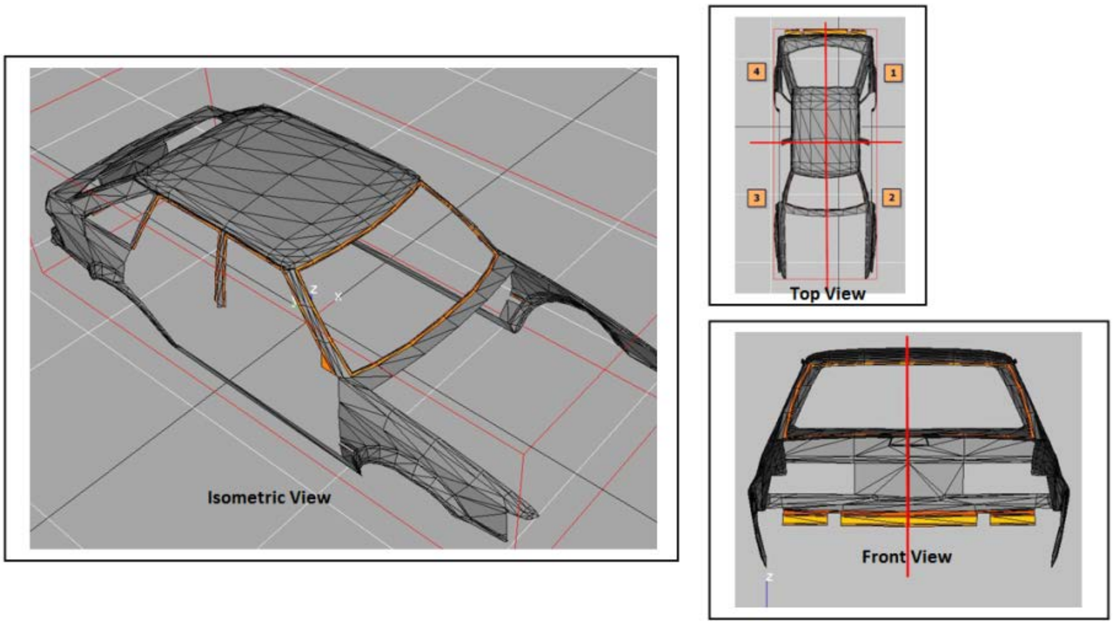


Figure 4-16 Cross section of CAR body for spot welding

4.4.3. Planning the Spot Welding Assembly Line

In this test bench a car BIW spot welding line was designed where there are three different spot welding stations in series. Each station was assigned with four spot welding robots. Also a general rule was assumed to identify each robot in each individual station. In every station towards the direction of movement spot welding model (car BIW in this case) the robot on upper left side is considered the first robot in the station. Then the other robots are counted clock-wise as 2nd, 3rd and 4th robot of that station consecutively. In the test bench if the StationID of a group of spot weld is Station_3_1 that means this group of spot welds belongs to the 1st robot in the 3rd station. The screenshot of figure 4-17 shows the robotic spot welding line testbench more elaborately:

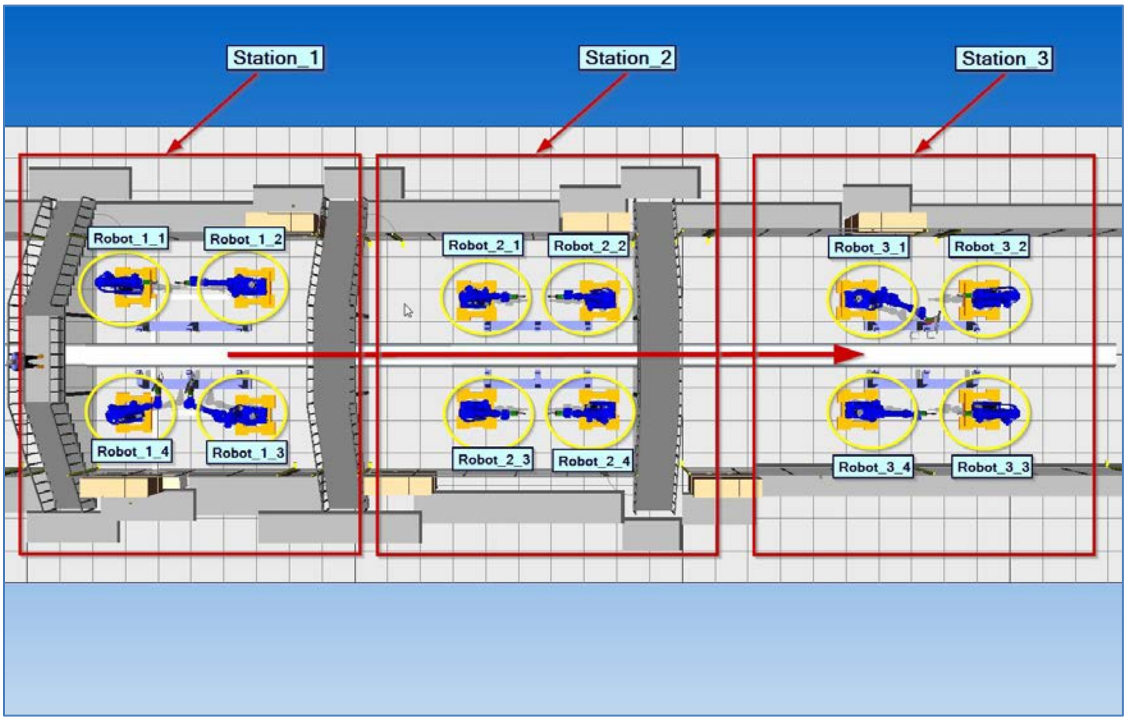
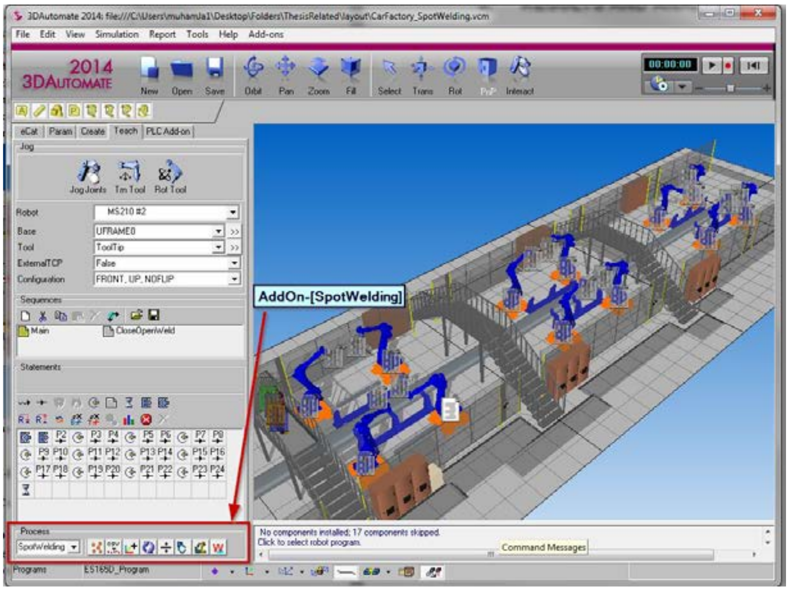


Figure 4-17 Robotic spot welding line setup.

4.4.4. Features of the Add-On [SpotWelding]

When 3D Automate is opened then the AddOn-[SpotWelding] will appear in the lower left corner of the GUI window:

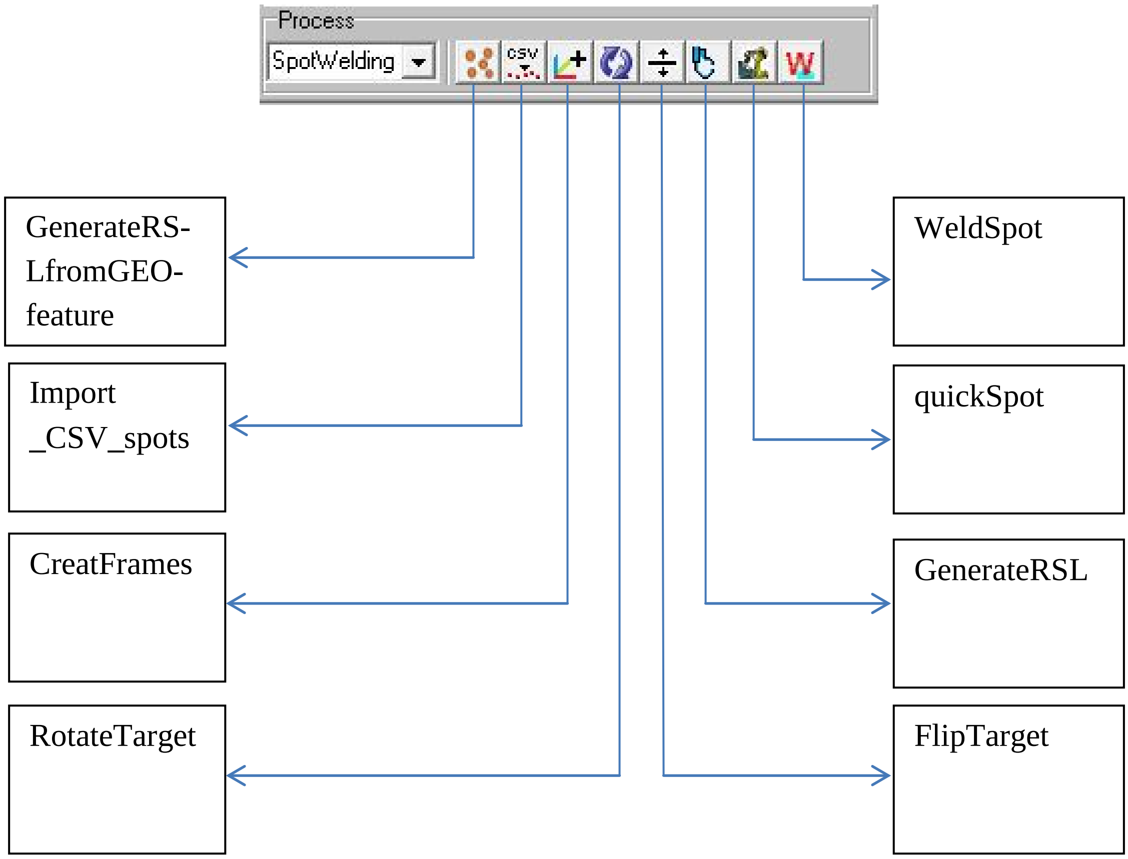

3DAutomate 2014: file:///C:\Users\muhamJa1\Desktop\Folders\ThesisRelated\layout\CarFactory_SpotWelding.vcm

Robot: MS210 #2
Base: UFRAME0
Tool: ToolTip
ExternalTCP: False
Configuration: FRONT, UP, NOFLIP
Process: SpotWelding
Programs: ES165D_Program

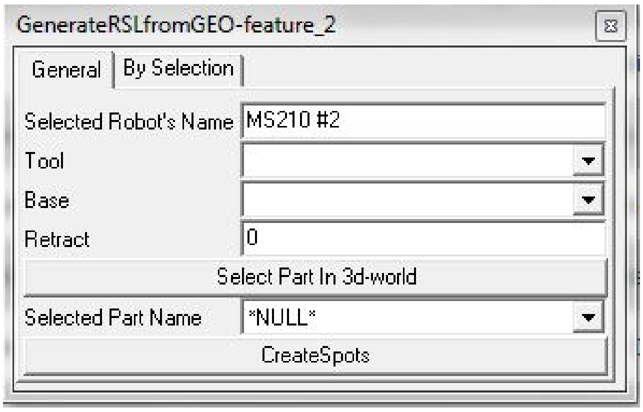
No components installed; 17 components skipped.
Click to select robot program.

Figure 4-18 AddOn location in the GUI of 3D Automate

There are 8 different tools in the whole functionality of the AddOn [SpotWelding].



GenerateRSLfromGEO-feature


GenerateRSLfromGEO-feature_2

Selected Robot's Name: MS210 #2
Tool:
Base:
Retract: 0

Selected Part Name: *NULL*

Figure 4-19 SpotWeldingTool : GenerateRSLfromGEO-feature

In the manufacturing industry CAD modellers usually models the CAD geometry with the spot welding points included as small geometry features. This makes the OLP job a lot easier as the tag point for robot can be easily created from the CAD data directly. Here in the car BiW model there are almost 150 spot weld points which are represented by small circular geometry features.

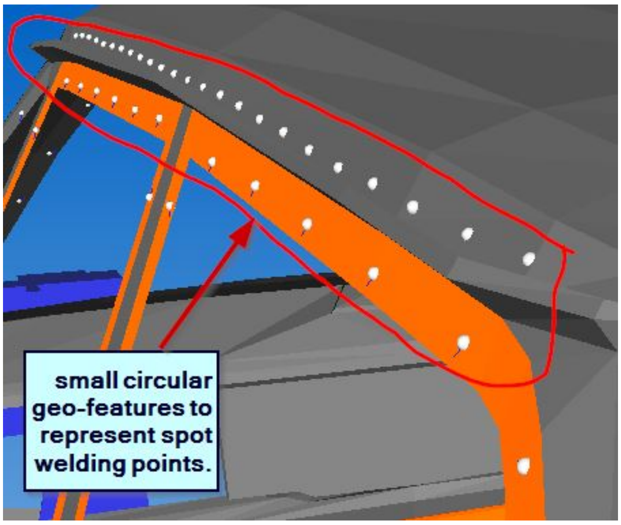

small circular geo-features to represent spot welding points.

Figure 4-20 Geometry feature pointing spot location

To run this AddOn a robot needs to be selected first. When the user runs this AddOn the name of the selected robot pops up in the box [Selected Robot’s Name]. Then the user has to enter the name of the station for which the selected robot is going to generate the RSL statements for the spot welding operation. While the geo spots were created in the car BiW CAD model it was kept in mind that there are four robots in each station and the spot welds were grouped in a way so that the reachability of spot weld points in each robot in each station are conveniently distributed.

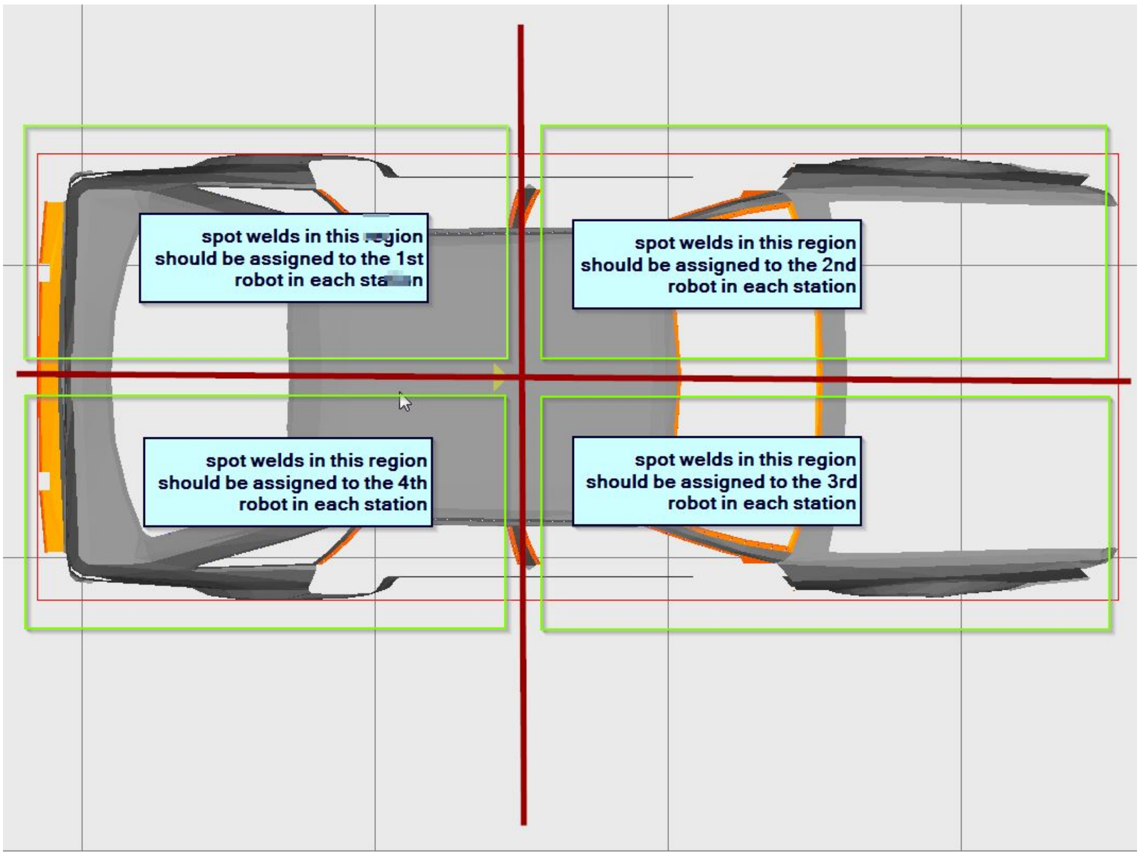
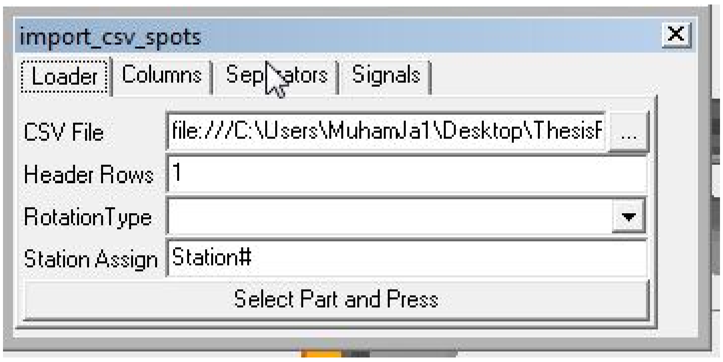

spot welds in this region should be assigned to the 1st robot in each station
spot welds in this region should be assigned to the 2nd robot in each station
spot welds in this region should be assigned to the 4th robot in each station
spot welds in this region should be assigned to the 3rd robot in each station

Figure 4-21 Distribution of spot welds according to robot position

Import_CSV_spots



import_csv_spots

CSV File: file:///C:\Users\MuhamJa1\Desktop\ThesisF

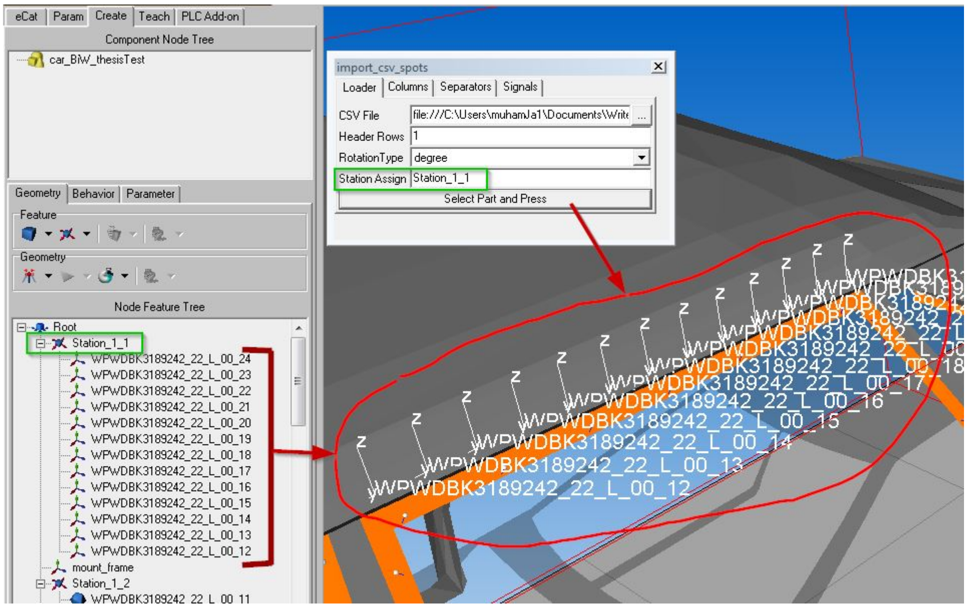
Header Rows: 1

RotationType:

Station Assign: Station#

Figure 4-22 SpotWeldingTool : Import_CSV_spots

This tool is developed to read from the CSV file the spot welding information and create frames in the CAD model. This tool has the flexibility to select one definite station for a group of spot welds, to select the rotation type (degree or radian) to define the columns of the CSV file, to define the I/O signals of the spot weld gun. For example when the user gives the value in the field [Station Assign] – “Station_1_1” then after selecting the part it searches for all the spot welds in the CSV file with the StationID as “Station_1_1” and creates frames in the CAD model within the same group name. An example is given below.



import_csv_spots

CSV File: file:///C:\Users\muhamJa1\Documents\Writ

Header Rows: 1

RotationType: degree

Station Assign: Station_1_1

Component Node Tree

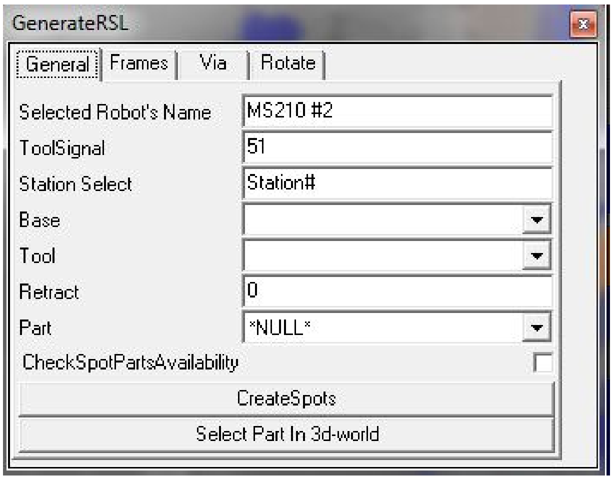
car_BiW_thesisTest

Node Feature Tree

Root
Station_1_1
WPWDBK3189242_22_L_00_24
WPWDBK3189242_22_L_00_23
WPWDBK3189242_22_L_00_22
WPWDBK3189242_22_L_00_21
WPWDBK3189242_22_L_00_20
WPWDBK3189242_22_L_00_19
WPWDBK3189242_22_L_00_18
WPWDBK3189242_22_L_00_17
WPWDBK3189242_22_L_00_16
WPWDBK3189242_22_L_00_15
WPWDBK3189242_22_L_00_14
WPWDBK3189242_22_L_00_13
WPWDBK3189242_22_L_00_12
mount_frame
Station_1_2
WPWDBK3189242_22_L_00_11

Figure 4-23 Frames created in CAD file from the CSV file

GenerateRSL

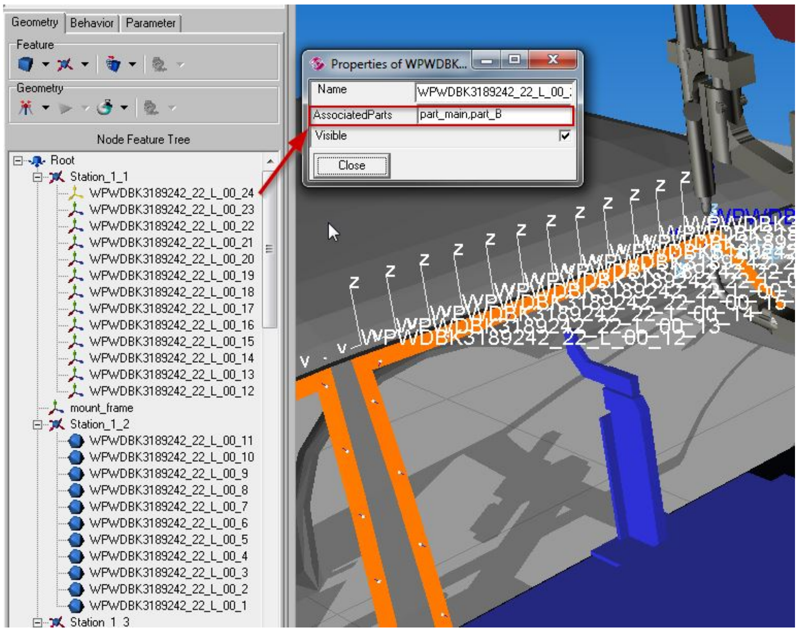


Selected Robot's Name: MS210 #2
ToolSignal: 51
Station Select: Station#
Base:
Tool:
Retract: 0
Part: *NULL*
CheckSpotPartsAvailability ☐

Figure 4-24 SpotWeldingTool : GenerateRSL

After the frames have been created in the CAD model of the spot welding workpiece, the user needs to run this tool to create the RSL statements. Options in this tool are:

- Via: The user can select this tab and then enter values to define the approach and retract via points. For retract motion after each spot weld the user has to enter value in the [Retract] section under the tab [General].
- Rotate: There is the need for re-orientating the collection of spot welding points if the reachability is confined or there are collisions or the joint configuration is not appropriate. So the user can just enter the value in [RotateRelX], [RotateRelY] or [RotateRelZ] field, re-generate the RSL statements and the position matrices of each RSL statement gets updated.
- CheckSpotPartsAvalability: The program reads the value of the property [AssociatedParts] in each weld frame and then check if all the geometry features (as mentioned earlier, different parts are represented as features) are available related to each spot welding point.



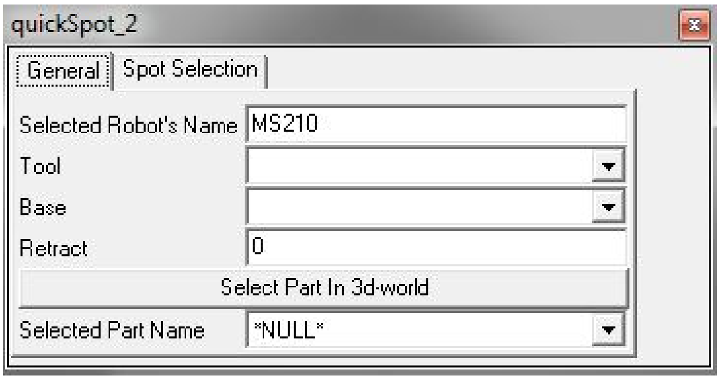
Name: WPWDBK3189242_22_L_00_

AssociatedParts: part_main,part_B

Visible ☑

Figure 4-25 Option to check spot weld associated parts

quickSpot



Selected Robot's Name: MS210

Tool:

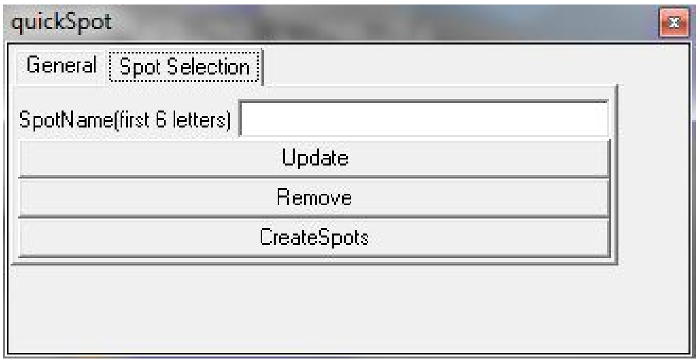
Base:

Retract: 0

Selected Part Name: *NULL*

Figure 4-26 SpotWeldingTool : quickSpot

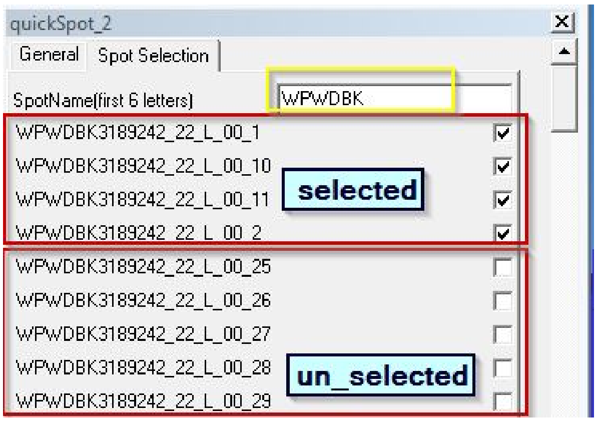
There can be also cases where the user just wants to load all the spot welding related geometry features, regardless of the station or groups. So the user simply wants to load all the points and then make selection from there, and finally create RSL statement for the selected points. So here the user needs to enter just the first six letters of the names of the spot weld geometry feature or frame feature and then the program runs through the whole tree of the root node of the component for searching all the features which matches the criteria.



SpotName(first 6 letters):

Figure 4-27 quickSpot option : Spot Selection

After loading all the spot points whose name matches with the first six letters, the user needs to unselect all those points which he doesn't desire to generate RSL, needs to press [Update] and next when pressed [CreateSpots] it generates RSL statements for the selected points. An example is shown in figure 4-28.



SpotName(first 6 letters): WPWDBK

☑ WPWDBK3189242_22_L_00_1
☑ WPWDBK3189242_22_L_00_10
☑ WPWDBK3189242_22_L_00_11
☑ WPWDBK3189242_22_L_00_2
□ WPWDBK3189242_22_L_00_25
□ WPWDBK3189242_22_L_00_26
□ WPWDBK3189242_22_L_00_27
□ WPWDBK3189242_22_L_00_28
□ WPWDBK3189242_22_L_00_29

Figure 4-28 quickSpot weld point selection method

CreateFrames

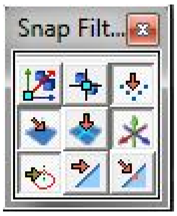
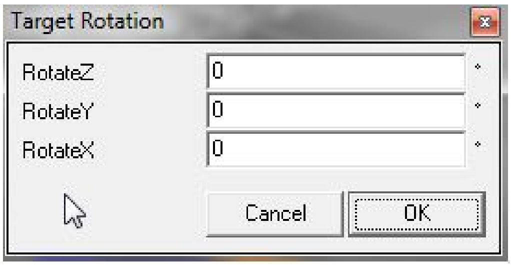


Figure 4-29 SpotWeldingTool : CreateFrames

If the user needs to create some quick spot welding points then he can just utilize this handy tool to generate frames on the CAD model with 9 different interactive snapping options. This creates frames with the names spot_1, spot_2, spot_3 and so on.

RotateTarget



RotateZ 0 °
RotateY 0 °
RotateX 0 °

Figure 4-30 SpotWeldingTool : RotateTarget

Once the RSL statements have been created there are often requirement that the orientation of the points needs to be updated. The user just needs to run this simple command and enter values in the fields-(RotateX/RotateY/RotateZ), this will rotate all the targets/RSL points in the current program.

FlipTarget

Just by pressing this button the program rotates the Z direction of the target points to complete opposite direction. This means that this command rotates all the RSL target point in the current program by Ry=180^0.

WeldSpot

This feature is under development and for future implementation. There are some limitations of RSL statements. We don’t have circular motion in RSL and there are also no conditional loops. There will be needs for implementation of changing of weld gun, changing of parameters like semi-open position, weld time, tip pressure, voltage and conditional loops like if-else loop or for loop during the program execution without the need of going to the parameter tab of robot or spot weld gun .All these can be achieved by RSL process handler. RSL process handler contains a python script and many operations like 1) create/delete/update property of robot or other components and 2) include conditional loops or algorithm for some critical task (fly by points, collision detection etc.) can be done. [WeldSpot] tool is an approach to implement more intelligent in this spot welding AddOn in the near future.

4.5. Demonstration

In this chapter the working principle of the developed AddOn [SpotWeld] is discussed. A line consisting of three robotic assembly stations with each station consisting of four Motoman MS210 robots are set in the virtual environment. This automotive assembly line didn’t take into consideration the details of part transfer, fixture assembly and intermediate process. The main aim is to provide an overview of the proposed spot welding tool.

4.5.1. Robot Selection

The robot which was selected for the spot welding line in this thesis work is a new generation spot welding robot MS210 from Motoman. It is claimed to be the next generation ultra-fast, high-density robot for spot welding application.

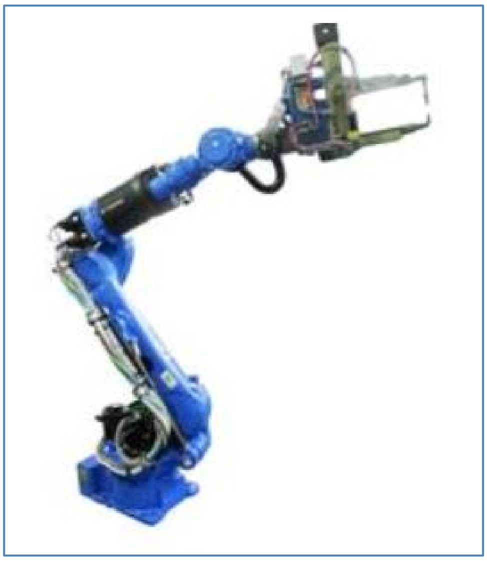


Figure 4-31 Motoman MS210 spot welding robot.

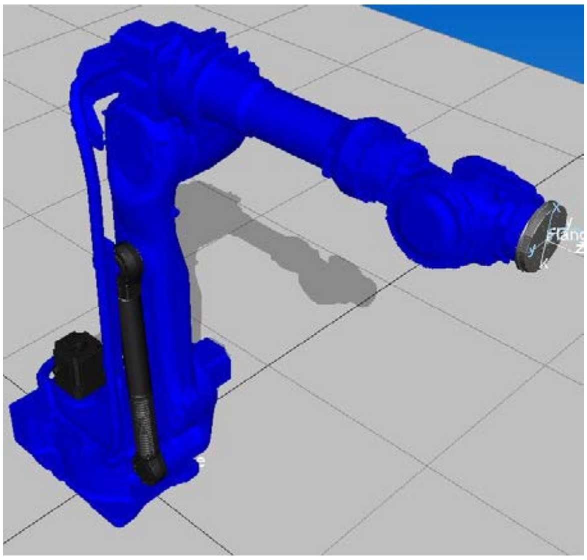


Figure 4-32 Motoman MS210 in 3D CAD model

Features of this recently released robot are given below [58]:

- Ultra fast operation with new vibration control feature. By implementing new lightweight components the joint speeds has been increased 25% and cycle time decreased 30% from previous models. In future by utilizing new sensor free learning control feature cycle time can be reduced by more 10%.
- Gas balancer has been implemented in joint-2 reducing the robot width by 25% and the robot base also been reduced by 43% from previous models. This has made the robot highly recommendable for high density installation.
- It has a high payload of 210 kg, which makes possible to use large spot weld guns. It has only one power cable, instead of 2 power cables in the previous model.
- It has advanced spot welding functionality which comes with the new controller DX200. It has automatic spot weld gun tuning feature for guns with different

specifications. Also DX200 controller has enhanced security functions like tool conversion surveillance function etc. The cabinet for the controller has been compactly designed which reduced the installation area.

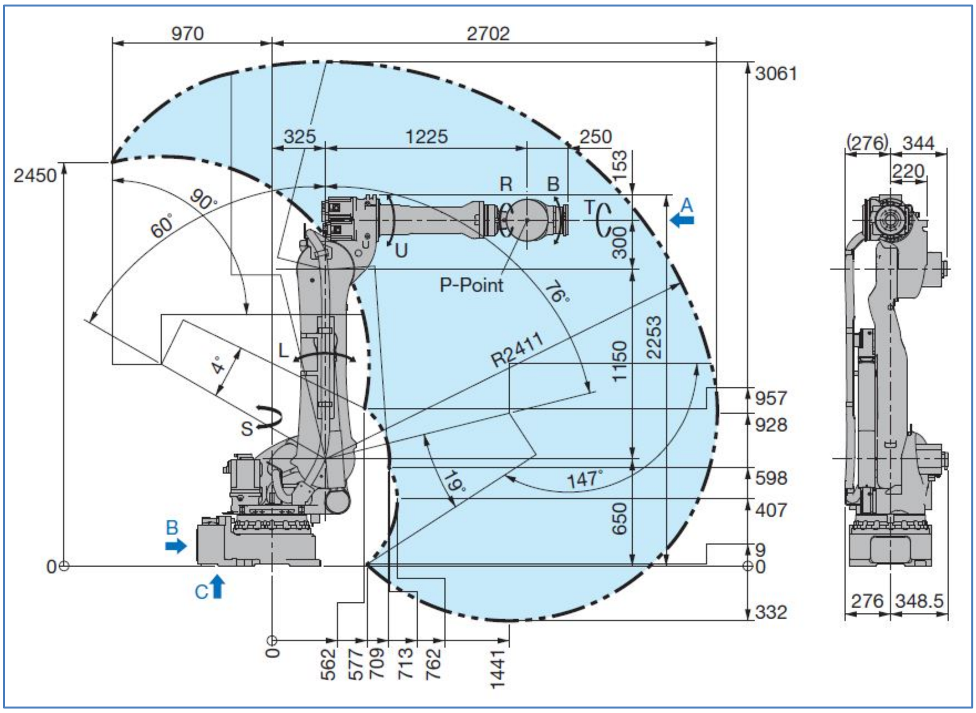


Figure 4-33 Work envelope of Motoman MS210

4.5.2. Simulation of the Spot Welding Line

The spot welding line was implemented for a car with 3 different parts to be spot welded by utilizing three spot welding stations and each station having four spot welding robots. The car frame has three different parts which are going to be spot welded in the line. By utilizing the developed AddOn-[SpotWelding] the RSL statements have been created in the robots. There were nearly total 156 spot welds in the whole simulation. In the sequence the car part comes on a carrier to the spot welding station and the conveyor gives an output signal to the robots that the part has arrived. After that each robot gives an output signal to the conveyor to stop. The spot welding operation starts and after completion of all the spot welds the robot gives an output signal to the conveyor to start again. When the conveyor gets input signal from each of the robot’s in one station it starts and the carrier (which carries the car part) goes to the next station. The spot welds are created by using the AddOn-[SpotWelding]. Both the functions of creating RSL from the CSV file or creating RSL from geometry features were used to demonstrate the functionality of the AddOn. There are several parameters that count for the cycle time of spot welding:

- Speed of conveyor
- Speed of servo of spot welding gun
- Number of spot welds

- Motion/path planning
- Synchronous and non-synchronous movement of the robot and spot welding part
- Time of each spot weld
- Reachability and joint configuration.

The whole simulation which included all the 156 spot welds in three different station and 16 spot welding robots takes 4 min only. The variables in the spot weld gun are a major factor in the cycle time. Those are:

- Stroke speed
- Weld Time
- Voltage
- Tip Pressure
- Synchronous motion with the external axis
- Complexity and distribution of the spot welding points.

The AddOn [SpotWelding] is based on a CAD based approach. This tool enables the user to work on the CAD file, define welding path and approach escape path between two consecutive welds, organize the welding sequence. When the definition is complete the designer can export all the spot welding data to an external *.csv file using the developed COM application [CreateCSVspot] which is written in C#. This tool extracts all the spot weld information from the CAD file and creates the *.csv file. After that utilizing the tools in the AddOn the user can extract information from the CAD file or the *.csv file and create robot command (RSL statements) and can be immediately tested for detailed tuning. Then utilizing the modifying tools in the AddOn there can be necessary modification, correction of the spot welding parameters. After a few simulation, checking collision points and robot movement paths needs to be updated, speed updating for different inter process points for faster operation it needs tuning of the robot program and the whole robot program in RSL is ready to be converted to robot specific (in this case Motoman MS210) program by using the OLP tool. The whole process can take from a few minutes to few hours depending on the complexity of the process.

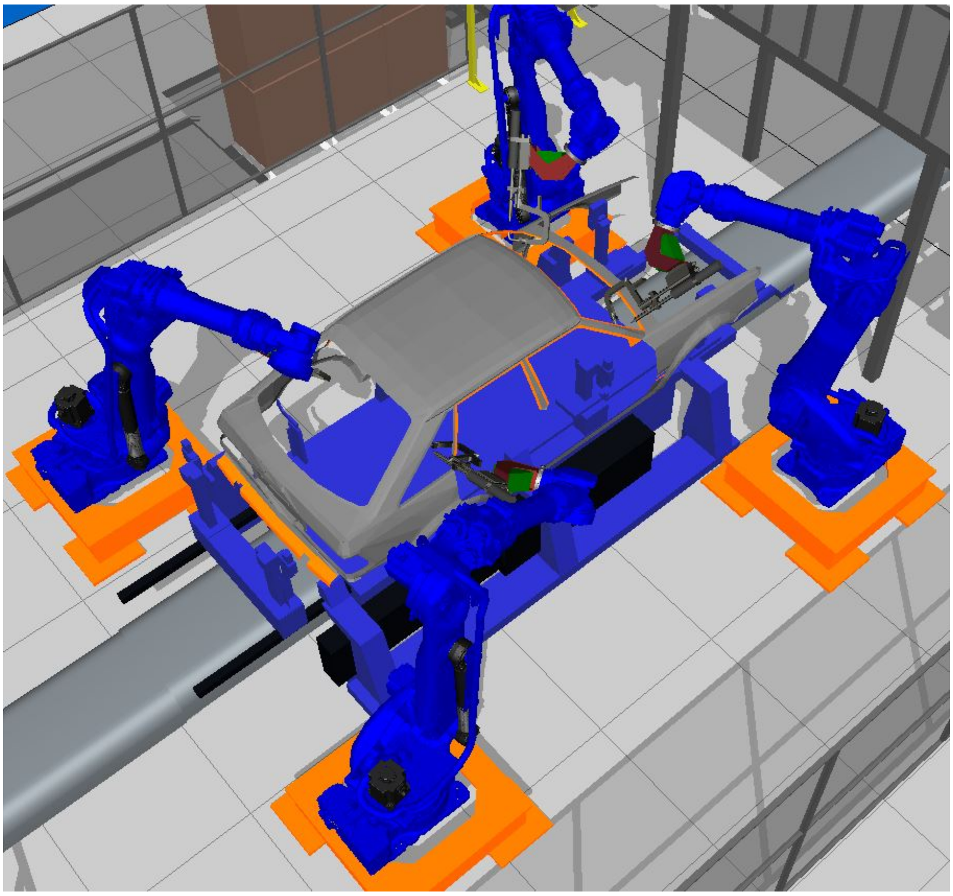


Figure 4-34 One Station with four spot wlding robots

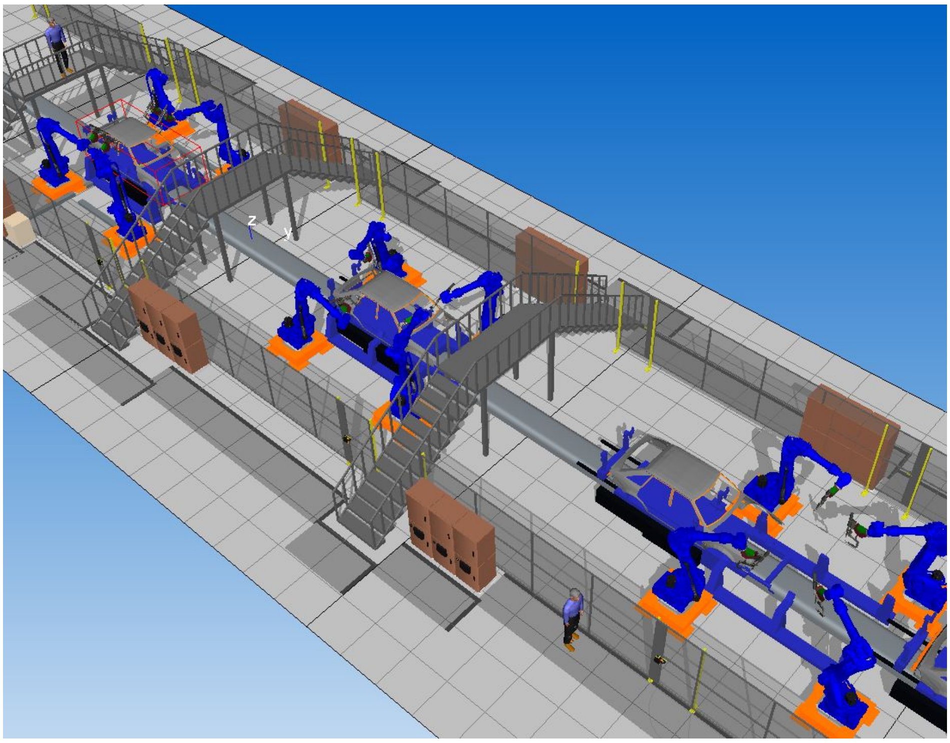


Figure 4-35 Isometric view of the spot welding line

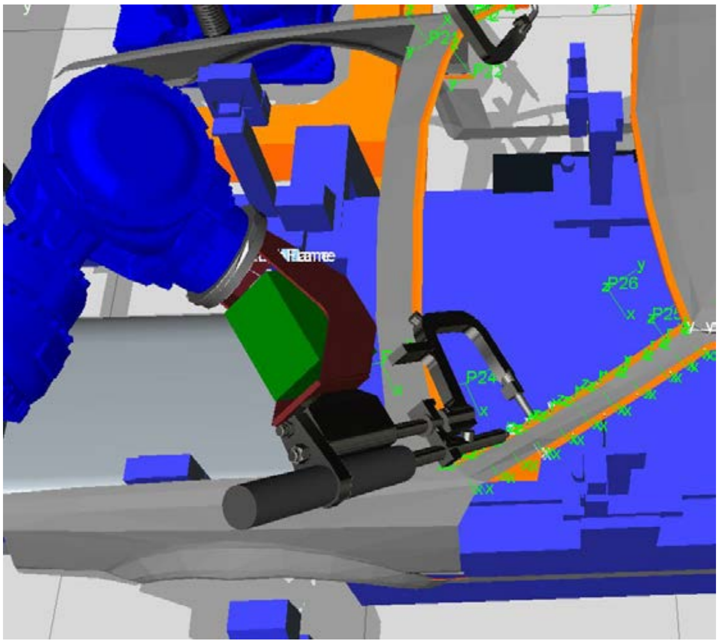


Figure 4-36 Close view of the spot welding operation

4.5.3. Simulation Result

The aim of the developed AddOn > [SpotWelding] was to make a closed loop solution for the automotive spot welding line. In previous practice while doing the spot weld process planning only one station was kept in consideration and then there was not enough feedback from the system to quickly make compensation or update in deviation the CAD geometry or the assembly line. The steps proposed for a better work process is shown below.

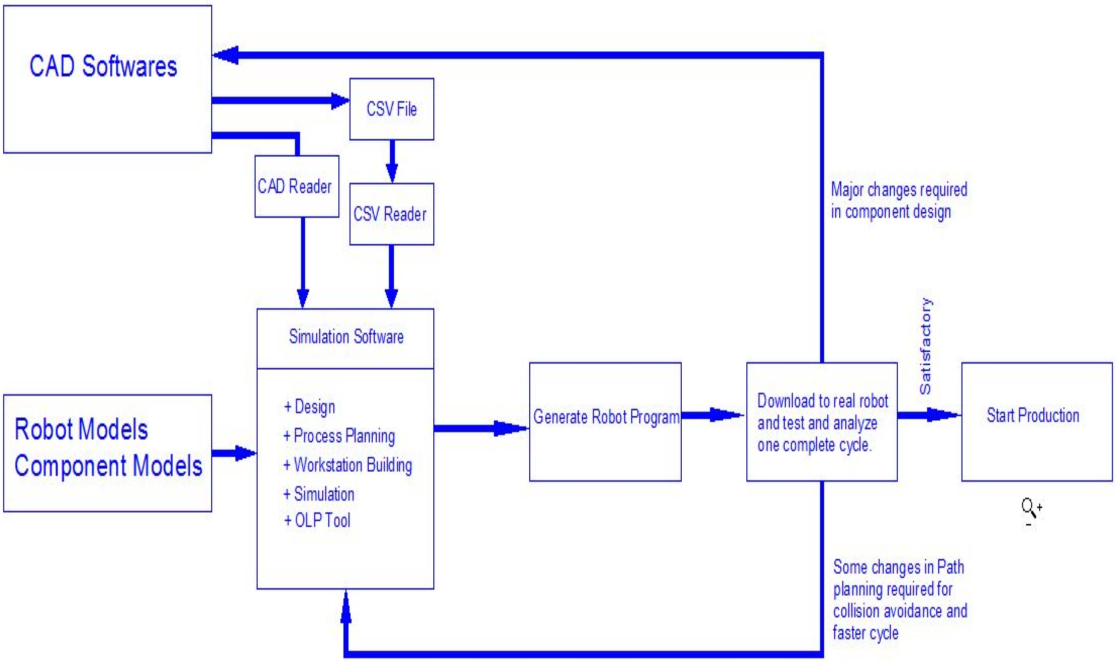


Figure 4-37 Proposed method for a spot welding assembly line

A spot welding line in virtual environment was made in 3D Automate and consisted of three assembly stations and each assembly station with four Motoman MS210 robots. In previous practices the storing and interchange of spot welding data were ignored. A method is proposed here which can be described in following steps:

- The CAD software (AutoCAD, Solid Works, CATIA, ProE etc.) generates the model of the assembly of the car body
- The details of spot welding operation are taken into account and the data are stored in 2 ways: a) In the CAD file or b) In a separate CSV or Excel file.
- As different CAD modellers may generate different kind of assembly information so there needs to be an intermediate converting AddON which will interpret the spot welding data and import the information in the simulation software.
- CAD readers makes it possible to import almost all the different CAD formats(Step, Igs, dwg, CATIA, SolidWorks etc.) to the simulation software. If the CAD files contains the spot weld data inside then these are also imported.
- The robot workcell environment is created (by using point cloud or imported CAD model) in the simulation software.
- The workcell is calibrated.
- The assembly parts are placed on the fixture and taken to the assembly station where the developed AddOn [SpotWeld] is used to generate the robot program.
- As path planning was done without any collision avoidance algorithm so the PTP (point to point) motions were directly assigned.
- Manual touch up and PTP motions needs to be added to the robot program to avoid collision with the assembly part and the neighbouring robot.
- Simulation is run and checked if right spot weld task was assigned to the right robot in the right assembly station, if not the AddOn facilitates to quickly delete the current faulty program and generate the spot sequence correctly again.
- A closed loop method is proposed which starts from the as CAD design and goes through the 3D process simulation , trajectory planning and finally the robot the program is uploaded to the real robots in the factory floor.

5. RESULTS AND ANALYSIS

5.1. Conclusions

Current welding robot technology research focuses on the seam tracking technology, off line programming and path planning technology, multi-robot coordination control technology, dedicated arc welding power technology, welding robot systems simulation technology, robotic welding processes and remote welding technology. The proposed solution gives an overview of how to make the discrete automotive spot welding process into an ordered flow. The development of the AddON > [SpotWelding] required the knowledge of inverse kinematics and python programming. Also for developing the COM AddOn it required the knowledge of basic C# programming. The pilot implementation can be used directly as a basis for development of more advanced and robust implementations for production use or the design can be utilized for further development in collision free trajectory planning and intelligent via point generation. To reach the quality needed to propose a unique spot welding tool which has automatic collision free trajectory planner, which will reduce the currently huge time consuming part of manual touch up in the path planning phase of the spot weld sequence a lot of work still needs to be done.

Personally the work progress taught me a lot. I had a strong background in using CAD software, industrial robotics and production/assembly process thanks to my university studies and former work experience in the field, which helped me carry out the work. However programming with Python and C# was completely new to me and required a lot of work to get familiar with. It was a rewarding learning process which opened up my ideas and gave me motivation for further acquiring knowledge on virtual manufacturing, factory automation and using API extensively for developing programming and automation skills.

5.2. Future Work

With the continuous development of computer technology, network technology, intelligent control technology, artificial intelligence theory and industrial production systems, welding robot technology is constantly upgrading especially by the vision control technology, fuzzy control technology, intelligent control technology, embedded control technology, virtual reality technology & network control technology are the main direction for future research. Future research can be devoted to the application of neural learning mechanisms applied on the fuzzy and adaptive neurofuzzy planner. Potential direction for future research is needed in offline programming of industrial manipulators to generate faster robot programs in offline mode which also includes collision free motion planning for a single robot and multiple robot workcell. Trajectory planning is the field which autonomous systems researchers and engineers have been dealing for years. From time to time various algorithms have been proposed and some of them have been proved to be very effective. The developed AddOn for the thesis work can be effectively used in industry but still there is the bottleneck of time consuming step of manual touch up of the robot movement path for making the robot's total trajectory of the spot welding sequence collision free. In automatic robot path planning numerous researches has been going on for years and different approaches have been presented which can be categorized into potential field, cell decomposition and roadmap/skeleton method. After assigning the location and orientation of the spot welds in the robot program by the AddOn there should be an intelligent algorithm which automatically checks through all the point to point (PTP) movement of the robot and if collisions are found, checks the geometry of the assembly part/assembly and generates via points plus modified TCP orientation of the end of arm tooling. Also there are scopes for upgrading the workcell calibration method, robot calibration method for OLP, faster build-up of virtual models by using point cloud data from 3D scanner. In future the simulation software should be able to handle the whole sequence of importing CAD files of assembly parts, workcell then setting up the virtual workcell and running simulation, automatic collision-free path planning for minimum cycle time, generating OLP for the robots as well as generating PLC program for the fixture and conveyors in the assembly line and a faster calibration method.

BIBLIOGRAPHY

[1] Z. Li, Z. Huang and Y. Huang, “Design of Spot Welding Robot,” vol. 11, 2013.

[2] Z. J. Pasek, “Adaptive assembly system for automotive applications.,” 1993.

[3] P. Neto, J. Pires and A. Moreira, “CAD based offline robot programming,” in *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*, Singapore, 2010.

[4] S. Y. Nof, Handbook of Industrial Robotics, Wiley, 1999.

[5] D. Nitzan and C. Rozen, “Programmable Industrial Automation,” *IEEE Transactions On Computers*, Vols. C-25, no. 12, 2006.

[6] I. F. o. Robots, “www.ifr.org,” 2012. [Online]. Available: http://www.ifr.org/fileadmin/user_upload/downloads/forms___info/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf.

[7] “http://www.worldrobotics.org/,” 2012. [Online]. Available: http://www.worldrobotics.org/index.php?id=home&news_id=265. [Accessed July 2013].

[8] A. Gonzalez-Rodriruez and A. Gonzalez-Rodriguez, “Collision free motion planning and scheduling,” *Robotics and Computer integrated manufacturing*, pp. 657-665, 2010.

[9] T. S. and Z. P, Industrial Robotics: Theory, Modelling and Control, National Technical University of Athens, 2006.

[10] B. Siciliano and O. Khatib, Springer Handbook of Robotics, Springer Handbook, July 9; 2008.

[11] V. C. Oy.

[12] M. Urho, “Developing real-time communication between 3D simulation software components and electronics manufacturing equipment,” Department of Mechanical Engineering' TUT, Tampere, 2004.

[13] J. Eshbaugh, “Expediting equipment sales with 3D Simulation,” [Online]. Available: http://www.visualcomponents.com. [Accessed 2013].

[14] J. Browne and K. Rathmill, Simulation in Manufacturing, IFS Publications/Springer-Verlag, 1998.

[15] W. Chou, L. You and T. Wang, Automatic Path Planning for Welding Robot Based on Reconstructed Surface Model, Beijing: Robotics Institute" Beijing Institute of Aeronautics and Astronautics, 2007.

[16] H. Geng, Fundamentals and trends in robotic automation, McGraw-Hill, 2004.

[17] M. Hedelind and M. Jackson, “The need for reconfigurable system,” in *2nd*

international conference on changeable,reconfigurable and virtual production, Windsor, July 22-24; 2007.

[18] G. Reinhart, "Current State Model for Easy Reconfiguration of Robot Systems and Offline Programming Environments," Munich, Germany, 7-9 June; 2010.

[19] H. Nakamura, K. Yamamoto, T. Itaya and T. Koyama, "Development of off-line programming system for spot welding robot," in *IEEE 2nd International Workshop on Emerging technology and Factory automation*, Palm Cove-Cairns, Qld, 1993.

[20] L. a. H. T.-W. Everett, "The Theory of Kinematic Parameter Identification for Industrial Robots," *Transaction of ASME*, pp. 96-100, 1988.

[21] H. Zhuang, "A Complete and Parametrically Continuous Kinematic Model for Robot," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 451-63, 1992.

[22] M. J. M. S. T. and M. R. S. , "Modeling, Optimizing and Simulating Robot Calibration with Accuracy Improvements," *Journal of the Brazilian Society of*, vol. 21, pp. 386-402, 1999.

[23] M. J.M.S.T, Carvalho and McMaster.R.S, "Robot Calibration using a 3D Vision-Based Measurement System With a Single Camera," 1999.

[24] D. D. S, Z. and M. C, "Compensation of geometric and elastic errors in large manipulators with an application to a high accuracy medical system," *Robotica*, vol. 7, pp. 341-352, 2002.

[25] E. Park, W. X. and J. M. , "Calibration-based absolute localization of parts for Multi robot assembly," *Robotica*, vol. 20, pp. 359-366, 2002.

[26] J. M. S.T.Motta, Industrial Robotics: Programming, Simulation and Application, 2006.

[27] "www.3ds.com," Dassult Systems, 2013. [Online]. Available: http://ol.cadfamily.com/delmia/online/olpug_D2/olpugbt0415.htm.

[28] Z. Pan, J. Polden, N. Larkin, S. V. Duin and J. Norrish, "Recent progress on programming methods for industrial robots," 2010.

[29] Z. Bi and L. S.Y.T, "A framework for CAD and sensor based Robotic Coating Automation," *IEEE transactions on industrial informatics*, vol. 3, no. 1, pp. 84-91, Feb 2007.

[30] K. J.Y, "CAD based automated robot programming in adhesive spray systems for shoe outsoles and uppers," *Journal of robotic systems*, vol. 21, pp. 625-634, 2004.

[31] T. Pulkkinen, "2D CAD based robot programming for processing metal profiles in short series manufacturing," in *International Conference on Control, Automation and Systems*, Seoul, Korea, 2008.

[32] J. Pries, T. Godinho and P. Ferreira, "CAD interface for automatic robotic welding programming," *Industrial Robot:An international journal*, vol. 31, pp. 71-76, 2004.

[33] K. Kim, D. Kim and B. Nnaji, "Robot arc welding task sequencing using genetic algorithm," *IEEE Transactions*, pp. 865-880, 2002.

[34] H. Kang and J. Park, “Work planning using genetic algorithm and 3D simulation at a subassembly line of shipyard,” *MITS/IEEE Techno-Oceans,* Vols. 9-12, pp. 218-222, Nov 2004.

[35] F. S. Cheng, Advanced Techniques of Industrial Robot Programming, Advances in Robot Manipulators, Michigan: Ernest Hall, 2010.

[36] F. Cheng, “The Simulation Approach for Designing Robotic Workcells,” *Journal of Engineering Technology,* vol. 20, no. 2, pp. 42-48, 2003.

[37] C. Connolly, “ Delmia Robot Modeling Software Aids Nuclear and Other Industries,” *International Journal of Industrial Robot*, vol. 33, no. 4, pp. 259-264, 2006.

[38] Z. L, “Simulation in robotics,” vol. 79, pp. 879-897, 2008.

[39] M. S., B. K-D, M. G, S. D and M. G, “Off-line programming of an industrial robot for manufacturing,” *The International Journal of Advanced Manufacturing Technology,* vol. 26, no. 3, pp. 262-267, August 2005.

[40] S. M and K. I, “Generation of continuous tool paths based on CAD models for Friction Stir Welding in 3D,” in *Proceedings of the 15th Mediterranean Conference on Control& Automation,* Greece, 2003.

[41] C.-s. Kim and K.-S. Hong, “PC based offline programming using VRML for welding Robots in Shipbuilding,” *Proceedings of IEEE 2004 Conference on Robotics,Automation and Mechatronics,* vol. 2, pp. 949-954, 1-3 December 2004.

[42] “www.aws.org,” [Online]. Available: http://www.aws.org/wj/nov03/feature.html. [Accessed 10 2013].

[43] E. Group, “www.esigmbh.de,” February 2006. [Online]. Available: http://www.esigmbh.de/downloads/ESI/Dokumente/Welding/old/The_Welding_Simulation_Solution_090406.pdf. [Accessed July 2013].

[44] “www.robot-welding.com,” [Online]. Available: http://www.robot-welding.com/spot_welding.htm. [Accessed July 2013].

[45] M. Welds, “www.millerwelds.com,” June 2012. [Online]. Available: http://www.millerwelds.com/pdf/Resistance.pdf.

[46] A. W. Society, Specification for Automotive Resistance Spot Welding Electrodes, American Welding Society, 2005.

[47] W. Dong, H. Li and X. Teng, “Offline programming of Spot weld robot for car body in White based on Robcad,” *International Conference on Mechatronics and Automation,* August 2007.

[48] P. Rogadas and D. McMaster, “A robot cell calibration algorithm and its use with a 3D measuring system,” 1997.

[49] “www.csee.umbc.edu,” Robotic Simulations Ltd, [Online]. Available: http://www.csee.umbc.edu/~nicholas/676/files/101.html. [Accessed 01 10 2013].

[50] “http://www.microsoft.com/,” Microsoft, [Online]. Available: http://www.microsoft.com/com/default.mspx. [Accessed 03 08 2013].

[51] “www.cs.umd.edu,” [Online]. Available: http://www.cs.umd.edu/~pugh/com/. [Accessed 03 08 2013].

[52] “www.python.org,” Python Software Foundation (PSF). [Online]. [Accessed 07 08 2013].

[53] M. Dawson, Python Programming for the absolute beginner, Third ed., Boston: Course Technology, 2010.

[54] J. N. Pires, Industrial Robots Programming, Springer, 2007.

[55] J. J. Craig, Introduction to Robotics, 3rd ed., Prentice Hall, 2004.

[56] A. W. Technologies, “http://www.arotechnologies.com/,” [Online]. Available: http://www.arotechnologies.com/Product3G. [Accessed July 2013].

[57] “www.technodat.eu,” TECHNODAT, 2013. [Online]. Available: http://www.technodat.eu/spot-welds. [Accessed 09 2013].

[58] M. I. Robots, “www.yaskawa.co.jp,” 09 2013. [Online]. Available: http://www.yaskawa.co.jp/php/newsrelease/contents_en.php?id=74&year=2013&.

[59] Q. Zhu and Q. Deng, “Simulation Planning of Robot Welding Line,” *Second International Conference on Mechanic Automation and Control*, pp. 763-768, 2011.

[60] J. Belanger, P. Venne and J.-N. Paquin, “www.opal-rt.com,” [Online]. Available: http://www.opal-rt.com/sites/default/files/technical_papers/PES-GM-Tutorial_04%20-%20Real%20Time%20Simulation.pdf. [Accessed 02 08 2013].

[61] M. Lutz, Programming Python, Fourth ed., O'Reilly, 2011.

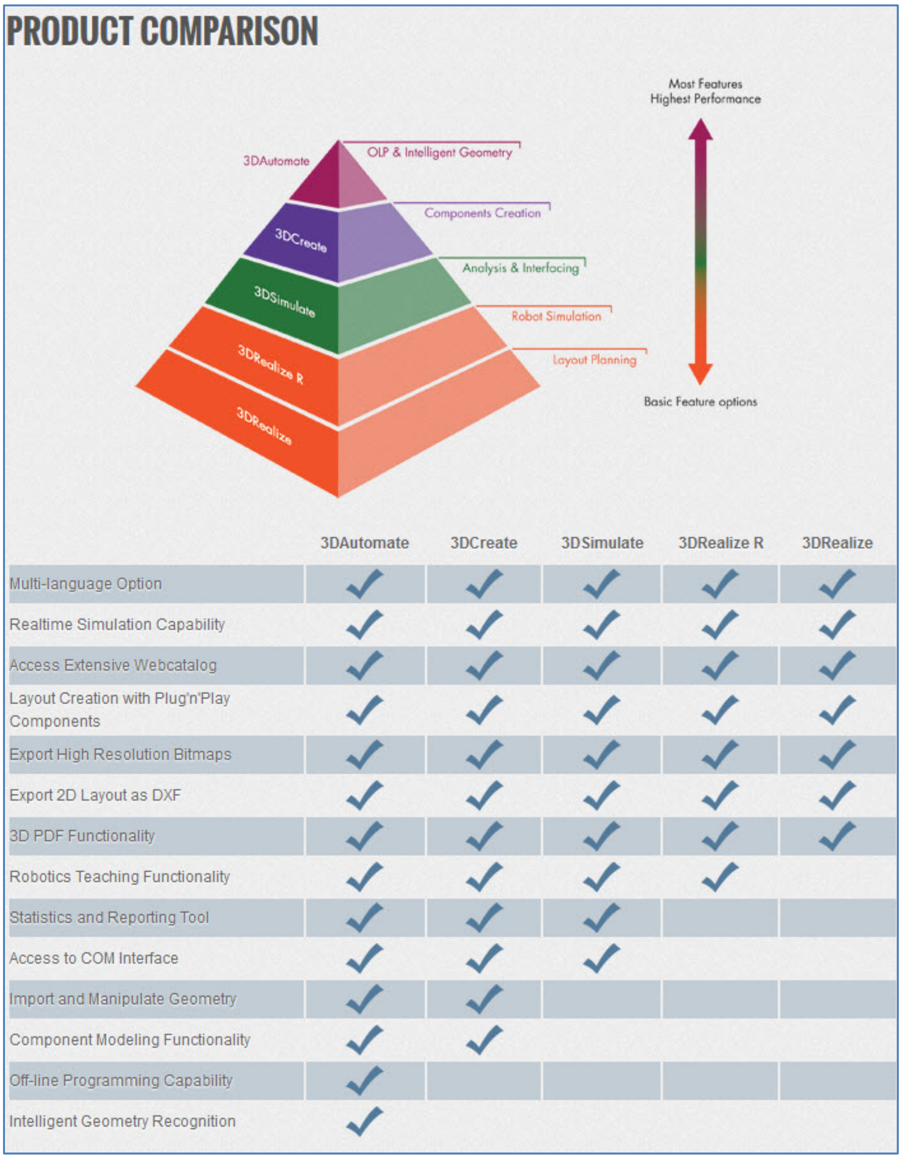
[62] B. Pavol and K. Marek, “Path correction algorithm for spot welding robot in body-in-white applications,” Strbske Pleso, 2013.

[63] C. Xing, C.-H. Ling and K. Bo-Seon, “A Robot Simulation System Basing on AutoLisp,” in *IEEE Conference on Industrial Electronics and Application*, Harbin, 2007.

APPENDICES

A. Introduction to Visual Components product family

Visual components product family consists of 5 levels of software application. The product family has been designed to meet the requirement of different user groups. These are 3D Automate, 3D Create, 3D Simulate, 3D Realize R and 3D Realize. The idea of the different levels of the product is to set right function for the different user groups and making it conveniently accessible to different level of users.



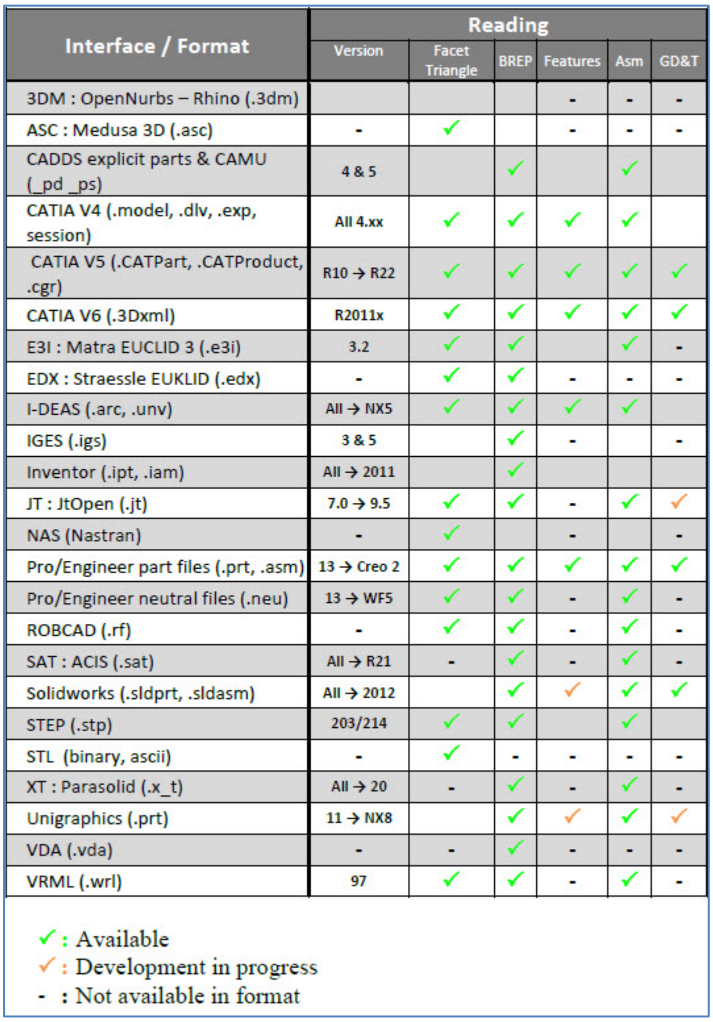
	3DAutomate	3DCreate	3DSimulate	3DRealize R	3DRealize
Multi-language Option	✓	✓	✓	✓	✓
Realtime Simulation Capability	✓	✓	✓	✓	✓
Access Extensive Webcatalog	✓	✓	✓	✓	✓
Layout Creation with Plug'n'Play Components	✓	✓	✓	✓	✓
Export High Resolution Bitmaps	✓	✓	✓	✓	✓
Export 2D Layout as DXF	✓	✓	✓	✓	✓
3D PDF Functionality	✓	✓	✓	✓	✓
Robotics Teaching Functionality	✓	✓	✓	✓	
Statistics and Reporting Tool	✓	✓	✓		
Access to COM Interface	✓	✓	✓		
Import and Manipulate Geometry	✓	✓			
Component Modeling Functionality	✓	✓			
Off-line Programming Capability	✓				
Intelligent Geometry Recognition	✓				

Figure A-1 Visual Components products comparison

a. Multi-Language Options: Currently in 2014 version of all the products on Visual Components there are 2 language options – English and German. In the next generation product (NextGen) of visual components there are going to be more language options – Japanese, Chinese, Spanish and more.

b. Real-time Simulation Capability: In Real-time simulation the 3D models in the software runs at the same rate as actual physical system. For example if a 10 meter conveyor takes 2min 2sec to transport a box from one end to another end, the simulation of the same in visual components software would take 2min 2sec as well.
c. Access Extensive Webcatalog: In the Web eCatalog of Visual Componets there is a huge collection of robots up to 25 different robot manufacturers, different types of conveyors, Grippers, Machine Tools, Template components, Feeders, Assembly components, ASRS components etc.
d. Layout Creation with Plug'n'Play Components: There are different types of managers (robot manager, resource manager), controllers, resources (human, machine, transport), components (feeders, conveyors etc.). They are very useful to quickly build simple working layout and robotic workcells.
e. Export High Resolution Bitmaps: By using this tool screenshots of the components or layout can be easily captured at any angle. They are very handy to use in marketing and documentation purposes.
f. Export 2D layout as DXF: DXF mean Drawing Exchange Format which was developed by Autodesk to enable file exchange between AutoCAD and other CAD software. Many times it is also required to represent the Visual Components layout in 2D view in AutoCAD to demonstrate the process flow and sequences in a simple way. So any 3D layout can be translated to the *.dxf format by the conversion tool.
g. 3D-PDF & Video Recording functionality: It enables the recording of simulation in 3D pdf format which improved the memory consumption issue to a great extent. Also there are various video recording formats into which the simulation can be recorded.
h. Robotics Teaching Functionality: Robot movements can be easily teached in RSL format. RSL stands for Robot Scripting Language. This tool is available starting from 3D Realize R®.
i. Statistics and Reporting Tool: Visual Components offers very useful tools for statistical analysis and reporting. It can produce Gantt chart, line/bar chart for production rate, production WIP, storage utilization etc. Also each individual state of a working robot like busy, idle, parts count, utilization, interval utilization etc. can be represented by Line/Bar chart.
j. Access to COM interface: Starting from 3D Simulate to upper pyramid access to COM interface is provided. With COM (Component Object Model) interfaces makes customization and application extensions possible. By using COM many of the Visual Components partners have been able to develop their own customized application (e.g. external controller of robots) on top of 3D Create® or 3D Automate®.

k. Import and Manipulate Geometry: Visual Component offers wide variety of CAD readers from different CAD software. The files can be imported with their original hierarchical tree structure or as a collapsed geometry block. It can utilize a wide variety of existing CAD geometry to present an entire product family with one lightweight simulation model that uses parameters to modify geometry appearance and behaviour.CAD formats which can currently be imported to 3D Create and 3D Automate are as following:



Interface / Format	Reading					
	Version	Facet Triangle	BREP	Features	Asm	GD&T
3DM : OpenNurbs – Rhino (.3dm)				-	-	-
ASC : Medusa 3D (.asc)	-	✓		-	-	-
CADDS explicit parts & CAMU (_pd _ps)	4 & 5		✓		✓	
CATIA V4 (.model, .dlv, .exp, session)	All 4.xx	✓	✓	✓	✓	
CATIA V5 (.CATPart, .CATProduct, .cgr)	R10 → R22	✓	✓	✓	✓	✓
CATIA V6 (.3Dxml)	R2011x	✓	✓	✓	✓	✓
E3I : Matra EUCLID 3 (.e3i)	3.2	✓	✓		✓	-
EDX : Straessle EUKLID (.edx)	-	✓	✓	-	-	-
I-DEAS (.arc, .unv)	All → NX5	✓	✓	✓	✓	
IGES (.igs)	3 & 5		✓	-		-
Inventor (.ipt, .iam)	All → 2011		✓			
JT : JtOpen (.jt)	7.0 → 9.5	✓	✓	-	✓	✔
NAS (Nastran)	-	✓		-		-
Pro/Engineer part files (.prt, .asm)	13 → Creo 2	✓	✓	✓	✓	✓
Pro/Engineer neutral files (.neu)	13 → WF5	✓	✓	-	✓	-
ROBCAD (.rf)	-	✓	✓	-	✓	-
SAT : ACIS (.sat)	All → R21	-	✓	-	✓	-
Solidworks (.sldprt, .sldasm)	All → 2012		✓	✔	✓	✓
STEP (.stp)	203/214	✓	✓		✓	
STL (binary, ascii)	-	✓	-	-	-	-
XT : Parasolid (.x_t)	All → 20	-	✓	-	✓	-
Unigraphics (.prt)	11 → NX8		✓	✔	✓	✔
VDA (.vda)	-	-	✓	-	-	-
VRML (.wrl)	97	✓	✓	-	✓	-

✓ : Available
✔ : Development in progress
- : Not available in format

Figure A-2 CAD formats eligible with 3D Automate

l. Component Modelling Functionality: Geometry feature can be created and modified with this functionality. There are basic geometry features (block, cylinder, cone, wedge etc.) and other features (e.g. curve) and complicated surfaces can be created by python API. Also the modification options e.g. transform, cloning, mirror, revolve, extrude etc. makes it possible to modify any created or imported geometry and make completely parametric components as per design requirement.

m. Off-line Programming Capability: 3D Automate® has offline programming capability. So the created RSL programs can be translated to various robot specific languages (Kuka, Staubli, Mitsubishi, Motoman, Nachi etc.) using the post processor. Also new post processor's can be developed as per customer requirement.

n. Intelligent Geometry Recognition: This feature is only available in 3D Automate®. It has mathematical representation and surface recognition capability which aids in more complicated offline programming. 3D Automate® has topology. Topology is a branch of mathematics concerned with spatial properties preserved under continuous deformation (stretching without tearing or gluing); these properties are the topological invariants.

3D Automate®

In my thesis work I have worked on 3D Automate as it has more versatile feature for topology and offline programming. Also it has geometry optimization capability, which was essential to import large size CAD files; the tessellation reduction feature reduces the file size to a great extent. After geometry optimization the plug and play interfaces and behaviours are added to the components. These components are then attached to each other according to process sequence to make the complete layout. 3D Automate also has 2 types of API's, a. COM and b. Python. This enables development, customization and application extension of the software possible.

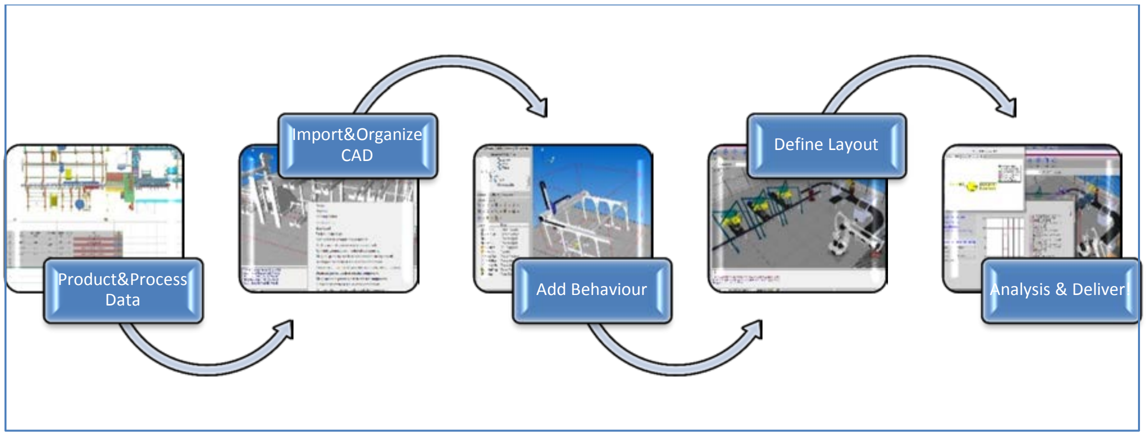


Figure A-3 Simulation Layout Creation Workflow

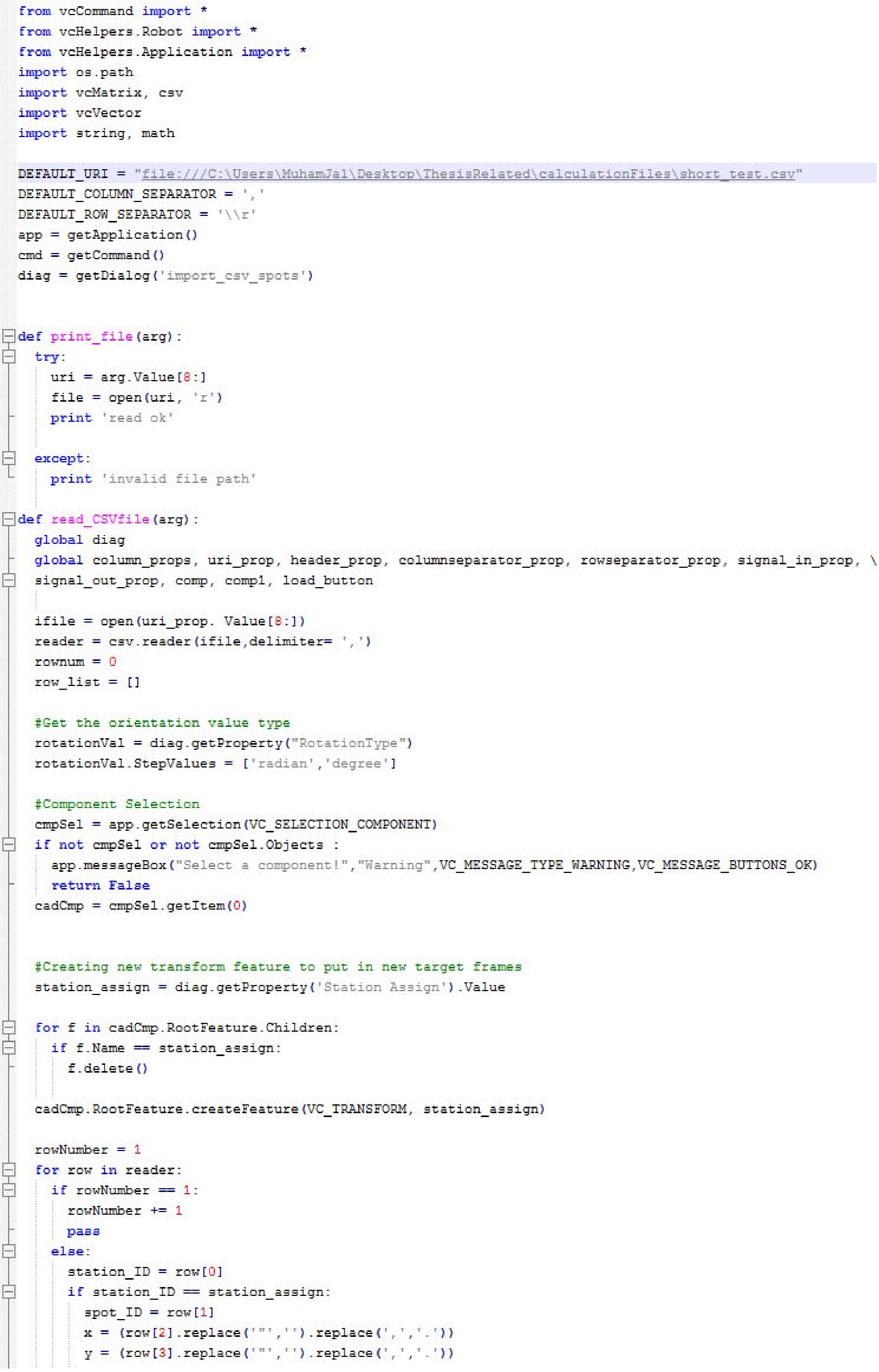
B. Spot Weld Data in an excel file generated by a macro from a CATIA CAD model.

ASSEMBLY:	Test_Product_CECAB1-0001_123															
CREATED:	25.05.2012															
												PART INFORMATION				
POINT INFORMATION												PART			PART	
ID	X	Y	Z	I	J	K	PTY	ROB	NTH	SPHERE	FEATURE LOC.	NAME	GAUGE	MATERIAL	NAME	GAUGE
1001-0001	-37,993	12,000	10,713	-0,686	0	-0,727	Resistance	Ordinary	2	3	PIA	Test_Part_2_CECAB-0004_123	2mm	MSW 1200	Test_Part_4_CECAB-0004_123	2.8mm
1001-0002-L	-42,025	4,000	14,517	-0,686	0	-0,727	Projection	Safety	3	3	OEM	Test_Part_3_CECAB-0004_123	1.4mm	DC03	Test_Part_4_CECAB-0004_123	2.8mm
1001-0003-R	-46,056	-4,000	18,322	-0,686	0	-0,727	Rivet	Ordinary	2	3	PIA	Test_Part_CECAB-0004_123	1.2mm	DC04	Test_Part_2_CECAB-0004_123	2mm
1001-0004-L/R	-50,087	-12,000	22,126	-0,686	0	-0,727	Other	Ordinary	4	3	PIA	Test_Part_4_CECAB-0004_123	2.8mm	DC03	Test_Part_CECAB-0004_123	1.2mm
1001-0005	-7,307	8,276	2,000						0							
1001-0006	-3,507	-3,125	2,000						0							
1001-0007	-14,908	-6,925	2,000						0							
1001-0008	-18,709	4,476	2,000						0							

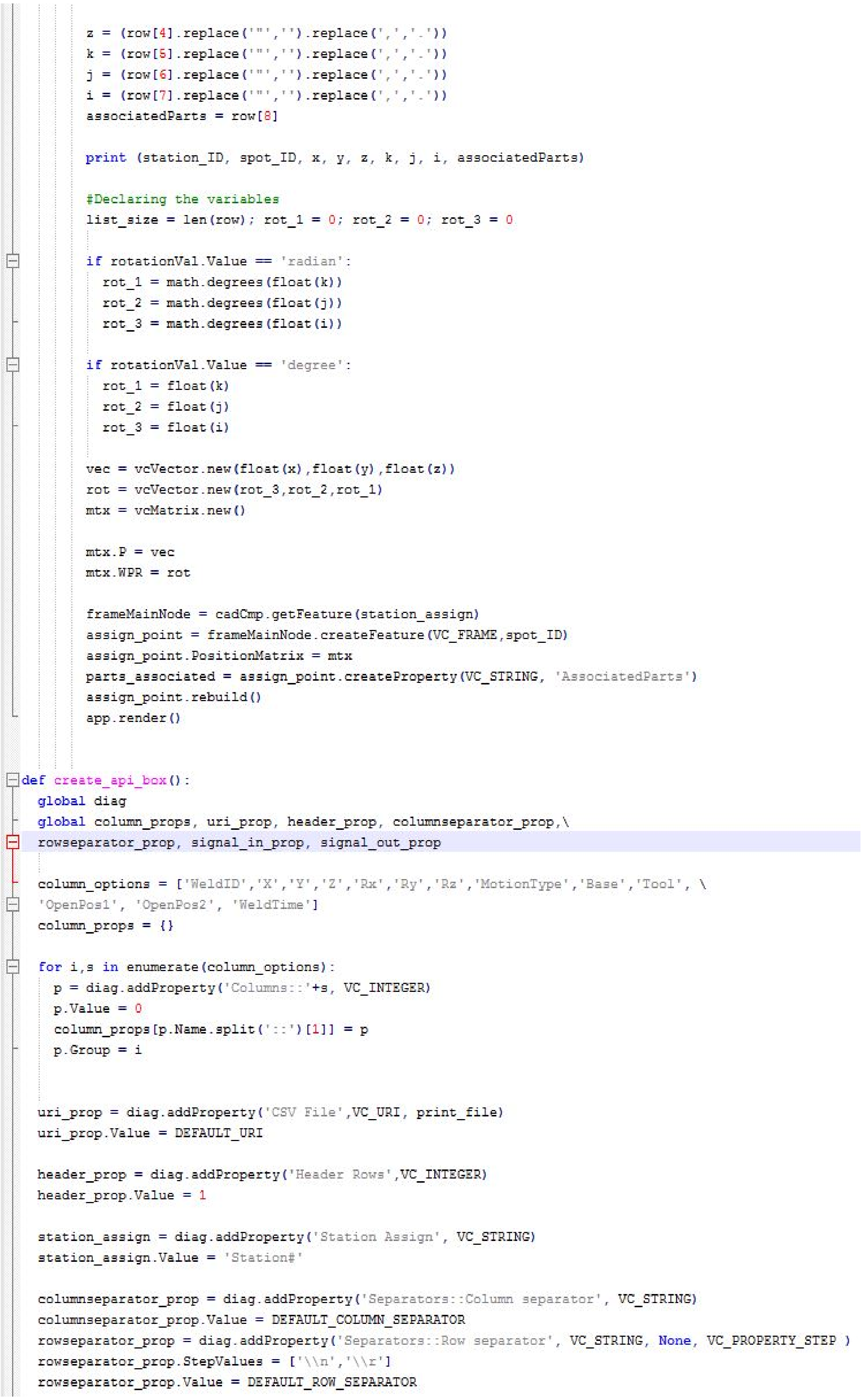
				MATERIAL	DC03	DC03	MSW 1200	DC04				
			PART	NAME		Test_Part_2_CECAB-0004_123		Test_Part_2_CECAB-0004_123				
				GAUGE		2mm		2mm				
				MATERIAL		MSW 1200		MSW 1200				
			PART	NAME				Test_Part_3_CECAB-0004_123				
				GAUGE				1.4mm				
				MATERIAL				DC03				
				STACK	4.8mm	6.2mm	3.2mm	7.4mm	0mm	0mm	0mm	0mm
				Weld Type	Spot	Spot	Spot	Spot	Spot	Spot	Spot	Spot

Table B-1 Excel file containing spot welding information

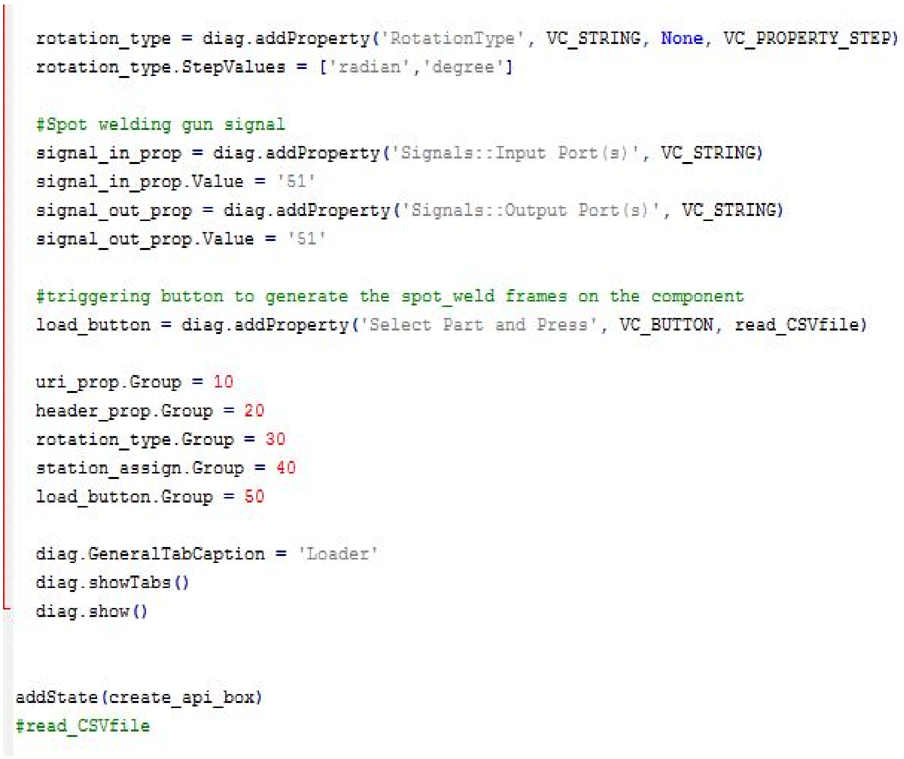
C. Python program of the AddOn Tool [Spot Welding] > [import_csv_spot]



```python
from vcCommand import *
from vcHelpers.Robot import *
from vcHelpers.Application import *
import os.path
import vcMatrix, csv
import vcVector
import string, math

DEFAULT_URI = "file:///C:\Users\MuhamJa1\Desktop\ThesisRelated\calculationFiles\short_test.csv"
DEFAULT_COLUMN_SEPARATOR = ','
DEFAULT_ROW_SEPARATOR = '\\r'
app = getApplication()
cmd = getCommand()
diag = getDialog('import_csv_spots')


def print_file(arg):
  try:
    uri = arg.Value[8:]
    file = open(uri, 'r')
    print 'read ok'

  except:
    print 'invalid file path'

def read_CSVfile(arg):
  global diag
  global column_props, uri_prop, header_prop, columnseparator_prop, rowseparator_prop, signal_in_prop, \
  signal_out_prop, comp, comp1, load_button

  ifile = open(uri_prop. Value[8:])
  reader = csv.reader(ifile,delimiter= ',')
  rownum = 0
  row_list = []

  #Get the orientation value type
  rotationVal = diag.getProperty("RotationType")
  rotationVal.StepValues = ['radian','degree']

  #Component Selection
  cmpSel = app.getSelection(VC_SELECTION_COMPONENT)
  if not cmpSel or not cmpSel.Objects :
    app.messageBox("Select a component!","Warning",VC_MESSAGE_TYPE_WARNING,VC_MESSAGE_BUTTONS_OK)
    return False
  cadCmp = cmpSel.getItem(0)


  #Creating new transform feature to put in new target frames
  station_assign = diag.getProperty('Station Assign').Value

  for f in cadCmp.RootFeature.Children:
    if f.Name == station_assign:
      f.delete()

  cadCmp.RootFeature.createFeature(VC_TRANSFORM, station_assign)

  rowNumber = 1
  for row in reader:
    if rowNumber == 1:
      rowNumber += 1
      pass
    else:
      station_ID = row[0]
      if station_ID == station_assign:
        spot_ID = row[1]
        x = (row[2].replace('"','').replace(',','.'))
        y = (row[3].replace('"','').replace(',','.'))
```

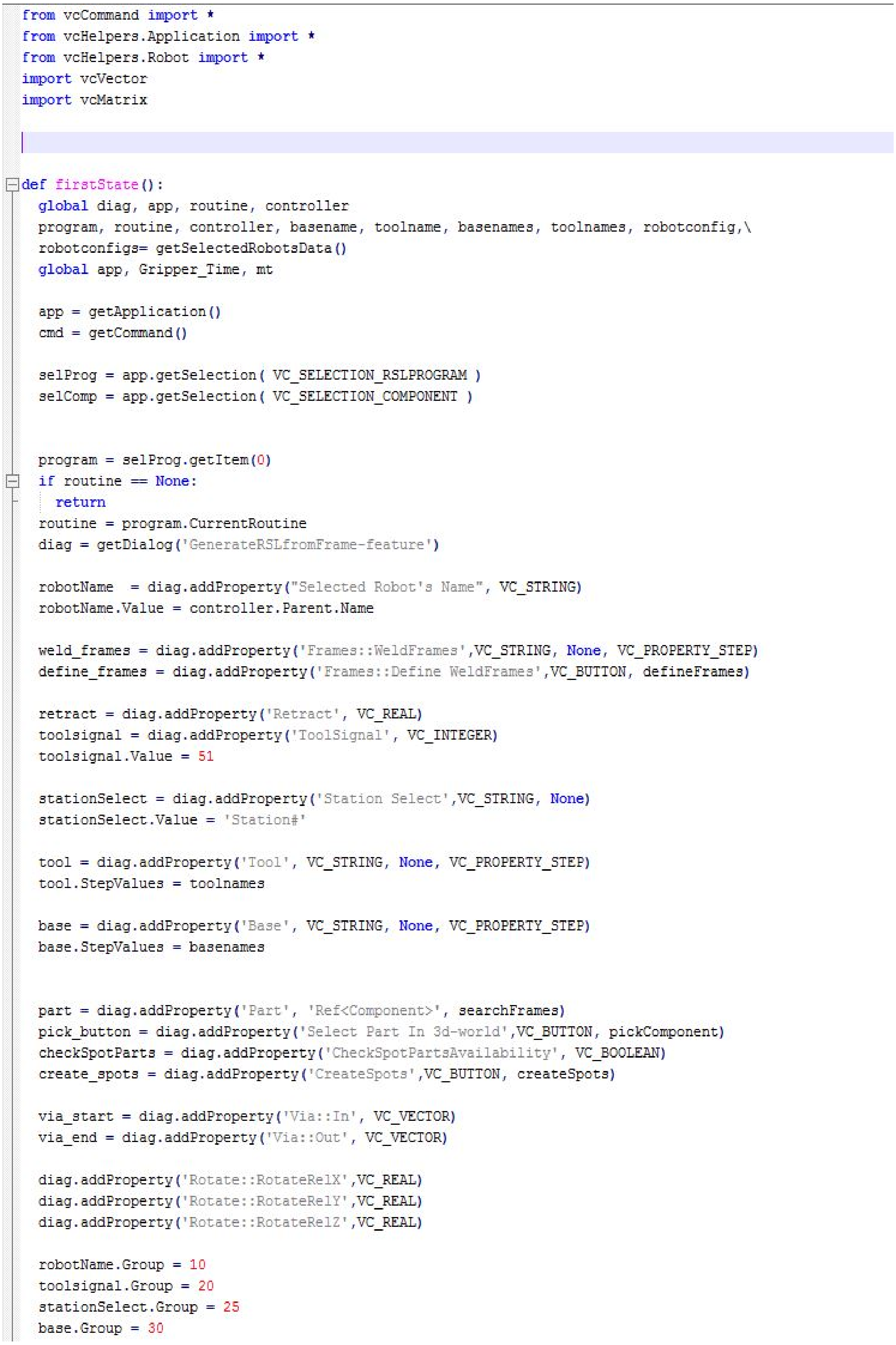


```
        z = (row[4].replace('"','').replace(',','.'))
        k = (row[5].replace('"','').replace(',','.'))
        j = (row[6].replace('"','').replace(',','.'))
        i = (row[7].replace('"','').replace(',','.'))
        associatedParts = row[8]

        print (station_ID, spot_ID, x, y, z, k, j, i, associatedParts)

        #Declaring the variables
        list_size = len(row); rot_1 = 0; rot_2 = 0; rot_3 = 0

        if rotationVal.Value == 'radian':
          rot_1 = math.degrees(float(k))
          rot_2 = math.degrees(float(j))
          rot_3 = math.degrees(float(i))

        if rotationVal.Value == 'degree':
          rot_1 = float(k)
          rot_2 = float(j)
          rot_3 = float(i)

        vec = vcVector.new(float(x),float(y),float(z))
        rot = vcVector.new(rot_3,rot_2,rot_1)
        mtx = vcMatrix.new()

        mtx.P = vec
        mtx.WPR = rot

        frameMainNode = cadCmp.getFeature(station_assign)
        assign_point = frameMainNode.createFeature(VC_FRAME,spot_ID)
        assign_point.PositionMatrix = mtx
        parts_associated = assign_point.createProperty(VC_STRING, 'AssociatedParts')
        assign_point.rebuild()
        app.render()


def create_api_box():
  global diag
  global column_props, uri_prop, header_prop, columnseparator_prop,\
  rowseparator_prop, signal_in_prop, signal_out_prop

  column_options = ['WeldID','X','Y','Z','Rx','Ry','Rz','MotionType','Base','Tool', \
  'OpenPos1', 'OpenPos2', 'WeldTime']
  column_props = {}

  for i,s in enumerate(column_options):
    p = diag.addProperty('Columns::'+s, VC_INTEGER)
    p.Value = 0
    column_props[p.Name.split('::')[1]] = p
    p.Group = i


  uri_prop = diag.addProperty('CSV File',VC_URI, print_file)
  uri_prop.Value = DEFAULT_URI

  header_prop = diag.addProperty('Header Rows',VC_INTEGER)
  header_prop.Value = 1

  station_assign = diag.addProperty('Station Assign', VC_STRING)
  station_assign.Value = 'Station#'

  columnseparator_prop = diag.addProperty('Separators::Column separator', VC_STRING)
  columnseparator_prop.Value = DEFAULT_COLUMN_SEPARATOR
  rowseparator_prop = diag.addProperty('Separators::Row separator', VC_STRING, None, VC_PROPERTY_STEP )
  rowseparator_prop.StepValues = ['\\n','\\r']
  rowseparator_prop.Value = DEFAULT_ROW_SEPARATOR
```

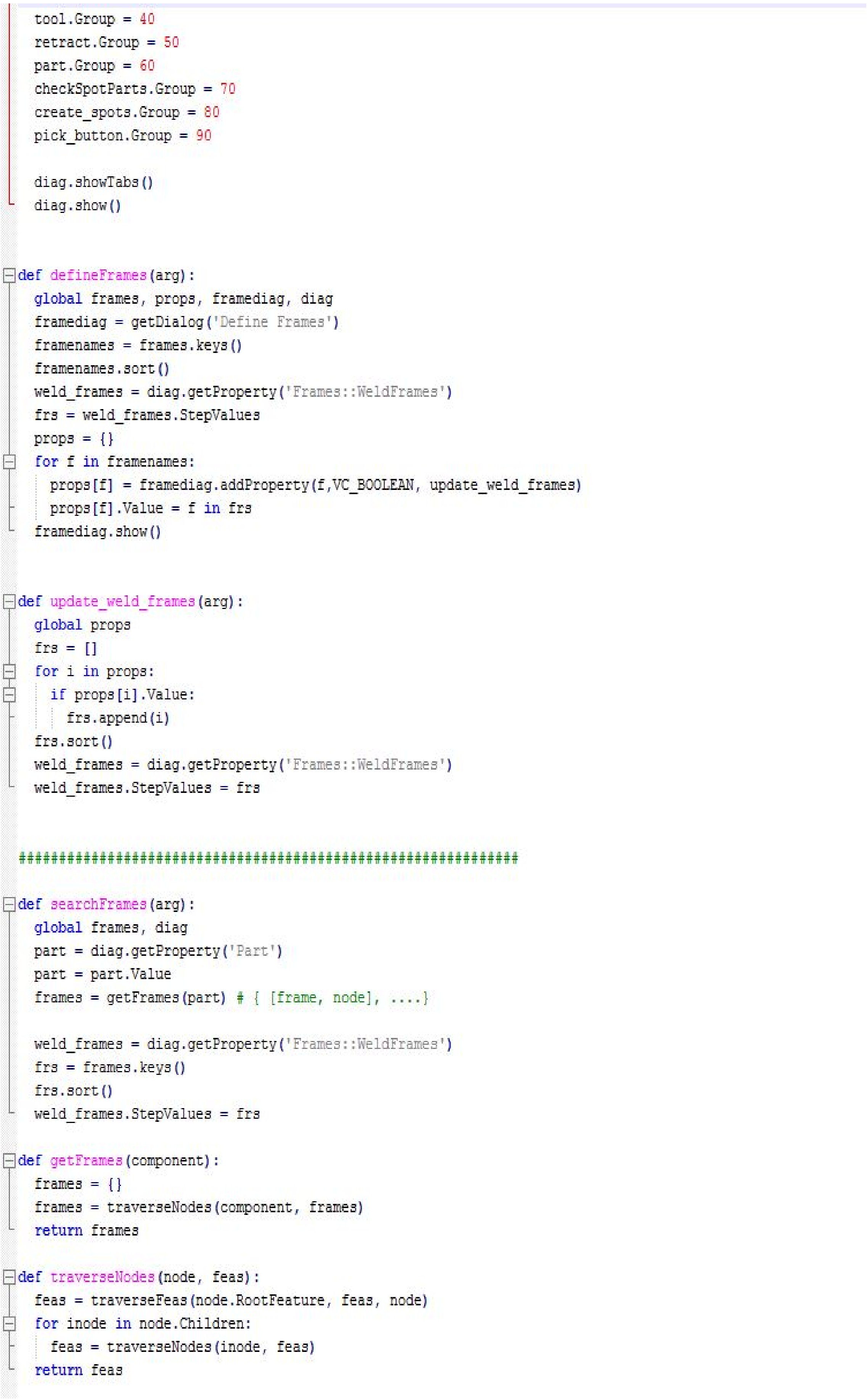


```
  rotation_type = diag.addProperty('RotationType', VC_STRING, None, VC_PROPERTY_STEP)
  rotation_type.StepValues = ['radian','degree']

  #Spot welding gun signal
  signal_in_prop = diag.addProperty('Signals::Input Port(s)', VC_STRING)
  signal_in_prop.Value = '51'
  signal_out_prop = diag.addProperty('Signals::Output Port(s)', VC_STRING)
  signal_out_prop.Value = '51'

  #triggering button to generate the spot_weld frames on the component
  load_button = diag.addProperty('Select Part and Press', VC_BUTTON, read_CSVfile)

  uri_prop.Group = 10
  header_prop.Group = 20
  rotation_type.Group = 30
  station_assign.Group = 40
  load_button.Group = 50

  diag.GeneralTabCaption = 'Loader'
  diag.showTabs()
  diag.show()


addState(create_api_box)
#read_CSVfile
```

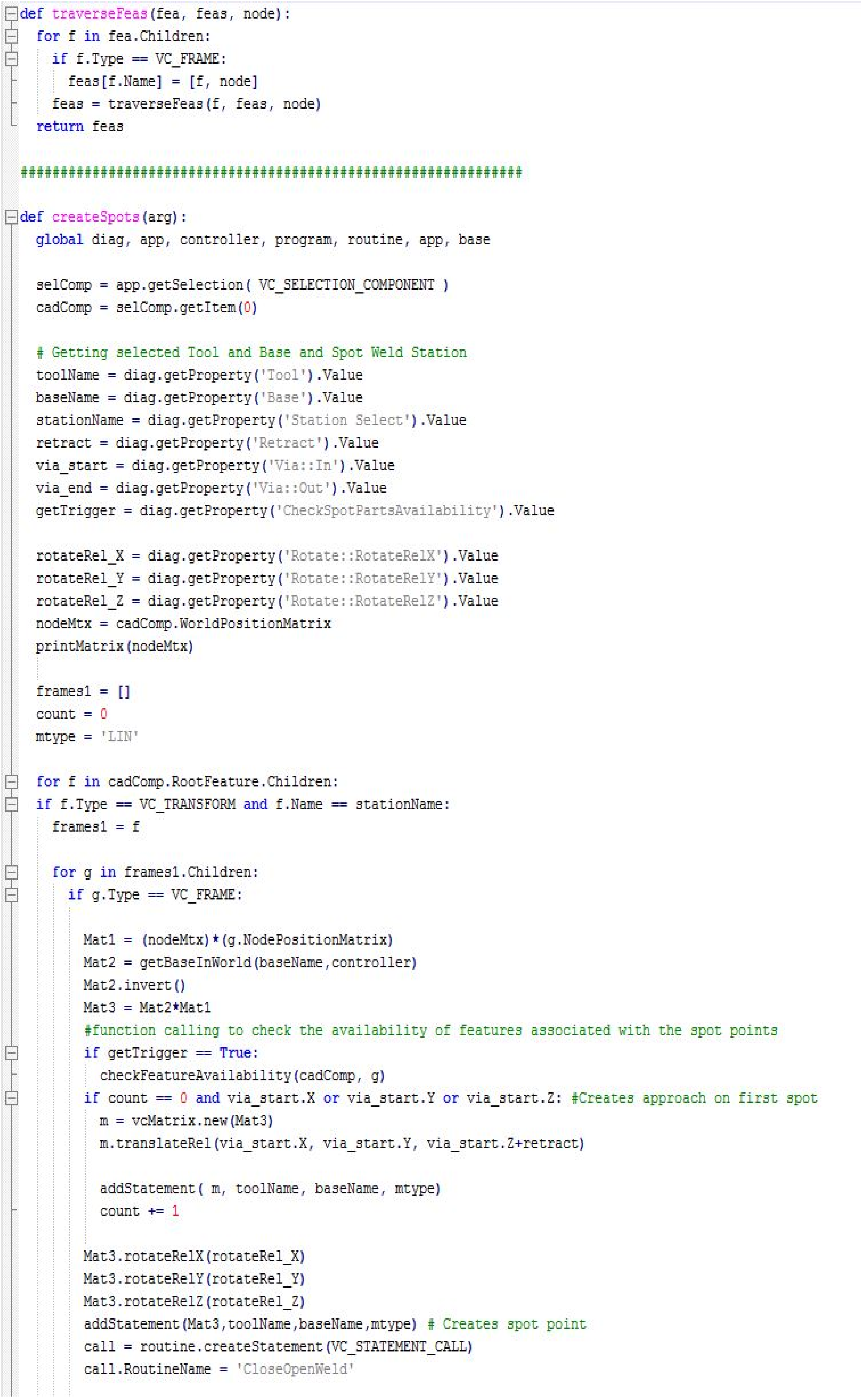
D. Python program for the AddOn Tool [SpotWelding]>[generateRSLfrom-Frame-Feature]



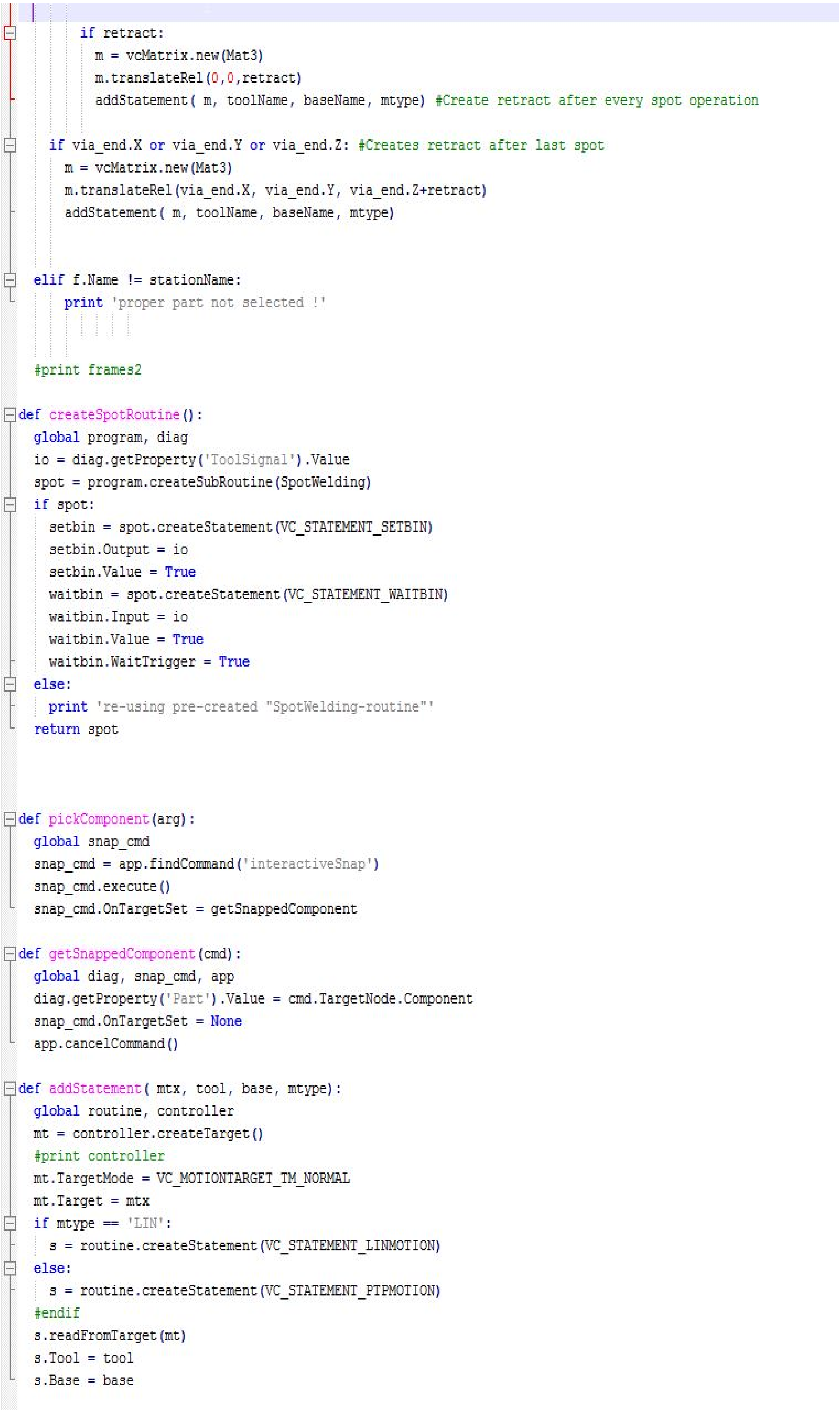
```
from vcCommand import *
from vcHelpers.Application import *
from vcHelpers.Robot import *
import vcVector
import vcMatrix


def firstState():
  global diag, app, routine, controller
  program, routine, controller, basename, toolname, basenames, toolnames, robotconfig,\
  robotconfigs= getSelectedRobotsData()
  global app, Gripper_Time, mt

  app = getApplication()
  cmd = getCommand()

  selProg = app.getSelection( VC_SELECTION_RSLPROGRAM )
  selComp = app.getSelection( VC_SELECTION_COMPONENT )


  program = selProg.getItem(0)
  if routine == None:
    return
  routine = program.CurrentRoutine
  diag = getDialog('GenerateRSLfromFrame-feature')

  robotName  = diag.addProperty("Selected Robot's Name", VC_STRING)
  robotName.Value = controller.Parent.Name

  weld_frames = diag.addProperty('Frames::WeldFrames',VC_STRING, None, VC_PROPERTY_STEP)
  define_frames = diag.addProperty('Frames::Define WeldFrames',VC_BUTTON, defineFrames)

  retract = diag.addProperty('Retract', VC_REAL)
  toolsignal = diag.addProperty('ToolSignal', VC_INTEGER)
  toolsignal.Value = 51

  stationSelect = diag.addProperty('Station Select',VC_STRING, None)
  stationSelect.Value = 'Station#'

  tool = diag.addProperty('Tool', VC_STRING, None, VC_PROPERTY_STEP)
  tool.StepValues = toolnames

  base = diag.addProperty('Base', VC_STRING, None, VC_PROPERTY_STEP)
  base.StepValues = basenames


  part = diag.addProperty('Part', 'Ref<Component>', searchFrames)
  pick_button = diag.addProperty('Select Part In 3d-world',VC_BUTTON, pickComponent)
  checkSpotParts = diag.addProperty('CheckSpotPartsAvailability', VC_BOOLEAN)
  create_spots = diag.addProperty('CreateSpots',VC_BUTTON, createSpots)

  via_start = diag.addProperty('Via::In', VC_VECTOR)
  via_end = diag.addProperty('Via::Out', VC_VECTOR)

  diag.addProperty('Rotate::RotateRelX',VC_REAL)
  diag.addProperty('Rotate::RotateRelY',VC_REAL)
  diag.addProperty('Rotate::RotateRelZ',VC_REAL)

  robotName.Group = 10
  toolsignal.Group = 20
  stationSelect.Group = 25
  base.Group = 30
```



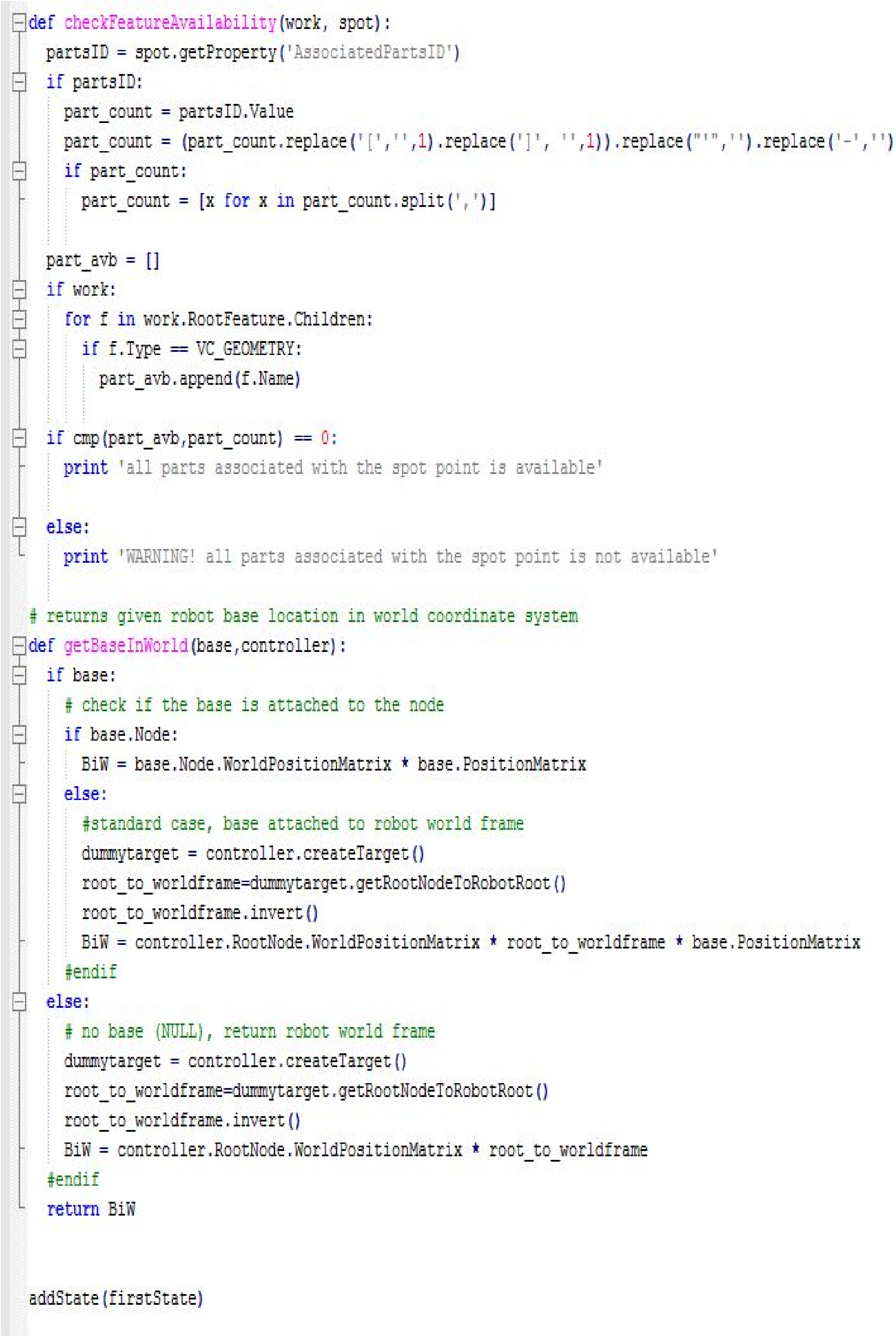
```
  tool.Group = 40
  retract.Group = 50
  part.Group = 60
  checkSpotParts.Group = 70
  create_spots.Group = 80
  pick_button.Group = 90

  diag.showTabs()
  diag.show()


def defineFrames(arg):
  global frames, props, framediag, diag
  framediag = getDialog('Define Frames')
  framenames = frames.keys()
  framenames.sort()
  weld_frames = diag.getProperty('Frames::WeldFrames')
  frs = weld_frames.StepValues
  props = {}
  for f in framenames:
    props[f] = framediag.addProperty(f,VC_BOOLEAN, update_weld_frames)
    props[f].Value = f in frs
  framediag.show()


def update_weld_frames(arg):
  global props
  frs = []
  for i in props:
    if props[i].Value:
      frs.append(i)
  frs.sort()
  weld_frames = diag.getProperty('Frames::WeldFrames')
  weld_frames.StepValues = frs


##############################################################

def searchFrames(arg):
  global frames, diag
  part = diag.getProperty('Part')
  part = part.Value
  frames = getFrames(part) # { [frame, node], ....}

  weld_frames = diag.getProperty('Frames::WeldFrames')
  frs = frames.keys()
  frs.sort()
  weld_frames.StepValues = frs

def getFrames(component):
  frames = {}
  frames = traverseNodes(component, frames)
  return frames

def traverseNodes(node, feas):
  feas = traverseFeas(node.RootFeature, feas, node)
  for inode in node.Children:
    feas = traverseNodes(inode, feas)
  return feas
```



```
def traverseFeas(fea, feas, node):
  for f in fea.Children:
    if f.Type == VC_FRAME:
      feas[f.Name] = [f, node]
    feas = traverseFeas(f, feas, node)
  return feas

###############################################################

def createSpots(arg):
  global diag, app, controller, program, routine, app, base

  selComp = app.getSelection( VC_SELECTION_COMPONENT )
  cadComp = selComp.getItem(0)

  # Getting selected Tool and Base and Spot Weld Station
  toolName = diag.getProperty('Tool').Value
  baseName = diag.getProperty('Base').Value
  stationName = diag.getProperty('Station Select').Value
  retract = diag.getProperty('Retract').Value
  via_start = diag.getProperty('Via::In').Value
  via_end = diag.getProperty('Via::Out').Value
  getTrigger = diag.getProperty('CheckSpotPartsAvailability').Value

  rotateRel_X = diag.getProperty('Rotate::RotateRelX').Value
  rotateRel_Y = diag.getProperty('Rotate::RotateRelY').Value
  rotateRel_Z = diag.getProperty('Rotate::RotateRelZ').Value
  nodeMtx = cadComp.WorldPositionMatrix
  printMatrix(nodeMtx)

  frames1 = []
  count = 0
  mtype = 'LIN'

  for f in cadComp.RootFeature.Children:
  if f.Type == VC_TRANSFORM and f.Name == stationName:
    frames1 = f

    for g in frames1.Children:
      if g.Type == VC_FRAME:

        Mat1 = (nodeMtx)*(g.NodePositionMatrix)
        Mat2 = getBaseInWorld(baseName,controller)
        Mat2.invert()
        Mat3 = Mat2*Mat1
        #function calling to check the availability of features associated with the spot points
        if getTrigger == True:
          checkFeatureAvailability(cadComp, g)
        if count == 0 and via_start.X or via_start.Y or via_start.Z: #Creates approach on first spot
          m = vcMatrix.new(Mat3)
          m.translateRel(via_start.X, via_start.Y, via_start.Z+retract)

          addStatement( m, toolName, baseName, mtype)
          count += 1

        Mat3.rotateRelX(rotateRel_X)
        Mat3.rotateRelY(rotateRel_Y)
        Mat3.rotateRelZ(rotateRel_Z)
        addStatement(Mat3,toolName,baseName,mtype) # Creates spot point
        call = routine.createStatement(VC_STATEMENT_CALL)
        call.RoutineName = 'CloseOpenWeld'
```

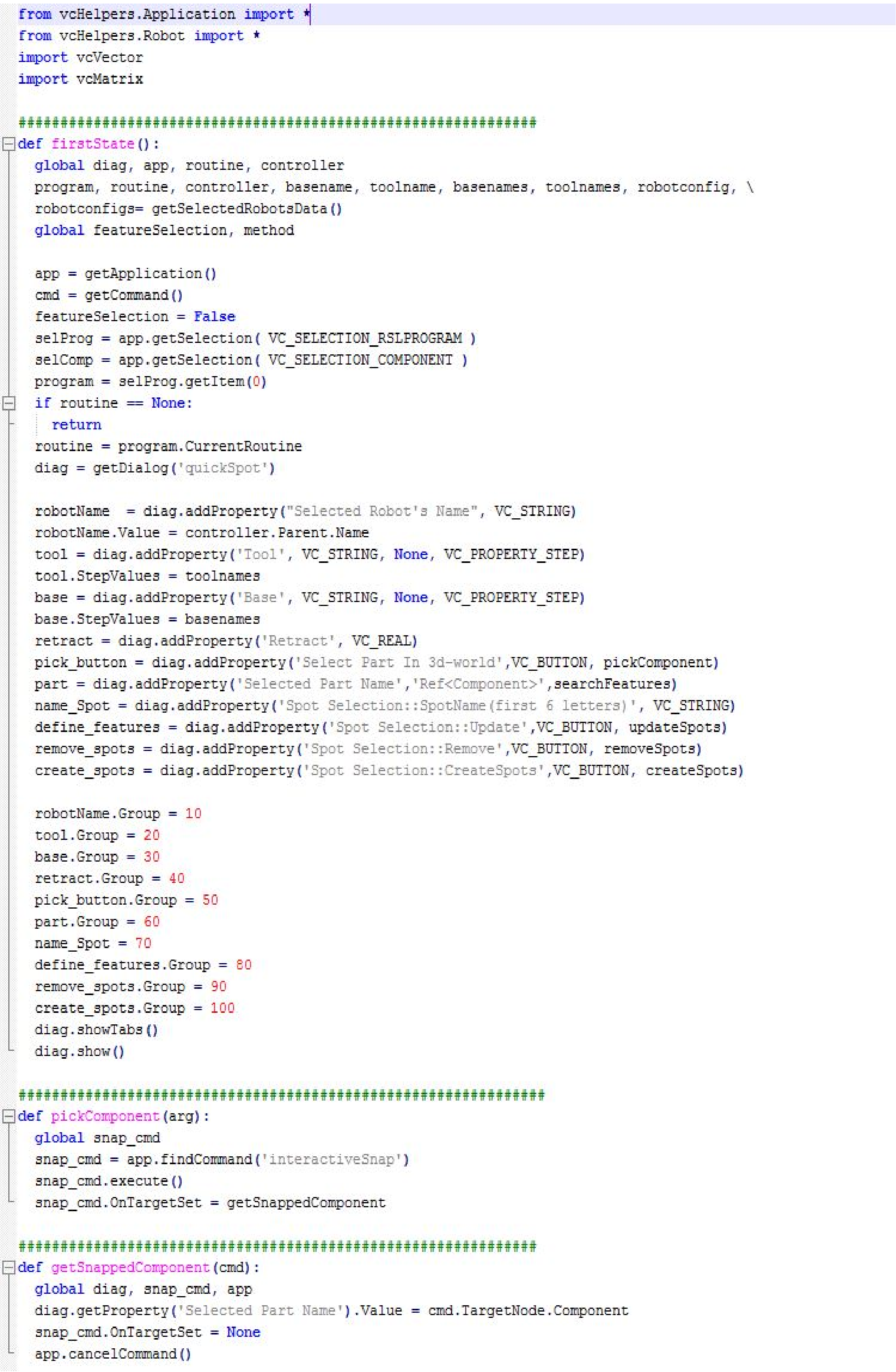


```
        if retract:
          m = vcMatrix.new(Mat3)
          m.translateRel(0,0,retract)
          addStatement( m, toolName, baseName, mtype) #Create retract after every spot operation

    if via_end.X or via_end.Y or via_end.Z: #Creates retract after last spot
      m = vcMatrix.new(Mat3)
      m.translateRel(via_end.X, via_end.Y, via_end.Z+retract)
      addStatement( m, toolName, baseName, mtype)


  elif f.Name != stationName:
      print 'proper part not selected !'


  #print frames2

def createSpotRoutine():
  global program, diag
  io = diag.getProperty('ToolSignal').Value
  spot = program.createSubRoutine(SpotWelding)
  if spot:
    setbin = spot.createStatement(VC_STATEMENT_SETBIN)
    setbin.Output = io
    setbin.Value = True
    waitbin = spot.createStatement(VC_STATEMENT_WAITBIN)
    waitbin.Input = io
    waitbin.Value = True
    waitbin.WaitTrigger = True
  else:
    print 're-using pre-created "SpotWelding-routine"'
  return spot



def pickComponent(arg):
  global snap_cmd
  snap_cmd = app.findCommand('interactiveSnap')
  snap_cmd.execute()
  snap_cmd.OnTargetSet = getSnappedComponent

def getSnappedComponent(cmd):
  global diag, snap_cmd, app
  diag.getProperty('Part').Value = cmd.TargetNode.Component
  snap_cmd.OnTargetSet = None
  app.cancelCommand()

def addStatement( mtx, tool, base, mtype):
  global routine, controller
  mt = controller.createTarget()
  #print controller
  mt.TargetMode = VC_MOTIONTARGET_TM_NORMAL
  mt.Target = mtx
  if mtype == 'LIN':
    s = routine.createStatement(VC_STATEMENT_LINMOTION)
  else:
    s = routine.createStatement(VC_STATEMENT_PTPMOTION)
  #endif
  s.readFromTarget(mt)
  s.Tool = tool
  s.Base = base
```

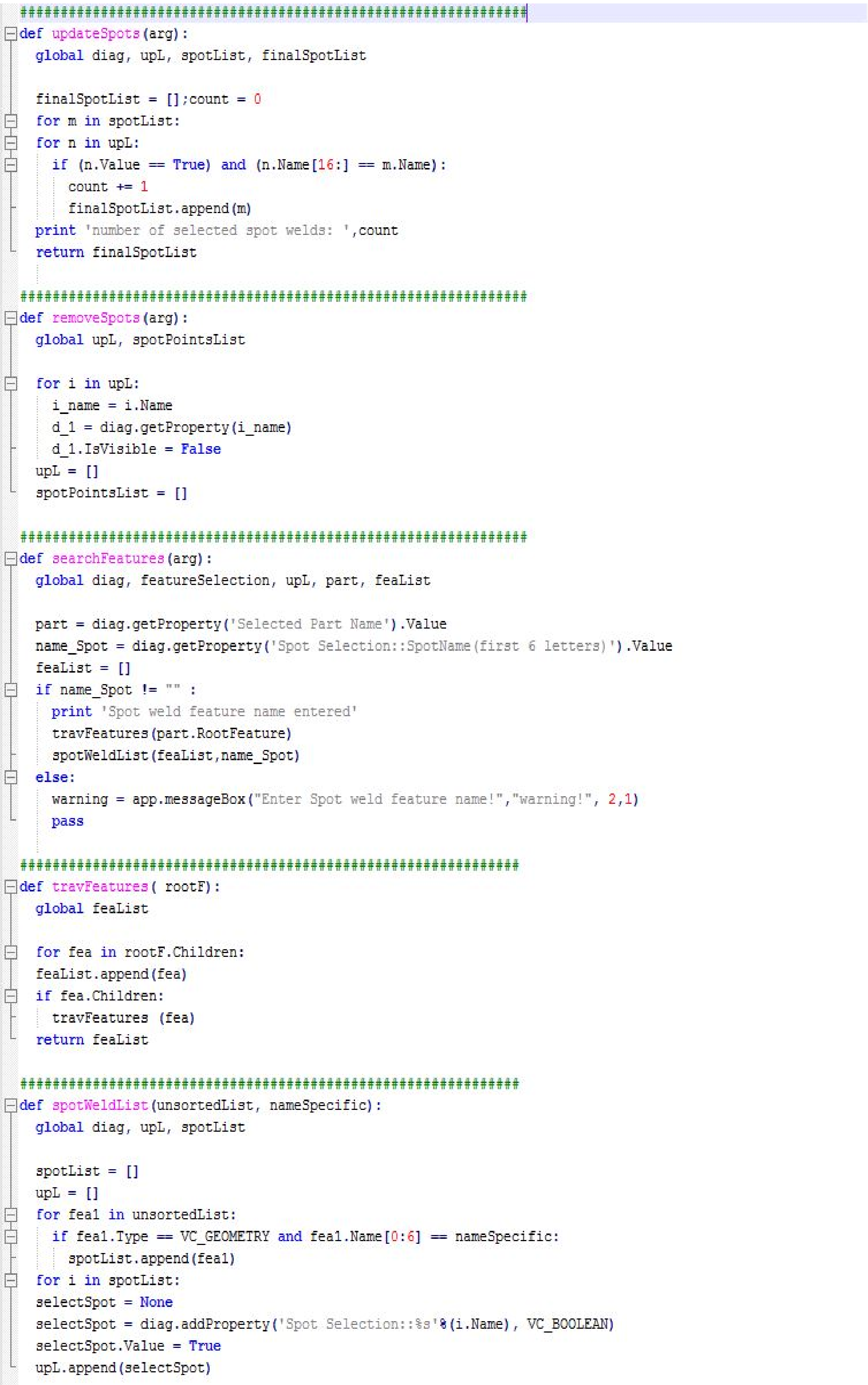


```
def checkFeatureAvailability(work, spot):
  partsID = spot.getProperty('AssociatedPartsID')
  if partsID:
    part_count = partsID.Value
    part_count = (part_count.replace('[','',1).replace(']', '',1)).replace("'",'').replace('-','')
    if part_count:
      part_count = [x for x in part_count.split(',')]

  part_avb = []
  if work:
    for f in work.RootFeature.Children:
      if f.Type == VC_GEOMETRY:
        part_avb.append(f.Name)

  if cmp(part_avb,part_count) == 0:
    print 'all parts associated with the spot point is available'

  else:
    print 'WARNING! all parts associated with the spot point is not available'

# returns given robot base location in world coordinate system
def getBaseInWorld(base,controller):
  if base:
    # check if the base is attached to the node
    if base.Node:
      BiW = base.Node.WorldPositionMatrix * base.PositionMatrix
    else:
      #standard case, base attached to robot world frame
      dummytarget = controller.createTarget()
      root_to_worldframe=dummytarget.getRootNodeToRobotRoot()
      root_to_worldframe.invert()
      BiW = controller.RootNode.WorldPositionMatrix * root_to_worldframe * base.PositionMatrix
    #endif
  else:
    # no base (NULL), return robot world frame
    dummytarget = controller.createTarget()
    root_to_worldframe=dummytarget.getRootNodeToRobotRoot()
    root_to_worldframe.invert()
    BiW = controller.RootNode.WorldPositionMatrix * root_to_worldframe
  #endif
  return BiW


addState(firstState)
```

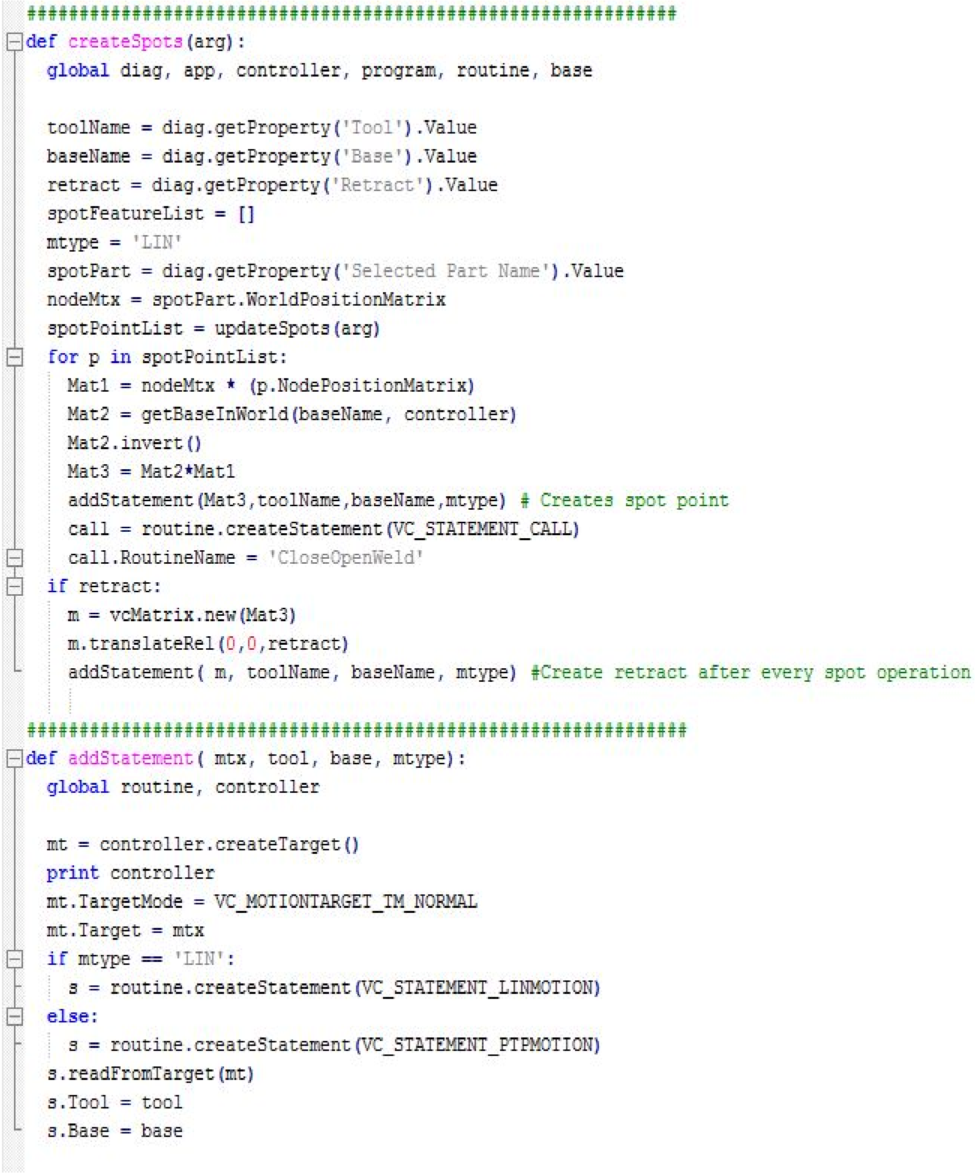
E. Python program for the AddOn Tool [SpotWelding]>[quickSpot]



```
from vcHelpers.Application import *
from vcHelpers.Robot import *
import vcVector
import vcMatrix

#################################################################
def firstState():
  global diag, app, routine, controller
  program, routine, controller, basename, toolname, basenames, toolnames, robotconfig, \
  robotconfigs= getSelectedRobotsData()
  global featureSelection, method

  app = getApplication()
  cmd = getCommand()
  featureSelection = False
  selProg = app.getSelection( VC_SELECTION_RSLPROGRAM )
  selComp = app.getSelection( VC_SELECTION_COMPONENT )
  program = selProg.getItem(0)
  if routine == None:
    return
  routine = program.CurrentRoutine
  diag = getDialog('quickSpot')

  robotName  = diag.addProperty("Selected Robot's Name", VC_STRING)
  robotName.Value = controller.Parent.Name
  tool = diag.addProperty('Tool', VC_STRING, None, VC_PROPERTY_STEP)
  tool.StepValues = toolnames
  base = diag.addProperty('Base', VC_STRING, None, VC_PROPERTY_STEP)
  base.StepValues = basenames
  retract = diag.addProperty('Retract', VC_REAL)
  pick_button = diag.addProperty('Select Part In 3d-world',VC_BUTTON, pickComponent)
  part = diag.addProperty('Selected Part Name','Ref<Component>',searchFeatures)
  name_Spot = diag.addProperty('Spot Selection::SpotName(first 6 letters)', VC_STRING)
  define_features = diag.addProperty('Spot Selection::Update',VC_BUTTON, updateSpots)
  remove_spots = diag.addProperty('Spot Selection::Remove',VC_BUTTON, removeSpots)
  create_spots = diag.addProperty('Spot Selection::CreateSpots',VC_BUTTON, createSpots)

  robotName.Group = 10
  tool.Group = 20
  base.Group = 30
  retract.Group = 40
  pick_button.Group = 50
  part.Group = 60
  name_Spot = 70
  define_features.Group = 80
  remove_spots.Group = 90
  create_spots.Group = 100
  diag.showTabs()
  diag.show()

##################################################################
def pickComponent(arg):
  global snap_cmd
  snap_cmd = app.findCommand('interactiveSnap')
  snap_cmd.execute()
  snap_cmd.OnTargetSet = getSnappedComponent

#################################################################
def getSnappedComponent(cmd):
  global diag, snap_cmd, app
  diag.getProperty('Selected Part Name').Value = cmd.TargetNode.Component
  snap_cmd.OnTargetSet = None
  app.cancelCommand()
```



```
###############################################################
def updateSpots(arg):
  global diag, upL, spotList, finalSpotList

  finalSpotList = [];count = 0
  for m in spotList:
  for n in upL:
    if (n.Value == True) and (n.Name[16:] == m.Name):
      count += 1
      finalSpotList.append(m)
  print 'number of selected spot welds: ',count
  return finalSpotList

###############################################################
def removeSpots(arg):
  global upL, spotPointsList

  for i in upL:
    i_name = i.Name
    d_1 = diag.getProperty(i_name)
    d_1.IsVisible = False
  upL = []
  spotPointsList = []

###############################################################
def searchFeatures(arg):
  global diag, featureSelection, upL, part, feaList

  part = diag.getProperty('Selected Part Name').Value
  name_Spot = diag.getProperty('Spot Selection::SpotName(first 6 letters)').Value
  feaList = []
  if name_Spot != "" :
    print 'Spot weld feature name entered'
    travFeatures(part.RootFeature)
    spotWeldList(feaList,name_Spot)
  else:
    warning = app.messageBox("Enter Spot weld feature name!","warning!", 2,1)
    pass

##############################################################
def travFeatures( rootF):
  global feaList

  for fea in rootF.Children:
  feaList.append(fea)
  if fea.Children:
    travFeatures (fea)
  return feaList

##############################################################
def spotWeldList(unsortedList, nameSpecific):
  global diag, upL, spotList

  spotList = []
  upL = []
  for fea1 in unsortedList:
    if fea1.Type == VC_GEOMETRY and fea1.Name[0:6] == nameSpecific:
      spotList.append(fea1)
  for i in spotList:
  selectSpot = None
  selectSpot = diag.addProperty('Spot Selection::%s'%(i.Name), VC_BOOLEAN)
  selectSpot.Value = True
  upL.append(selectSpot)
```



```
##############################################################
def createSpots(arg):
  global diag, app, controller, program, routine, base

  toolName = diag.getProperty('Tool').Value
  baseName = diag.getProperty('Base').Value
  retract = diag.getProperty('Retract').Value
  spotFeatureList = []
  mtype = 'LIN'
  spotPart = diag.getProperty('Selected Part Name').Value
  nodeMtx = spotPart.WorldPositionMatrix
  spotPointList = updateSpots(arg)
  for p in spotPointList:
    Mat1 = nodeMtx * (p.NodePositionMatrix)
    Mat2 = getBaseInWorld(baseName, controller)
    Mat2.invert()
    Mat3 = Mat2*Mat1
    addStatement(Mat3,toolName,baseName,mtype) # Creates spot point
    call = routine.createStatement(VC_STATEMENT_CALL)
    call.RoutineName = 'CloseOpenWeld'
  if retract:
    m = vcMatrix.new(Mat3)
    m.translateRel(0,0,retract)
    addStatement( m, toolName, baseName, mtype) #Create retract after every spot operation

###############################################################
def addStatement( mtx, tool, base, mtype):
  global routine, controller

  mt = controller.createTarget()
  print controller
  mt.TargetMode = VC_MOTIONTARGET_TM_NORMAL
  mt.Target = mtx
  if mtype == 'LIN':
    s = routine.createStatement(VC_STATEMENT_LINMOTION)
  else:
    s = routine.createStatement(VC_STATEMENT_PTPMOTION)
  s.readFromTarget(mt)
  s.Tool = tool
  s.Base = base
```

F. COM Add-On (written in C#) for storing spot welding information into a CSV file

```
using System;
using System.Windows.Forms;
using vcCOM;
using vc3DCreate;
using ReadWriteCsv;
using System.IO;

namespace SendingSignals
{
    public partial class Form1 : Form
    {
        IvcApplication app;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            app = (IvcApplication)new vcc3DCreate();
        }
        private void label1_Click(object sender, EventArgs e)
        {
        }
        private void textBox3_TextChanged(object sender, EventArgs e)
        {
            //textBox3
        }
        private void button1_Click_1(object sender, EventArgs e)
        {
            string weldpart = textBox3.Text;
            IvcComponent comp = app.findComponent(weldpart);
            long childCount_1 = comp.RootNode.RootFeature.ChildCount;

            for (int i = 0; i < childCount_1; i++)
            {
                 IvcFeature feat = (IvcFeature)comp.RootNode.RootFeature.getChild(i);
                 Console.WriteLine(feat.getProperty("Name"));
                 double[] pos = (double[])feat.getProperty("NodePositionMatrix");
                 Console.WriteLine(pos[0]);
            }
        }
        private void textBox5_TextChanged(object sender, EventArgs e)
        {
        }
        private void CreateCSVspot_Click(object sender, EventArgs e)
        {
            string weldpart = textBox3.Text;
            IvcComponent comp = app.findComponent(weldpart);
            long childCount_1 = comp.RootNode.RootFeature.ChildCount;
            IvcBehaviour header = (IvcBehaviour)comp.findBehaviour("Note");
            string header_note = header.getProperty("Note");
            CsvFileWriter writer = new CsvFileWriter("WriteTest.csv");
            CsvRow row = new CsvRow();
            row.Add(header_note);
            writer.WriteRow(row);
            textBox4.Text = header_note;
            for (int i = 0; i < childCount_1; i++)
            {
                IvcFeature feat = (IvcFeature)comp.RootNode.RootFeature.getChild(i);
                string stationID = feat.getProperty("Name");
                long childCount_2 = feat.ChildCount;
                for (int j = 0; j < childCount_2; j++)
                {
                    IvcFeature feat1 = (IvcFeature)feat.getChild(j);
                    double[] pos = (double[]) feat1.getProperty ("NodePositionMatrix");
                    string spotID = feat1.getProperty("Name");
                    bool hasProperty = HasProperty(feat1, "AssociatedParts");

                    if (hasProperty)
                    {
                        string associatedParts = feat1.getProperty("AssociatedParts");
                        Console.WriteLine(spotID);
                        createCSVfile(writer, stationID, spotID, pos, associatedParts);
                    }
                    else
                    {
                        Console.WriteLine("Feature Reading Complete!");
                    }
                }
            }
        }

        private bool HasProperty(IvcPropertyList propertyList, string propertyName)
        {
            for (int i = 0; i < propertyList.PropertyCount; ++i)
            {
                string tmp = propertyList.getPropertyName(i);
                if (tmp.Equals(propertyName, StringComparison.OrdinalIgnoreCase))
                {
                    return true;
                }
            }

            return false;
        }
        private void createCSVfile(CsvFileWriter writer, string stationID, string spotID, double[] positionMatrix, string associatedParts)
        {

            CsvRow row = new CsvRow();
            row.Add(stationID);
            row.Add(spotID);
            row.Add(positionMatrix[0].ToString());
            row.Add(positionMatrix[1].ToString());
            row.Add(positionMatrix[2].ToString());
            row.Add(positionMatrix[3].ToString());
            row.Add(positionMatrix[4].ToString());
            row.Add(positionMatrix[5].ToString());
            writer.WriteRow(row);


        }
    }
}
```