

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

1999

### Economical industrial workcell modeling: Simulation and layout design.

Ana. Djuric  
*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

#### Recommended Citation

Djuric, Ana., "Economical industrial workcell modeling: Simulation and layout design." (1999). *Electronic Theses and Dissertations*. 1811.  
<https://scholar.uwindsor.ca/etd/1811>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

**Bell & Howell Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600**

**UMI<sup>®</sup>**



## **NOTE TO USERS**

**Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.**

**139-140**

**This is reproduction is the best copy available**

**UMI**



**ECONOMICAL INDUSTRIAL WORKCELL MODELING  
SIMULATION AND LAYOUT DESIGN**

by

**Ana Djuric**

A Thesis

Submitted to the College of Graduate Studies and Research  
through the Industrial and Manufacturing Systems Engineering Program  
in Partial Fulfillment of the Requirements for  
the Degree of Master of Applied Science at the  
University of Windsor

Windsor, Ontario, Canada

1999

© Ana Djuric



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52537-6

Canada

## **Abstract**

The main objective of this research was to find an optimal solution for the simulation and analysis of workcells for some automated production processes. Two processes were chosen for the study – assembly of a car fender, and spot welding of a front support engine/transaxle mount. In closer defining of this task certain constraints and requirements were postulated in order to make these processes economical and productive. For safety reasons an operator was foreseen to be included in the production process, and the possible collisions in the process had to be detected. To meet this objective and postulated requirements, the simulation and analysis were designed for a low-cost microcomputer, instead of expensive hardware systems often used previously. The commercial Workspace software package was used because of its flexible modeling, designing and process analyzing capabilities, including off-line programming, calibration, animation, and collision avoidance.

The workcell simulation approach described in this thesis is a step forward in providing a task-oriented solution to the problem of robotic cell design and programming. It integrates off-line programming techniques with significant features of welding technology. This particular graphical simulation and layout design of spot welding and assembly proved the importance of production planning and analysis. Simulation was running in real-time and each aspect of the production could be easily tracked and analyzed. Changing throughput-setting variables could simulate various production scenarios. For simple simulation the input is only run time, and for complex simulation the input represents run time with down time. Another important feature provided by – collision avoidance was also included into the simulation.



A unique feature of this study was the introduction of an operator into the simulation, for the first time in the Workspace® software. The operator was defined as a mechanism with eighteen joints.

All originally given constraints and requirements were successfully met in this simulation.

# **DEDICATION**

To my father

## **ACKNOWLEDGMENTS**

I wish to express my sincere and deep appreciation to my supervisor, Dr. Waguih H. ElMaraghy, who has been giving me encouragement and support from the very beginning of my study in this department and throughout the development of this thesis.

I am also cordially indebted to Dr. John Owens, technical director of Robot Simulation Ltd., who helped me to solve some robot simulation technology issues.

Thanks are also due to Dr. Filippo Salustri and Dr. Jerry Sokolowski, my committee members, for supporting my work and giving previous advice on preparing this thesis.

Thanks are due to Ms. Jacquie Mummery, staff of the Industrial and Manufacturing Systems Engineering Department and Izabela Kojic staff IMS Centre, for their support and friendly assistance during my study in this department.

The encouragement and suggestions from my group members are unforgettable. Finally, I convey my deep gratitude to all others not named here who helped me in various ways.

# TABLE OF CONTENTS

TABLE OF CONTENTS.....	VI
LIST OF FIGURES .....	IX
LIST OF TABLES .....	XII
1. INTRODUCTION .....	1
1.1. Introduction to robotics.....	1
1.2. The need for simulation .....	4
1.3. The history of commercially available simulations .....	5
1.4. Research into Robot Simulation .....	7
1.5. Literature survey .....	7
1.6. The objective of the research .....	10
2. MODELING AND SIMULATION.....	11
2.1. The Workspace software.....	11
2.1.1. Creating and simulating robot programs.....	11
2.1.2. The 3D CAD system and kinematics modeler.....	13
2.1.3. Defining positions.....	13
2.1.4. Geometry Points.....	14
2.1.5. Capabilities of Workspace .....	14
2.1.6. Workspace Features .....	15
2.2. Robot simulation.....	16
2.3. Modeling workcell components.....	18
2.4. Creating mechanisms .....	19
2.4.1. Modeling dependant joints to create a gripper.....	19
2.4.2. Auxiliary Axis and the Turntable .....	21
2.5. Creating robots.....	21

2.5.1. Robot links .....	22
2.5.2. Robot axes.....	24
2.5.3. Robot joint limits .....	24
2.5.4. Forward kinematics.....	24
2.5.5. Inverse kinematics .....	25
2.5.6. Creating moving cables.....	25
2.5.7. Procedure for Robot Modeling through the example of the Motoman SK120...	26
3. ROBOT PROGRAMMING.....	37
3.1. On-line programming.....	37
3.2. Off-line programming.....	37
3.3. Programming languages of industrial robots .....	39
4. INDUSTRIAL APPLICATIONS FOR ASSEMBLY AND WELDING.....	40
4.1. Assembly application.....	40
4.1.1. The application of robotics to assembly .....	40
4.1.2. A Workspace model of on automotive assembly workcell .....	41
4.1.3. Off-line programming for the assembly operation using Karel 2 language .....	47
4.1.4. Running multiple programs simultaneously for robot and mechanism .....	49
4.2. Welding application .....	54
4.2.1. The application of robotics to spot welding.....	54
4.2.2. Problem description of Industrial application for assembly and welding.....	56
4.2.3. Proposed scenario for workcell layout design .....	58
4.2.4. Operation sequences .....	60
4.2.5. Defining robot positions on the modeling parts.....	61
4.2.6. Solution for storing parts .....	62
4.2.7. Envelope plotting .....	63

4.2.8. Creating an animation .....	66
5. DETECTION OF POTENTIAL COLLISIONS AND OPTIMIZING SYSTEM LAYOUTS IN WORKSPACE .....	68
5.1. Throughput statistics.....	68
5.2. Preparing a statistics run .....	68
5.3. Analyses of workcell using Workspace .....	70
5.3.1. Throughput options.....	71
5.3.2. Simple analysis of a workcell without using down-time.....	73
5.3.3. Complex analysis of workcell considering down-time.....	79
5.4. Real-time collision avoidance.....	90
5.5. Analysis of Results .....	91
CONCLUSION.....	96
REFERENCES .....	98
APPENDIX.....	101
VITA AUCTORIS .....	164

# LIST OF FIGURES

TABLE OF CONTENTS.....	VI
LIST OF FIGURES .....	IX
LIST OF TABLES .....	XII
1. INTRODUCTION .....	1
1.1. Introduction to robotics.....	1
1.2. The need for simulation .....	4
1.3. The history of commercially available simulations .....	5
1.4. Research into Robot Simulation .....	7
1.5. Literature survey .....	7
1.6. The objective of the research .....	10
2. MODELING AND SIMULATION.....	11
2.1. The Workspace software.....	11
2.1.1. Creating and simulating robot programs.....	11
2.1.2. The 3D CAD system and kinematics modeler.....	13
2.1.3. Defining positions.....	13
2.1.4. Geometry Points.....	14
2.1.5. Capabilities of Workspace .....	14
2.1.6. Workspace Features .....	15
2.2. Robot simulation.....	16
2.3. Modeling workcell components.....	18
2.4. Creating mechanisms .....	19
2.4.1. Modeling dependant joints to create a gripper.....	19
2.4.2. Auxiliary Axis and the Turntable .....	21
2.5. Creating robots.....	21

2.5.1. Robot links .....	22
2.5.2. Robot axes.....	24
2.5.3. Robot joint limits .....	24
2.5.4. Forward kinematics.....	24
2.5.5. Inverse kinematics .....	25
2.5.6. Creating moving cables.....	25
2.5.7. Procedure for Robot Modeling through the example of the Motoman SK120...	26
3. ROBOT PROGRAMMING.....	37
3.1. On-line programming.....	37
3.2. Off-line programming .....	37
3.3. Programming languages of industrial robots .....	39
4. INDUSTRIAL APPLICATIONS FOR ASSEMBLY AND WELDING.....	40
4.1. Assembly application.....	40
4.1.1. The application of robotics to assembly .....	40
4.1.2. A Workspace model of on automotive assembly workcell .....	41
4.1.3. Off-line programming for the assembly operation using Karel 2 language .....	47
4.1.4. Running multiple programs simultaneously for robot and mechanism .....	49
4.2. Welding application .....	54
4.2.1. The application of robotics to spot welding.....	54
4.2.2. Problem description of Industrial application for assembly and welding.....	56
4.2.3. Proposed scenario for workcell layout design .....	58
4.2.4. Operation sequences .....	60
4.2.5. Defining robot positions on the modeling parts.....	61
4.2.6. Solution for storing parts .....	62
4.2.7. Envelope plotting.....	63



4.2.8. Creating an animation .....	66
5. DETECTION OF POTENTIAL COLLISIONS AND OPTIMIZING SYSTEM	
LAYOUTS IN WORKSPACE .....	68
5.1. Throughput statistics .....	68
5.2. Preparing a statistics run .....	68
5.3. Analyses of workcell using Workspace .....	70
5.3.1. Throughput options .....	71
5.3.2. Simple analysis of a workcell without using down-time .....	73
5.3.3. Complex analysis of workcell considering down-time .....	79
5.4. Real-time collision avoidance .....	90
5.5. Analysis of Results .....	91
CONCLUSION .....	96
APPENDIX .....	98
REFERENCES .....	98
VITA AUCTORIS .....	164

# LIST OF TABLES

TABLE 1: A LIST OF MAJOR AND SOME MINOR ROBOT MANUFACTURERS AND THEIR PROGRAMMING LANGUAGES .....	15
TABLE 2: SUMMARY OF MANUFACTURING CAPACITY RESULTS .....	104
TABLE 3: CALCULATION FOR CELL COST .....	105
TABLE 4: WORKSPACE SIMPLE AND COMPLEX ANALYSIS RESULTS COMPARISON .....	6 106
TABLE 5: REQUIREMENTS AND REQUESTS .....	107

# **1. INTRODUCTION**

## **1.1. Introduction to robotics**

Research during the past twenty years has been directed towards creating soft (flexible) automation techniques, which could be applied to small and medium batch runs. This has culminated in the development of numerically-controlled milling machines, flexible manufacturing systems, automatic guided vehicles, and industrial robot-arms.

The development of the industrial robot-arm was of particular importance. Its ability to emulate the serial-link manipulator nature of the human-arm has led to a whole new class of production tasks, which may be profitably automated. Examples are: paint-spraying, welding, PCB component insertion, assembly, and pick-and-place operations. More recently, advances in image-recognition, tactile-sensing, and artificial-intelligence have expanded the number of potential applications into those areas where sensory feedback from the environment is required, such as picking up objects which have been randomly placed on a conveyor-belt.

‘Dumb’ manipulators (controlled directly by a human instead of following a computer program) have found further applications in carrying out tasks which might be potentially hazardous to humans. Examples are: handling radioactive materials, and manipulating objects underwater or in space [Owens, 1990].

The first robots were introduced in manufacturing industry in the USA in 1961. Since that time there has been a significant growth in the number of robots in use and the range of applications in which they are used. Today robots play a key role in industrial and manufacturing engineering. In developed countries, many large-scale

national and international research and development programs have been launched and still are going on. For example, ESPRIT, BRITE, and EUREKA are large European research programs that focus on robotics. NASA and the European Space Agency also conduct robotics research for tele-operations in outer space. The governments in United States, Canada, and Japan are currently funding large research projects to develop the so-called fourth-generation robots. In the private sector, several hundreds of companies are manufacturing and servicing robots of various types, as well as supporting robotic systems and peripherals [Ulrich, 1990].

The main driving force in the development and application of robots and mechanisms has been the automotive industry. The benefits of flexible automation have been incorporated wholeheartedly into the automotive manufacturing process. The automotive companies have also assisted the utilization of robots in other areas by encouraging their suppliers to utilize the same technologies.

In general, robots may substitute for human operators in the following situations [Groover, 1987]:

- Hazardous work environment for human operators
- Repetitive and heavy work cycles
- Parts difficult to handle by human operators
- Infrequent product changeovers
- Part positioning and orientating

Many commercially available industrial robots are widely used in manufacturing and assembly tasks, such as material handling, spot/arc welding, parts assembly, paint spraying, loading, and unloading numerically controlled machines. Some robots are also used in space and undersea exploration, and prosthetic arm research. According to a search of the Internet [[www.ros1.com](http://www.ros1.com)], there are over 100

robot manufacturers around the world. Table 1 is a list of some manufacturers and their programming languages [Gong, 1998].

**Table 1 A list of major and some minor robot manufacturers and their programming languages**

Manufacturer	Programming Languages
ABB	ARLA, RAPID
Adept	V+
Comau	PDL2
Eshed	ACL
Fanuc	RG2, Karel 2, Karel 3, TP
IBM (sold their robotics business to Sankyo over ten years ago)	AML/2
Kawasaki	AS
Motoman	Inform 1, Inform 2
Nachi	SLIM
Panasonic	Parl-1, Parl-2
PSI	PSI
RTX (ceased trading five years ago. RTX now handled by OXIM)	FRTX
Samsung	FARL-II
Seiko	DARL 4
Toyoda	TL-1
TQ	TQ
Unimation (now called Staubli, and now use V+)	VAL I, VAL II

## **1.2. The need for simulation**

The techniques of Computer Aided Design have found extensive use in replacing and improving the process of engineering drawing, architectural drawing, and many other applications. However, an engineering process involving moving parts can only be understood fully through the process of simulation.

Early techniques have involved taking a CAD engineering drawing of a machined part and creating an animated simulation of the movements that the machining centre must go through to create the part from a 'raw' block. As well as providing a visualization of the process, a file can be created containing the required instructions to the machining centre. The file can then be executed to create the part.

This extension of Computer Aided Design to Computer Aided Manufacturing seemed desirable for Industrial Robotics. However, the kinematics involved in robot movement is considerably more complex than that related to XYZ machining centres, and the relation between the curves swept by the robot end-effector and the joint variables is not straight-forward. Additionally, for a robot simulation to be of general use, it must be capable of simulating a wide variety of robot types and configurations.

Despite these difficulties, simulation appears to provide a suitable interactive graphical environment to improve the ease with which industrial robots are programmed. Benefits such as the ability to detect off-line collisions between robots and objects, and the ability to evaluate and optimize the time taken for a sequence of movements off-line, have also provided major incentives to research and development into robot simulation [Owens, 1990].

### **1.3. The history of commercially available simulations**

Perhaps the first commercially available robot simulation was GRASP, marketed by the UK Company BYG, which was developed at Nottingham University over a period of seven years. GRASP is used in the UK by PA, Taylor-high-tech, and a number of academic and research institutions. It has undergone a number of major improvements, the most significant one was the introduction of solid modeling. GRASP cannot model non-serial robot structures (e.g. parallelogram joints-frequently found in robots) in its present formulation.

Computervision (CV) – a UK company better known for its CAD packages marketed a simulation named "Robographics" which was used at Austin-rover and Unimation.

The McAuto CAD division of McDonnell-Douglas company marketed a series of packages designed for robot simulation, that were used by Cincinnati-Millacron. PLACE is used for Robot cell layout and evaluation, BUILD is used for robot modeling and can be used for real-time 3-D dynamic studies, COMMAND is used for off-line programming, and ADJUST for robot calibration.

McAuto had the advantage that it was compatible with McDonnell-Douglas CAD systems that were in widespread use. However the package does not provide such basic features as hidden-line removal and collision-detection.

Technamatics, an Israeli-based company with a European base in Brussels, marketed ROBCAD. The package is currently used by many of the European carmakers (Ford, BMW, Volkswagen, and OPEL). ROBCAD can automatically define object face – a useful feature if IGES CAD standard geometric information is to be input, since IGES does not pass face information.

Deneb IGRIP is also currently used by many large manufacturing organizations. Like ROBCAD, it is graphics workstation based. Some robot manufacturers have created their own robot simulation software with limited functionality specifically for selling alongside their own range of robots. Motoman sells the Rotsy software based on a customization of a CAD package from a company named 3D-Eye.

A number of other robot simulations are commercially available but have made little impact on the market, including the Dassault “CATIA Robotics” package which is compatible with its CATIA CAD system and the General Electric “CALMA” package. Dassault recently purchased Deneb and dropped their CATIA robotics package.

ROSI marketed by Cambridge Control is a robot simulation, simulating the dynamics of a robot – a different type of simulation, of little use in evaluating workspaces, but of great use in design and evaluation of new robot structures. It can also provide extremely accurate estimates of cycle times since it takes into account not only the demanded joint position but also the lag between the demand and the actual position due to the dynamics of the robot system. This is important for very fast high-load applications, such as the robots produced by Lamberton Robotics (in Coatbridge near Glasgow) for handling very large payloads (e.g. palletizing three barrels at a time) [Owens, 1990].



## **1.4. Research into Robot Simulation**

A considerable amount of academic research is devoted to the general techniques involved in robot simulation, with many institutions developing their own package.

Researchers at Brunel University use a language called UPL to implement a robot simulation on a microcomputer-first-order reasoning to be used by the robot to find a series of robot actions that will enable the robot to achieve its goal. Others have concentrated on the design of a task-description language, possibly based on one of the “modern” algorithmic programming languages. A third approach has been to examine the geometric nature of a robot task and reduce the degrees-of-freedom via spatial reasoning [Owens, 1990].

## **1.5. Literature survey**

A great deal of work can be found that relates to robotic research and development. Some of this work deals with the control of industrial robots, which enables automatization of robot systems and improves their application reliability in manufacturing processes [Craig 1989, Vukobratovic 1986, Critchlow 1985, Groover 1987, Ulrich 1990, and Gong 1998].

For the modeling, simulation, off-line programming, collision detection, and layout analysis, industrial companies are currently using high-end workstations that are in the high price range. There are some commercially available simulation software packages with different capabilities and price.

XAnimate: An educational software for robot graphical simulation, has been developed at Ohio University [Marhefka and Orin, 1996], to provide portable

graphical simulation at no cost. XAnimate's graphical user interface, C library of functions, and set of predefined objects enable users to easily display animation wireframe images or solid color object systems of any topological structure.

ROBO\_SIM: A robotics simulation environment on personal computers has been developed for the MATLAB matrix manipulation program. The package consists of callable routines for performing specific calculations: the functions for Forward and Inverse Kinematics, Arm dynamics, trajectory Planing, Control and Simulation.

Neither software has enough capabilities for simulation, off-line programming, and workcell analysis.

A group of scientists [Kukareko, Pashkevich, Khmel, Korzun, and Yurkevich, 1994], presented the results of their research devoted to computer-aided welding workcell design, path planning and programming. This paper presents specially developed algorithms and software tools for the simulation of arc and spot welding robotic cells and the generation of technological programs. This algorithm cannot be applied to the other industrial applications. The developed algorithm has been implemented in the ROBARC package. This software has been used for welding workcells design in automotive industry, and can be run only on IBM compatible workstations. The workpiece that they used in this simulation has been designed using the CATIA CAD system, and was transferred in DXF format to ROBARC. In their workcell simulation, they did not provide a layout analysis.

Rooks [1997] examines the developing uses of off-line programming in the automotive industry, particularly in the area of robot welding and robot painting. One of the major reasons for the success of robot OLP in automotive production is the

development of graphical simulation tools. These analyses are provided in Robcad and IGRIP software packages, but these are mainly PC-based.

PROWELD, a PC-based software package for low-level programmers is developed for off-line programming of welding robots, and is evaluated for arc welding using a six-degree of freedom industrial robot [Balkan, Arikan and Bulut, 1997]. This software has some limitations, such as the lack of Collision detection, workcell analysis, advanced robot languages, import and export facility, direct and inverse kinematics modeler, etc.

DENEB developed separate software for human factor and ergonomic analysis, called ERGO [Alexopoulos, 1995]. For graphical simulation and off-line programming, DENEB developed software TELEGRIP-Tele-interactive Graphics Robot Instruction Program [Hewer, 1996]. Both software packages are based on PC stations, and do not have capability for layout analysis.

Some research has been done in discrete event system simulation for planning and design purposes. The motivation for using simulation is that it can often capture and describe the complex interactions within a particular FMS where analytical methods fail. Simulation is commonly used to gain insight into manufacturing systems [Glover, Kelly, Laguna 1996], [Peters, Smith, Curry, LaJimore 1996], [Kelton, Sadowski, and Sadowski 1998]. Voss and Haddock [1994] presented a discrete event simulation approach to the analysis of intelligent robotic systems. The model is intended to help balance system design decisions.

## **1.6. The objective of the research**

The objective of this research was to design and analyze a workcell for automated assembly and spot welding of front-support member engine/transaxel mount. The Workspace software is used to develop and monitor this particular workcell. The main advantages of Workspace are capability for graphical simulation, off-line programming of robots, mechanisms and human beings, and layout designs. It is a PC-based software for Windows®, which makes it user-friendly and affordable for average computer user.

In solving this problem the following requirements were considered:

- To automate the process with IRB6000 robots,
- To include one operator (for union and safety reasons),
- A total cell cost of no more than \$ 600,000,
- Production must be at least 40,000 parts per year,
- Size of workcell is 24 X 30 sq. ft,
- Layout design (the number of jobs, the run-in time, the run time, the number of jobs per hour),
- Collision detection.

According to the literature survey and preliminary research in this thesis, it was concluded that Workspace software could satisfy the above requirements and the price.

The simulation and workcell for front support member engine/transaxel mount is unique, because this industrial process has not been automated before and also because this workcell includes a operator in the simulation, for the first time in the Workspace software.

## **2. Modeling and simulation**

Many of the largest users of robotic production equipment, predominantly in the automotive sector, are well aware of the benefits that effective simulation systems can offer as a part of the design and specification process for their robotic systems.

Simulation is a powerful tool in the design of the manufacturing systems, which can be analyzed, optimized and verified before the purchase or installation of any capital equipment. It is a means to avoiding costly errors, speeding up commissioning and ensuring the plant operates right first time [Rooks, 1997].

### **2.1. The Workspace software**

Workspace is the first industrial robot simulation software package to be based on a microcomputer. Since its commercial release in 1989 it has undergone many revisions. With the latest software release (version 4.103) Workspace is in use on over 1000 industrial and educational sites worldwide as both a graphic simulation system and a means of off-line programming a robot workcell.

#### **2.1.1. Creating and simulating robot programs**

Workspace will create and simulate robot programs in the native language of the robot. There is, therefore, no need for postprocessors to translate from a simulation language to the robot language: the full power of the robot language is available through off-line programming. It is also possible to transfer existing robot programs from the robot controller back into Workspace. Off-line programming using Workspace is, therefore, a two-way process. This is particularly facilitated by the use

of portable microcomputers which may be taken down onto the factory floor and used next to a robot workcell.

The user first selects the target robot language. To create a robot program, the user clicks on robot commands in a pull-down menu. These commands are immediately written to an ASCII text file. In this way a robot program is created without the necessity for the user to know the precise syntax of the commands, though he may also use an in-built text editor if desired. As each command is written to the file, it is simulated graphically on the computer screen in color 3D solid graphics.

Programs may be debugged by tracing through line by line seeing at each line what is happening in the simulation. The value of variables may be watched or examined throughout the program. If an error occurs at any point, the user will immediately be placed in the Workspace text editor at the position in the robot program where the error occurred, enabling him to promptly correct the error.

The full structure of the robot languages is implemented, including typed variables, teachpoints, subroutines, looping, branching on condition, signals, and condition handler interrupts.

Several robots working in co-ordination, each under the control of a separate robot program may be simulated to produce one animation. During the simulation of the robot program any collision that occurs between any objects in the workcell may be automatically reported, and a cycle time is calculated for the overall sequence of movements. The 3D volume of the envelope may be plotted, or 2D slices through the envelope displayed on the screen.

All the main industrial and educational robot languages are implemented, and a library of over 230 robot models is available to the user (though it is also possible for users to create their own robots).

### **2.1.2. The 3D CAD system and kinematics modeler**

The first step in any simulation is to load the CAD models of the parts to be processed into the simulation.

3D solid object (including combinations created using Constructive Solid Geometry) or surfaces (including Bspline, Parametric, or Bezier surface) may be created using Workspace's own 3D CAD system, or else imported into Workspace from an external CAD system using the DXF file format (common amongst microcomputer based CAD systems such as AutoCAD) or the IGES file format (common amongst graphics workstation based CAD systems such as CATIA). The use of an unlimited number of layers for storing different levels of detail makes it possible to turn off the display of object irrelevant to the current task, or display them in wireframe for speed.

The movement of any mechanism may be modeled using a kinematics modeler. The mechanism may have any number of joints in any serial or tree-structure combination. Conveyors, automatic vehicle, and other independently moving objects may also be modeled [User Guide Manual Workspace, 1997].

### **2.1.3. Defining positions**

Positions and paths for the robot tool to move may be defined in several ways. A software emulation of a teach pendant is available to move the robot either by stepping individual joint angles or by stepping the xyz Cartesian position of the tool relative to the robot world co-ordinate frame. Positions may be saved as teachpoint for use in a robot program later. These are displayed on the screen graphically as coordinate frames. If the robot is placed in a different position in the workcell then the

teachpoints will change color if they are no longer achievable due to the limits on the joints of the robot or due to the limited reach of the robot.

Positions may also be defined by clicking at different positions on the computer screen (changing the view if necessary to define a 3D position). Orientations may be defined in many different ways: for example by dragging the approach vector of the teachpoint co-ordinate frame in a particular direction (using the mouse).

#### **2.1.4. Geometry Points**

The easiest and most powerful way of defining robot positions is to use the objects geometry. Points may be defined along the seam between two surfaces in such a way that the position of the point on the seam, the distance of the point from the seam, the lean of the approach vector of the tool towards the seam, or the angle between the adjacent surfaces and the tool approach vector is set by the user. This is of use in applications such as arc welding.

Points may also be defined normal to a surface so that the position on the surface and the distance from the surface to the point is set by the user. As the point is moved over the surface the point is always maintained normal to the surface. This is of use in applications such as spot-welding [Owens, 1994].

#### **2.1.5. Capabilities of Workspace**

- To model new workcell layouts involving conveyors, AGV's and evaluate their performance
- Communicate design concepts using state of the art 3D graphics and demonstrate



process and plant design

- Prepare analytical statistics of workcell performance
- Prepare real time animations for presentations
- Training and education in complete safety
- Manage complex projects
- Calibrate the robot to an accuracy of less than 1mm
- Generate robot programs off-line using a simple mouse driven menu system

#### **2.1.6. Workspace Features**

Some of Workspace Features are:

- 3D CAD system featuring Constructive Solid Geometry and Swept Polylines.
- Advanced Robot languages of the major manufacturers.
- Off-line Programming.
- DXF/IGES import and export facility.
- Robot and workcell calibration to less than 1mm accuracy using Calibration Plus®
- Kinematics and inverse kinematics modeler for mechanisms with up to 22 joints.
- For the rest of the features see Appendix A.1. [User Guide Manual for Workspace, 1997].

## **2.2. Robot simulation**

The simulation of industrial robots has become an important means to increase the application efficiency of robots. In advanced manufacturing, it is desirable to simulate the manufacturing systems and processes before installing physical equipment. Simulation enables us to rectify errors in design and manufacturing processes before the manufacturing system is set on the floor. Animation and simulation can be packaged with multimedia production with graphical/textural information [Gong, 1998].

Designing efficient work cells is a difficult task. The need to predict robot movement, to avoid interference between robots and determine the optimum placement for maximum reach is essential for the workcell design engineer. The ability to design within confined spaces where there is a high risk of collision together with identification of possible production bottlenecks is of utmost importance. It is also important to know how long each process within a cycle will take.

When a new design is to be incorporated, the concepts must be clearly communicated to all team members. Simulation allows the planning of safety factors because operating robots can carry a risk of injury.

The benefits of using simulation can easily be measured. With a faster design of new workcell layouts, the engineering effort is reduced creating less pressure on the team. Faster redesign means reduced downtime in line shutdowns for on-line programming. New ideas can be tested on the computer, eliminating costly mistakes. The cycle time can be examined to produce optimum schedules while verifying robot reach and collision detection. More importantly, the optimization of workcells can be achieved off-line, allowing the robot to continue operating during the planning phase.

Off-line programming is available for similar reasons, and allows downloading to the robot controller [Bernhardt, Schreck and Willnow, 1995].

Workspace can increase productivity and reduce system downtime. System programming takes place before robot installation or while an existing workcell is in operation. Complex layouts can be tested in complete safety either in an office environment or on the shop floor. Simulation highlights potential problems before they happen, allowing quick and easy modifications.

3D simulation has revolutionized the ways that engineers can work. Workspace lets you add equipment to a workcell, delete or move objects for the most complex of robotics applications, reducing development time and ensuring a complete solution.

A graphical simulation system for the validation and specification of the robot program is an integral part of an advanced programming system. It must provide a library of emulated robots, transport devices, and end-effectors to build up a cell model quickly. Modeling software for the robot's environment and of the robot itself must be available. The graphic representation enables the operator to check the programmed operation. So the system contains specific tools for [Gong, 1998]:

- Robot modeling
- World modeling
- Description of motion
- Collision detection
- Control code generation

There is a number of graphical simulation systems available in the market, such as IGRIP, ROBCAD, CATIA, ADAMS, and Workspace.

One of the aims of Workspace is to provide a realistic, 3D animation showing how the machinery on a production line will work together. In addition to create animation, we can test the robot's reach, detect collision and get an indication of cycle time for a workcell. The total cycle time is displayed in the information window after a track has finished being recorded or replayed. During replay or recording the elapsed cycle time can also be seen on the far right of the status line.

### **2.3. Modeling workcell components**

A Workcell model presents the significant features and behavior of the workcell components and defines the structure of relations among them. Here an object-oriented approach is used for developing the model. The model is split up into a hierarchy of classes characterizing certain properties of components such as geometrical, physical and functional properties relevant to their mode of operation. A Geometrical submodel is constructed by composition of rigid solids using a CAD graphical editor. The relational aspect is modeled using a graph structure characterizing spatial links between objects. Robot and positioner kinematics parameters are also included in the geometrical submodel. A physical submodel provides simulation of the mechanical system [Kukareko, Pashkevich, Khmel, Korzun, and Yurkevich, 1994].

Workspace is a powerful modeling package, that enables engineers and designers to create their own 'virtual' world. The majority of robots and workcell reside in factory, workshop, or laboratory environments. A typical application of Workspace might be to test the feasibility of a chosen robot or robots to perform set tasks within a given space. Workspace includes a library of common shapes, which may be used to construct complex worlds with minimum effort.

## **2.4. Creating mechanisms**

A mechanism is a collection of dependent joints that control the motion of a stand-alone structure (See Appendix A2). This structure can be serial or parallel, or a combination of both.

To create a mechanism one has to model the flesh first (there are no named limitations for the objects that make up a mechanism, unlike a robot). Next, one should create the dependant joints defining the motion of the objects, which make up the mechanism structure. It is necessary to create joints in between the component objects making up the mechanism. In addition, it is also necessary to specify an equation of motion for each joint so that Workspace knows how to simulate the mechanism correctly.

The joint axes should be positioned using the same commands that were used to modify robot axes. The expressions that define a particular joint's movement should be set.

Finally, one has to select the base of mechanism and issue the Create/Create Mechanism command giving the total number of axes when prompted.

A mechanism resembles an ordinary robot as it has auxiliary joints and can be controlled from the Pendant menu. However, it also differs from an ordinary robot since it does not have any ordinary joints (it has dependent ones) or robot links [User Guide Manual Workspace, 1997].

### **2.4.1. Modeling dependant joints to create a gripper**

Dependant joints work by defining an attachment between a parent and child object, and by defining a mathematical expression, which Workspace evaluates during simulation to decide what the value of the joint displacement should be. The joint coordinate axes are moved to the correct position and orientation using the commands

on the 'Edit Joint' menu, bearing in mind that the z-axis represents the rotation axis for the joint. There are actually two types of joint available: rotational and translational (or prismatic). In the case of the translational joint, the z-axis defines the direction of the joint motion.

To create a dependant joint between two objects, a decision needs to be made on which object remains fixed (the parent) and which object actually moves (the child) and the child object has to be selected. Then, the 'Create dependent Joint' command is selected. The commands are used to move and turn the joint to the correct position.

The next step is to describe how the joints move by entering an 'Expression for dependent Joint'. The default is AXISPOS (1), which means that the joint will move as the joint 1 of the robot is changed from the pendant menu. In this work AXISPOS was not used because there was a special internal function for grippers called GRIPPOS. This variable change from 0 to 100 when an 'Open Gripper' command is issued and then goes back to 0 when the 'Close Gripper' command is given. Obviously, there is not always necessary to have a displacement of 100 in joint, so a scale factor has to be used. For instance, if the gripper consists of rotational joints which only move by 30 degrees from closed to open then the expression would be:  $GRIPPOS \cdot 0.3$ . If the gripper had translational joints which moved apart by a total aperture of 100mm ( $2 \cdot 50\text{mm}$ , one for each gripper) then the expression for each joint would be:  $GRIPPOS \cdot 0.5$ .

Finally, it has to be checked whether GRIPPOS has the value that was expected before modeling the gripper. For GRIPPOS to have zero value the gripper is modeled closed, and for  $GRIPPOS=100$  the command 'Open Hand' is issued. It is not possible to move dependent joints until they are attached to a part of the robot - for

example Link6. If there are several grippers that the robot can automatically attach and detach from its Link6 during a task, we don't get the grippers that are not in use opening and closing when the robot issues gripper commands [User Guide Manual, Workspace, 1997].

#### **2.4.2. Auxiliary Axis and the Turntable**

The turntable involves techniques similar to those used when modeling the gripper, except that an auxiliary axis (number 7) of the robot controls the table instead of GRIPPOS. The turntable will need auxiliary axis teachpoints storing for the two positions that it will move to (AUXPOS variables in the teachpoint file).

### **2.5. Creating robots**

The definition of robot: "A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks". In the treatment presented here, a robot will be taken to mean an industrial robot, also called a robotic manipulator.

Workspace incorporates a wide range of facilities for the design, building, testing and calibration of robot structures.

At the simplest level the mechanical structure of a robot can be considered as a serial of links bolted together at the joints. Each bolt is sufficiently loose to enable the jointed links to move relative to each other. In Workspace, robots are special objects which have additional "LINK" objects and data associated with them.

By convention, the robot object is the object that is usually attached to the floor at the base of the robot. This object should be given the model number of robot. (E.g. IRB6000, Comau-S2).

### **2.5.1. Robot links**

Workspace links are special objects that make up part of a robot. They are named: Link1, Link2, Link3, and so on. Link1 is attached to the robot base object. Link2 is attached to Link1 and so on.

These links are objects attached in a special way. A robot link has an axis associated with it that defines how the next link in the chain will move.

For this axis, there are two simple rules:

- For a translational joint, the next link moves along Z.
- For a rotational joint, the next link moves around Z.

The last link in the chain, usually Link6 has an axis even though there are no more Link objects after it. The position of this axis determines the positions of the robot's end effector.

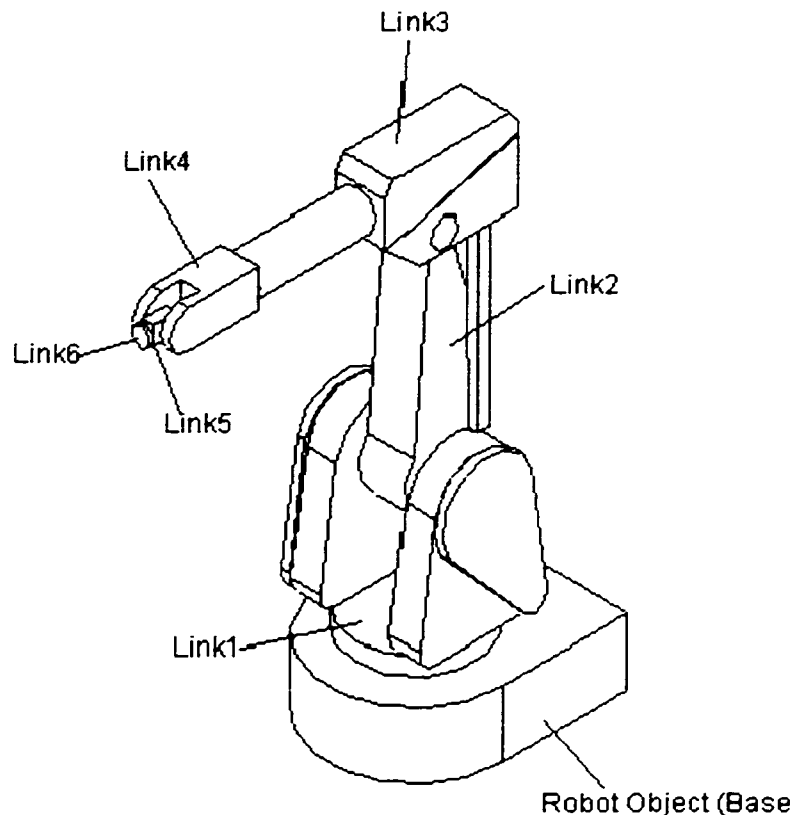
To create a typical 6-axis robot, the following is needed:

A robot object (base) – for the “waist” axis

- Link1 - for the “shoulder” axis
- Link2 - for the “elbow” axis
- Link3 - for the “roll” axis
- Link4 - for “pitch” axis
- Link5 - for “yaw” axis
- Link6 – for the end effector frame



So, at least seven objects are necessary to start with, (Figure 1). We cannot attach any of the base or link objects together, this will be done automatically when we issue the Create/Create Robot command. To each of these links, one may wish to attach additional objects, e.g. motor housing. Once we have positioned the body of the robot, issue the Create Robot command, we will be prompted to enter the number of



joints for this new robot. This will be equal to the highest link number [User Guide Manual Workspace, 1997].

**Figure 1 Link objects in the IRB2000 model**

### **2.5.2. Robot axes**

Although it is possible to specify the location and orientation of an axis directly in the robot A-matrix, there are two commands to do this directly in the CAD editor. These commands are in the Edit/Edit Joint submenu:

- Move Axis
- Turn Axis

We cannot always move or turn an axis in the way we want. Workspace will not allow us to turn or move an axis, so that the robot A – matrix becomes invalid. The range of permissible angle and positions is determined by the application of Denavit Hartenberg rules to the axes of the robot. To help us position all the joints we can check the Edit/Edit Robot/ Settings/Show kinematics command. This option will display the distance and offsets between the joints on our robot [User Guide Manual Workspace, 1997].

### **2.5.3. Robot joint limits**

Our robot model stores information for each of its joints. This information includes the maximum and minimum values for the joint that is by default in mm for a prismatic joint or in degrees for a revolute one. Also stored are the maximum speed and acceleration for each joint.

### **2.5.4. Forward kinematics**

Once we have defined the position and orientation of all our joint axes, including the axis stored on the last link of our robot defining the robot's end effector, Workspace can calculate forward kinematics for our robot. This means that, given all

the joint values, Workspace can calculate the position and orientation of the robot's end effector relative to the robot base. We can change these angles directly by using the Pendant menu.

#### **2.5.5. Inverse kinematics**

Inverse kinematics is effectively the reverse of forward kinematics. That is, given a desired x, y, z location and orientation for the robot's end effector, we can find the robot joint values required to make the robot's end effector attain that position.

#### **2.5.6. Creating moving cables**

A moving cable can be used (for example) to simulate the power supply cable to a spot welding gun. As the robot is moved, one can see if the cable is likely to get wrapped around the wrist or tool.

Before a moving cable is created, one must define two objects (one for each end of the cable). Both these objects must have a co-ordinate frame defined on them. To add a co-ordinate frame to an object, select the object and issue the command Edit/Edit Object/Add Co-ordinate Frame. To edit an object's co-ordinate frame, the command on the Edit/Edit Joint submenu is used.

When the two cable end objects (and their co-ordinate frames) have been defined, one of the objects is selected and the command Create/Create Moving Cable is issued. Then the other end object has to be selected and a name entered. The cross-section of the cable is oval, so the dimensions that can be varied are:

- cable width slot
- cable length slot
- cable length

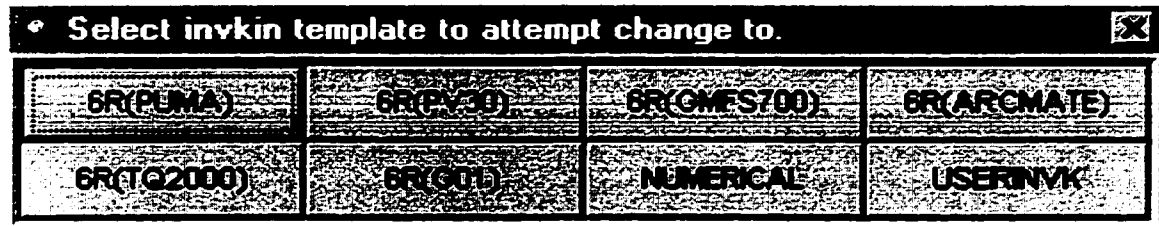
The ends of the cable will be created normal to the z-axes of the co-ordinate frames.

### **2.5.7. Procedure for Robot Modeling through the example of the Motoman**

#### **SK120**

- Create the "flesh" of the robot.
- Rename all robot components to the appropriate names; for base use robot name, for links use Link1, Link2, Link3, Link4, Link5, Link6.
- The next stage of the construction is to join the links into a robot structure. For this operation co-ordinate frames must be placed at the joint centers and oriented correctly. Attach all components to Link1 that are belonging to Link1, the same for Link2, Link3, and, Link6.
- Do not attach dependant joints.
- For the robot base use the robot's name (such as Motomansk16, Comau, IRB6000, ...)
- Select the base of the robot and go to Create/Create Robot, Enter the number of joint variables.
- Each joint must be in the correct position. To change their positions, go to Edit/Edit Joint/Move Axes, or Edit/Edit Joint/Turn Axes.
- Edit the Kinematics of the robot: Edit/Edit Robot/Exact Kinematics, or if a similar robot exists, copy Kinematics, Edit/Edit Robot/Copy Kinematics from another

robot. If the robot has six degrees of freedom, normally one of two types of Exact Kinematics is selected. When Joint2 and Joint3 rotate in same direction (e.g. forward and down respectively), 6R(PUMA) will be used. When Joint2 and Joint3 rotate in opposite directions (e.g. forward and up respectively), 6R(GMFS700) will be used (Figure2).



**Figure 2 Invkin template**

If another robot model exists, that has a similar shape, and rotates in the same directions, then the second model has to be imported into the current file. The operating model should be moved to a different location, and the command switched to File/Load Model and Add to Current Model. The second model has to be selected from the directory that it is in. It will be imported into the current file at the (0,0,0) location. By selecting the base of the robot and the CTRL button it will be possible to operate that particular robot. Then the following commands should be applied: selecting the operating robot; select Edit/Edit Robot/ Copy Kinematics from Another Robot; confirm the copy; delete the imported robot, and move all of the joints into place. Double check that the Exact Kinematics is correct.

- Match the template of the Exact Kinematics and make modifications to the robot model if necessary.

- By choosing (for SK120) 6R(GMFS700), matrix 'A' is optaned (Figure3).

Information Window

6R(GMFS700) inverse kinematics require these A matrices:

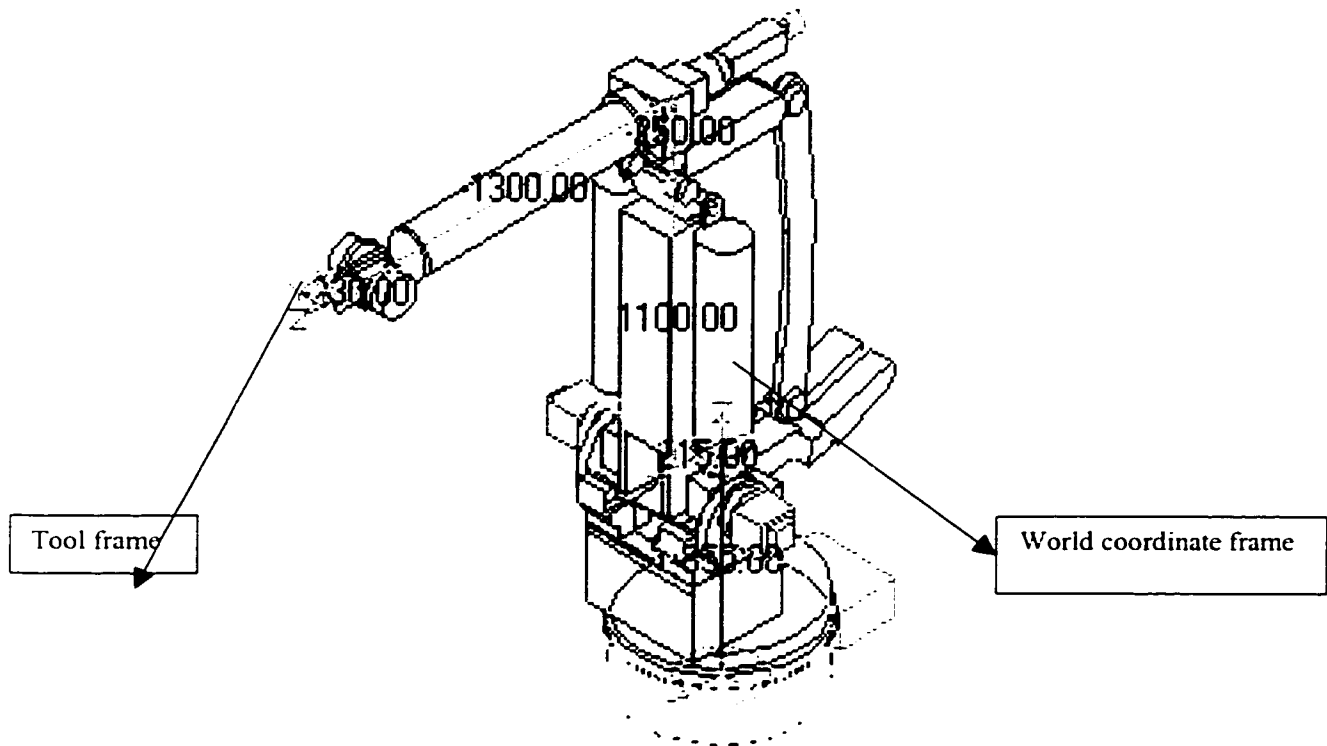
Th1:dontcare D1:dontcare A1:dontcare AI1: -90.00  
 Th2:dontcare D2:dontcare A2:dontcare AI2: 180.00  
 Th3:dontcare D3:dontcare A3:dontcare AI3: 90.00  
 Th4:dontcare D4:dontcare A4: 0.00 AI4: -90.00  
 Th5:dontcare D5: 0.00 A5: 0.00 AI5: 90.00  
 Th6:dontcare D6:dontcare A6:dontcare AI6:dontcare

• A matrix constants

Th1: 0.00	D1: 0.00	A1: 215.00	AI1: 90.00
Th2: 90.00	D2: 0.00	A2: 500.00	AI2: 180.00
Th3: 180.00	D3: 0.00	A3: 250.00	AI3: 90.00
Th4: 0.00	D4: 1100.00	A4: 0.00	AI4: 90.00
Th5: 0.00	D5: 0.00	A5: 0.00	AI5: 90.00
Th6: 180.00	D6: 230.00	A6: 0.00	AI6: 180.00
Tw1: 0.00			

Figure 3 A matrix constants

- Add the World Coordinate Frame to the robot, Edit/Edit Robot/Add Robot World Frame. In order to move or rotate World Coordinate Frame to correct position, one has to select the base of the robot and go to Edit/Edit Joint, chose the option World Coordinate Frame, and go to Move/Turn Joint. The CP of the base must be placed at the World Coordinate frame. To make this easier, one has to create the base of the robot about that point (0,0,0).
- In order to change the Tool Frame orientation for Joint6, enter: Modify: Th6, or Tw6, or AI6 in matrix 'A', (Figure4).



**Figure 4 Tool and World coordinate frame**

- Enter the robot limits for each joint, Edit/Edit Robot/Limits/Enter the given limits, and make a modification if the Joint23 mechanical linkage is required.
- Edit the robot Velocity and Acceleration values, Edit/Edit Robot/Velocity and acceleration. See (Figure 5).

Joint limits <span style="float: right;">X</span>			
Min1: -180.00	Max1: 180.00	Mxv1: 110.00	Mxa1: 220.00
Min2: -70.00	Max2: 70.00	Mxv2: 125.00	Mxa2: 250.00
Min3: -75.00	Max3: 75.00	Mxv3: 125.00	Mxa3: 250.00
Min4: -350.00	Max4: 350.00	Mxv4: 150.00	Mxa4: 300.00
Min5: -130.00	Max5: 130.00	Mxv5: 150.00	Mxa5: 300.00
Min6: -355.00	Max6: 355.00	Mxv6: 250.00	Mxa6: 500.00
Mxlv: 400.00	Mxla: 400.00	Mn23: -115.00	Mx23: 35.00

**Figure 5 Joint Limits**

- Min1, Min2, ... , Min6: is minimum rotation in degrees for Joint1, Joint2, ... , Joint6
- Max1, Max2, ... , Max6: is maximum rotation in degrees for Joint1, Joint2, ... , Joint6
- Mxv1, Mxv2, ... , Mxv6: is maximum velocity in degrees/sec. for Joint1, Joint2, ... , Joint6
- Mxa1, Mxa2, ... , Mxa6: is maximum acceleration in degrees/sec<sup>2</sup>. for Joint1, Joint2, ... , Joint6
- Mxlv: is maximum linear trajectory velocity in mm/sec.
- Mxla: is maximum linear trajectory acceleration in mm/sec<sup>2</sup>.
- Create all dependant joints if they are required (i.e. if the robot has a parallelogram structure on joints 2 and 3). Go to Create/Create Dependant Joint; select O.K. to attach the part to the parent.



- Checking of the working range information is sometimes outlined by particular joint centre. Normally the center of joint5 is used. If nothing is specified, the end of joint6 can be used. The Tool Center Point (TCP) has to be at the center of the required joint. To move TCP to required joint center, command: Action/Variables/Assign System Variables/ SUTOOL. Then value for SUTOOL has to be set.

If Z is the approach vector and the distance between joint5 and joint6 is for example +230mm, then change Pos to (0,0,230,0,0,0"). In order to create an accurate range for the 2/3 limits, one of several processes may be used. Now the information of the working range can be checked.

(a) If information on the work envelope, indicating any angles (either above or below horizontal), is available, then those values are taken as positive and negative 2/3 limits.

(b) If all information on the work envelope, that indicates any point (x, y) along the outer edge of the envelope is available, then the (x, y) coordinates are used as (x, z) locations. The robot is moved to reach these points (through joint2 and joint3 only). Follow the x and z values in the Teach Pendant. Once a point is reached, the values indicated for joint2 and joint3 should be recorded. If this point cannot be reached because of limited joints2 and 3, one has temporarily to increase/decrease the limit to reach the final point. After having obtained all given points, the largest +/- values are taken and those 2/3 limits made (there is a back strap attached). If no back strap is attached, the largest +/- values of joint3 are taken and made the 2/3 limits. This is a trial and error process. Once the values for the 2/3 limits are obtained, they are turned ON through Robot Settings. Then the maximum and minimum values of the 2/3 limits in Robot Limits are changed, to the new values. The Work Envelope is

then compared to the given envelope. If the envelope is not correct, this process is repeated as many times as needed until the correct values are obtained. In order to move the TCP back to its correct position, the same steps are followed and Pos set back to (0,0,0,0,0,0”).

For robot SK120, the procedure is following: select CSG143; go to Create/Create Dependant Joint; select link2 for parent. Select BKSTRP, and go to Create/Create Dependant Joint; select CSG143 for parent. Go to Edit/Edit Joint, and check type of joint Rotational/Translational, Direction of Axes and expression for Dependant joint (for SK120 it will be AXISPOS (3) for both dependant joints).

Adding a Moving Cable into the model (if needed). The cable must come out from one object and go to another (separate from the first object). The part from which the cable comes out is selected by: Edit/ Edit Object/ Add coordinate Frame.

(a) This coordinate frame can be treated as a joint. Select it and move or turn it accordingly. When turning the coordinate frame, the Z vector is the approach vector. This means that the Z vector must be pointing in the direction that the cable must follow.

(b) Select the part where the cable ends. Repeat steps (b) and (c). To keep twists in the cable to a minimum, try to point the X and Y vectors in the same directions between steps (b) and (c).

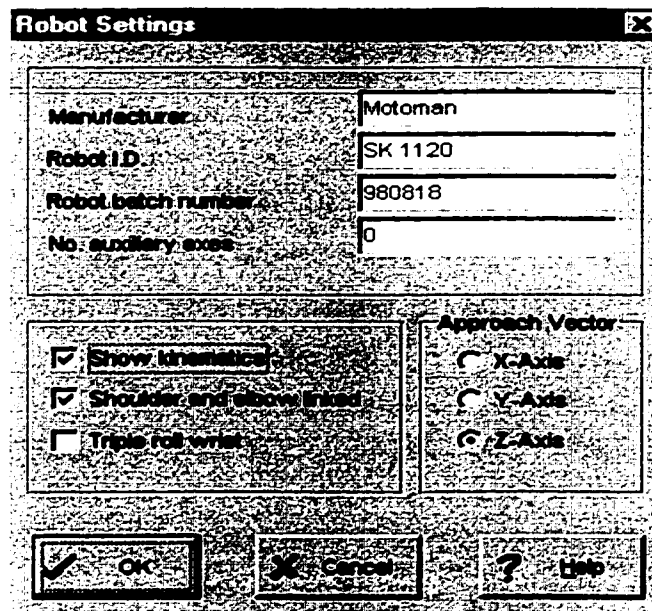
(c) Now the actual cable, can be created by going to Create/Create Moving Cable.

(d) The dimensions dialogue box appears, on the monitor. Modify the dimensions to fit the chosen model.

(e) Attach selected moving cable to the robot (select the part where the cable comes from and where it is ending).

(f) The moving cable can go through all objects! To avoid this situation, move the coordinate frames to a position where this will be avoided.

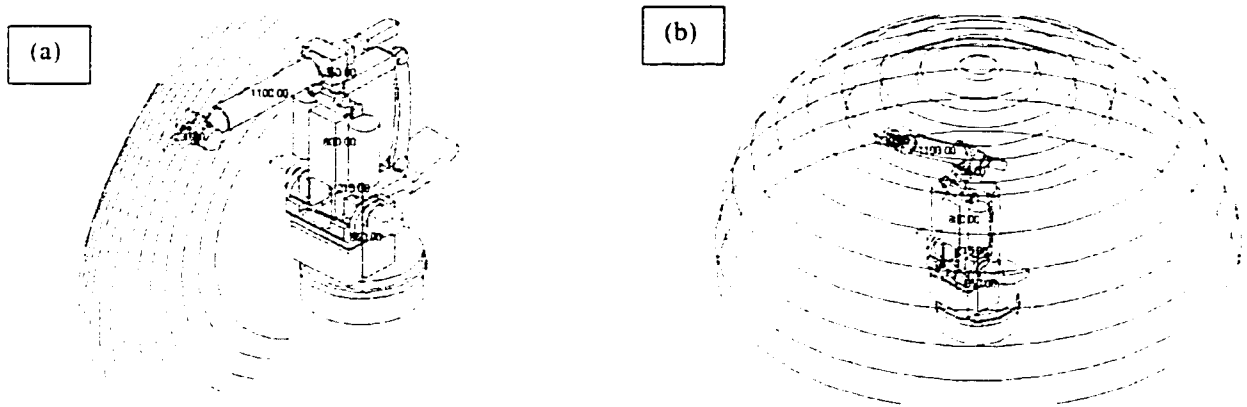
- Move the robot to see if all components are attached properly. Select the base of the robot and go to Edit/Move.
- Check all robot motions for possible collision. If a collision is detected, modify the model so that the robot is not allowed to collide with itself.
- Create the Home position of the robot. Edit/Edit Robot/home Position
- Create the Zero position of the robot. Edit/Edit Robot/Zero Position. Another screen will appear, showing you the coordinates of the robot. Just select exit.
- When the robot model is complete, we must enter the Robot Settings, go to Edit/Edit Robot Settings and fill up the form about the robot for Manufacturer, Robot I.D., Robot batch number, No. Auxiliary axes. Select box for Show kinematics, Shoulder and elbow linked. Choose the approach vector. (See Figure 6).



Robot Settings	
Manufacturer:	Motoman
Robot I.D.:	SK 1120
Robot batch number:	980818
No. auxiliary axes:	0
<input checked="" type="checkbox"/> Show kinematics	
<input checked="" type="checkbox"/> Shoulder and elbow linked	
<input type="checkbox"/> Triple roll wrist	
Approach Vector	
<input type="radio"/> X-Axis	
<input type="radio"/> Y-Axis	
<input checked="" type="radio"/> Z-Axis	
<input checked="" type="checkbox"/> OK	
<input type="checkbox"/> Cancel	
<input type="checkbox"/> Help	

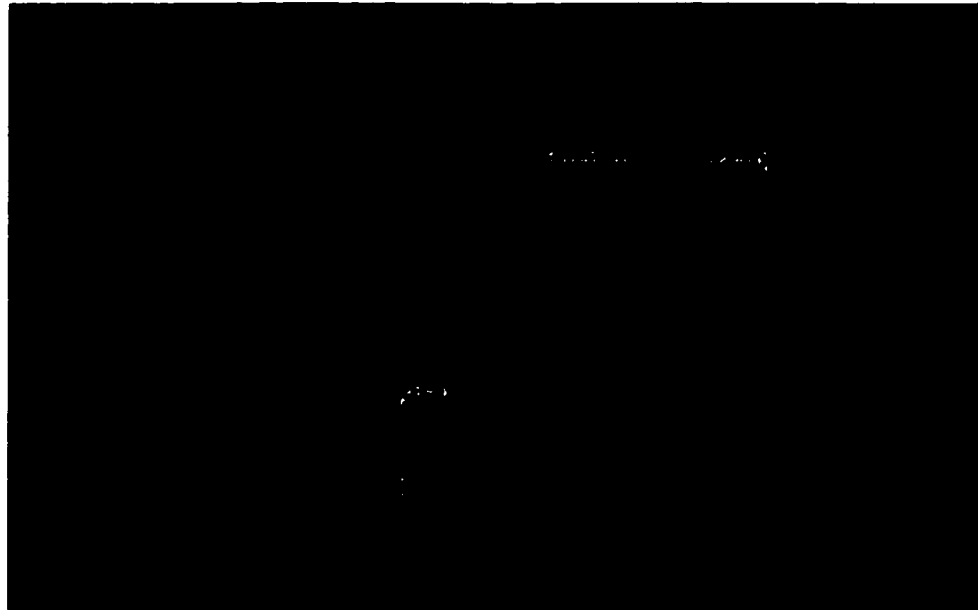
Figure 6 Robot Settings

Check the robot-working envelope, Options/Robot Utilities/Envelope. Workspace asks for the first and then for the second joint to vary. For joint2 and joint3 the reachable envelope is on Figure7 (a), and for joint1 and joint2 is on Figure7 (b). The working envelope is important to show the limits of operation of the robot and also for safety considerations.



**Figure 7 Working Envelope**

- There are circumstances where it is more useful to view a "slice" through the working envelope. Such a "slice" view, known as a "Config Map", shows all possible robot positions in the plane of interest. Put the CP at the world axes, go to Options/Robot Utilities/Create Config Map. Use side view. The CP scrolls vertically across the screen placing small crosses in the working envelope of robot (Figure 8).



**Figure 8 Configuration Map**

When the robot is defined, this object will hold all the information Workspace requires about the robot. This information includes:

- An A-matrix defining the relationship between the robot's joints.
- The name of the inverse kinematics template used by the robot.
- The number of joints.
- The type of each joint (rotational or translational).
- Joint limits.
- Joint maximum velocities.
- Joint maximum accelerations.
- Robot zero position.
- Robot home position.

Robot structures in Workspace, are stored using the Denavit-Hartenberg (D-H) convention. The D-H convention is a means of assigning a co-ordinate system to each link of an articulated chain. This results in a 4 x 4 homogeneous transformation

matrix representing each link's co-ordinate system at its joint, with respect to the previous link's co-ordinate system.

Through sequential transformations, performed by Workspace, the end effector frame can be expressed in terms of the robot base co-ordinates.

The resulting 'A' matrices provide a precise mathematical means of storing the structure of a robot at any position.

### **3. ROBOT PROGRAMMING**

#### **3.1. On-line programming**

On-line programming uses either the walk-through or the lead-through method. In the walk-through method, the programmer moves the robot manually, while information on position, velocity and other related variables is recorded by the robot's control system. This recorded motion can be played back whenever required. In the lead-through method, the robot is moved to a number of desired positions by actuating its drive mechanism, and these positions are recorded by using the teach pendant. The last recorded position is compared with the previous position, and the appropriate move command is automatically generated by the control system of the robot.

In on-line programming, the human programmer may be exposed to an unpleasant atmosphere and the motions taught are, at best, at the programmer's skill level. This method seems to be suitable for pick-and-place or materials handling type of tasks, but it becomes a time-consuming process for operations like arc welding or spray painting, in which the entire path must be taught [Gong, 1998].

#### **3.2. Off-line programming**

In off-line programming, new robot programs can be prepared on a computer and downloaded to a robot without interrupting its production. The major benefits of off-line programming include: reduced downtime, giving greater capability to robots, better understanding of process through simulation, and reduced risk of damage to expensive equipment or of injury to operator. Using either textual languages or graphics-based simulation and programming systems can do off-line programming.

With increased use of industrial robots in a variety of production applications, great attention has been devoted to investigation and development of off-line programming of industrial robots. Advanced integrated off-line programming systems include a CAD modeler, and, many of them also contain the components for a geometric modeler and graphic animation system, an off-line programming language and simulator, and an interface to the target robot system. In general, off-line programming languages should have the following capabilities [Gong 1998]:

- User-definable tasks and subroutines
- User-definable end-effectors and robot arms
- Complex data structures and predefined state variables
- Coordinate transformation between frames
- Runtime definition of variables
- High level instructions for tactile sensors and vision
- Decision-making capabilities allowing the robot to recover from unexpected events
- Use of CAD data

One of the major reasons for the success of robot off-line programming in automotive production is the development of graphical simulation tools. Simulation allows the cell or line to be visualized on the computer screen, particularly the interactions between the moving robots and other production equipment as well as the components, so that potential collisions and incorrect system layouts can be detected [Rooks 1997].



### **3.3. Programming languages of industrial robots**

Presently, there are hundreds of commercially available robot languages. Most of them are based on such classical programming languages as Pascal, C, Modula-2, BASIC, and Assembler. Robot programming languages can be classified according to the robot reference model, the type of control structure used for data, the type of motion specification, the interfaces to external machines, and the peripherals used. The following types of robot programming languages are available [Gong, 1998]:

- Point-to-point motion languages
- Basic motion languages at the Assembler level
- Non-structured high level programming languages
- Structured high level programming languages
- NC-type languages
- Task-oriented languages

## **4. INDUSTRIAL APPLICATIONS FOR ASSEMBLY AND WELDING**

### **4.1. Assembly application**

#### **4.1.1. The application of robotics to assembly**

The term assembly is defined here to mean the fitting together of two or more discrete parts to form a new subassembly. The process usually consists of the sequential addition of components to a base part or existing subassembly to create a more complex subassembly or a complete product. As such, assembly operations involve a considerable amount of handling and orienting of parts to mate them together properly. The difference between assembly tasks and other material-handling tasks is that value is added to the product through the assembly operation. Also, there are often interactions that take place between two parts being assembled, between the gripper and a part, and between other elements of the workcell. When parts are fastened together (called parts joining), there are often additional interactions with the medium used to join the components. All of these potential interactions can make assembly operations considerably more complex compared to the simpler task of moving a part from one location to another.

There are a variety of assembly processes used in industry today. These include mechanical fastening operations (using screws, nuts, bolts, rivets, etc.), welding, brazing, and bonding by adhesives. Some of these processes are more adaptable to automatic assembly.

There is a growing interest in automated assembly because of the high manual labor content of most assembly operations today. Automated assembly systems have

traditionally been applied to high-volume products in which a large investment is made in custom-engineering equipment, designed to perform the specific operations required for those products. However, there is a large number of products, representing a majority of the assembly operations performed in the U.S.A., where the volume of production is low or medium. In these cases, it is not economically feasible to make large investments in specialized assembly equipment. Programmable and flexible systems, including robotics, must be applied to these low-and medium-volume assembly operations if automation is to be successfully achieved [Critchlow, 1985].

#### **4.1.2. A Workspace model of an automotive assembly workcell**

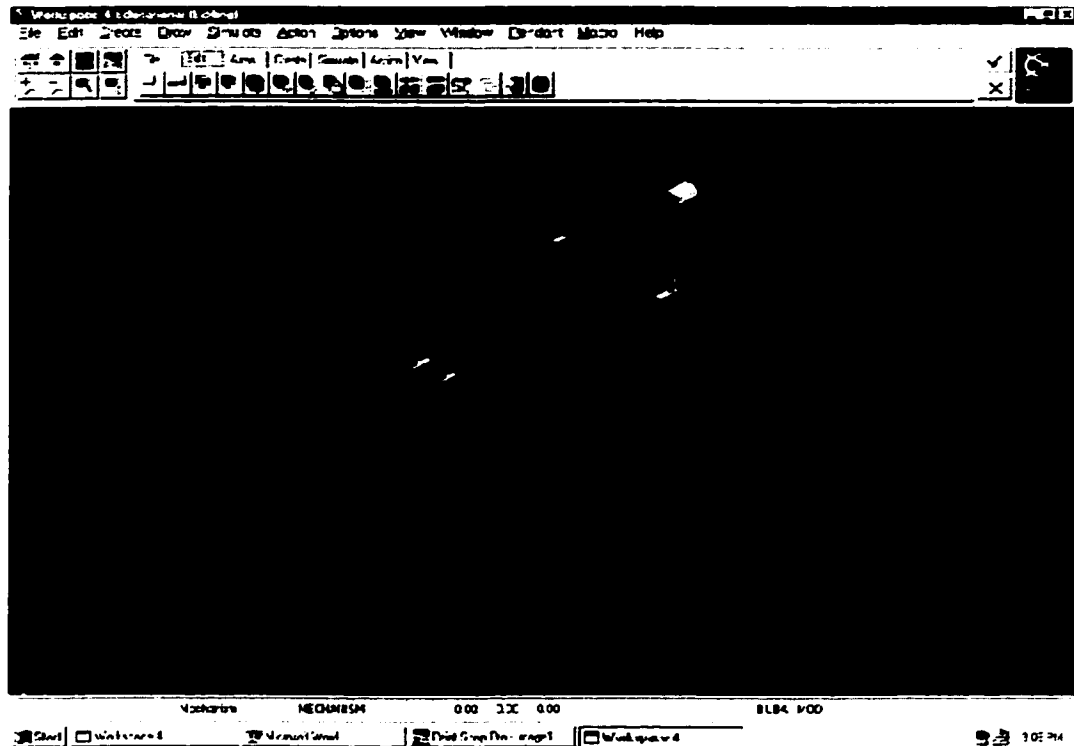
##### **Mechanism (fixture)**

##### **Introduction**

Mechanisms are non-robotic devices found in a workcell, such as turn tables, conveyors, AGV's, stamping presses, etc. These devices can be simulated and given motion in Workspace. By simulating mechanisms, a more realistic simulation can be created. Mechanisms are simple robots without the inverse kinematics, and therefore share some of the same menus.

##### **Strategy**

- Create boxes, cylinders, to represent the fixture and finally fender on the top of fixture (made of 400 boxes), (Figure 9).

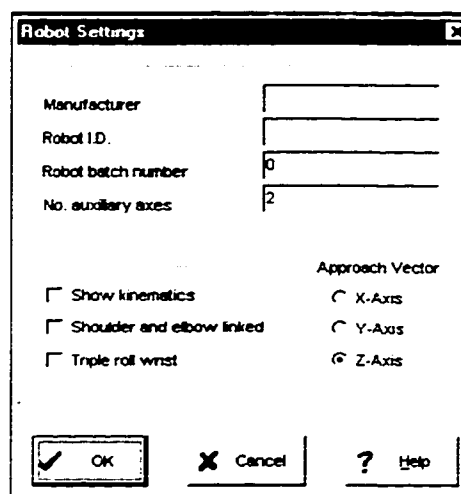


**Figure 9 Fixture**

- Attach the named parts correctly is extremely important, because this is required for correct motions. The attachment takes the form of a “parent-child” relationship. Each child may only be attached to one parent, but a parent may have many children. It is not permitted to attach a child to one of its ancestors, since that would create a circular loop, and this would lead to ambiguities during object movement etc. If a parent object is moved or turned then all of its children are moved or turned with it. If a child object is moved or turned, its parent object is not. The attachment therefore is in one direction only.
- Select the one leg for fixture base (named mechanism).
- Select the Create/Create Mechanism menu.
- Enter 2 for the number of Auxiliary Axis. One is for translation in X-direction, and second for translation in Y-direction.

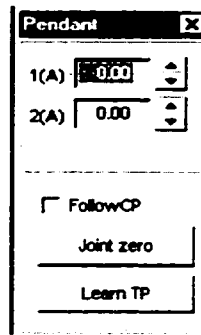
- There should be the name “mechanism” in the lower right-hand corner of the screen and the pendant will show two axes. A mechanism has been created, but the joints in the mechanism have yet to be defined.
- Select the slider on a right side (named link1), and select right, front, top leg (named link2). Now define translation in X-direction. For the translation in Y direction, select first right side (in left part of the fixture - named link3) and select left, front, top leg (named link4).
- Select link1 and go to the Create/Create Dependant Joint icon. Chose box45 as a parent object. Select link2 and go to the Create/Create Dependant Joint icon. Chose box73 as a parent object. Select link3 and go to the Create/Create Dependant Joint icon. Chose box36 as a parent object. Select link4 and go to the Create/Create Dependant Joint icon. Chose box88 as a parent object. Press enter always after having selected the object.
- A dependent joint is any joint that is not part of the serial robot structure linking the base to link1, link1 to link2, and so on. Dependent joints provide the user with a great flexibility in defining joint movement of objects in the workcell that are not parts of the main robot’s serial linkage. When a robot controls this movement, it is just referred to as a dependent joint and is usually controlled via auxiliary axes. If the structure needs to be a stand-alone system, it is called a mechanism. Workspace treats a mechanism as a special kind of robot that only has auxiliary axes.
- Select link1 and go to Edit/Edit joint/Type of joint, and select translational. Repeat the same for the other three joints.

- The position of a dependent joint is controlled by an arithmetic expression. This expression may be entered using the Expression of Dependent Joint. Select link1 and go to Edit/Edit joint/Expression of Dependent Joint.
- For link1 and link3 AXISPOS (1) has to be typed because this represents translation in the X-direction (Figure 9). For link2 and link4 AXISPOS (2) has to be typed because they represent translation in the Y-direction.
- Go to Edit/Edit Robot/Robot Settings. For number of auxiliary Axes type 2. For approach vector type Z, (Figure 10).



**Figure 10 No. of auxiliary axes**

Robots in Workspace may have auxiliary axes associated with them. These additional axes will appear on the pendant as “n (A)” (Figure 11), where n is the number of the axes. Auxiliary axes may be used to control axes on the tool, a positioner, slider or turntable.



**Figure 11 Pendant for Fixture**

- Select every link, one by one and orient the approach vector z in the desired direction. Go to Edit/Edit joint/Turn axes.

### Outcome

- A fully operational mechanism has been created. The fixture can be moved by decreasing or increasing the value of joint1 and joint3 together or of joint2 and joint4 on the pendant menu.
- Joint limits can be set through the Edit/Edit Robot/Limits, (Figure 12).

Joint limits			
Min1: -150.00	Max1: 150.00	Mxv1: 200.00	Mxa1: 400.00
Min2: -150.00	Max2: 150.00	Mxv2: 200.00	Mxa2: 400.00
Mxlv: 400.00	Mxla: 400.00		

**Figure 12 Joint limits for fixture**

Min1, Min2: is minimum rotation (in degrees) for joint1 and joint2

Max1, Max2: is maximum rotation (degrees) for joint1 and joint2

Mxv1, Mxv2: is maximum velocity (degrees/second) of joint1 and joint2

Mxa1, Mxa2: is maximum acceleration (degrees/second<sup>2</sup>)

Mxlv: maximum linear trajectory velocity (mm/second)

Mxla: maximum linear trajectory acceleration (mm/second<sup>2</sup>)

## **Robot Comau-S2**

### Strategy

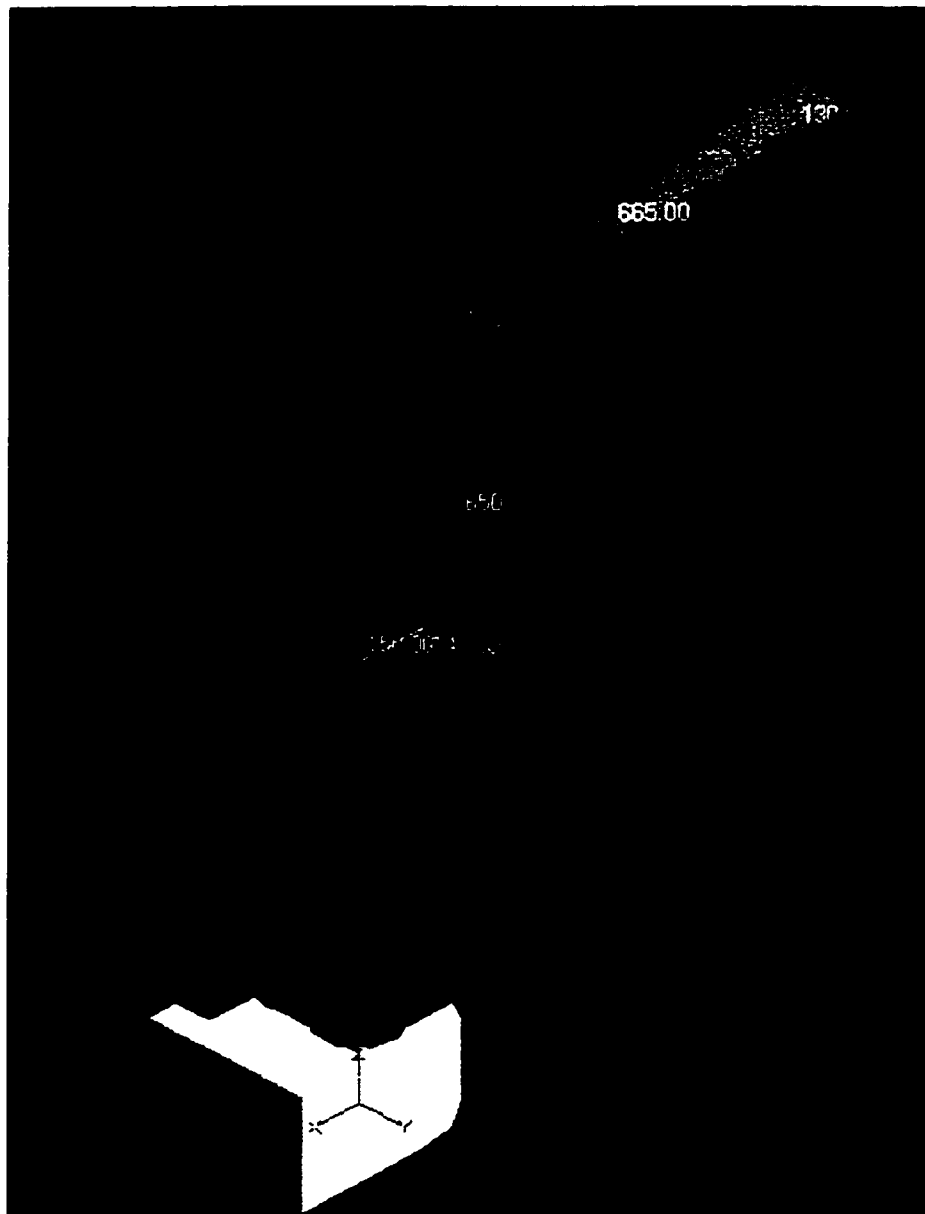
- Create different solids to represent shape of robot.
- Attach and name parts correctly: Link1, Link2... Link6.
- Select robot base and named robot.
- Activate the Create Robot menu, and confirm 6 as the number of joints.
- Once the axes are positioned graphically, it is usually best to make sure that the 'A'-matrices numbers are exact. To show these numbers we can use Edit/Edit Robot/Exact kinematics command (Figure 13).

A matrix constants			
Th1: 0.00	D1: 0.00	A1: 150.00	Al1: -90.00
Th2: 0.00	D2: -147.00	A2: 650.00	Al2: 0.00
Th3: 0.00	D3: 0.00	A3: -100.00	Al3: 90.00
Th4: 0.00	D4: 665.00	A4: 0.00	Al4: -90.00
Th5: 0.00	D5: 0.00	A5: 0.00	Al5: 90.00
Th6: 0.00	D6: 130.00	A6: 0.00	Al6: 0.00
Tw1: 0.00			

**Figure 13 A matrix for robot Comau-S2**

Check the Show kinematics box. The dimension of the robot model, i.e. the distance and offsets between the joints of the robot, is shown in Figure 14.





**Figure 14 Dimensions of robot Comau-S**

#### **4.1.3. Off-line programming for the assembly operation using Karel 2 language**

##### **Introduction**

One of Workspace's main features is off-line programming. Workspace writes all of its robot commands in the robot's native language. All robot language commands are found under the Action main menu. There is no need to post-process a program written in Workspace; it can be downloaded directly to a robot controller.

This capability also enables the user to upload and modify a program currently running on a robot (mechanism). All the popular robot languages are available.

### **Strategy**

- Start from the Pendant menu. Go to learn TP (teach points) on a Teach pendant, save teach points file as B1#.kl
- Go to Simulate/Start Track.
- Give a name to the track: B3.kl
- Select Action/Begin/Robot Move Commands/Move Aux. Now chose Aux positions for a fixture, that was saved in B1#.kl file.
- When the track is finished, close the Robot Move Commands menu and select End to end the track.
- After selecting the Simulation/Edit Teachpoints icon, the teachpoint file should appear in the Editing screen.
- Selecting Edit Track icon, the Karel2 program should appear in the Editing screen.

Workspace is also capable of creating AVI files. These animation files can be replayed using the media player supplied with Microsoft Window, or using Workspace's internal Media Player. AVI files are slow to create, but will play back in real time and can be resized.

- Go to Simulate/Replay track and Create AVI. Choose B2.kl for the track file, B1#.kl for teach point file, and B2.AVI, for AVI file.
- The last step is to go to Simulate/Media Player. The fixture should move through the teachpoints in order, then go back to HOME POSITION.

#### 4.1.4. Running multiple programs simultaneously for robot and mechanism

##### Introduction

Workspace has the ability to run more than one program simultaneously. This is useful when there is more than one robot or device in a workcell. In this project we have robot and fixture.

The main program (track for robot) is referred to as the “Foreground Track”, and all other programs (track for fixture) are referred to as “Background Tracks” (Figure15).

The screenshot shows the 'Simulate Options' dialog box with the following settings:

- Track file: C:\PROGRAM FILES\WORKSPACE 4\TRK\AD1.KL
- Teach point file: C:\PROGRAM FILES\WORKSPACE 4\TRK\AD1#.K
- AVI file: C:\PROGRAM FILES\WORKSPACE 4\AVI\AD1.AVI
- Background Tracks (1): C:\PROGRAM FILES\WORKSPACE 4\TRK\B2.KL
- Stack Size: 8192
- Real time interval: ☐
- Equal intervals: ☐
- Frame Interval: 0.5
- Interp. Interval: 0.5
- Robot: ROBOT
- Language: KAREL 2
- End Condition: FALSE

Buttons at the bottom: OK, Cancel, Help.

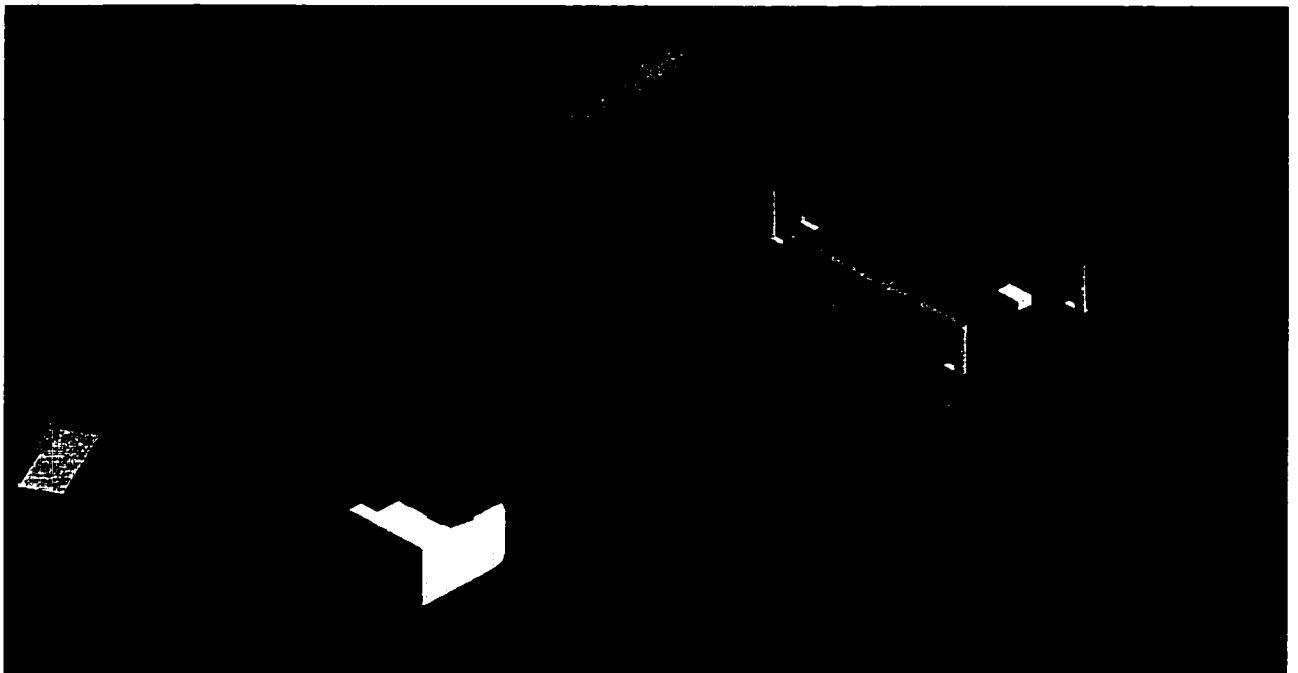
Figure 15 Simulation Options

There is no limit to the number of background tracks in a simulation. All of the programs do not have to be written in the same language. Communication between programs is accomplished through signals.

The two main types of signals are “Digital Outs” (DOUT) and “Digital Ins” (DIN). Before a program can use a DIN, the signal must be defined through a Workspace command called SIGNALDEF.

## Strategy

- Go to File/Load model, ADFINPROJECT1.mod. Then again go to File/Load model and add to current model; choose BUBA3.mod. Now fixture and robot are in one file.
- This step is very important to attach the mechanism to the robot. The first idea for attaching is to attach the mechanism base to the robot base, but it will not give a solution for this problem. The second idea is to attach links parents to the robot base: Link1 is attached to box45 and box45 will be put together with the robot base; Link2 is attached to box73 and box73 will be put together with the robot base; Link3 is attached to box36 and box36 will be put together with the robot base; Link4 is attached to box88 and box88 will be put together with the robot base; (Figure 16).



**Figure 16 Workcell model for assembly operation**

- Now it is necessary to “connect” the device to the simulated robot controller. There are already two auxiliary axes, and when the Pendant menu is pulled down, additional axes are displayed. For a six axes robot, this axis will appear as “7(A)” and “8(A)”. Using Edit/Edit joint/Expression for dependent joint, for link1 and link3 one has to type AXISPOS (7), because they represent translation in X-direction; for link2 and link4 one has to type AXISPOS (8), because they represent translation in Y-direction. Now the joint depends upon the values “7(A)” and “8(A)” on the Pendant menu. Finally we can show joint limits for the mechanism and robot are brought together (Figure 17).

Joint limits			
Min1: -152.00	Max1: 152.00	Mxv1: 200.00	Mxa1: 400.00
Min2: -195.00	Max2: 15.00	Mxv2: 200.00	Mxa2: 400.00
Min3: -35.00	Max3: 245.00	Mxv3: 200.00	Mxa3: 400.00
Min4: -180.00	Max4: 180.00	Mxv4: 200.00	Mxa4: 400.00
Min5: -125.00	Max5: 125.00	Mxv5: 200.00	Mxa5: 400.00
Min6: -210.00	Max6: 210.00	Mxv6: 200.00	Mxa6: 400.00
Min7: -150.00	Max7: 150.00	Mxv7: 200.00	Mxa7: 400.00
Min8: -150.00	Max8: 150.00	Mxv8: 200.00	Mxa8: 400.00
Mxlv: 400.00	Mxla: 400.00		

**Figure 17 Joint limits for robot and mechanism**

There are already two separate programs for robot (AD2.kl) and mechanism (B3.kl) tracks. Now it is necessary create a signal from the robot to the mechanism. Thus, the mechanism starts moving after the robot stops (Figures 18a and 18b).

```

PROGRAM AD2 (for robot)
-- ! LANGUAGE KAREL 2
-- ! MEMORY 8192
-- ! ROBOT
-- TEACHPOINT DECLARATIONS
VAR
  TP1 : POSITION
  TP2 : POSITION
  TP3 : POSITION
  TP4 : POSITION
  TP5 : POSITION
  TP6 : POSITION
  TP7 : POSITION
  TP8 : POSITION
  TP9 : POSITION
  AUX1 : AUXPOS
  AUX2 : AUXPOS
  AUX3 : AUXPOS
BEGIN
  SUSEMAXACCEL=TRUE
  %INCLUDE AD1#
  MOVE TO TP2
  MOVE TO TP3
  MOVE TO TP4
  MOVE TO TP5
  CLOSE HAND 1
  -- ! GRASPOBJ 'CSG18'
  MOVE TO TP6
  MOVE TO TP7
  MOVE TO TP8
  OPEN HAND 1
  MOVE TO TP9
  MOVE TO SHOME:SUTOOL
  --!Attachobject.1,'CSG18','CYLINDER50'

  DOUT[1]=ON
  --!SIGNALDEF DIN[2],TRK\B3.KL,2
  WAIT FOR DIN[2]=ON

  MOVE TO TP7
  MOVE TO TP8
  -- ! Detachobject.1,'CSG18'
  CLOSE HAND 1
  -- ! GRASPOBJ 'CSG18'
  MOVE TO TP7
  MOVE TO TP3
  MOVE TO TP4
  MOVE TO TP5
  OPEN HAND 1
  MOVE TO TP4
  MOVE TO TP3
  MOVE TO TP2
  MOVE TO SHOME:SUTOOL
END AD2

```

**Figure 18(a) Robot program**

**PROGRAM B3(for mechanism)**

```
-- ! LANGUAGE KAREL 2
-- ! MEMORY 8192
-- ! ROBOT
-- TEACHPOINT DECLARATIONS
VAR
  TP1 : POSITION
  TP2 : POSITION
  TP3 : POSITION
  TP4 : POSITION
  TP5 : POSITION
  TP6 : POSITION
  TP7 : POSITION
  TP8 : POSITION
  TP9 : POSITION
  AUX1 : AUXPOS
  AUX2 : AUXPOS
  AUX3 : AUXPOS
BEGIN

--!SIGNALDEF DIN[1],TRK\AD2.KL,1
WAIT FOR DIN[1]=ON

  SUSEMAXACCEL=TRUE
  %INCLUDE AD1#
  WITH SMOTYPE=JOINT
    MOVE AUX TO AUX2
  WITH SMOTYPE=JOINT
    MOVE AUX TO AUX2
  WITH SMOTYPE=JOINT
    MOVE AUX TO AUX3
  WITH SMOTYPE=JOINT
    MOVE AUX TO AUX1

DOUT[2]=ON

ENDB3
```

**Figure 18(b) Mechanism program**

## **4.2. Welding application**

### **4.2.1. The application of robotics to spot welding**

Spot welding was one of the first applications of robots because the required accuracy and speed were available in early hydraulic robots, and the robot's reach and load-carrying capacity were far superior to those of a human operator. In spot welding, two pieces of conductive metal are joined together by passing a large electric current through them. This high current heats the contact point sufficiently to cause local melting so that molten metal is formed momentarily. The current is on for a short time only (pulsed), so that the molten metal quickly solidifies, forming a strong joint.

Overheating occurs at the joint because of the resistance of the metal to the passage of current through it. This is often called the I<sup>2</sup>R heating effect.

I-current (amperes)

R-resistance (ohms)

The robot end effector is the welding gun, which has powerful pincer electrodes to force the metal together and provide good electrical contact during the welding operation. Power for forcing the electrodes together is usually pneumatic or hydraulic from a separate power source.

Computer-controlled sequencing of the robot and the spot-welding operation is used today to improve the quality of the welds by accurately controlling the duration and intensity of the current pulses used.

Fabrication of automobile bodies is the most common applications of spot welding. Over 1200 spot-welding robots are used in the automobile application at present time.



The weight of the spot-welding gun and cable system was reduced by about a half, so that smaller robots could be used and the improved accuracy of electric drives could be obtained [Critchlow, 1985].

A successful spot-weld is one that is normal to both surfaces - clearly a geometric problem. Many cells are still designed by the 'chalk' method where a man with a piece of chalk marks crosses on the part where the spot-weld is required before a robot programmer uses a teach pendant to teach each spot weld point and its approach and depart points.

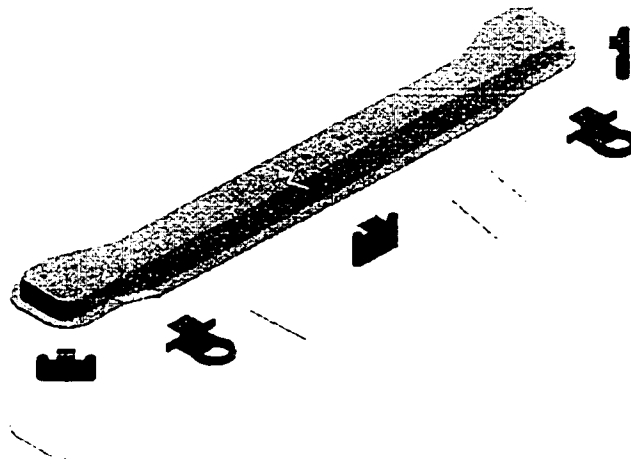
Workspace has commands for orienting geometry points (which are like robot teachpoints except they are stored as part of the geometric model of the cell) normal to a surface. The points can be turned automatically about their approach vectors until they become achievable - all the designer needs to do is to specify a point on the surface and the distance from the surface and Workspace does the rest. The best position for the robot relative to the part can also be automatically chosen to make all the points achievable.

The problem becomes more complex when several robots are working in close proximity within each other's working envelope (the volume enclosing all the points that the robot can physically reach). The task becomes a question of how can all the spot-welds be reached in the shortest possible time, thereby ensuring that the cell does not create a bottleneck on the production line. Collisions between objects are highlighted in red so that they are clearly visible within the simulation and can be avoided on the real robot.

Workspace stores geometry points linked together in a path. Sections of each robot path can be cut and pasted between robots to try the effect of allocating different spot-welds to different robots. A Cycle time can be reported either for a given

sequence of movements or for the overall cycle. Statistics for the effect on throughput of different cycle arrangements can be viewed graphically using a Gantt (bar) chart and the resulting graph printed out together with a report validating the cycle time. The effect of machine downtime and repair time can also be introduced using customizable probability distributions and the effect of random fluctuations in production over a year investigated [Owens, 1994].

#### **4.2.2. Problem description of Industrial application for assembly and welding**

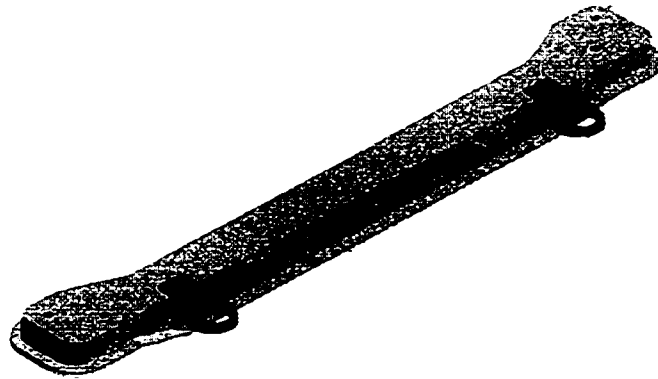


**Figure 19 Parts for Front support member engine/ transaxle mount**

Front support member engine/transaxle mount has nine different parts. They are all made of sheet metal (Figure 19).

The problem is to design an automated cell that performs the welding and assembly operations, considering the requirements from manufacturer.

The final product is shown on (Figure 20).



**Figure 20 Front support member engine/ transaxle mount**

The workcell under consideration is a collection of industrial equipment that is spatially arranged for spot welding operations. It usually consists of a robot or several robots with welding guns, positioner or conveyor, a workpiece, welding fixture, control and sensing system.

Tehnological operation is described as a welding gun motion along given points (spot welding) as well as motions between them.

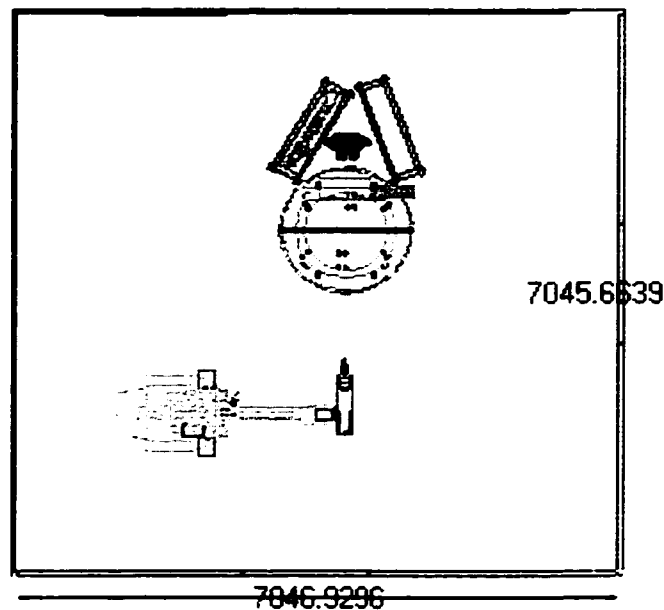
A general formulation of the problem can be stated as follows: for a given set of welding dots along with a complete model of the robot world, find a robot motion allowing to reach the goal without any collisions between the robot, welding tool and the objects belonging to the robot workspace.

The main difficulty in solving the general problem lies in its high algorithmic complexity. So a more “practical” way is to consider several instances of the problem that can be solved using specific approaches. As a result, the problem is decomposed into the following tasks: to determine the number of robots in a workcell and their

relative positions, to choose workpiece position, to select welding guns, to distribute welding targets between robots, and to find the corresponding robot path.

#### 4.2.3. Proposed scenario for workcell layout design

Having in mind the previously given constraints and requirements, it was decided to choose in this work the following equipment in the simulation (Figure 21).



**Figure 21 Proposed scenario for workcell layout design**

- ABB IRB 6000 robot for spot-welding. This robot is required by the customer.
- Welding gun from 'Centerline' model no: CLTG-9535-12 (for spot welding). The gun is appropriate for this application, because it can reach all points on the parts.
- Rotary table from 'Ferguson' model: HPM 60". The size of the table is suitable for size of the fixture, which is used for positioning, parts. The length of part is over 1m, and on the rotary table must be enough space for positioning two parts at

the same time.

- Part local storage area must be big enough for one pallet of prefinished parts and one pallet of finished parts.
- Calculation for cell cost: Robot =\$150,000

Rotary Table =\$30,000

Fixture =\$5,000

Welding Gun =\$20,000

Pallets for prefinished parts =\$5,000

Pallets for finished parts =\$5,000

Operators salary = \$70,000

---

Sum = \$285,000 < \$600,000

Prices and costs are obtained from manufacturer (ABB, Ferguson, Centerline, and Robot Simulations Ltd.)

- Calculation for number of parts in the production:

Cycle time = 149 sec. per part

Pieces per hour:

@ 100% →  $(3600\text{sec}) / (149\text{sec}) = 24$  pieces per hour.....( 1 )

@ 70% →  $[(3600\text{sec}) / (149\text{sec})] \times 0.7 = 17$  pieces per hour.....( 2 )

Pieces per shift:

@ 100% →  $(8 \text{ hours}) \times (24 \text{ pieces}) = 192$  pieces per shift .....( 3 )

@ 70% →  $(8 \text{ hours}) \times (17 \text{ pieces}) = 136$  pieces per shift.....( 4 )

Pieces per year:

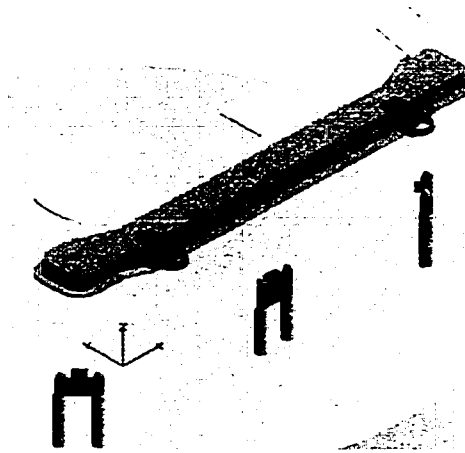
5 days X 8 hours X 17 pieces per shift X 48 weeks = 32,640 pieces per year.....( 5 )

To satisfy yearly requirements of 40,000 pieces, production has to run 10 hours shift:

**5 days X 10 hours X 17 pieces per shift X 48 weeks = 40,800 pieces per year...(6)**

- The size of the cell matches in required dimensions of 24 X 30sq.ft (8000mm X 10000mm), (Figure 22).

The next problem is how to handle and position the parts that the robot will operate on. For positioning the part we recommend location pins on a rotary table (Figure 22). Operator will handle the parts from local storage to the rotary table.



**Figure 22 Pins for positioning parts**

#### **4.2.4. Operation sequences**

The workcell includes a robot IRB6000 for spot welding, welding gun, operator for manual handling and assembly, rotary table with fixture for positioning parts, local storage for parts, and pallets for finished parts.

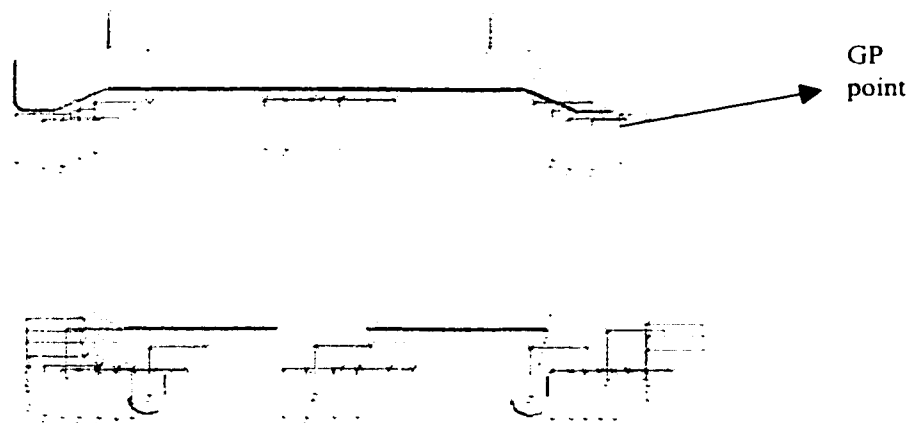
For this workcell the operation sequence will be:

- The operator will handle parts from local storage and positioned on rotary table
- The table will rotate for 180 °
- The robot will weld parts

- The table will rotate for 180 °
- The operator will put three plates on the top of other parts (it is still positioned with same pins)
- The table will rotate for 180 °
- The robot will weld that three plates
- The table will rotate for 180 °
- The operator will take finished part and put on pallet

#### 4.2.5. Defining robot positions on the modeling parts

The easiest and most powerful way of defining robot positions is to use the geometry of the objects. Points may be defined normal to a surface so that the position on the surface and the distance from the surface to the point is set by the user, using the Create/Create GPs on Surface command. As the point is moved over the surface it is always maintained normal to the surface. This is of use in applications such as spot-welding. Figure 23 shows geometry points on the modeling part that the robot will follow in the track program.

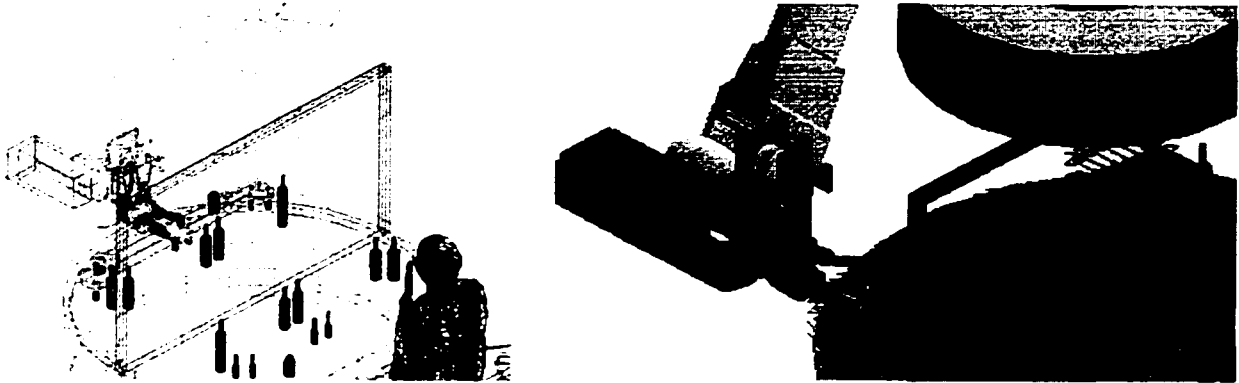


**Figure 23 Geometric Points (GP) points on surface**

One can then see immediately on the screen if the GPs are reachable. They will be green if they are reachable, or red if they are not reachable.

By running a track one can see how a robot follows GP points on the surface.

Figure 24 represents wireframe and shade view of robot's following GP points.

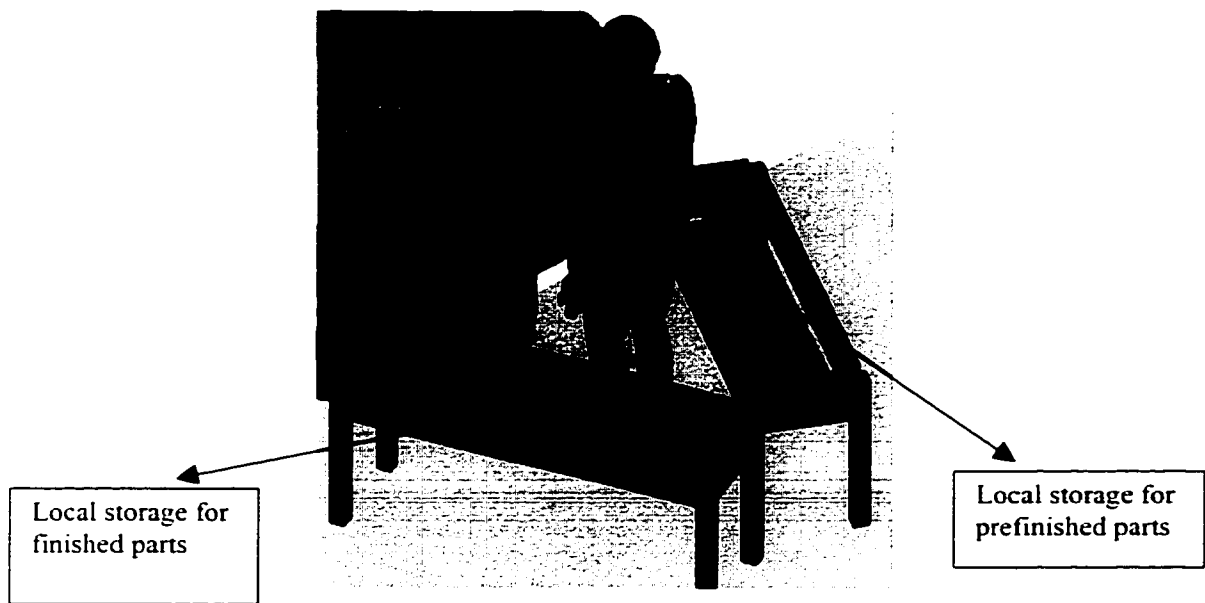


**Figure 24 Robot following GP points**

#### **4.2.6. Solution for storing parts**

Local storage is used for storing parts in a workcell. In this simulation the representation of local storage is very simple, but in real production it is supposed to be big enough for one shift (eight hours). The first storage is for raw material and the second storage is for finished parts (Figure 25). The size of local storage must be enough for  $136 \times 9 = 1224$  prefinished parts. The storage for finished parts must be organized for 136 finished parts.

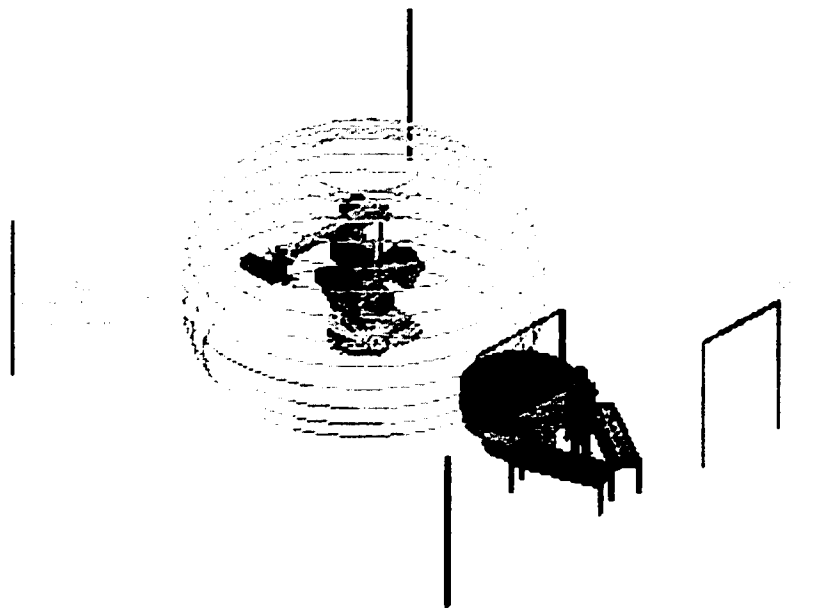




**Figure 25 Local storage**

#### **4.2.7. Envelope plotting**

Workspace has a capability to plot the envelopes of all the robots in the workcell in three dimensions, as shown on Figure 26.

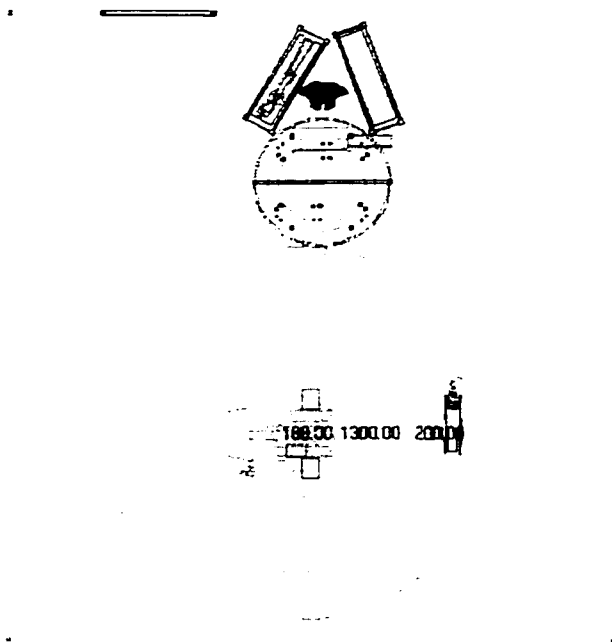


**Figure 26 Envelope plotting**

This enables to check which areas of the workspace may be deemed safe for a user to work in (i.e. outside that volume which the robot is physically capable to reach), or to check that there is no overlap between the envelopes of two robots or a robot and other moving equipment. This option is important for improved planning of the installation for safety.

To get the working envelope, use the Options/Robot Utilities/Envelope command. Enter first joint variable to vary 1, and second joint variable to vary 2, and the robot will achieve its maximum reach.

Display the Envelope for the robot, varying joints 1 and 2, and check if the whole equipment lies within that envelope, viewing from the top of the workcell. This will provide a quick first layout design of the workcell (Figure 27).



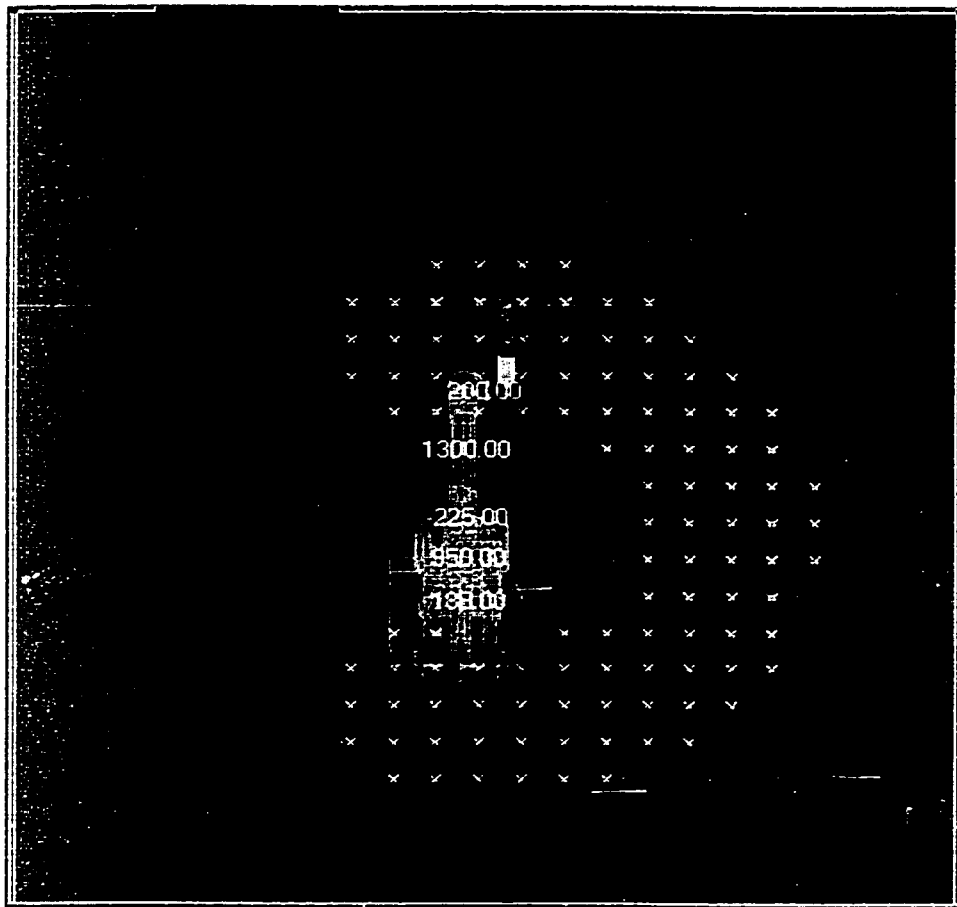
**Figure 27 Layout design of workcell**

A more sophisticated approach to robot and workcell layout design is to use the 'Config Map'. This is more detailed than an Envelope, because it shows only positions that the robot can reach with a particular orientation. To see all the positions

that robot can reach with the tool, in the z-elevation corresponding to the top of the workpiece, one has to do following steps:

- Point the robot in the wanted orientation.
- Snap the CP onto the GP on the top of the workpiece, to define the z-elevation of the configuration map.
- Change to top view and zoom out to define that the a configuration map is seen.
- To activate config map go to Option/Robot Utilities/Create Config Map.

Now, it is possible to see the area where the robot can reach the workpiece and one may decide to make some small adjustments in the position of the workpiece or turntable (Figure 28).



**Figure 28 Configuration Map**

#### 4.2.8. Creating an animation

An animation is created when track files are “replayed”. The track files are programs written in the appropriate language for robots, mechanisms, operators, or any device in workcell. In this simulation it is Karel2 language. Before running the simulation one has to go to the Simulate/Simulate Options submenu to select language, the teach point file, and the track files. ROBOT.kl is the foreground track, the AZUR.kl and TABLE.kl are background tracks, and ANA-ALL#. kl is the teach point file. Also one has to select the type of animation file, in this case it is an AVI file. The Frame Interval (the number of frames calculated per second of simulation time) is 0.5s. If one wants to move the simulation images at the same time as “real life”, then he should select Real time interval check box. Workspace tries to synchronize the cycle time with PC’s clock time (Figure 29).

Simulate Options

Track file C:\MAGISTRATURA\CHANGE 4\ROBOT.KL Browse

Teach point file C:\MAGISTRATURA\CHANGE 4\ANA-ALL#.KL Browse

Animation file C:\MAGISTRATURA\Change 4\WONAME.AVI Browse

Background Tracks (2)

C:\MAGISTRATURA\CHANGE 4\TABLE.KL  
C:\MAGISTRATURA\CHANGE 4\AZUR.KL Add Delete Clear

Animation Format  
☒ AVI ☐ FLC

Stack Size 8192

Robot IRB6000

☐ Real time interval ☒ Equal intervals

Frame Interval 0.5

Interp Interval 1

Language KAREL 2 Select

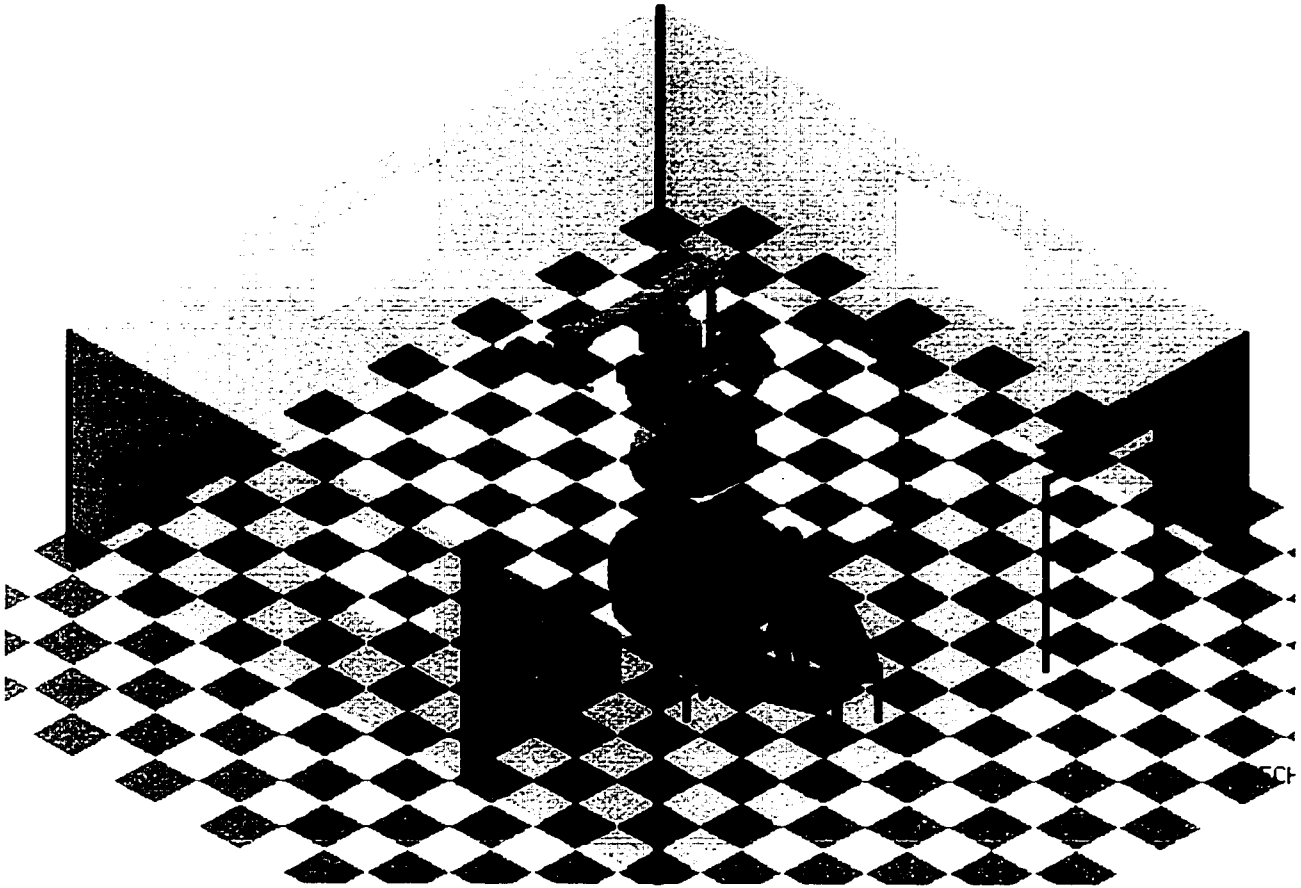
End Condition FALSE

✓ OK ✗ Cancel ? Help

Figure 29 Simulate Options

When all file names are set, one can replay the track and create an AVI file.

Go to Simulate/Replay Track and Create Animation (Figure 30).



**Figure 30 Spot-welding application workcell**

## **5. DETECTION OF POTENTIAL COLLISIONS AND OPTIMIZING SYSTEM LAYOUTS IN WORKSPACE**

### **5.1. Throughput statistics**

The statistics module in Workspace allows making accurate estimates of a process cycle time. Effects such as run-in time, down-time, repair-time and machine storage buffers may be included in the simulated production run. The results are presented in text files containing statistics on each activity generated and Gantt charts of time activity plotted. Those results are important in identifying unnecessary delays in the process and adapting the model and/or track files accordingly [User Guide Manual, Workspace4, 1997].

### **5.2. Preparing a statistics run**

Workspace has three commands that should be inserted at appropriate places in the track files. They are all available on the Action/WS Miscellaneous submenu, or can be manually inserted to the existing tracks. Those commands are: “LABELUSAGE”, “LABLEVENT”, and “ENDOFJOB”.

“LABELUSAGE ” tells Workspace which statistics text file relates to which robot or mechanism. The string after this command is the robot or mechanism’s name, and the integer (2, 3, 4, etc.) represents the text file. The declarations need only be made once, usually at the start of the main track. For example:

```
--! LABELUSAGE MAN, 2
```

```
--! LABELUSAGE ROBOT, 3
```

```
--! LABELUSAGE TABLE, 4
```

The values 0 or 1 are reserved in Workspace for wait time and breakdown respectively.

“LBELEVENT”, inserts a line in one of the statistics text files. The string is the text to be written and the integer indicates in which text file the line will be placed into, as define by “LABELUSAGE ”. For example:

```
--! LBELEVENT 'Go pick up part from pallet', 2
```

```
--! LBELEVENT 'Go to weld parts', 3
```

```
--! LBELEVENT 'Turn table for 180 degrees', 4
```

```
--! LBELEVENT 'Wait for man to pick up parts from pallet', 0
```

“ENDOFJOB” in a track tells Workspace that the track has completed one cycle of the process. The command will therefore appear in several. This command has no effect if the process is still in its run-in time. For example:

```
--! ENDOFJOB
```

Workspace replays a track (or set of tracks) over the run-in period as specified on the throughput menu. When an “ENDOFJOB” command is encountered in the track for the first time after the run-in time then Workspace starts a counter of counting the number of jobs. Each subsequent time the job is encountered, the count is incremented. At the end of the run time as specified on the throughput menu a report is generated in a text file whose name can be specified by the user. This report contains information on the number of jobs, the run-in time, the run time, and the number of jobs per hour.

The downtime settings on the throughput menu are also written into the file. These settings can be used to cause tracks to simulate a breakdown by entering a wait state at random, in accordance with the efficiency of the track, as specified on the throughput menu.

The ENDOFJOB command is also used in recording usage for one cycle (from the end of one job to the end of the next job). The usage is recorded as timings in text files, which have the same name as the track files, and which are stored in the track directory. These timings are created by each LABELEVENT command encountered in the track. They include a step number, a description of the action as defined by the LABELEVENT command, the duration of the action, the type of usage as referenced by a usage number in the LABELEVENT command. This number refers to a usage string defined by the LABELUSAGE command. LABELUSAGE commands should always be at the beginning of one of the tracks as their only purpose is to make a LABELEVENT usage number refer to a usage string. This allows putting a useful comment when a particular part of the cycle is encountered. Once a replay of the tracks has been made then a bar graph showing each track as a horizontal bar representing time passing along the x-axis is also produced. Each bar changes color as the usage changes for that track. A table is displayed above the graph, showing the percentage of each type of usage over the track.

### **5.3. Analyses of workcell using Workspace**

To analyze the workcell we should insert LABELUSAGE, LABELVENT and ENDOFJOB commands into the tracks. This will then automatically generate a report at the end of the simulation to show how the cycle time breaks down into individual portions. Then we can view the behaviour of the cell when the robot or turntable, men break down by turning on usedowntime.

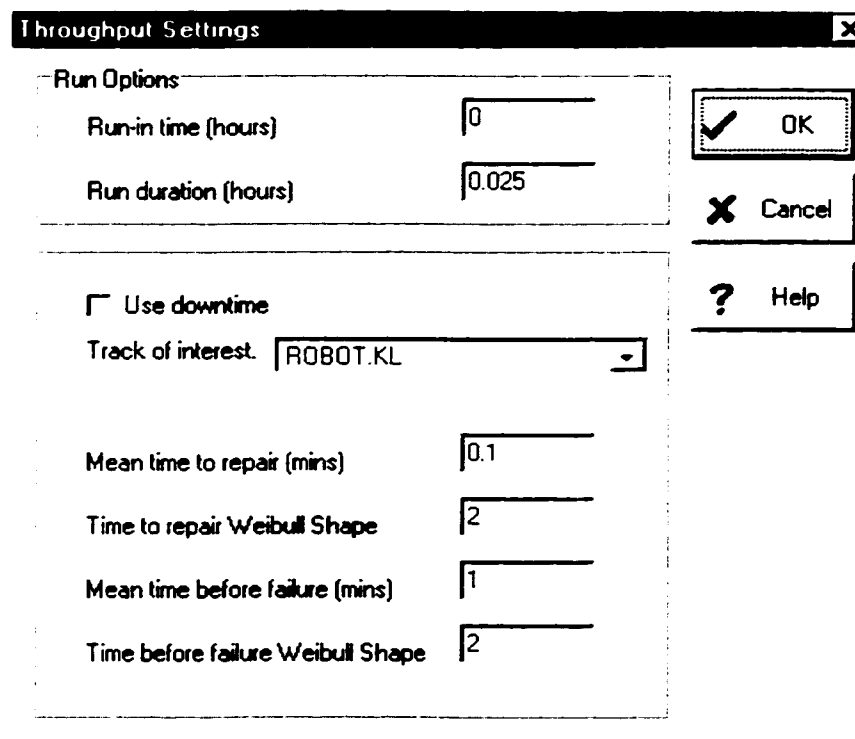


### **5.3.1. Throughput options**

Before a workcell – analyzing track is initiated, the following options need to be set up using the Options/Throughput options dialogue box.

- Run-in time – this is a value in hours for the time taken for the process to reach its normal production rate. The default value is 0. Set the time for which a track is allowed to run before the "ENDOFJOB" commands are used in recording an event cycle or the number of jobs. It is necessary for most simulations to have a brief run-in time before the timing of the cycle of events settles into a repeatable pattern.
- Run duration - this is a value in hours for the time that the process runs for. The default value is 0.025 (90 sec). Set the time for which the number of jobs is recorded. The number of jobs is incremented each time an "ENDOFJOB" command is encountered in a track during this period.
- Use down-time – choose whether to simulate process down-time (due to failures). The default value is false. When it is off Efficiency, Mean time to repair, and Mean time before failure have no effect on the track. When it is on, periods of downtime are simulated in each track at random, depending on the probabilities calculated using the Mean time before failure. The length of the downtime is based on the Mean time to repair.
- Track of interest – select which of the current tracks the statistical analysis will be concentrated on.
- Main time to repair - this is a value (in minutes) for the time that it takes for failure to be repaired and the process to restart. The mean time taken to repair a broken track (which may be controlling a robot or mechanism or may simply be moving objects using the Moveobject and Turnobject commands).

- Time to repair Weibull shape – this value defines the spread of the “time to repair PD”. Display time to repair PD is the “time to repair the Probability Density Function”. It allows a positive real number, which represents the desired shape of the probability distribution for the time to repair, to be entered. This probability distribution governs the random generation of downtimes for the track of interest.
- Main time before failure - this is a value (in minutes) for the time before a failure occurs.
- Main time before Weibull shape - this value defines the spread of the “time before failure PD”. Display time before failure PD is the “time before failure Probability Density Function” (Figure 31).



**Throughput Settings**

**Run Options**

Run-in time (hours)

Run duration (hours)

☐ Use downtime

Track of interest

Mean time to repair (mins)

Time to repair Weibull Shape

Mean time before failure (mins)

Time before failure Weibull Shape

☒ OK

☐ Cancel

☐ Help

**Figure 31 Throughput Settings**

If down time is not used in simulation, results will be just cycle time. In case that down time is considered, the results will be more complex.

### 5.3.2. Simple analysis of a workcell without using down-time

In simple analysis of a workcell, down-time is not included. The result will be just cycle time. For run-in-time 0.5 hours have been chosen, and for run duration 8 hours (Figure 32).

**Throughput Settings**

**Run Options**

Run-in time (hours) 0.5

Run duration (hours) 8

☐ Use downtime

Track of interest: ROBOT.KL

Mean time to repair (mins) 0.1

Time to repair Weibull Shape 2

Mean time before failure (mins) 1

Time before failure Weibull Shape 2

OK Cancel Help

**Figure 32 Throughput Settings for simple simulation**

“Generated on 3/2/1999(d/m/y) at 17:41:19(h/m/s) by WORKSPACE(c)  
Robot Simulations Ltd.”

Run-in time (hours)	0.5
Run duration (hours)	8
Use downtime	Off

---

Number of jobs during run	96
Number of jobs per hour	12

A comparison of these calculations and those of Workspace shows that the restates are the same. For one hour, the number of pieces is 24 (see equation 1). Workspace has accomplished 12.375 jobs, but in every job there are two finished parts. The number of jobs during run is 96, which means 192 parts for 8 hours.

“Generated on 3/2/1999(d/m/y) at 17:41:19(h/m/s) by WORKSPACE(c)  
Robot Simulations Ltd.” : **AZUR.KL**

<u>Step</u>	<u>Description</u>	<u>Time(sec)</u>	<u>Usage</u>
1	'Go to pick up part from pallet'	24.69	MAN
2	'Wait for table to rotate'	3.725	WAIT
3	'Go pick up part from pallet'	24.69	MAN
4	'Wait for table to rotate'	3.725	WAIT
5	'Go pick up part from pallet'	10.375	MAN
6	'Wait for table to rotate'	9.6393	WAIT
7	'Go pick up part from pallet'	10.375	MAN
8	'Wait for table to rotate'	50.8005	WAIT
9	'Turn a part'	3.3	MAN
10	'Wait for robot to finish welding'	55.3474	WAIT
11	'Turn a part'	3.3	MAN
12	'Wait for robot to finish welding'	40.0582	WAIT
13	'Put a part in a basket'	6.22	MAN
14	'Wait for robot to finish welding'	37.1791	WAIT
15	'Put a part in a basket'	9.945	MAN
-----		-----	
	CYCLE	293.3695	

Those results describe the operation sequences and the time for each sequence. The final cycle time for the operator is 293.3695 seconds. Workspace will give the results for the throughput analysis of the turntable and the robot.

“Generated on 3/2/1999(d/m/y) at 17:41:19(h/m/s) by WORKSPACE(c)

Robot Simulations Ltd”: **ROBOT.KL**

<u>Step</u>	<u>Description</u>	<u>Time(sec)</u>	<u>Usage</u>
1	'Wait for table to stop'	32.415	WAIT
2	'Go to weld parts'	15.1331	ROBOT
3	'Move robot to home position'	1.1643	ROBOT
4	'Wait for table to stop'	12.1176	WAIT
5	'Go to weld parts'	15.1267	ROBOT
6	'Move robot to home position'	1.1626	ROBOT
7	'Wait for table to stop'	3.725	WAIT
8	'Go to weld parts'	56.2378	ROBOT
9	'Move robot to home position'	1.2127	ROBOT
10	'Wait for table to stop'	3.725	WAIT
11	'Go to weld parts'	53.7138	ROBOT
12	'Move robot to home position'	1.2086	ROBOT
13	'Wait for table to stop'	3.725	WAIT
14	'Go to weld parts'	38.4055	ROBOT
15	'Move robot to home position'	1.2276	ROBOT
16	'Wait for table to stop'	3.725	WAIT
17	'Go to weld parts'	38.4399	ROBOT
18	'Move robot to home position'	1.2342	ROBOT
19	'Wait for table to stop'	13.67	WAIT

-----  
CYCLE

-----  
297.3695

The cycle time for the robot is 297.3695 seconds.

“Generated on 3/2/1999(d/m/y) at 17:41:19(h/m/s) by WORKSPACE(c)

Robot Simulations Ltd”: **TABLE.KL**

<u>Step</u>	<u>Description</u>	<u>Time(sec)</u>	<u>Usage</u>
1	'Wait for man to pick up parts from pallet'	28.69	WAIT
2	'Turn table for 180 degrees'	3.725	TABLE
3	'Wait for a part'	24.69	WAIT
4	'Turn table for 180 degrees'	3.725	TABLE
5	'Wait for a part'	16.2893	WAIT
6	'Turn table for 180 degrees'	3.725	TABLE
7	'Wait for a part'	57.4505	WAIT
8	'Turn table for 180 degrees'	3.725	TABLE
9	'Wait for a part'	54.9224	WAIT
10	'Turn table for 180 degrees'	3.725	TABLE
11	'Wait a for part'	39.6332	WAIT
12	'Turn table for 180 degrees'	3.725	TABLE
13	'Wait for a part'	39.6741	WAIT
14	'Turn table for 180 degrees'	3.725	TABLE
15	'Wait for a part'	6.22	WAIT
16	'Turn table for 180 degrees'	3.725	TABLE
17	'Wait for man to pick up parts from pallet'	0	WAIT

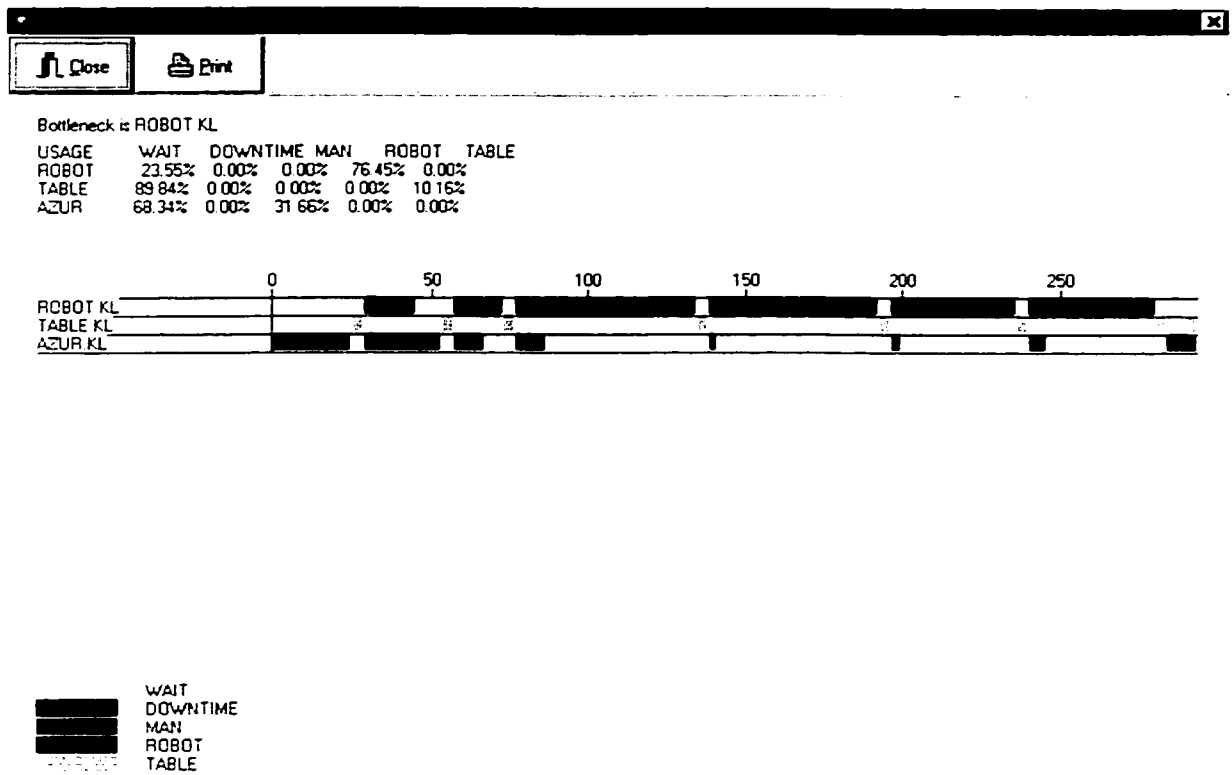
-----  
CYCLE

-----  
297.3695

The cycle time for the table is 297.3695 seconds.

Once a replay of the tracks has been made, a bar graph showing each track as a horizontal bar representing time passing along the x-axis is produced. Each bar has a different color for each track. Yellow represents WAIT, black represents DOWNTIME, red is for Operator's track, blue is for Robot's track, and green is for Table's track. A table is displayed above the graph, showing the percentage of each type of usage over the track. Bottleneck is the Robot track (Figure 33). Analyzing the bar graph and Workspace's calculation for the Operator, Robot and the Table, it can be concluded that the numbers match on the graph. The Robot and the Table are waiting about 25 seconds while the operator is working. The Robot is still waiting, and the Operator stops until the Table rotates for 180 degrees. When the Table stops, the Robot will start to weld, and the Operator will load parts. In this graph we can see that the Table can rotate only when the Robot and the Operator are not working. The Robot and the Operator can work at the same time.

The Robot will wait 23.55%, the Operator will wait for 68.34%, and the Table will wait for 89.84%.



**Figure 33 Bar graph showing each track as a horizontal bar representing time passing along the x-axis for simple simulation**



### **5.3.3. Complex analysis of workcell considering down-time**

To run a complex simulation one has to include downtime, mean time before failure, time before failure Weibull Shape, mean time to repair, and time to repair Weibull Shape for each device.

Time before failure is the key variable in reliability analysis. A variety of probability distributions characterizes how long an entity will survive. The most flexible distribution that is quite useful in reliability analysis is the Weibull distribution. Failure time  $T$  is a continuous random variable with probability density function  $f(x)$  [Lapin, 1997]. Throughput statistic in Workspace is based on the Weibull distribution.

One of the most attractive features of robots is their reliability-98% up-time is normal. Nevertheless, those 2% down-time may be critical to the company's production and could make the difference between profit and loss.

Although any component in a robotic system could wear out or be defective, certain components are more susceptible than others to such problems. The power system is the most important factor that affects maintenance. Other components that often require attention are the mechanical parts, particularly moving parts. Failures may be caused by corrosion, lack of lubrication, or leaks in seals. Wrists are particularly subject to wear because of their extensive movement. End-effector, such as grippers, spray paint heads, and welding tips are likewise sources of potential problems. If spot weld gun tips are not periodically dressed, they will not produce high-strength welds and will stick to the workpiece.

The controllers may be another source of breakdown. Electronic components could malfunction because of power surges, excessive heat, microwaves, or vibration.

In addition, contaminants in the atmosphere, including water vapor, could have adverse effects [Nof, 1997].

Regarding different effects, which can cause the break down of the workcell, reference will be made to the system designed in this work. Regular maintenance, negligible influence of software and end-effector interface are assumed. Referring to the research of Smith, [1992], mean time to failure for the robot and turntable is 3,000 hours. Concerning the operator, it is assumed that replacement is available.

The procedures for troubleshooting and diagnosing failures in robotics system are generally carefully outlined in the service manual. An effective manual will be carefully formatted with a table of contents that allows the technician to quickly locate the section related to the problem at hand. To help the technician to solve troubleshooting robotic problems quickly an expert system can be used. The advantage of the expert system is that it puts a specialized knowledge base at the disposal of an employee on the plant floor. The employee brings a portable computer up to the malfunctioning robot, switches on the computer screen and follows a simple step-by-step artificial intelligence reasoning program that leads to the trouble and explains how to fix it. Referring to previous research [Pasi, 1996], mean time to repair for the robot and turntable is 2 hours (Figure 34).

In the present experiments the simulation run for 2000 hours. Replay Track Blind command was used in order to check that tracks run correctly, and to calculate cycle time, number of parts per hour and number of parts during the simulation. This command will not calculate pictures or store animations for the tracks and is, therefore, faster.

Throughput Settings

Run Options

Run-in time (hours) 0

Run duration (hours) 2000

☒ Use downtime

Track of interest: AZUR KL

Mean time to repair (mins) 3

Time to repair Weibull Shape 2

Mean time before failure (mins) 240

Time before failure Weibull Shape 2

OK Cancel Help

Throughput Settings

Run Options

Run-in time (hours) 0

Run duration (hours) 2000

☒ Use downtime

Track of interest: ROBOT KL

Mean time to repair (mins) 120

Time to repair Weibull Shape 2

Mean time before failure (mins) 180000

Time before failure Weibull Shape 2

OK Cancel Help

Throughput Settings

Run Options

Run-in time (hours) 0

Run duration (hours) 2000

☒ Use downtime

Track of interest: TABLE KL

Mean time to repair (mins) 120

Time to repair Weibull Shape 2

Mean time before failure (mins) 180000

Time before failure Weibull Shape 2

OK Cancel Help

Figure 34 Throughput Settings for Robot, Turntable, and Operator

Parameter  $\beta$  designates time before failure of the Weibull shape and parameter  $\lambda$  represents time to repair the Weibull shape in the Weibull probability density function. In this simulation default values:  $\beta=2$  and  $\lambda=2$  will be used. The Mean time before failure represents the time before the breakdown of the track (Figure 35).

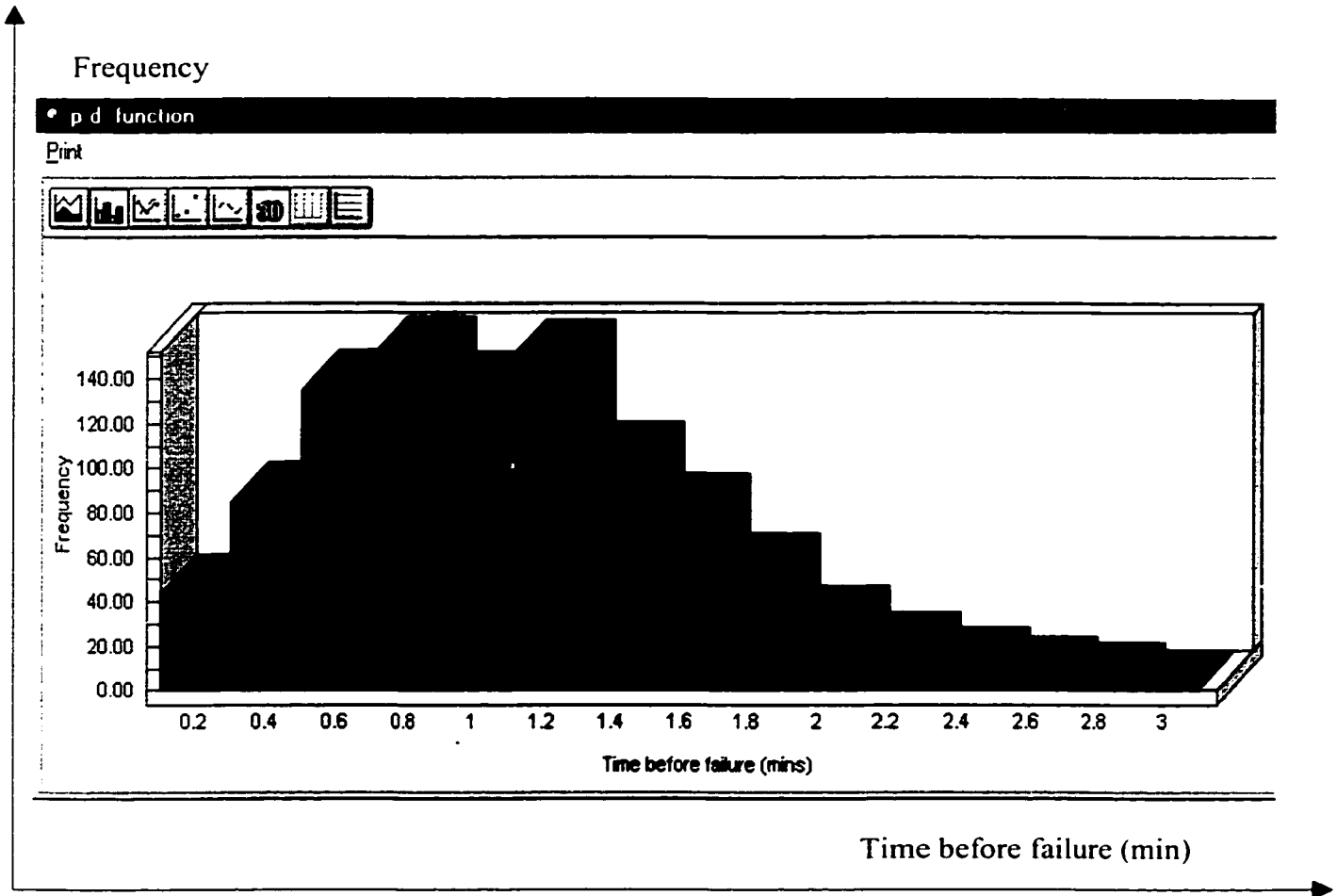


Figure 35 Probability Density function for time before failure

The Mean time to repair is the time taken to repair a broken track which may be controlling a robot or mechanism or may simply be moving objects using the moveobject or turnobject commands (Figure 36).

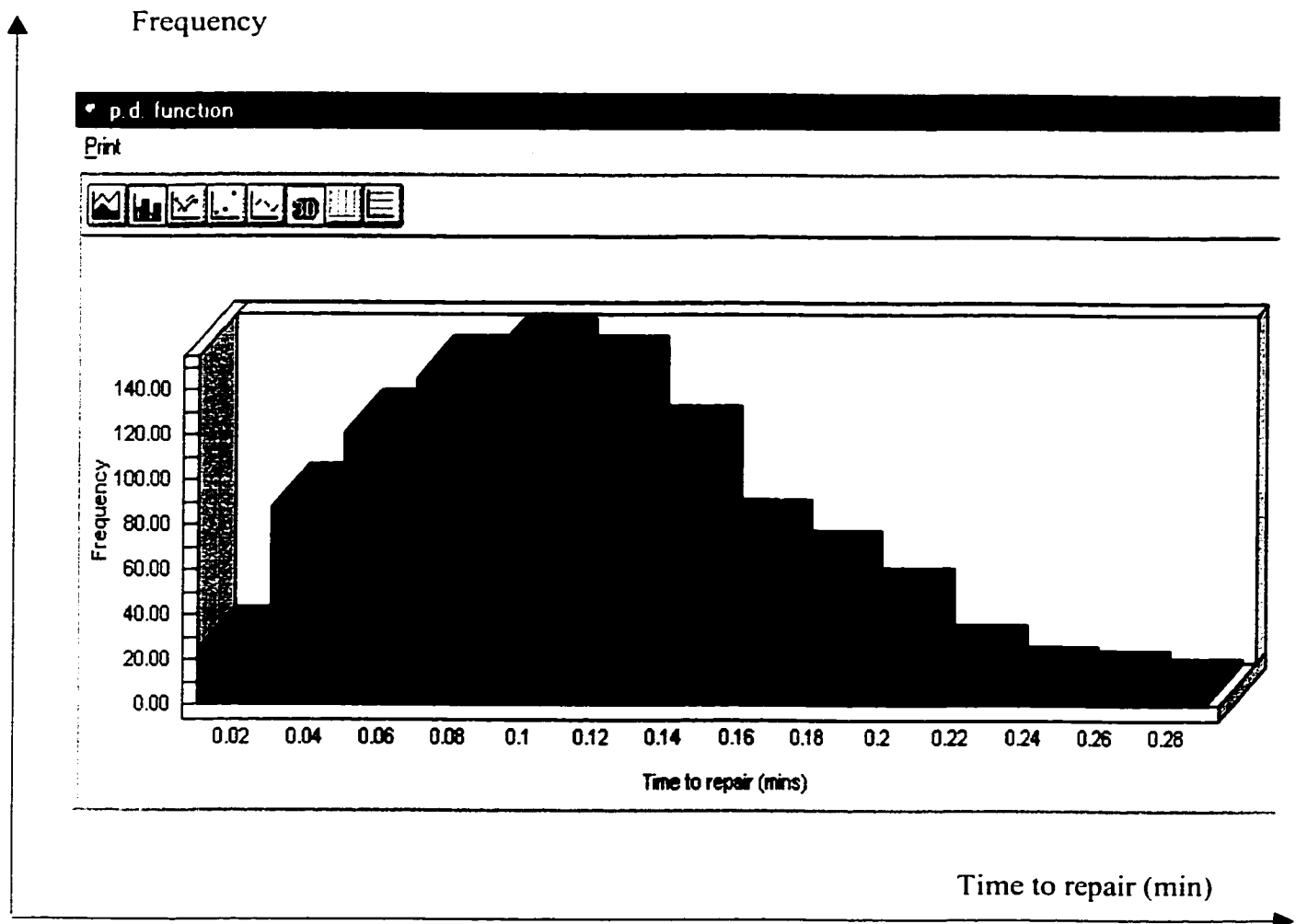


Figure 36 Probability Density function for time to repair

Run-in time (hours)	0
Run duration (hours)	2000
Use downtime	On

ROBOT.kl

Mean time before failure (min)	180000
Time before failure Weibull Shape	2
Mean time to repair (min)	120
Time to repair Weibull Shape	2
Efficiency (%)	90

TABLE.kl

Mean time before failure (min)	180000
Time before failure Weibull Shape	2
Mean time to repair (min)	120
Time to repair Weibull Shape	2
Efficiency (%)	90

AZUR.kl

Mean time before failure (min)	240
Time before failure Weibull Shape	2
Mean time to repair (min)	3
Time to repair Weibull Shape	2
Efficiency (%)	90

-----

Number of jobs during run	21600
Number of jobs per hour	10.8

# Workspace's calculation for complex simulation for Robot, Table, and

Operator :

“Generated on 15/2/1999(d/m/y) at 18:20:54(h/m/s) by WORKSPACE(c)

## AZUR.KL

Step	Description	Time(sec)	Usage
1	'Go to pick up part from pallet'	24.69	MAN
2	'Wait for table to rotate'	3.725	WAIT
3	'Go pick up part from pallet'	24.69	MAN
4	'Wait for table to rotate'	3.725	WAIT
5	'Go pick up part from pallet'	10.375	MAN
6	'Wait for table to rotate'	9.6393	WAIT
7	'Go pick up part from pallet'	10.375	MAN
8	'Wait for table to rotate'	50.8005	WAIT
9	'Turn a part'	3.3	MAN
10	'Wait for robot to finish welding'	55.3474	WAIT
11	'Turn a part'	3.3	MAN
12	'Wait for robot to finish welding'	40.0582	WAIT
13	'Put a part in a basket'	6.22	MAN
14	'Wait for robot to finish welding'	37.1791	WAIT
15	'Put a part in a basket'	9.945	MAN
----		-----	
	CYCLE	293.3695	

**ROBOT.KL**

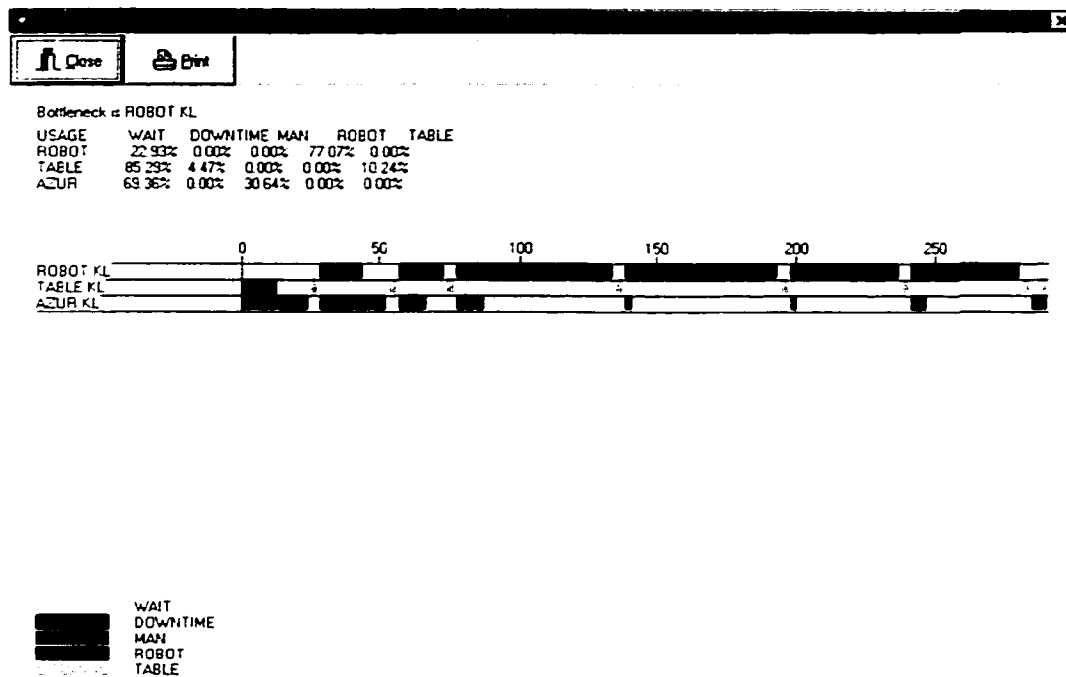
Step	Description	Time(sec)	Usage
1	'Wait for table to stop'	32.415	WAIT
2	'Go to weld parts'	15.1331	ROBOT
3	'Move robot to home position'	1.1643	ROBOT
4	'Wait for table to stop'	12.1176	WAIT
5	'Go to weld parts'	15.1267	ROBOT
6	'Move robot to home position'	1.1626	ROBOT
7	'Wait for table to stop'	3.725	WAIT
8	'Go to weld parts'	56.2378	ROBOT
9	'Move robot to home position'	1.2127	ROBOT
10	'Wait for table to stop'	3.725	WAIT
11	'Go to weld parts'	53.7138	ROBOT
12	'Move robot to home position'	1.2086	ROBOT
13	'Wait for table to stop'	3.725	WAIT
14	'Go to weld parts'	38.4055	ROBOT
15	'Move robot to home position'	1.2276	ROBOT
16	'Wait for table to stop'	3.725	WAIT
17	'Go to weld parts'	38.4399	ROBOT
18	'Move robot to home position'	1.2342	ROBOT
19	'Wait for table to stop'	13.67	WAIT
-----		-----	
CYCLE		297.3695	



**TABLE.KL**

Step	Description	Time(sec)	Usage
1	Downtime	13	DOWNTIME
2	'Wait for man to pick up parts from pallet'	24.69	WAIT
3	'Turn table for 180 degrees'	3.725	TABLE
4	'Wait for a part'	24.69	WAIT
5	'Turn table for 180 degrees'	3.725	TABLE
6	'Wait for a part'	16.2893	WAIT
7	'Turn table for 180 degrees'	3.725	TABLE
9	'Wait for a part'	57.4505	WAIT
10	'Turn table for 180 degrees'	3.725	TABLE
11	'Wait for a part'	54.9224	WAIT
12	'Turn table for 180 degrees'	3.725	TABLE
13	'Wait a for part'	39.6332	WAIT
14	'Turn table for 180 degrees'	3.725	TABLE
15	'Wait for a part'	39.6741	WAIT
16	'Turn table for 180 degrees'	3.725	TABLE
17	'Wait for a part'	6.22	WAIT
18	'Turn table for 180 degrees'	3.725	TABLE
170	'Wait for man to pick up parts from pallet'	0	WAIT
-----		-----	
	CYCLE	306.3695	

A complex simulation has downtime option switched on. When it is on, periods of downtime are simulated in each track at random, depending on the probabilities calculated using the Mean time before failure. The length of the downtime is based on the Mean time to repair (Figure 37). Numbers from analyzing the bar graph and Workspace calculation for the Operator, the Robot and the Table match calculation numbers. The simulation will start with the Operator, until the Robot and the Table are waiting for signals to start. On the graph one can see 13 seconds Down-time for the Table. Table will start to rotate when the Operator and the Robot stop. The Robot will start to weld parts at the same time when the Operator starts to load the parts. The number of jobs (2 completed parts) per hour is 10.8, and the number of jobs for 2000 hours is 21600 or 43200 parts for one year (2000 hours of run time). The annual production of 43200 pieces satisfies the original production request of 40000 pieces per year.



**Figure 37 Bar graph showing each track as a horizontal bar representing time passing along the x-axis for complex simulation**

## 5.4. Real-time collision avoidance

When planing a trajectory between goal points, it is clearly desirable to have some method of choosing a path that is collision free [Owens, 1990].

Collision detection can help prevent expensive crashes before they happen. There are three types of collision detection: Full, Partial and Near Miss. Full detection detects all the collisions in any view or rendering mode. Partial detection only detects collisions when in Hiddenline mode and Diagonal view. Near Miss detection will notify the user when two objects get closer than a user defined distance. A Collision List can be also used to select objects for which the collision is probable [User Guide Manuel, Workspace4, 1997]. Figure 38 represents full collision detection between the operator's head and welding gun. In the collision analysis one can include all objects that are in workcell.

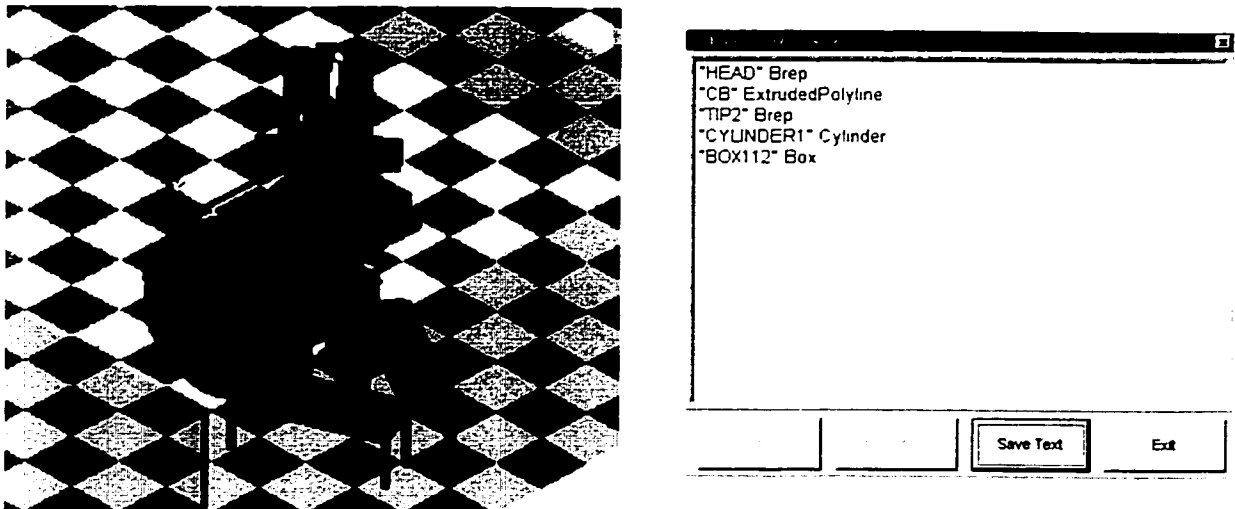
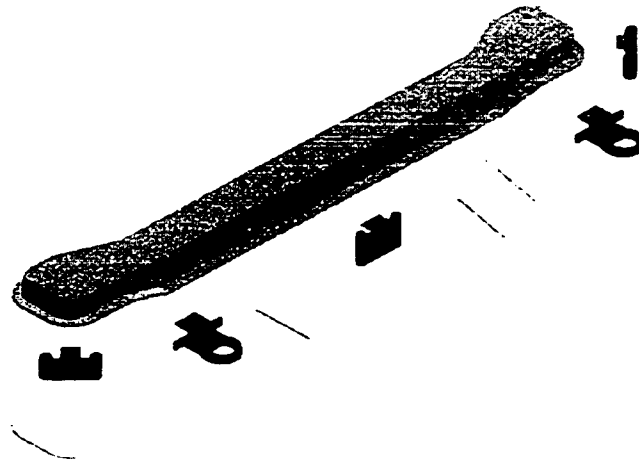


Figure 38 Collision detection

## 5.5. Analysis of Results

The reason for modeling, simulation, off-line programming and layout design of this workcell is to enable the production of defined automotive parts, under required conditions. The starting point are nine components for front support member engine/ transaxle mount (Figure 39).



**Figure 39 Parts for Front support member engine/transaxle mount**

Pre-requirements for workcell simulation:

- To automate the process with IRB6000 robots,
- To include one operator (for union and safety reasons),
- A total cell cost of no more than \$ 600,000,
- Production must be at least 40,000 parts per year,
- Size of workcell is 24 X 30 sq. ft,

The Table 2 represents a Summary of Manufacturing Capacity Results which include cycle time, number of pieces per hour, number of pieces per shift, and number of pieces per year. The calculation was made both for 100% and 70% of production

capacity. These results were obtained from calculation and they did not satisfy given requirements (production was less than 40,000 parts per year).

**Table 2 Summary of Manufacturing Capacity Results:**

	Cycle Time [sec]	Pieces/Hour	Pieces/Shift	Pieces/Year One shift per day 5 days per week 48 weeks per year
@100%	149	24	192	46,080
@70%	135	17	136	32,640

Calculation for 70% of production capacity does not reflect robot performance. It represents down-time connected with normal production cycle that manufacturer had in the past. E.g. down-times due to regular production breaks, change over-time, set-up time, and unplanned delay. This calculation is usual production tool for quick determination of “real life” capacity.

Table 3 represents the list of prices for all devices in a workcell. The price for whole cell is \$285,000 which is double less than the amount of money that was planed for developing workcell. The numbers listed in the Table were obtained from manufacturers.

**Table 3: Calculation for cell cost**

Robot	\$150,000
Rotary table	\$30,000
Fixture	\$5,000
Welding gun	\$20,000
Pallets for prefinished parts	\$5,000
Pallets for finished parts	\$5,000
Operator's salary	\$70,000
Sum	\$285,000<\$600,000

Table 4 represents Workspace Simple and Complex Analysis Results Comparison. For simple analysis, number of jobs during a run is 96, which means 192 parts during run. Run duration is 8 hours. Number of jobs per hour is 12, which means 24 parts per hour. For complex analysis run duration is 2000 hours, number of parts is 21.6, and number of parts during run (2000 hours) is 43200, which satisfies the required number of parts (40000 per year).

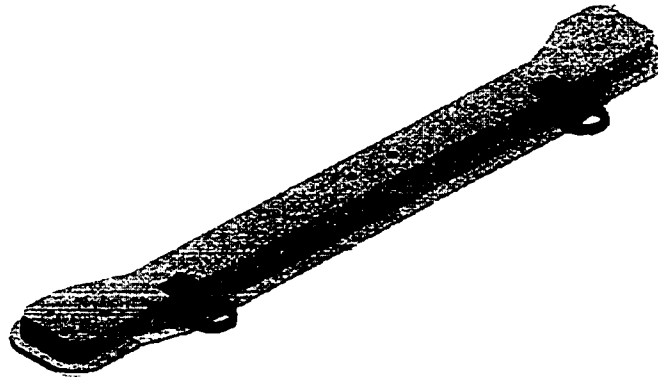
**Table 4: Workspace Simple and Complex Analysis Results Comparison:**

	Simple Analysis	Complex Analysis
Run-in Time [hours]	0.5	0
Run duration [hours]	8	2000
Mean Time to Repair [min]	/	Robot: 120 Table: 120 Operator: 3
Mean Time before Failure [min]	/	Robot: 180000 Table: 180000 Operator: 240
Number of jobs during run	$96 \times 2 = 192$	$21600 \times 2 = 43200$
Number of jobs per hour	$12 \times 2 = 24$	$10.8 \times 2 = 21.6$

One job includes two finished parts.



Cell output consist of finished part, Robot Program (see Appendix A3), Work Cell Layout Design (Figure 40).



**Figure 40 Front support member engine/transaxle mount**

The results of workcell analysis satisfy given requirements. It includes ABB IRB6000 robot, operator, size of workcell (24 X 30 sq. ft), number of parts per year (40000), and price of workcell (\$600,000), (Table 5).

**Table 5: Requirements and results**

To automate the process with IRB6000 robots,	ABB IRB6000
To include one operator (for union and safety reasons),	Operator
A total cell cost of no more than \$ 600,000,	\$285,000
Production must be at least 40,000 parts per year	43200 parts per year
Size of workcell is 24 X 30 sq. ft	23.4 X 23.4 sq. ft
Collision detection	No collision

## **CONCLUSION**

The greater acceptance, use and reliance of robot simulators are being driven mainly by the increasing demands from the lower priced sector of the market. For a fraction of the cost of an assembly cell, the user can now purchase a fully featured simulator that will run happily on a modestly powered PC platform - one that in most cases can already be found within the organization running a wordprocessor or spreadsheet. The possibility for simulating changes to system software before downloading a known working copy of the robot's program in its native language is a major contribution to safety in terms of personnel and damaged equipment.

Modeling, simulation and off-line programming has been conducted for an automotive assembly workcell.

The workcell simulation approach described in this thesis is a step forward in providing a task-oriented solution to the problem of robotic cell design and programming. It integrates off-line programming techniques with significant features of welding technology. The approach has been implemented in Workspace.

This particular graphical simulation and layout design of spot welding and assembly proved the importance of production planning and analysis. This simulation involves robot, rotary table and operator. The operator was defined as a mechanism with eighteen joints.

Simulation was running in real-time and each aspect of the production could be easily tracked and analyzed. Changing throughput-setting variables could simulate various production scenarios. For simple simulation the input is only run time, and for complex simulation the input represents run time with down time. Another important feature provided by Workspace - collision avoidance is also included into the simulation.

Part of this simulation was presented to Ford Motor Company, and to General Dynamic Company. The technical personnel were impressed with simulation capabilities and performances.

All originally given constraints and requirements were successfully met in this simulation.

# REFERENCES

Alexopoulos C., Kang K., Lilegdon W. R., Goldsman D., 1995; Proceedings of the 1995 Winter Simulation Conference ed.

Balkan Tuna, Arikan M. A. Sahir, Bulut Murat, 1997; Prowled: an off-line welding robot programming package with an interactive graphical interface

Bernhard R., Schreck G. and Willnow C., 1994; "*The Realistic robot simulation (RRS) interface*" Copyright IFAC Intelligent Manufacturing Systems, Vienna, Australia.

Craig J. John, 1986; "*Introduction to Robotics, Mechanics and Control*", 2<sup>nd</sup> edition SILMA Inc; Addison-Wesley Publishing Company;

Critchlow J. Arthur, 1985; "*Introduction To Robotics*"; McMillan Publishing Company; New York;

D'Andrea Tony, Dolata Mark, B. Eng., 1994, "*Operating Characteristics of Voltza® Transgun Assembly CLTD-9941-1*", OHMA® Systems incorporated.

Erickson D. Jon, Aucoin J. Jr Paschal, Dragg L. James., 1992; "*Tool for Simulating Space Exploration Initiative (SEI) Operations As A Means Of Obtaining Intelligent Systems Requirements*"; Cooperative Intelligent Robotics in Space, n.3, pp. 482-494.

Glover Fred, Kelly P. James, and Laguna Manuel, 1996 "*New advances and applications of combining simulation and optimization*", Proceeding of the Winter Simulation Conference

Gong Wenwei, 1998 "*Automatic Robot Path Generation for Manufacturing on Sculptured Surfaces*", Master Thesis, University of Windsor

Groover P. Mikell, 1987 "*Automation, Production Systems, and Computer-Integrated manufacturing*" Printed and published by the Prentice-Hall Inc., Englewood Cliffs, New Jersey.

Hewer D. Neal, Smith L. Alan, 1996; Telegrid: A Vitrual Tool for Real World Control Problems

Honey E. William, Jamshidi Mohammed 1992; "*ROBO\_SIM: A Robotic Simulation Environment On Personal Computers*"; Robotics and Autonomous Systems; n.9, pp. 305-317.

Kelton W. David, Sadowski P. Randall, and Sadowski A. Deborah, 1998, "*Simulation with Arena*" Copyright by The McGraw-Hill Companies, Inc.

Kukareko Evgeni, Pashkevich Anatoly, Khmel Dmitry, Korzun Alexander, Yurkevich Yury, 1994, "*Computer-aided programming of automated welding workcells*", SPIE Vol. 2247 sensors and Control for Automation (1994)/271

Lapin L. Lawrence, "*Modern engineering statistics.*", COPYRIGHT 1997 by Wadsworth Publishing Company, A Division of International Thomson Publishing Inc.

Marhefka W. Duane, Orin E. David., 1996; "*Xanimate: AN Educational Tool For Robot Graphical Simulation*"; IEEE Robotics and Automation Magazine; June pp. 6-14.

Nof Shimon Y., "Industrial assembly", London; New York: Chapman & Hall, 1997, first edition,

Owens John, 1994; "*Workspace, A Microcomputer Industrial Robot Simulator And Off Line Programming System*"; The Institution of Electrical Engineers; London, UK,

Owens John, 1990; "*Industrial robot simulation*"; Phd. Thesis, Newcastle, UK

Pasi D. 1996, "*A formal structure and implementation of an equipment level controller in a shop floor Control system*", Master Thesis, Industrial Engineering department, Pan State University.

Peters A. Brett, Smith S. Jeffery, Curry James, LaJimodire Cynthila, Charnes J. M. Ed. Morrice D. J., Brunner D. T., and Swain J. J., 1996 "*Advanced tutorial – simulation – based scheduling and control*", Proceedings of the 1996 Winter Simulation Conference

Rembold Ulrich, 1990 "*Robot Technology and Applications*" Printed and published by the Marcel Dekker Inc., New York.

Rooks W. Brian, 1997, "*Off-line programming: a success for the automotive industry*", Industrial Robot, Vol. 24, Number, 1997.

Smith J. 1992, "*A formal design and development methodology for shop floor control in computer integrated manufacturer*", Phd. Thesis, Industrial Engineering department, Pan State University

The specifications and applications of industrial robots in Japan, 1984  
Japan industrial robot Association

User Guide Manuel, Workspace4, 1997

Voss Pieter, Haddock Jorge, 1993, "*Simulation based performance analysis of an intelligent robotic system control architecture*", Proceedings of the 1993 Winter simulation Conference

Vukobratovic M., Stokic D., Kircanski N., Kircanski M., Hristic D., Karan B., Vujic D., Durovic M., 1986; "*UVOD U ROBOTIKU*"; Institut Mihajlo Pupin; Belgrade;

Weisel K. Walter, 1997, "*Controllers Give New Life To old Robots*", Foundry Management & Tehnology.

Wilson Mike, 1994; "*Robot Applications – Towards The Year 2000*"; FANUC Robotics Ltd.; The Institute of Electrical Engineers; London, UK;

# **APPENDIX**

## **A.1 Workspace Features**

- Sophisticated 3D CAD system featuring Constructive Solid Geometry and Swept Polylines.
- High resolution 3D rendering in OpenGL. Textures specular highlights and transparency with potential for 16 million colors.
- Advanced Robot languages of the major manufacturers.
- Special language development.
- Off-line Programming.
- Friendly mouse driven interface with Windows 95 and NT.
- DXF/IGES/STEP import and export facility.
- Robot and workcell calibration to less than 1mm accuracy using Calibration Plus®
- Kinematics and inverse kinematics modeler for mechanisms with up to 22 joints.
- Computer Aided Learning, sub-system
- Dynamic Link Libraries to external Pascal, C or C++ routines
- Collision and near miss detection
- Easy to use application macro's
- Moving Cable simulation
- Geometry Point (GP) to Teach Point (TP) conversion. Import and export of geometry points to external
- Automatic geometry path generation
- Optimum positioning of robot
- Simulation to real time animation conversion
- Virtual reality for the Internet. Animated 3D scenes can be created using Internet

## A.2 Key words

### **sim-u-la-tion (sim yuh lay'shuhn) n.**

1. imitation or enactment, as of conditions anticipated.
2. the act or process of pretending; feigning.
3. an assumption or imitation of a particular appearance or form; counterfeit.
4. the representation of the behavior or characteristics of one system through the use of another system, esp. using a computer.
5. a conscious attempt to feign some mental or physical disorder.

[1300-50; ME simulacion < L simulatio a pretense.

See SIMULATE, - TION]

### **ro-bot (roh'buht, -bot) n.**

1. a machine that resembles a human and does mechanical, routine tasks on command.
2. a person who acts and responds in a mechanical, routine manner; automaton.
3. any machine or mechanical device that operates automatically with humanlike skill.



[< Czech, coined by Karel Capek in the play R.U.R.

(1920) from the base robot-, as in robota

compulsory labor, robotník peasant owing such labor]

Derived words

--ro-bot'ic, ro bot-is'tic(-buh tis'tik, -bo-), adj.

**mech-an-ism (mek'uh niz uhm) n.**

1. an assembly of moving parts performing a complete functional motion.
2. the agency or means by which an effect is produced or a purpose is accomplished.
3. machinery; mechanical appliances.
4. the structure or arrangement of parts of a machine or similar device.
5. routine methods or procedures.
6. the theory that everything in the universe is produced by matter in motion. Compare DYNAMISM (def. 1), VITALISM (def. 1).
7. a. the view that all biological processes may be described in physicochemical terms.
8. a mode of behavior that helps an individual deal with the physical or psychological environment. Compare DEFENSE MECHANISM, escape mechanism .

[1655-65; < NL mechanismus; LL mechanisma a  
contrivance < Gk mechan (é) MACHINE + NL -ismus,  
LL -isma - ISM]

Derived words

--mech a-nis'mic, adj.

### A.3 Track programs and teachpoint file

#### **PROGRAM ROBOT**

PROGRAM ROBOT

-- ! LANGUAGE KAREL 2

-- ! MEMORY 8192

-- ! ROBOT IRB6000

-- TEACHPOINT DECLARATIONS

VAR

AUX1 : AUXPOS

AUX2 : AUXPOS

AUX3 : AUXPOS

AUX4 : AUXPOS

AUX5 : AUXPOS

AUX6 : AUXPOS

AUX7 : AUXPOS

AUX8 : AUXPOS

AUX9 : AUXPOS

AUX10 : AUXPOS

AUX11 : AUXPOS

```
AUX12 : AUXPOS
AUX13 : AUXPOS
AUX14 : AUXPOS
AUX15 : AUXPOS
AUX16 : AUXPOS
AUX17 : AUXPOS
AUX18 : AUXPOS
AUX19 : AUXPOS
AUX20 : AUXPOS
AUX21 : AUXPOS
AUX22 : AUXPOS
AUX23 : AUXPOS
AUX24 : AUXPOS
AUX25 : AUXPOS
AUX26 : AUXPOS
AUX27 : AUXPOS
AUX28 : AUXPOS
AUX29 : AUXPOS
AUX30 : AUXPOS
AUX31 : AUXPOS
BEGIN
REPEAT
  SUTOOL=POS(123.2222,681.1944,43.7018,90,90,0,")
  SUSEMAXACCEL=TRUE
  %INCLUDE ANA-ALLA#
```

```

--! DINSIGNALNAME DIN[1]
--! DINSIGNALNAME DIN[2]
--! DINSIGNALNAME DIN[3]
--! DINSIGNALNAME DIN[4]
--! DINSIGNALNAME DIN[5]
--! DINSIGNALNAME DIN[6]
--! DINSIGNALNAME DIN[7]
--! DINSIGNALNAME DIN[8]
--! DINSIGNALNAME DIN[9]
--! DINSIGNALNAME DIN[10]
--! DOUTSIGNALNAME DOUT[1]
--! DOUTSIGNALNAME DOUT[2]
--! DOUTSIGNALNAME DOUT[3]
--! DOUTSIGNALNAME DOUT[4]
--! DOUTSIGNALNAME DOUT[5]
--! DOUTSIGNALNAME DOUT[6]
--! DOUTSIGNALNAME DOUT[7]
--! DOUTSIGNALNAME DOUT[8]
--! DOUTSIGNALNAME DOUT[9]
--! DOUTSIGNALNAME DOUT[10]
--! DOUTSIGNALNAME DOUT[11]

--! SIGNALDEF DIN[1], TABLE.KL,1
--! SIGNALDEF DIN[2], TABLE.KL,2
--! SIGNALDEF DIN[3], TABLE.KL,3

```

--! SIGNALDEF DIN[4], TABLE.KL,4  
--! SIGNALDEF DIN[5], TABLE.KL,5  
--! SIGNALDEF DIN[6], TABLE.KL,6  
--! SIGNALDEF DIN[7], TABLE.KL,7  
--! SIGNALDEF DIN[8], TABLE.KL,8  
--! SIGNALDEF DIN[9], TABLE.KL,9  
--! SIGNALDEF DIN[10], TABLE.KL,10  
--! SIGNALDEF DIN[11], TABLE.KL,11

--! SIGNALDEF DOUT[1], TRACK,1  
--! SIGNALDEF DOUT[2], TRACK,2  
--! SIGNALDEF DOUT[3], TRACK,3  
--! SIGNALDEF DOUT[4], TRACK,4  
--! SIGNALDEF DOUT[5], TRACK,5  
--! SIGNALDEF DOUT[6], TRACK,6  
--! SIGNALDEF DOUT[7], TRACK,7  
--! SIGNALDEF DOUT[8], TRACK,8  
--! SIGNALDEF DOUT[9], TRACK,9  
--! SIGNALDEF DOUT[10], TRACK,10  
--! SIGNALDEF DOUT[11], TRACK,11

DOUT[1]=OFF

DOUT[2]=OFF

DOUT[3]=OFF

DOUT[4]=OFF

DOUT[5]=OFF

DOUT[6]=OFF

DOUT[7]=OFF

DOUT[8]=OFF

DOUT[9]=OFF

DOUT[10]=OFF

DOUT[11]=OFF

-- Path 1

WITH SMOTYPE=JOINT

MOVE TO \$HOME:\$SUTOOL

DOUT[1]=ON

REPEAT

WAIT FOR DIN[1]=ON

UNTIL DIN[1]=ON

--!LABLEVENT 'Go to weld parts' , 3

--!USEMOVINGVIEW view1, view2,2

WITH SMOTYPE=JOINT

MOVE TO POLYLINE93GP6

-- ! SPOTWELD 100,100

MOVE TO CSG343GP1

-- ! SPOTWELD 100,100

MOVE TO POLYLINE93GP4

-- ! SPOTWELD 100,100

MOVE TO CSG336GP1

-- ! SPOTWELD 100,100

```

MOVE TO POLYLINE93GP2

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robot to home position' , 3

WITH SMOTYPE=JOINT

MOVE TO $HOME:SUTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[2]=ON


-- Path 2

REPEAT

WAIT FOR DIN[2]=ON

UNTIL DIN[2]=ON

--!LABELEVENT 'Go to weld parts' , 3

WITH SMOTYPE=JOINT

MOVE TO CSG216GP3

-- ! SPOTWELD 100,100

MOVE TO CSG344GP1

-- ! SPOTWELD 100,100

MOVE TO CSG186GP1

-- ! SPOTWELD 100,100

MOVE TO CSG351GP1

-- ! SPOTWELD 100,100

MOVE TO CSG215GP2

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robt to home position' , 3

```

```

WITH SMOTYPE=JOINT

MOVE TO SHOME:$UTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[3]=ON


-- Path 3

REPEAT

WAIT FOR DIN[3]=ON

UNTIL DIN[3]=ON

--!LABELEVENT 'Go to weld parts' , 3

MOVE TO EXTRUS58GP6

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP7

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP8

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP9

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP10

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP11

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP12

-- ! SPOTWELD 100,100

MOVE TO EXTRUS60GP6

```



-- ! SPOTWELD 100,100  
MOVE TO EXTRUS58GP13  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP10  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP9  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP8  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP7  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP5  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP1  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP2  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP4  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP3  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP5  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP6  
-- ! SPOTWELD 100,100

```

MOVE TO EXTRUS63GP7

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robt to home position' , 3

WITH $MOTYPE=JOINT

MOVE TO $HOME:$SUTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[4]=ON


--Path 4

REPEAT

WAIT FOR DIN[4]=ON

UNTIL DIN[4]=ON

--!LABELEVENT 'Go to weld parts' , 3

MOVE TO EXTRUS58GP20

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP21

-- ! SPOTWELD 100,100

MOVE TO EXTRUS60GP22

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP23

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP24

-- ! SPOTWELD 100,100

MOVE TO EXTRUS58GP25

-- ! SPOTWELD 100,100

```

MOVE TO EXTRUS60GP17  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS60GP18  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP18  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP20  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP17  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP19  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS225GP16  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP12  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP13  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP14  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP15  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS62GP16  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS63GP12

```

-- ! SPOTWELD 100,100

MOVE TO EXTRUS63GP13

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robt to home position' , 3

WITH SMOTYPE=JOINT

MOVE TO SHOME:$UTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[5]=ON


-- Path 5

REPEAT

WAIT FOR DIN[5]=ON

UNTIL DIN[5]=ON

--!LABELEVENT 'Go to weld parts' , 3

MOVE TO EXTRUS834GP1

-- ! SPOTWELD 100,100

MOVE TO EXTRUS834GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS828GP3

-- ! SPOTWELD 100,100

MOVE TO EXTRUS828GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS828GP1

-- ! SPOTWELD 100,100

MOVE TO EXTRUS833GP1

```

```

-- ! SPOTWELD 100,100

MOVE TO EXTRUS833GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS833GP3

-- ! SPOTWELD 100,100

MOVE TO EXTRUS833GP4

-- ! SPOTWELD 100,100

MOVE TO EXTRUS825GP3

-- ! SPOTWELD 100,100

MOVE TO EXTRUS825GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS825GP1

-- ! SPOTWELD 100,100

MOVE TO EXTRUS796GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS796GP1

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robt to home position' , 3

WITH SMOTYPE=JOINT

MOVE TO SHOME:SUTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[6]=ON

-- Path 6

```

REPEAT  
WAIT FOR DIN[6]=ON  
UNTIL DIN[6]=ON  
--!LABLEVENT 'Go to weld parts' , 3  
MOVE TO EXTRUS792GP1  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS792GP2  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS786GP3  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS786GP2  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS786GP1  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS791GP1  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS791GP2  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS791GP3  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS791GP4  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS783GP3  
-- ! SPOTWELD 100,100  
MOVE TO EXTRUS783GP2

```

-- ! SPOTWELD 100,100

MOVE TO EXTRUS783GP1

-- ! SPOTWELD 100,100

MOVE TO EXTRUS754GP2

-- ! SPOTWELD 100,100

MOVE TO EXTRUS754GP1

-- ! SPOTWELD 100,100

--!LABELEVENT 'Move robt to home position' , 3

WITH SMOTYPE=JOINT

MOVE TO SHOME:SUTOOL

--!LABELEVENT 'Wait for table to stop' , 0

DOUT[7]=ON

REPEAT

WAIT FOR DIN[7]=ON

UNTIL DIN[7]=ON

REPEAT

WAIT FOR DIN[8]=ON

UNTIL DIN[8]=ON

DOUT[1]=ON

REPEAT

WAIT FOR DIN[1]=ON

UNTIL DIN[1]=ON

-- DELAY 35000

UNTIL FALSE

END ROBOT

```

## **PROGRAM TABLE**

### **PROGRAM TABLE**

-- ! LANGUAGE KAREL 2

-- ! MEMORY 8192

-- ! ROBOT MECHANISM

-- TEACHPOINT DECLARATIONS

VAR

AUX1 : AUXPOS

AUX2 : AUXPOS

AUX3 : AUXPOS

AUX4 : AUXPOS

AUX5 : AUXPOS

AUX6 : AUXPOS

AUX7 : AUXPOS

AUX8 : AUXPOS

AUX9 : AUXPOS

AUX10 : AUXPOS

AUX11 : AUXPOS

AUX12 : AUXPOS

AUX13 : AUXPOS

AUX14 : AUXPOS

AUX15 : AUXPOS

AUX16 : AUXPOS

AUX17 : AUXPOS

AUX18 : AUXPOS



```

AUX19 : AUXPOS
AUX20 : AUXPOS
AUX21 : AUXPOS
AUX22 : AUXPOS
AUX23 : AUXPOS
AUX24 : AUXPOS
AUX25 : AUXPOS
AUX26 : AUXPOS
AUX27 : AUXPOS
AUX28 : AUXPOS
AUX29 : AUXPOS
AUX30 : AUXPOS
AUX31 : AUXPOS
BEGIN
  SUSEMAXACCEL=TRUE
  %INCLUDE ANA-ALLA#
  --! DINSIGNALNAME DIN[1]
  --! DINSIGNALNAME DIN[2]
  --! DINSIGNALNAME DIN[3]
  --! DINSIGNALNAME DIN[4]
  --! DINSIGNALNAME DIN[5]
  --! DINSIGNALNAME DIN[6]
  --! DINSIGNALNAME DIN[7]
  --! DINSIGNALNAME DIN[8]
  --! DINSIGNALNAME DIN[9]

```

--! DINSIGNALNAME DIN[10]

--! DINSIGNALNAME DIN[11]

--! DINSIGNALNAME DIN[12]

--! DINSIGNALNAME DIN[13]

--! DINSIGNALNAME DIN[14]

--! DINSIGNALNAME DIN[15]

--! DINSIGNALNAME DIN[16]

--! DINSIGNALNAME DIN[17]

--! DINSIGNALNAME DIN[18]

--! DINSIGNALNAME DIN[19]

--! DINSIGNALNAME DIN[20]

--! DINSIGNALNAME DIN[21]

--! DOUTSIGNALNAME DOUT[1]

--! DOUTSIGNALNAME DOUT[2]

--! DOUTSIGNALNAME DOUT[3]

--! DOUTSIGNALNAME DOUT[4]

--! DOUTSIGNALNAME DOUT[5]

--! DOUTSIGNALNAME DOUT[6]

--! DOUTSIGNALNAME DOUT[7]

--! DOUTSIGNALNAME DOUT[8]

--! DOUTSIGNALNAME DOUT[9]

--! DOUTSIGNALNAME DOUT[10]

--! DOUTSIGNALNAME DOUT[11]

--! DOUTSIGNALNAME DOUT[12]

--! DOUTSIGNALNAME DOUT[13]  
--! DOUTSIGNALNAME DOUT[14]  
--! DOUTSIGNALNAME DOUT[15]  
--! DOUTSIGNALNAME DOUT[16]  
--! DOUTSIGNALNAME DOUT[17]  
--! DOUTSIGNALNAME DOUT[18]  
--! DOUTSIGNALNAME DOUT[19]  
--! DOUTSIGNALNAME DOUT[20]

--! SIGNALDEF DIN[1], ROBOT.KL,1  
--! SIGNALDEF DIN[2], ROBOT.KL,2  
--! SIGNALDEF DIN[3], ROBOT.KL,3  
--! SIGNALDEF DIN[4], ROBOT.KL,4  
--! SIGNALDEF DIN[5], ROBOT.KL,5  
--! SIGNALDEF DIN[6], ROBOT.KL,6  
--! SIGNALDEF DIN[7], ROBOT.KL,7  
--! SIGNALDEF DIN[8], ROBOT.KL,8  
--! SIGNALDEF DIN[9], ROBOT.KL,9  
--! SIGNALDEF DIN[10], ROBOT.KL,10

--! SIGNALDEF DIN[11], AZUR.KL,11  
--! SIGNALDEF DIN[12], AZUR.KL,12  
--! SIGNALDEF DIN[13], AZUR.KL,13  
--! SIGNALDEF DIN[14], AZUR.KL,14  
--! SIGNALDEF DIN[15], AZUR.KL,15

--! SIGNALDEF DIN[16], AZUR.KL,16  
--! SIGNALDEF DIN[17], AZUR.KL,17  
--! SIGNALDEF DIN[18], AZUR.KL,18  
--! SIGNALDEF DIN[19], AZUR.KL,19  
--! SIGNALDEF DIN[20], AZUR.KL,20  
--! SIGNALDEF DIN[21], AZUR.KL,21

--! SIGNALDEF DOUT[1], TRACK,1  
--! SIGNALDEF DOUT[2], TRACK,2  
--! SIGNALDEF DOUT[3], TRACK,3  
--! SIGNALDEF DOUT[4], TRACK,4  
--! SIGNALDEF DOUT[5], TRACK,5  
--! SIGNALDEF DOUT[6], TRACK,6  
--! SIGNALDEF DOUT[7], TRACK,7  
--! SIGNALDEF DOUT[8], TRACK,8  
--! SIGNALDEF DOUT[9], TRACK,9  
--! SIGNALDEF DOUT[10], TRACK,10

--! SIGNALDEF DOUT[11], TRACK,11  
--! SIGNALDEF DOUT[12], TRACK,12  
--! SIGNALDEF DOUT[13], TRACK,13  
--! SIGNALDEF DOUT[14], TRACK,14  
--! SIGNALDEF DOUT[15], TRACK,15  
--! SIGNALDEF DOUT[16], TRACK,16  
--! SIGNALDEF DOUT[17], TRACK,17

```

--! SIGNALDEF DOUT[18], TRACK,18

--! SIGNALDEF DOUT[19], TRACK,19

--! SIGNALDEF DOUT[20], TRACK,20

REPEAT

DOUT[1]=OFF

DOUT[2]=OFF

DOUT[3]=OFF

DOUT[4]=OFF

DOUT[5]=OFF

DOUT[6]=OFF

DOUT[7]=OFF

DOUT[9]=OFF

DOUT[10]=OFF

DOUT[11]=OFF

DOUT[12]=OFF

DOUT[13]=OFF

DOUT[14]=OFF

DOUT[15]=OFF

DOUT[16]=OFF

DOUT[17]=OFF

DOUT[19]=OFF

DOUT[20]=OFF

--!LABLEVENT 'Wait for man to pick up parts from pallet' , 0

```

REPEAT  
WAIT FOR (DIN[1]=ON) AND (DIN[11]=ON)  
UNTIL (DIN[1]=ON) AND (DIN[11]=ON)  
DOUT[8]=OFF  
DOUT[18]=OFF  
--!LABELEVENT 'Turn table for 180 degrees' , 4  
WITH SMOTYPE=JOINT  
MOVE AUX TO AUX15  
--!LABELEVENT 'Wait for a part' , 0  
DOUT[1]=ON  
DOUT[11]=ON

REPEAT  
WAIT FOR (DIN[2]=ON) AND (DIN[12]=ON)  
UNTIL (DIN[2]=ON) AND (DIN[12]=ON)  
DOUT[1]=OFF  
DOUT[11]=OFF  
--!LABELEVENT 'Turn table for 180 degrees' , 4  
WITH SMOTYPE=JOINT  
MOVE AUX TO AUX31  
--!LABELEVENT 'Wait for a part' , 0  
DOUT[2]=ON  
DOUT[12]=ON

```
REPEAT
WAIT FOR (DIN[3]=ON) AND (DIN[13]=ON)
UNTIL (DIN[3]=ON) AND (DIN[13]=ON)
DOUT[2]=OFF
DOUT[12]=OFF
--!LABELEVENT 'Turn table for 180 degrees' , 4
WITH SMOTYPE=JOINT
MOVE AUX TO AUX15
--!LABELEVENT 'Wait for a part' , 0
DOUT[3]=ON
DOUT[13]=ON
```

```
REPEAT
WAIT FOR (DIN[4]=ON) AND (DIN[14]=ON)
UNTIL (DIN[4]=ON) AND (DIN[14]=ON)
DOUT[3]=OFF
DOUT[13]=OFF
--!LABELEVENT 'Turn table for 180 degrees' , 4
WITH SMOTYPE=JOINT
MOVE AUX TO AUX31
--!LABELEVENT 'Wait for a part' , 0
DOUT[4]=ON
DOUT[14]=ON
```

REPEAT

WAIT FOR (DIN[5]=ON) AND (DIN[15]=ON)

UNTIL (DIN[5]=ON) AND (DIN[15]=ON)

DOUT[4]=OFF

DOUT[14]=OFF

--!LABELEVENT 'Turn table for 180 degrees' , 4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX15

--!LABELEVENT 'Wait a for part' , 0

DOUT[5]=ON

DOUT[15]=ON

REPEAT

WAIT FOR (DIN[6]=ON) AND (DIN[16]=ON)

UNTIL (DIN[6]=ON) AND (DIN[16]=ON)

DOUT[5]=OFF

DOUT[15]=OFF

--!LABELEVENT 'Turn table for 180 degrees' , 4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX31

--!LABELEVENT 'Wait for a part' , 0

DOUT[6]=ON

DOUT[16]=ON



```

REPEAT

WAIT FOR (DIN[7]=ON) AND (DIN[17]=ON)

UNTIL (DIN[7]=ON) AND (DIN[17]=ON)

DOUT[6]=OFF

DOUT[16]=OFF

--!LABELEVENT 'Turn table for 180 degrees' , 4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX15

--!LABELEVENT 'Wait for a part' , 0

DOUT[7]=ON

DOUT[17]=ON

```

```

REPEAT

WAIT FOR DIN[18]=ON

UNTIL DIN[18]=ON

DOUT[7]=OFF

DOUT[17]=OFF

--!LABELEVENT 'Turn table for 180 degrees' , 4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX31

DOUT[8]=ON

DOUT[18]=ON

UNTIL FALSE

END TABLE

```

## **PROGRAM AZUR**

PROGRAM AZUR

-- ! LANGUAGE KAREL 2

-- ! MEMORY 8192

-- ! ROBOT MR\_AZUR

-- TEACHPOINT DECLARATIONS

VAR

AUX1 : AUXPOS

AUX2 : AUXPOS

AUX3 : AUXPOS

AUX4 : AUXPOS

AUX5 : AUXPOS

AUX6 : AUXPOS

AUX7 : AUXPOS

AUX8 : AUXPOS

AUX9 : AUXPOS

AUX10 : AUXPOS

AUX11 : AUXPOS

AUX12 : AUXPOS

AUX13 : AUXPOS

AUX14 : AUXPOS

AUX15 : AUXPOS

AUX16 : AUXPOS

AUX17 : AUXPOS

AUX18 : AUXPOS

```

AUX19 : AUXPOS
AUX20 : AUXPOS
AUX21 : AUXPOS
AUX22 : AUXPOS
AUX23 : AUXPOS
AUX24 : AUXPOS
AUX25 : AUXPOS
AUX26 : AUXPOS
AUX27 : AUXPOS
AUX28 : AUXPOS
AUX29 : AUXPOS
AUX30 : AUXPOS
AUX31 : AUXPOS
-- NUMBER OF PARTS PRODUCED
PART : INTEGER
BEGIN
REPEAT
    SUSEMAXACCEL=TRUE
    %INCLUDE ANA-ALLA#
    --! DINSIGNALNAME DIN[1]
    --! DINSIGNALNAME DIN[2]
    --! DINSIGNALNAME DIN[3]
    --! DINSIGNALNAME DIN[4]
    --! DINSIGNALNAME DIN[5]
    --! DINSIGNALNAME DIN[6]

```

```

--! DINSIGNALNAME DIN[7]
--! DINSIGNALNAME DIN[8]
--! DINSIGNALNAME DIN[9]
--! DINSIGNALNAME DIN[10]
--! DOUTSIGNALNAME DOUT[1]
--! DOUTSIGNALNAME DOUT[2]
--! DOUTSIGNALNAME DOUT[3]
--! DOUTSIGNALNAME DOUT[4]
--! DOUTSIGNALNAME DOUT[5]
--! DOUTSIGNALNAME DOUT[6]
--! DOUTSIGNALNAME DOUT[7]
--! DOUTSIGNALNAME DOUT[8]
--! DOUTSIGNALNAME DOUT[9]
--! DOUTSIGNALNAME DOUT[10]
--! DOUTSIGNALNAME DOUT[11]

--! SIGNALDEF DIN[11], TABLE.KL,11
--! SIGNALDEF DIN[12], TABLE.KL,12
--! SIGNALDEF DIN[13], TABLE.KL,13
--! SIGNALDEF DIN[14], TABLE.KL,14
--! SIGNALDEF DIN[15], TABLE.KL,15
--! SIGNALDEF DIN[16], TABLE.KL,16
--! SIGNALDEF DIN[17], TABLE.KL,17
--! SIGNALDEF DIN[18], TABLE.KL,18
--! SIGNALDEF DIN[19], TABLE.KL,19

```

--! SIGNALDEF DIN[20], TABLE.KL,20

--! SIGNALDEF DOUT[11], TRACK,11

--! SIGNALDEF DOUT[12], TRACK,12

--! SIGNALDEF DOUT[13], TRACK,13

--! SIGNALDEF DOUT[14], TRACK,14

--! SIGNALDEF DOUT[15], TRACK,15

--! SIGNALDEF DOUT[16], TRACK,16

--! SIGNALDEF DOUT[17], TRACK,17

--! SIGNALDEF DOUT[18], TRACK,18

--! SIGNALDEF DOUT[19], TRACK,19

--! SIGNALDEF DOUT[20], TRACK,20

DOUT[11]=OFF

DOUT[12]=OFF

DOUT[13]=OFF

DOUT[14]=OFF

DOUT[15]=OFF

DOUT[16]=OFF

DOUT[17]=OFF

DOUT[18]=OFF

DOUT[19]=OFF

DOUT[20]=OFF

```

WITH SMOTYPE=JOINT

MOVE AUX TO AUX4

--!LABLEVENT 'Go to pick up part from pallet' , 2

--!USEMOVINGVIEW view1, view3,2

WITH SMOTYPE=JOINT

MOVE AUX TO AUX20

-- ! Attachobject,1,'CSG254','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX19

-- ! Detachobject,1,'CSG254'

-- ! Attachobject,1,'CSG254','cylinder7'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX20

-- ! Attachobject,1,'CSG247','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX21

-- ! Detachobject,1,'CSG247'

-- ! Attachobject,1,'CSG247','cylinder7'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX3

-- ! Attachobject,1,'EXTRUS98','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX1

-- ! Detachobject,1,'EXTRUS98'

-- ! Attachobject,1,'EXTRUS98','cylinder7'

```

```

-- ! Detachobject,1,'CSG254'

-- ! Attachobject,1,'CSG254','EXTRUS98'

-- ! Detachobject,1,'CSG247'

-- ! Attachobject,1,'CSG247','EXTRUS98'

MOVE AUX TO AUX4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX6

-- ! Attachobject,1,'CSG90','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX5

-- ! Detachobject,1,'CSG90'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX8

-- ! Attachobject,1,'CSG89','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX7

-- ! Detachobject,1,'CSG89'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX10

-- ! Attachobject,1,'CSG91','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX9

-- ! Detachobject,1,'CSG91'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX4

```

```

-- ! Attachobject,1,'CSG89','extrus98'

-- ! Attachobject,1,'CSG90','extrus98'

-- ! Attachobject,1,'CSG91','extrus98'

--!LABELEVENT 'Wait for table to rotate' , 0

DOUT[11]=ON

```

REPEAT

```

WAIT FOR DIN[11]=ON

UNTIL DIN[11]=ON

--!LABELEVENT 'Go pick up part from pallet' , 2

WITH SMOTYPE=JOINT

MOVE AUX TO AUX20

-- ! Attachobject,1,'CSG255','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX19

-- ! Detachobject,1,'CSG255'

-- ! Attachobject,1,'CSG255','cylinder7'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX20

-- ! Attachobject,1,'CSG262','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX21

-- ! Detachobject,1,'CSG262'

-- ! Attachobject,1,'CSG262','cylinder7'

WITH SMOTYPE=JOINT

```



```

MOVE AUX TO AUX3

-- ! Attachobject,1,'EXTRUS266','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX1

-- ! Detachobject,1,'EXTRUS266'

-- ! Attachobject,1,'EXTRUS266','cylinder7'

-- ! Detachobject,1,'CSG262'

-- ! Attachobject,1,'CSG262','EXTRUS266'

-- ! Detachobject,1,'CSG255'

-- ! Attachobject,1,'CSG255','EXTRUS266'

MOVE AUX TO AUX4

WITH SMOTYPE=JOINT

MOVE AUX TO AUX6

-- ! Attachobject,1,'CSG213','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX5

-- ! Detachobject,1,'CSG213'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX8

-- ! Attachobject,1,'CSG211','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX7

-- ! Detachobject,1,'CSG211'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX10

```

```

-- ! Attachobject,1,'CSG212','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX9

-- ! Detachobject,1,'CSG212'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX4

-- ! Attachobject,1,'CSG211','extrus266'

-- ! Attachobject,1,'CSG213','extrus266'

-- ! Attachobject,1,'CSG212','extrus266'

--!LABLEVENT 'Wait for table to rotate' , 0

DOUT[12]=ON


REPEAT

WAIT FOR DIN[12]=ON

UNTIL DIN[12]=ON

--!LABLEVENT 'Go pick up part from pallet' , 2

WITH SMOTYPE=JOINT

MOVE AUX TO AUX6

-- ! Attachobject,1,'EXTRUS109','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX5

-- ! Detachobject,1,'EXTRUS109'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX8

-- ! Attachobject,1,'EXTRUS65','R_HAND'

```

```

WITH SMOTYPE=JOINT

MOVE AUX TO AUX7

-- ! Detachobject,1,'EXTRUS65'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX10

-- ! Attachobject,1,'EXTRUS106','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX9

-- ! Detachobject,1,'EXTRUS106'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX4

-- ! Attachobject,1,'EXTRUS109','extrus98'

-- ! Attachobject,1,'EXTRUS65','extrus98'

-- ! Attachobject,1,'EXTRUS106','extrus98'

--!LABELEVENT 'Wait for table to rotate' . 0

DOUT[13]=ON

```

REPEAT

```

WAIT FOR DIN[13]=ON

UNTIL DIN[13]=ON

--!LABELEVENT 'Go pick up part from pallet' , 2

WITH SMOTYPE=JOINT

MOVE AUX TO AUX6

-- ! Attachobject,1,'EXTRUS305','R_HAND'

WITH SMOTYPE=JOINT

```

```

MOVE AUX TO AUX5

-- ! Detachobject,1,'EXTRUS305'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX8

-- ! Attachobject,1,'EXTRUS302','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX7

-- ! Detachobject,1,'EXTRUS302'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX10

-- ! Attachobject,1,'EXTRUS303','R_HAND'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX9

-- ! Detachobject,1,'EXTRUS303'

WITH SMOTYPE=JOINT

MOVE AUX TO AUX4

-- ! Attachobject,1,'EXTRUS305','extrus266'

-- ! Attachobject,1,'EXTRUS302','extrus266'

-- ! Attachobject,1,'EXTRUS303','extrus266'

--!LABLEVENT 'Wait for table to rotate' , 0

DOUT[14]=ON

REPEAT

WAIT FOR DIN[14]=ON

UNTIL DIN[14]=ON

```

## **NOTE TO USERS**

**Page(s) not included in the original manuscript are unavailable from the author or university. The manuscript was microfilmed as received.**

**139-140**

**This is reproduction is the best copy available**

**UMI**

```

MOVE AUX TO AUX28

--!USEMOVINGVIEW view2, view3,3

WITH $MOTYPE=JOINT

MOVE AUX TO AUX29

-- ! Detachobject,1,'EXTRUS98'


WITH $MOTYPE=JOINT

MOVE AUX TO AUX4

--!USEMOVINGVIEW view3, view2,1


--!LABELEVENT 'Wait for robot to finished welding' , 0

DOUT[17]=ON


REPEAT

WAIT FOR DIN[17]=ON

UNTIL DIN[17]=ON

--!LABELEVENT 'Put a part in a basket' , 2

--Set the labels for the different types of usage

--in this simulation so that Display events can used

--afterwards to show useful information

-- ! LABELUSAGE MAN,2

-- ! LABELUSAGE ROBOT,3

-- ! LABELUSAGE TABLE,4

WITH $MOTYPE=JOINT

MOVE AUX TO AUX25

```

```

-- ! Detachobject,1,'EXTRUS266'

-- ! Attachobject,1,'EXTRUS266','R_HAND'

WITH $MOTYPE=JOINT

MOVE AUX TO AUX28

--!USEMOVINGVIEW view2, view3,3

WITH $MOTYPE=JOINT

MOVE AUX TO AUX29

--!USEMOVINGVIEW view3, view1,1

-- ! Detachobject,1,'EXTRUS266'

WITH $MOTYPE=JOINT

MOVE AUX TO AUX4

DOUT[18]=ON

DOUT[11]=OFF

REPEAT

WAIT FOR DIN[18]=ON

UNTIL DIN[18]=ON

-- ! ENDOFJOB


-- ! Detachobject,1,'CSG255'

-- ! Placeobject,1,'CSG255',-554.9978,3068.9585,-244.4402,-136.9801,- &
29.7309,-18.1192

-- ! Detachobject,1,'CSG254'

-- ! Placeobject,1,'CSG254',-554.9978,3068.9585,-244.4403,-136.9801,- &
29.7309,-18.1192

-- ! Detachobject,1,'CSG213'

```

```

-- ! Placeobject,1,'CSG213',-1485.6829,1810.1866,695.1567,-114.6363,2.0884, &
27.4466

-- ! Detachobject,1,'CSG90'

-- ! Placeobject,1,'CSG90',-1485.6829,1810.1867,695.1567,-114.6363,2.0884, &
27.4466

-- ! Detachobject,1,'CSG211'

-- ! Placeobject,1,'CSG211',-1590.4275,1983.8669,-1106.8143,-167.7188,- &
15.5303,32.3849

-- ! Detachobject,1,'CSG89'

-- ! Placeobject,1,'CSG89',-1590.4275,1983.8669,-1106.8143,-167.7188,- &
15.5303,32.3849

-- ! Detachobject,1,'CSG247'

-- ! Placeobject,1,'CSG247',-771.8863,1529.4629,-1949.4995,146.3416,- &
32.3962,49.7031

-- ! Detachobject,1,'CSG262'

-- ! Placeobject,1,'CSG262',-771.8863,1529.4629,-1949.4995,146.3416,- &
32.3962,49.7031

-- ! Detachobject,1,'CSG212'

-- ! Placeobject,1,'CSG212',-1339.1779,637.0717,-1592.3544,169.3178,- &
12.5851,79.1982

-- ! Detachobject,1,'CSG91'

-- ! Placeobject,1,'CSG91',-1339.1779,637.0717,-1592.3544,169.3178,- &
12.5851,79.1982

-- ! Detachobject,1,'EXTRUS305'

-- ! Placeobject,1,'EXTRUS305',-1485.6829,1810.1866,695.1567,-114.6363, &

```



2.0884,27.4466

-- ! Detachobject,1,'EXTRUS109'

-- ! Placeobject,1,'EXTRUS109',-1485.6829,1810.1867,695.1567,-114.6363, &

2.0884,27.4466

-- ! Detachobject,1,'EXTRUS302'

-- ! Placeobject,1,'EXTRUS302',-1590.4275,1983.8669,-1106.8143,-167.7188,- &

15.5303,32.3849

-- ! Detachobject,1,'EXTRUS65'

-- ! Placeobject,1,'EXTRUS65',-1590.4275,1983.8669,-1106.8143,-167.7188,- &

15.5303,32.3849

-- ! Detachobject,1,'EXTRUS303'

-- ! Placeobject,1,'EXTRUS303',-1339.1779,637.0717,-1592.3544,169.3178,- &

12.5851,79.1982

-- ! Detachobject,1,'EXTRUS106'

-- ! Placeobject,1,'EXTRUS106',-1339.1779,637.0717,-1592.3544,169.3178,- &

12.5851,79.1982

-- ! Placeobject,1,'EXTRUS266',-859.7103,2446.5938,-1455.1798,166.6203, &

2.5124,-6.5378

-- ! Placeobject,1,'EXTRUS98',-859.7103,2446.5938,-1455.1798,166.6203, &

2.5124,-6.5378

UNTIL FALSE

END AZUR

**-- WORKSPACE teachpoint file**

AUX1[1] = -10

AUX1[2] = 0

AUX1[3] = 0

AUX1[4] = 0

AUX1[5] = 0

AUX1[6] = 0

AUX1[7] = 50

AUX1[8] = 0

AUX1[9] = 20

AUX1[10] = 0

AUX1[11] = 0

AUX1[12] = 0

AUX1[13] = 30

AUX1[14] = 0

AUX1[15] = 35

AUX1[16] = 10

AUX1[17] = 0

AUX1[18] = 0

**-- END AUX1**

AUX2[1] = 0

AUX2[2] = -110

AUX2[3] = -10

AUX2[4] = 0

AUX2[5] = 0

```
AUX2[6] = 0
AUX2[7] = 50
AUX2[8] = 0
AUX2[9] = 20
AUX2[10] = 0
AUX2[11] = 0
AUX2[12] = 0
AUX2[13] = 30
AUX2[14] = 0
AUX2[15] = 35
AUX2[16] = 10
AUX2[17] = 0
AUX2[18] = 0
-- END AUX2

AUX3[1] = 10
AUX3[2] = -120
AUX3[3] = -50
AUX3[4] = 0
AUX3[5] = 0
AUX3[6] = 0
AUX3[7] = 50
AUX3[8] = 0
AUX3[9] = 20
AUX3[10] = 0
AUX3[11] = 0
```

AUX3[12] = 0

AUX3[13] = 30

AUX3[14] = 0

AUX3[15] = 35

AUX3[16] = 10

AUX3[17] = 0

AUX3[18] = 0

-- END AUX3

AUX4[1] = 0

AUX4[2] = 0

AUX4[3] = 0

AUX4[4] = 0

AUX4[5] = 0

AUX4[6] = 0

AUX4[7] = 0

AUX4[8] = 0

AUX4[9] = 0

AUX4[10] = 0

AUX4[11] = 0

AUX4[12] = 0

AUX4[13] = 0

AUX4[14] = 0

AUX4[15] = 0

AUX4[16] = 0

AUX4[17] = 0

```
AUX4[18] = 0
-- END AUX4
AUX5[1] = -30
AUX5[2] = 40
AUX5[3] = 0
AUX5[4] = 0
AUX5[5] = 30
AUX5[6] = 0
AUX5[7] = 43
AUX5[8] = -20
AUX5[9] = 30
AUX5[10] = 0
AUX5[11] = 0
AUX5[12] = 0
AUX5[13] = 39
AUX5[14] = -10
AUX5[15] = 30
AUX5[16] = 0
AUX5[17] = 0
AUX5[18] = 0
-- END AUX5
AUX6[1] = -10
AUX6[2] = -35
AUX6[3] = -55
AUX6[4] = 0
```

AUX6[5] = -40

AUX6[6] = 0

AUX6[7] = 40

AUX6[8] = 20

AUX6[9] = 10

AUX6[10] = 0

AUX6[11] = 0

AUX6[12] = 0

AUX6[13] = 0

AUX6[14] = 0

AUX6[15] = 0

AUX6[16] = 0

AUX6[17] = 0

AUX6[18] = 0

-- END AUX6

AUX7[1] = 0

AUX7[2] = 25

AUX7[3] = 0

AUX7[4] = 0

AUX7[5] = 0

AUX7[6] = 0

AUX7[7] = 45

AUX7[8] = 0

AUX7[9] = 26.5

AUX7[10] = 0

AUX7[11] = 0

AUX7[12] = 0

AUX7[13] = 0

AUX7[14] = 0

AUX7[15] = 0

AUX7[16] = 0

AUX7[17] = 0

AUX7[18] = 0

-- END AUX7

AUX8[1] = -10

AUX8[2] = -55

AUX8[3] = -50

AUX8[4] = 0

AUX8[5] = 0

AUX8[6] = 0

AUX8[7] = 15

AUX8[8] = 0

AUX8[9] = 26.5

AUX8[10] = 0

AUX8[11] = 0

AUX8[12] = 0

AUX8[13] = 0

AUX8[14] = 0

AUX8[15] = 0

AUX8[16] = 0

```
AUX8[17] = 0
AUX8[18] = 0
-- END AUX8

AUX9[1] = -15
AUX9[2] = -15
AUX9[3] = 0
AUX9[4] = 0
AUX9[5] = -20
AUX9[6] = -10
AUX9[7] = 50
AUX9[8] = 7
AUX9[9] = 21
AUX9[10] = 0
AUX9[11] = 0
AUX9[12] = 0
AUX9[13] = 0
AUX9[14] = 0
AUX9[15] = 0
AUX9[16] = 0
AUX9[17] = 0
AUX9[18] = 0
-- END AUX9

AUX10[1] = -5
AUX10[2] = -45
AUX10[3] = -50
```



AUX10[4] = 0  
AUX10[5] = -20  
AUX10[6] = -10  
AUX10[7] = 20  
AUX10[8] = 12  
AUX10[9] = 21  
AUX10[10] = 0  
AUX10[11] = 0  
AUX10[12] = 0  
AUX10[13] = 0  
AUX10[14] = 0  
AUX10[15] = 0  
AUX10[16] = 0  
AUX10[17] = 0  
AUX10[18] = 0  
-- END AUX10  
AUX11[1] = -5  
AUX11[2] = 50  
AUX11[3] = 0  
AUX11[4] = 0  
AUX11[5] = 0  
AUX11[6] = 0  
AUX11[7] = 40  
AUX11[8] = 0  
AUX11[9] = 33

AUX11[10] = 0

AUX11[11] = 0

AUX11[12] = 0

AUX11[13] = 0

AUX11[14] = 0

AUX11[15] = 0

AUX11[16] = 0

AUX11[17] = 0

AUX11[18] = 0

-- END AUX11

AUX12[1] = -5

AUX12[2] = -50

AUX12[3] = -60

AUX12[4] = 0

AUX12[5] = 0

AUX12[6] = 0

AUX12[7] = 30

AUX12[8] = 5

AUX12[9] = 33

AUX12[10] = 0

AUX12[11] = 0

AUX12[12] = 0

AUX12[13] = 0

AUX12[14] = 0

AUX12[15] = 0

```
AUX12[16] = 0
AUX12[17] = 0
AUX12[18] = 0
-- END AUX12

AUX13[1] = 0
AUX13[2] = -50
AUX13[3] = -50
AUX13[4] = 0
AUX13[5] = -10
AUX13[6] = -20
AUX13[7] = 30
AUX13[8] = 5
AUX13[9] = 15.5
AUX13[10] = 0
AUX13[11] = 0
AUX13[12] = 0
AUX13[13] = 0
AUX13[14] = 0
AUX13[15] = 0
AUX13[16] = 0
AUX13[17] = 0
AUX13[18] = 0
-- END AUX13

AUX14[1] = 0
AUX14[2] = 0
```

```
AUX14[3] = 0
AUX14[4] = 0
AUX14[5] = 0
AUX14[6] = 0
AUX14[7] = 30
AUX14[8] = 10
AUX14[9] = 45
AUX14[10] = 0
AUX14[11] = 0
AUX14[12] = 0
AUX14[13] = 0
AUX14[14] = 0
AUX14[15] = 0
AUX14[16] = 0
AUX14[17] = 0
AUX14[18] = 0
-- END AUX14
AUX15[1] = 170
-- END AUX15
AUX16[1] = 350
-- END AUX16
AUX17[1] = 530
-- END AUX17
AUX18[1] = 710
-- END AUX18
```

AUX19[1] = -10

AUX19[2] = 40

AUX19[3] = 0

AUX19[4] = 0

AUX19[5] = 10

AUX19[6] = 0

AUX19[7] = 40

AUX19[8] = -15

AUX19[9] = 29

AUX19[10] = 0

AUX19[11] = 0

AUX19[12] = 0

AUX19[13] = 0

AUX19[14] = 0

AUX19[15] = 0

AUX19[16] = 0

AUX19[17] = 0

AUX19[18] = 0

-- END AUX19

AUX20[1] = 0

AUX20[2] = -80

AUX20[3] = -40

AUX20[4] = 0

AUX20[5] = -50

AUX20[6] = -30

AUX20[7] = 40

AUX20[8] = 5

AUX20[9] = -21

AUX20[10] = 0

AUX20[11] = 0

AUX20[12] = 0

AUX20[13] = 0

AUX20[14] = 0

AUX20[15] = 0

AUX20[16] = 0

AUX20[17] = 0

AUX20[18] = 0

-- END AUX20

AUX21[1] = -10

AUX21[2] = -15

AUX21[3] = 0

AUX21[4] = 0

AUX21[5] = -10

AUX21[6] = -10

AUX21[7] = 20

AUX21[8] = 0

AUX21[9] = 56

AUX21[10] = 0

AUX21[11] = 0

AUX21[12] = 0

```
AUX21[13] = 0
AUX21[14] = 0
AUX21[15] = 0
AUX21[16] = 0
AUX21[17] = 0
AUX21[18] = 0
-- END AUX21
AUX22[1] = -10
AUX22[2] = 0
AUX22[3] = 0
AUX22[4] = 0
AUX22[5] = 0
AUX22[6] = 0
AUX22[7] = 70
AUX22[8] = 0
AUX22[9] = 20
AUX22[10] = 0
AUX22[11] = 0
AUX22[12] = 0
AUX22[13] = 50
AUX22[14] = 0
AUX22[15] = 35
AUX22[16] = 10
AUX22[17] = 0
AUX22[18] = 0
```

```
-- END AUX22

AUX23[1] = -20
AUX23[2] = 0
AUX23[3] = 0
AUX23[4] = 0
AUX23[5] = 0
AUX23[6] = 0
AUX23[7] = 51
AUX23[8] = 0
AUX23[9] = 20
AUX23[10] = 0
AUX23[11] = 0
AUX23[12] = 0
AUX23[13] = 51
AUX23[14] = 0
AUX23[15] = 21.5
AUX23[16] = 0
AUX23[17] = 0
AUX23[18] = 0
-- END AUX23

AUX24[1] = -10
AUX24[2] = 0
AUX24[3] = 0
AUX24[4] = 0
AUX24[5] = 0
```



AUX24[6] = 0  
AUX24[7] = 70  
AUX24[8] = 0  
AUX24[9] = 20  
AUX24[10] = 0  
AUX24[11] = 0  
AUX24[12] = 0  
AUX24[13] = 50  
AUX24[14] = 0  
AUX24[15] = 35  
AUX24[16] = 10  
AUX24[17] = 0  
AUX24[18] = 0  
-- END AUX24  
AUX25[1] = -20  
AUX25[2] = 0  
AUX25[3] = 0  
AUX25[4] = 0  
AUX25[5] = 0  
AUX25[6] = 0  
AUX25[7] = 50  
AUX25[8] = 0  
AUX25[9] = 35  
AUX25[10] = 0  
AUX25[11] = 0

```
AUX25[12] = 0
AUX25[13] = 39
AUX25[14] = 0
AUX25[15] = 35
AUX25[16] = 10
AUX25[17] = 0
AUX25[18] = 0
-- END AUX25
AUX26[1] = 890
-- END AUX26
AUX27[1] = 1070
-- END AUX27
AUX28[1] = -20
AUX28[2] = 0
AUX28[3] = 0
AUX28[4] = 0
AUX28[5] = 0
AUX28[6] = 0
AUX28[7] = 65
AUX28[8] = 0
AUX28[9] = 35
AUX28[10] = 0
AUX28[11] = 0
AUX28[12] = 0
AUX28[13] = 55
```

```
AUX28[14] = 0
AUX28[15] = 35
AUX28[16] = 10
AUX28[17] = 0
AUX28[18] = 0
-- END AUX28

AUX29[1] = 0
AUX29[2] = 110
AUX29[3] = -60
AUX29[4] = 0
AUX29[5] = 0
AUX29[6] = 0
AUX29[7] = 55
AUX29[8] = -5
AUX29[9] = 15
AUX29[10] = 0
AUX29[11] = 0
AUX29[12] = 0
AUX29[13] = 55
AUX29[14] = 0
AUX29[15] = 10
AUX29[16] = 10
AUX29[17] = 0
AUX29[18] = 0
-- END AUX29
```

AUX30[1] = 1250

-- END AUX30

AUX31[1] = -10

-- END AUX31

# **VITA AUCTORIS**

**NAME:** Ana Djuric

**PLACE OF BIRTH:** Uzice, Yugoslavia

**EDUCATION:** University of Windsor, Windsor, Ontario

1997 – 1999 M.A.Sc

University of Belgrade, Faculty of Mechanical Engineering,

Belgrade, Yugoslavia

B.Sc. 1993