1991

Automatic programming of arc welding robots

Srikanth Padmanabhan
Iowa State University

Order Number 9207252

Automatic programming of arc welding robots

Padmanabhan, Srikanth, Ph.D.

Iowa State University, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Automatic programming of arc welding robots

by

Srikanth Padmanabhan

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1991

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I express my sincere thanks to Professor Mohan S. Devgun for being my major professor, advisor, and friend. Throughout the course of my study, he had extended freedom to pursue my own research interests and ideas, while at the same time always providing helpful hints and invaluable guidance.

I gratefully acknowledge Dr. Scrutton's confidence-building-remarks and am indebted to him for critically reading the initial drafts of the various chapters of this thesis.

Many thanks to Professor James E. Bernard for serving on the committee as a co-major professor, and correcting my thesis. I thank Drs. Even, Strawn, and Vanderploeg for being on my committee and taking their valuable time to read through my dissertation.

Drs. Henkin and Bernard have inspired me to be better at whatever I am pursuing and I will forever remember the valuable lessons learnt through them.

I acknowledge Dr. Hall and the R. A. Engel CIM Laboratory, Department of Mechanical Engineering for providing the necessary financial support without which none of this would have been possible. My sincere thanks to the welding engineers at John Deere & Co. and Ford New Holland for their response and professional insight.

I deeply appreciate the many hours Dr. Flugrad, Thiag, and Varma spent with

me in discussing my research problem and helping me with their suggestions. I thank Bruce for his help on the figures.

I thank my roommate Bala for introducing me to some of the greatest music I have ever heard. I thank Julie, Bala, and Raman for the many wonderful discussions and rationalizations which always helped me to see life in a different perspective. The Raghavans, the Ramans, the Athreyas, Thiag, Anand, and Sridhar made life in Ames a pleasant and memorable experience and my many thanks to all of them. Special thanks to the Espelands and the Scandretts for all their help, and putting up with me and my idiosyncracies. And finally thanks to the lunch group – Brian, Bruce, Gamal, Gaylord, Peri, and Thiag for their wonderful company.

I owe most of what I have achieved until now to my parents and family. To my mother, who wanted me to be a "Doctor" more than anyone else – this one is for you mom!

CHAPTER 1. INTRODUCTION

Robotic arc welding is one of the fastest growing applications of industrial robots. Robotic arc welding operations provide flexibility for accomodating a family of products and are capable of providing consistent quality welds [Lane 1987]. The use of arc welding robots in small volume production is limited due to the time-consuming preparation of the welding process: programming the robot motions, specifying the process parameters for the weld schedule, and fixturing/positioning the workpiece. This research presents a solution to these problems by means of an integrated "computer-aided engineering environment" (CAEE) that will automatically generate robot arc welding programs.

Since the late 1970's, arc welding robots have been used primarily in the high volume industries such as automobile and farm equipment manufacturer. The majority of robotic welding installations fabricate relatively large batches of selected workpieces that are classified as "primarily robot-attractive." It is obvious that the principal reason for using arc welding robots is not their flexibility but the shortened cycle times compared to manual welding [Zimmerman 1990]. This seemingly contradicts the initial purpose and the advantages expected of industrial robots – flexibility. However, serious practical limitations exist pertaining to the fixturing of the workpiece and the programming of the robots for use in small volume production.

The robot industry has matured over the years and steady improvements are being made towards increasing the flexibility of application.

The common methods of robot programming have been: (1) by teaching, (2) by means of a high level textual language, (3) by means of a graphical off-line programming system. Robotic arc welding systems have been programmed primarily by on-line teaching. Although it is relatively easy to learn "teach pendant programming", as problems become more complicated (as in typical arc welding workcells), the programming time increases dramatically [Charny 1984]. The robot is occupied during teaching and the selection of proper parameters, both geometrical and technological, requires qualified personnel. This problem led to the development of high level textual languages such as VAL, RAIL, AML, and MCL [Gruver et al. 1984]. Although program logic can be developed without the use of robots, or prototype parts, the specification of the coordinates for the movement of the robot is typically attempted by teaching. Furthermore, there are no means of explicit selection of welding process parameters and there is no way of debugging the program without executing the program on-line. In response to these concerns, graphical off-line programming systems are being developed to generate simulation methods, graphical debugging procedures and layout techniques for robot workcells. However, programs generated by off-line programming systems are not used directly in arc welding workcells due to the inaccuracies of wire impingement on the workpiece relative to the robot tool center point.

Furthermore, contemporary CAD systems do not produce a complete product definition. The lack of complete data in the computer-aided design (CAD) database does not permit the necessary reasoning and automatic generation of process param-

eters for the welding schedule. Rather, the information is provided in the form of low level geometric primitives from which higher level abstractions cannot be easily inferred and does not also include specifications of welding features. As a result, even though various systems have addressed parts of the problem, robot programming for arc welding is still primarily undertaken by lead-through teaching methods.

Research Goals

Few attempts have been made to provide systems for automatic programming of arc welding robots. Also no attempt seems to have been made to provide complete definition of the weld features in a CAD database, or to use the geometric modeling system as the core for the automatic programming system for robotic arc welding. Furthermore, a mapping shell to generate welding procedures based on a CAD representation has not been attempted. Continuous path planning of an integrated welding robot and positioner has received very little attention. The overall objective of this research therefore is to build a computer-aided engineering environment that will generate robot arc welding programs automatically based on the geometry, process, and kinematic constraints. This sytem will eventually assist in the utilization of arc welding robots in small series production. The specific goals of this study are:

1. To develop methodologies and data abstractions by which all weld feature information will be encapsulated and incorporated in a solid modeling system based on constructive solid geometry;

2. To develop a methodology for the acquisition and representation of gas metal arc welding (GMAW) process knowledge and a mapping shell for the auto-

matic selection of the welding schedule based on the geometric and kinematic constraints;

3. To develop a method for planning continuously the trajectory of an integrated 5-axis robot/2-axis positioner;

4. To implement a prototype system based on the methodologies developed and to validate the system response with those of robotic welding experts.

Dissertation Organization

The second chapter of the thesis presents a literature review. The review focuses on three areas: contemporary efforts in modeling systems to incorporate high level geometric information, expert systems for welding, and trajectory planning and kinematics for automatic robot programming systems.

Details of the data abstractions and the representation of weld features and attributes in a solid modeling system are provided in Chapter 3. Chapter 4 explains the weld process knowledge mapping system, along with the methods of knowledge acquisition and representation. The implementation of the integrated welding representation structure and the weld process knowledge is described in Chapter 5. The method of planning the trajectory for an integrated robot/positioner using the novel modified continuation approach is discussed in Chapter 6. The conclusions and scope for future work are summarized in Chapter 7. Appendices provide additional details of the system development.

CHAPTER 2. LITERATURE REVIEW

The research accomplished in this dissertation can be broadly categorized into three areas: representation of weld feature information in a solid modeling system, development of a novel mapping system for weld process knowledge to obtain process schedules, and the continuous path planning of an integrated robot/positioner system. This chapter provides a summary of literature in these three areas. The first Section reviews efforts in geometric modeling and research related to the addition of process information in geometric models. The second Section is a review of knowledge-based systems in welding, including a discussion of means for acquiring and representing weld process knowledge. The final Section describes the status of off-line programming of industrial robots and gives a description of the various methods for obtaining the kinematics of an integrated robot and positioner system used typically in arc welding workcells.

Geometric Modeling For Automated Manufacturing

Computer-aided modeling systems have evolved over the past four decades, with early research propelled by advances in CAM. Figure 2.1 traces the evolution of this technology. A discussion of modeling methods is provided in two survey papers [Requicha 1980, Requicha and Voelcker 1982].

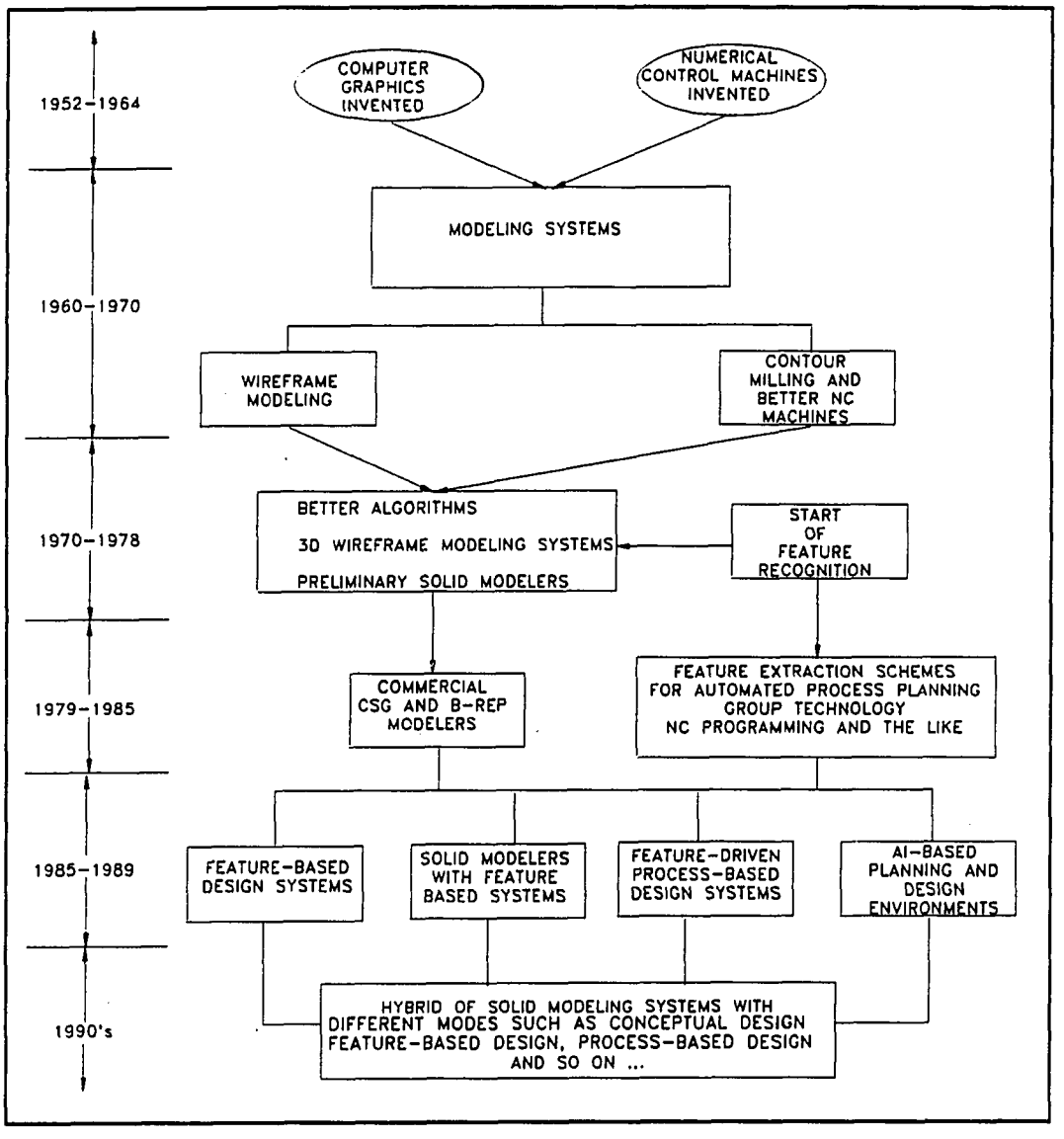


Figure 2.1: Evolution of geometric modeling technology

Until the advent of solid modeling systems wireframe modeling systems supplied the base for all commercial CAD/CAM systems. Wireframe was then replaced by solid modeling as the most unambiguous method of representing solids. Boundary representation (B-Rep) and constructive solid geometry (CSG) are the two most widely used techniques in current modeling systems. In a B-Rep model, a solid is represented by segmenting its boundary into a finite number of faces and representing each face by its bounding edges and vertices. The representation is a directed graph containing object, face, edge, and vertex nodes. The principal advantage of this method is the availability of information, which can be used for subsequent downstream applications. CSG relates to a family of schemes for representing solids as a boolean construction or a combination of solid components. The representation structure is a binary tree where nonterminal nodes represent operators, which may be either rigid motions or regularized union, intersection, or difference operations and terminal nodes, which are primitive leaves or transformation leaves that contain the defining arguments. CSG trees are less efficient than B-Rep, but are highly useful for automation from the rough machining perspective since volumes of material to be removed are implicitly represented by a CSG modeling system.

A shortcoming of B-Rep and CSG is that they do not provide means to model the effects of a process in the design stage that can also be used in the automatic planning of machine operations [Devgun and Padmanabhan 1990]. The manufacturing information that needs to be incorporated is wide ranging, from materials specifications to representation of tolerances, surface finish, and weld information which can be termed secondary feature information or variational information. Traditionally none of this data are held in the CAD database. This gap results in the inability to

support automatic querying and retrieval schemes, which in turn inhibit the development of automatic planning/programming in the machining, welding, and assembly disciplines of manufacturing.

This deficiency has led to two streams of research effort. The first concerned with the extraction of implicit information from a CAD database, and the second concerned with representing explicitly manufacturing information on a solid model. However, both efforts lack total automatic planning and programming systems in the manufacturing domain. Further details concerning feature extraction are given in the paper by Devgun and Padmanabhan [1990]. Representation of explicit information is often termed feature-based design and the next Section presents a literature review on this topic.

Feature-based modeling systems

The prime elements of a feature-based modeling system, so called features, are high-order abstract forms or entitites that are used in a reasoning system and are related to the topology and geometry of designed artifacts during various design and manufacturing activities [Cunningham and Dixon 1988]. The primary purpose of a feature is to represent the specific geometric entities of a product.

Features themselves have been primarily researched in the area of form features: those fixed geometries which can be produced by machining operations. Features for other areas of application are limited. Pratt and Wilson [1985] laid the fundamentals for the functional requirements of a form feature modeler in a solid modeling context. Representation in a solid model (both CSG and B-REP) was considered along with the conceptual facilities required for manipulating these features. This research set

the tone for much of the other related efforts in form feature modeling. At the same time, J. R. Dixon and his group at Amherst [Vaghul et al. 1985, Dixon et al. 1987] created sets of feature-based design systems (FBDS) for varied applications. Miner [1985] developed a prototype solid modeling system to support form features.

Hummel and Brooks [1986] used an object-oriented programming approach to represent features in a hierarchical format. They utilized the features to represent the volume of material that must be removed to form the final product from the rough stock. A conceptual hierarchical structure for the representation of form features in a database was introduced by Gindy [1989]. Shah and his group at Arizona State University [1988] extended Pratt and Wilson's [1985] scheme further to implement an FBDS for form features. Representational and interpretation issues were considered, and a prototype system was implemented for a solid modeling system. Other applications that have been implemented by this group include tolerancing [Shah and Miller 1989], group technology [Shah and Bhatnagar 1989], and manufacturability evaluation [Shah et al. 1990].

The new generation of feature-based design systems promises to close the gap between design and manufacturing. It represents the logical next step from existing solid modelers and captures the manufacturers' intent at the design stage. However, these systems have certain limitations including the difficulty involved in determining a closed set of features. Furthermore, agreement concerning which sets or classes of features can represent a modeling system requires compromise. Feature recognition and feature-based design systems have been used primarily for form features with minor exceptions for casting and extrusion. A considerable number of manufacturing disciplines are totally neglected. The scope of the CAD/CAM link should include

other manufacturing technologies such as metal forming, unconventional metal removal processes, metal joining processes, and others to enable horizontal integration in the overall context [Srinivasan and Liu 1987].

Secondary feature representation

Since CAD models are incomplete, when human manufacturing experts interpret geometry, they apply a great deal of ancillary information, analyze the intentions of the designers, and so on. This is particularly important when interpreting drawings that include tolerances, weld features, and other surface feature information which we term as secondary feature or variational information. In order to ensure that designs are interpreted unambiguously, standard methodologies have been set up to represent information that requires human visualization [Welding Handbook 1984, ANSI Y14.5M 1982]. The need for representing this information in CAD models has been recognized only recently. There has been active work in representing tolerancing information in geometric models [Requicha and Chan 1986, Shah and Miller 1989]. However, we are unaware of any research reports concentrating on the representation of welding information in solid models. One of the aims of the research presented in this thesis is to provide a structure for representing weld feature information. Since tolerancing is primarily variational attribute information that is added to geometric features, an analogy can be drawn between tolerance information and weld feature information. In order to understand the methods for representing secondary feature information, the following paragraphs describe the methods used to represent tolerance information in CAD models. These will be used to draw conclusions regarding the development of a welding representation scheme for solid modelers.

Tolerances represented on a drawing include size, form, orientation, position, and material conditions. Methods proposed to support tolerances either assume that features exist in the model (so that information can be attached to these features), or provide mechanisms for constructing features from a solid geometry model to which property information may be attached. Shah and Miller [1989] provide a useful summary classifying these approaches as non-parametric, parametric, evaluated entity structures, and hybrid CSG - B-Rep structures. The interest here lies in the non-parametric and evaluated entity approaches. The non-parametric approach which does not assume form features to exist apriori in the model [Requicha and Chan 1986], is based on a CSG modeler, while the evaluated entity approach used by Shah and Miller [1989] assumes that there exists a form feature modeler and attaches information to the features so that the meaning of the tolerance can be encapsulated.

The specification of tolerances by the non-parametric approach refers to evaluated geometric entities since the scheme is implemented in PADL-2 [Brown 1982] (a modeling system based on CSG). The basic structure proposed for representing features and attributes is called a variational graph (VGRAPH) and the structure is straightforward. The variational information associated with a solid is represented by a graph whose node types are: (1) VFACE: User defined portions of the object boundary (2) SFEATS: Surface feature which are group of VFACES (3) VEDGES: User defined subsets of an object (4) CFEATS: Curve features such as SFEATS (5) ATTRIBUTES: That associated with CFEATS and SFEATS (6) DATASYS: To determine datum systems.

To refer to the faces of a solid and to attach property information, these authors have used the 2-D intersection operator to intersect with a virtual object. Tolerance

attributes are attached to the VFACES or SFEAT nodes. These nodes provide the ability to add six classes of tolerances: Intrinsic, extrinsic surface and curve attributes, and measured entity attributes. Each provides a name, data fields, and pointers to the respective tolerance classes. In designing the system on PADL-2, means are provided for naming primitive instances and to incrementally create a VGRAPH by path names and a bottom-up inheritance mechanism. By using default values, checking consistency of centerplanes, axes, and centers of features with nominal geometry, and checking the validity of datum systems, the overall validity of the VGRAPH representation is assured. Requicha and Chan [1986] provide an excellent solution for the tolerancing problem that was not addressed previously. Although many research problems in tolerancing still remain, this work provides a good base to modify the existing structures of geometric models.

Another approach of interest is the evaluated entity method which assumes that the feature exists in a geometric model. The research of Shah and Miller [1989] presents a structure by which the meaning of each tolerance class is encapsulated. The tolerance modeler is part of the larger interactive design system that includes a feature modeler, solid modeler, and a wireframe modeler. Users access the tolerance modeler, choose features and instance the desired tolerance class. The template of each tolerance class is a dynamically linked list which provides a list of properties used by the class. It refers to the tolerance properties needed, the necessary rules to check validity and the means whereby the information is connected for evaluation. Each instance class stores the properties of the tolerance class corresponding to the feature-id (geometric entity). Datum systems are created by referring to the geometric entity (feature-id) and the material condition. Although the structure is simple, it captures

most of the requirements of a tolerance modeler. However the assumption is made that the entire geometry is created by features. For complicated shapes, it is not always possible to use features. The non-parametric approach provided earlier is better suited for general solid modelers. Nevertheless the structure developed allowed certain broad conclusions to be drawn concerning the direction of the work described in this thesis.

Based on published research in describing the representation of manufacturing information in a geometric model we conclude that:

1. Features are primarily created for elements that can be manufactured by machining (with certain exceptions for castings).

2. Tolerances are the only secondary feature representation that has been incorporated in a geometric model.

3. Incorporation of weld features and attributes in a solid model has not yet been attempted.

Knowledge-Based Systems For Arc Welding

Gas metal arc welding (GMAW) is a process wherein coalescence is produced by heating with an arc between a continuous consumable filler metal electrode and the workpiece. Shielding from oxidation or contamination by surrounding atmosphere is provided by a gas mixture [Welding Handbook 1984]. Arc welding operations can be manual or automatic. Automatic arc welding can either be hard automated or carried out by robots. One important aspect of arc welding is the specification of

Table 2.1: Parameters affecting gas metal arc welding

Product parameters	Process Parameters
Material Type	Current
Material Thickness	Voltage
Position of weld	Wire feed rate
Geometry	Travel speed
Type of joint	Electrode type
	Electrode size
	Arc length
	Gas mixture
	Gas flow rate
	Robot tool angle
	Number of passes
	Weave pattern
	Polarity
	Offset for multi-pass welds

process schedules for a given weld. The parameters that need to be considered in GMAW are given in Table 2.1.

The product parameters are decided by the design engineer and are functional in nature. However, the process parameters are obtained typically based on rule-of-thumb knowledge , data from empirical relationships and experience, and experimental data. Various methods have been proposed to specify weld schedules and most of them depend on empirical data [Tonkay and Knott 1989].

In robotic arc welding also, the programming of robots requires considerable knowledge of the process. The difficulties in programming arise due to lack of:

- definition of proper welding information

- definition of the orientation of the weld gun relative to the weld joint
- definition of the orientation of the workpiece when welding a particular joint
- definition of the proper sequence of welds for complicated shapes

It is crucial that the details of the weld process knowledge is consistently represented to automate these steps. However traditional algorithmic approaches are not always successful in characterizing the problem and providing solutions. A system that can simulate the human reasoning process and be consistent in its solutions will be an adequate model. To automate this decision making process, researchers have found the most valuable tool to be knowledge-based expert systems (KBES), the best developed subset of artificial intelligence (AI).

The AI-based approach is designed to make decisions or recommendations concerning industrial operations at a level of performance comparable to that of experienced humans [Hayes-Roth et al. 1983]. The field of KBES has been well established in the past few years and the basic structure is now clearly understood. Expert systems form a significant tool in automating human expertise, and successful standalone systems have been created in many areas of manufacturing [Kumara et al. 1985]. Manufacturing process control expert systems have been developed in the area of welding. The welding expert systems deal mainly with the control of process parameters for welding, the control of the nature of the weld, and the maintenance of the weld quality.

Figure 2.2 is a representation of the basic KBES used in welding. The prototypes developed are typically structured in two levels of hierarchy; the first level provides the supervisory program that serves as the user-interface for receiving the necessary

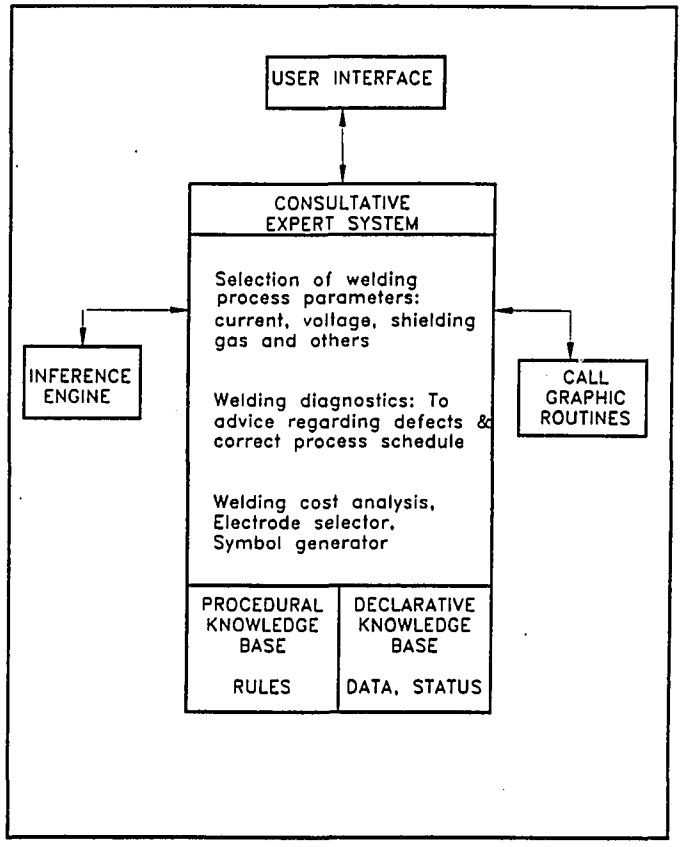


Figure 2.2: Overall structure of a welding expert system

information, and the second level performs the actual search for the correct solution space for the given problem. The module which specifies the process schedule receives the user-supplied information concerning the geometry of the weld. On the basis of this information, the module searches its knowledge base to provide the solution.

Acquisition of the knowledge required for building the expert system is the first step. Knowledge in any specialty is usually of two sorts; public and private. Public knowledge includes the published definitions, facts, and theories from textbooks and reference texts. But expertise normally involves more than this public knowledge.

Humans generally possess private knowledge that consists largely of rules-of-thumb and heuristics that have evolved from experience. In effect, the knowledge acquisition is the transfer and transformation of problem solving expertise from some knowledge source to a program. The various stages involved in the knowledge acquisition process are: (1) identification (2) conceptualization (3) formalization (4) implementation (5) testing. The knowledge acquired is then represented by a combination of data structure and interpretive procedures. This involves the design of several classes of data structure for storing information as well as the development of procedures for intelligent manipulation. During the consultation, the knowledge base provides the library of logical considerations (facts and rules) that govern the problem domain and the inference engine decides on the course of action. The knowledge base is normally controlled by the backward chaining mechanism that has the following characteristics: (1) goal driven (2) seeks only data needed for the solution (3) instantiates subframes on its path to the goal. Forward chaining strategy is normally used to override certain conditions and fire priority rules. Expert systems are normally written using AI languages such as LISP, or Prolog or by using expert system shells.

The various expert systems for welding which are commercially available or developed as prototype systems in universities are given in Table 2.2 [Barborak et al. [1991], Cary [1991], and Lucas [1987]]. The common problems solved by these expert systems are those of welding procedure generation, diagnostics-realtime and off-line, and cost analysis. Procedure generation programs prompt the user for relevant information including joint design, welding process and design characteristics. The program then recommends a weld procedure and weld parameters. Diagnostic expert systems obtain information concerning welding process variables and list

conclusions concerning probable causes of defects. On-line systems analyze pre-weld conditions based on joint design, weld process selection, material selection and others while weld phase analysis uses sensors to monitor the process, determining whether process conditions need to be modified and changes welding procedures accordingly. These systems can at best be as good as the engineer whose knowledge is captured. It is based on basic concepts in expert systems and many of these can be classified as small-to-medium sized expert systems. Expert systems are useful as a starting point for automation but good acquisition and representation techniques are the key to simulation of human reasoning. Frequently the reasoning process may not justify an expert system.

Tonkay and Knott [1989] developed a systematic methodology to select the welding parameters for the GMAW process. The method was incorporated in an expert system. The information was gathered from a series of experts in the field of arc welding and combined into a single model using statistical techniques. Several implementation criteria were examined to determine the structure of the expert system. The system was validated by comparing specimens welded by using the expert welding engineer specification and those welded by using the expert system. Encouraging results were reported. The strength of the paper lies in providing a systematic and detailed methodology of knowledge acquisition procedures and not in the implementation methods. To date no work in developing arc welding expert systems has detailed the means of coordinating knowledge from multiple sources into a single model and this topic deserves attention. However, the results of the analysis do not indicate a clear need for an expert system. Many of the values needed for arc welding are based on certain straightforward considerations and hence the justification for an expert

Table 2.2: Summary of KBES application research in welding

COMMERCIAL AND PROTOTYPE SYSTEMS

NAME	SOURCE
PROCEDURE GENERATION	
Weld Process Schedule Expert System	Hathaway and Finn [1986]
Weld Assist	Kuhne, Cary, Prinz [1987]
Steam Pipe Expert System	Alberry, Taylor, Yapp[1987]
Weld Procedure Selection Program	Ribeiro, Turner, and Farrar [1987]
Weldex	Dorn and Majumder[1988]
SAW Expert System	Taylor [1989]
Weld Scheduler Expert System	Larson [1985]
Expert Robot Welding System	Sicard and Levine [1988]
DIAGNOSTICS	
Weld Defect Diagnostic Expert System	Hathaway and Finn [1986]
Miller Expert Program	Miller Electric Company
CAMTECH 1000 and ADAPTITECH 1000	Kerth and Kerth [1984]
NEWCS	Reeves et al. [1988]
COST ANALYSIS	
Welding Estimating Expert System	Hathaway and Finn [1986]
Weld Costing System	James and Baker [1987]

system needs further thought.

From the research described it may be concluded that:

1. Welding process knowledge is claimed to be complicated and not to fit traditional algorithmic approaches required for the development of process schedules.

2. With one exception, no published research details any standard methodology of knowledge acquisition and representation for arc welding.

3. Expert systems have been created in specific domains of arc welding to provide schedules, without any direct coupling to the CAD system that designed the weld.

It is therefore desirable to develop a systematic understanding of the process of specifying welding schedules and to check whether there are any significant differences between robotic arc welding process specifications and manual arc welding process specifications. Further the need for such an approach is also to be critically examined.

Integrated Robot And Positioner Systems

Off-line programming of industrial robots

The premise on which off-line programming systems have evolved is that they are easy to use and that workcells can be created with minimum training [Chan et al. 1988]. However, very little information is available comparing conventional robot programming languages and off-line programming systems. A number of simulators are commercially available including GRASP, McDonnel Douglas Robot Software, RoboTeach, ROBCAD, and AutoPASS [Yong et al. 1985]. These provide sets of

modeling and simulation tools to model a robot and its associated equipment, to simulate the manufacturing task and to postprocess the sequence of motions to robot instructions. The methods adopted in the motion specification are very similar to those involved in teach pendant programming – the user specifies the desired positions of the tool center point of the robot along the path – and joint angle information is calculated. Subsequently a time-based simulator is executed to compute the movement and also to check for collisions. These systems are essentially kinematic in nature and limitations include: (1) calibration and feedback data from sensory inputs cannot be used (2) integration to already existing CAD models and process planning software is not possible (3) it is difficult to apply these systems as the workcell becomes complicated.

Trajectory planning of integrated robots and positioners

Robot programming for arc welding is primarily attempted by on-line lead-through teaching methods. Motion sequences and process parameters are specified and stored using a programming unit to be played back later. The welding torch in arc welding robots is typically programmed to move along a pre-defined path with a controlled orientation dictated by the welding process. This movement is normally defined in a frame that is fixed relative to the robot wrist. The demands set forth by the process often require additional movements in the orientation and position of the weld torch relative to the weld joint. This is accomplished by using a positioner in conjunction with the robot. Figure 2.3 provides the schematic of a typical integrated robot and positioner system used in industry. The advantages of having a positioner in a robotic workcell include: (1) the work space can be extended, (2)

proper orientations between the weld joint and the torch can be obtained without any interferences.

Trajectory planning is the task of designing a path to move the manipulator from an initial position to some desired final position as determined by the weld seam in the part that needs to be welded. This motion involves a change in orientation and position of the tool relative to the positioner station. Normally, the motion is specified by assigning a sequence of points, termed the knot points, between the initial and final points. Each of these knot points is a frame which specifies both the position and the orientation of the tool relative to the station. Between these points the motion of the manipulator is defined to be a smooth function. To guarantee this, constraints on the spatial and temporal qualities of the path are specified between the knot points. Joint space schemes achieve the desired position and orientation at the knot points. Between the knot points the shape of the path is simple in joint space but complex when described in Cartesian paths [Craig 1986].

Traditionally, the problem of path tracking and kinematics for a robot is converted from the Cartesian space to the joint space for ease of calculation. Taylor [1979], Goldenberg and Lawrence [1986], Hornick and Ravani [1986], Wang [1988], and Chand and Doty [1985] obtained the inverse kinematic solutions for a finite number of knot points on the trajectory and interpolated between knot points in the joint space to achieve the desired trajectory. Iterative schemes have been developed to evaluate the kinematic solutions simultaneously along trajectories by using Predictor-Corrector, and Modified Jacobian-Based Newton Raphson numerical methods [Gupta and Kazerounian 1985, Singh and Gupta 1989]. Angeles [1986] used a continuation method to obtain the inverse kinematics at the start point of the path

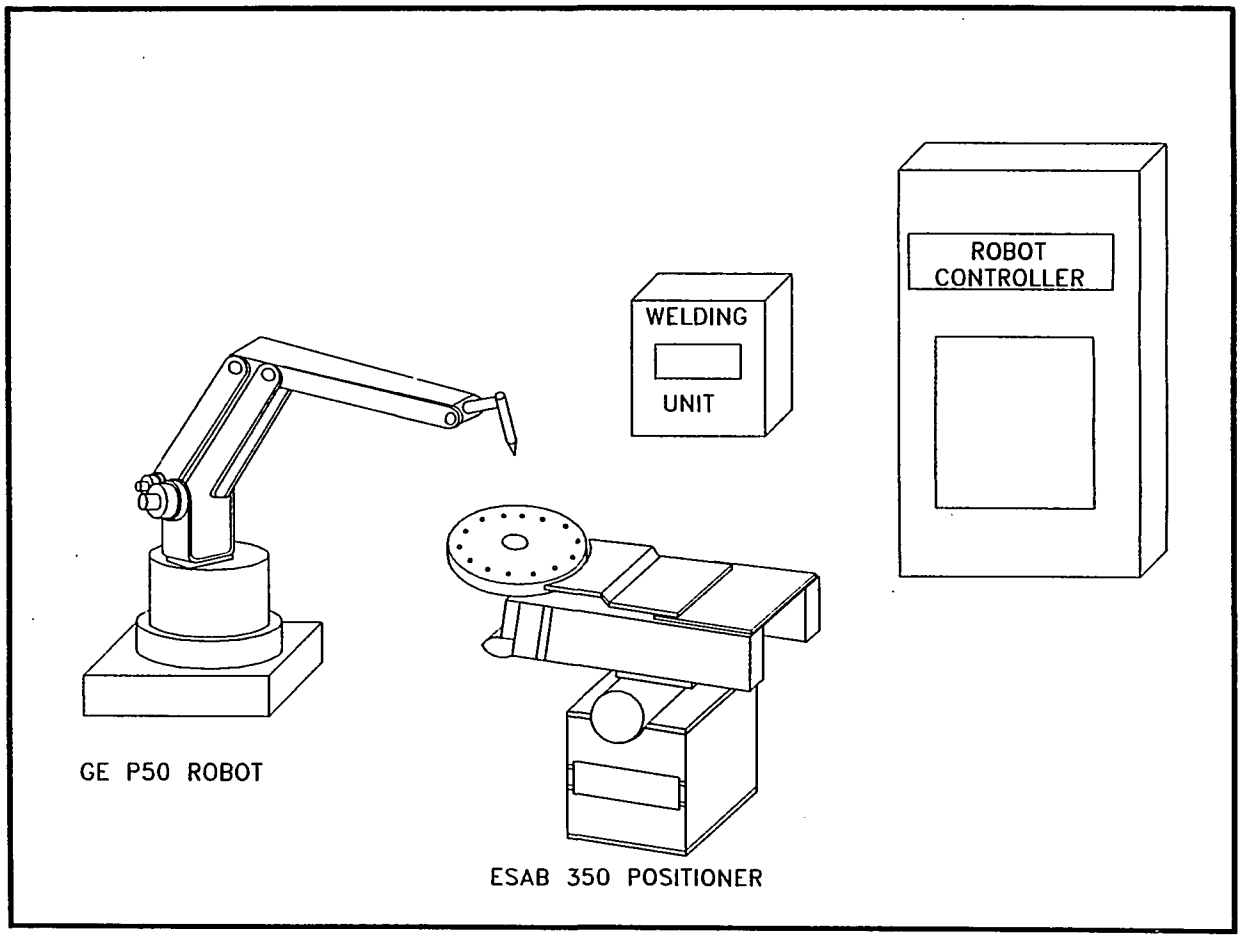


Figure 2.3: Schematic of an integrated robot and positioner

and then used a least squares approximation to determine the joint angles and velocity histories along the trajectory. Implementing such numerical techniques for on-line systems is difficult due to the computation time involved. To save time inverse kinematic solutions can only be obtained at specific intermediate points to save time. In typical manipulator systems, the path update rate lies between 20 and 200 Hz. A method that would plan the trajectory and provide inverse kinematic solutions along the trajectory would be very useful.

In an attempt to solve this problem, a continuation method approach was proposed by Subbian et al. [1991a] to plan continuously the trajectory of a robot by reducing the path generating problem to that of solving a system of first order ordinary differential equations for the joint angles with respect to a path variable. This was the first work which provided position and velocity information for all available points in a path. In establishing a continuation formulation for the problem the path parameter rather than time was used as the independent variable. This method was applied to solve the path planning of a circular trajectory for a 3-revolute and a six degree of freedom PUMA robot. The continuation method was adopted in the present work to solve our trajectory planning problem. The method is explained in Chapter 6.

All the above mentioned methods are used primarily to solve the problem of path planning for robots alone. However, as shown in Figure 2.3, a robot along with a two or three axis positioner is typical of arc welding workcells. Trajectory planning of the integrated system becomes quite involved. In current commercial robotic arc welding systems, the robot controller is used to control and move both the robot and the positioner during welding. The common method of planning a path for a

weld is to orient the positioner first to provide the best possible welding conditions and then teach the robot to reach that position and orientation. However, a large number of points need to be taught to obtain a coordinated motion of the robot and the positioner. This technique results in increase in the programming time at the robot station. Also, in an off-line programming system, this method results in defining local frames so that the object can be defined at all locations of the path. A planning system that could automatically synthesize the coordinated motions for a robot and positioner would be quite valuable [Craig 1987].

Very few research efforts have concentrated on reducing the programming time by means of an integrated kinematic description of the robot and positioner for a welding system [Bolmsjo 1989, Buchal et al. 1989 Craig 1987]. Yet another work [Thompson et al. 1988] deals with the trajectory planning of a robot and positioner problem without considering them as an integrated unit. All previous efforts obtain kinematic solutions only at intermediate knot points and in between knot points the shape of the path is interpolated in joint space as usually attempted in commercial systems. However the method of obtaining a coordinated motion of the robot and positioner using an integrated kinematic description in these efforts is described in detail to highlight the concept.

Thompson et al. [1988] present a prototype development of a hierarchically structured knowledge-based system for coordinated control of a welding robot and a positioning table. The system developed is modular and divided into four major areas: (1) weld planning (2) movement planning (3) robot control and (4) positioner control. The movement planner which is of interest in the present context, makes use of a combination of rules and algorithms to obtain the feasible orientations for the

positioning table. Certain process-related constraints are used to select the appropriate path for the robot. The kinematics and trajectory are calculated for a desk top robot (microbot) and a generic two axis positioner. Although these authors claim the system to be coordinated, problems considered are in a single plane and continuous motion of the robot and positioner is not attempted. The positioner is first oriented to pre-specified angles provided by the user in order to obtain a down-hand position of the weld gun and the robot joint orientations are then calculated. The technique follows the traditional method of providing solutions at intermediate points followed by linear interpolation in joint space between knot points.

Bolmsjo [1989] arrived at the forward kinematics of the robot and positioner in three steps: (1) obtain the homogeneous transformation from the base to the ith frame of the positioner, (2) obtain the homogeneous transformation from the robot base to the ith frame of the positioner, and (3) obtain the homogeneous transformation from the weld seam to the torch tip. The synchronization of the movements between the robot and the positioner is carried out by using the positioner as a leader and the robot as a follower.

In this work [Bolmsjo 1989], the ESAB 500 2-axis positioner is used to demonstrate the procedure for the inverse kinematic solution. The postioner does not have a closed form solution due to its mechanical structure. Also, in many cases in welding it is not possible to obtain the desired solution or the optimal orientation of the joint due to the physical limitations of the robot and the process specifications. This results in having allowable tolerances for a solution, and an optimization procedure is used to search for the most acceptable solution in the joint space. An approximate solution to the orientation of the given joint is obtained in the horizontal plane such

that a weld gun oriented normal to the joint is in a vertical line. An initial guess was assumed and using it as a start value and adopting a binary search in two dimensions yields the closest possible orientation for the positioner is found. Having found the orientation for the positioner, the robotic orientation for the corresponding position is searched using the closed form kinematic solution of the robot. The author alludes in the final part of his paper to the importance of having a CAD database to provide the geometric input and process details and predicts that a significant contribution could be made to off-line programming of welding robots if the information were complete and available.

Buchal et al. [1989] obtained the kinematics history of the robot and positioner for a PUMA 560 robot and a generic three-axis positioner. The forward kinematics was obtained as a product of intermediate transformations:

$$H_{RG} = H_{RT} H_{TB} H_{BC} H_{CG}$$

where H_{RG} corresponds to the homogeneous transformation from the Frame F_R to the Frame F_G. The homogeneous coordinate frames defined were:

F_R = The robot base frame

F_T = The positioning table frame

F_B = The workpiece frame

F_C = The weld seam frame

F_G = The tool frame

Once the intermediate transformations were determined, the location and orientation of the tool relative to the robot base were calculated. Details regarding the inverse kinematics were not provided excepting that a closed form solution based on the matrix method of Paul [1981] was used. A simple strategy was adopted to

find the interference free trajectory of the robot required to follow the weld path. The kinematically feasible solution was found for each point on the seam. When an obstacle was found, the torch was rotated in increasing arcs in both directions until an alternate path was found. Since the objective of this research was interference detection, the traditional method of finding inverse kinematic solutions at knot points was used and in between the points the shape of the path was interpolated in joint space. However, none of the process conditions was taken into account to orient the workpiece or the welding torch. The research was more a system to demonstrate the application of interference detection than to continuously plan the trajectory for weld seams.

From prior research in off-line programming and trajectory planning, the following may be concluded:

1. A robot and positioner are normally used together in robotic arc welding workcells. So far only limited research has been attempted to provide an integrated kinematic description of a robot and positioner.

2. Trajectory planning research is normally restricted to robots alone. Inverse kinematic solutions are obtained only at specific knot points and interpolated between the points. Continuous trajectory planning of weld seams for an integrated robot and positioner system has not been attempted.

Based on these observations and work (to be described later) in providing a complete definition of weld paths in a CAD system, Chapter 6 proposes a novel approach using continuation methods to solve the trajectory planning problem of an integrated GE P50 process robot and a two-axis positioner.

Need For The Research

Although several parts of the problem of automatic programming of arc welding robots have been attempted, there is no single environment that has attempted to provide a solution in an integrated manner. At the beginning of this research, the need for such a computer-aided engineering environment was identified. During the development of such a system based on a CAD representation, a more fundamental understanding of the process of programming a robot was obtained and this understanding prompted a reexamination of the concepts currently in use. Overall the present research has emphasized the need for developing robot programming systems on the basis of modeling systems with application systems being structured to suit the overall representation structure. The problem to be solved in this dissertation is defined as:

- **"Given a complete description of a part to be welded, design appropriate techniques and representations to incorporate, retrieve, and reason geometry, and process information which will automatically yield an acceptable strategy and generate code in a language understandable to the welding robot"**

The previous Sections have clearly identified the need for the present work and solutions are provided in the following chapters.

CHAPTER 3. REPRESENTATION OF WELD FEATURES AND ATTRIBUTES IN SOLID MODELERS BASED ON CSG

Systems for planning automatically the welding of mechanical parts by robots require computer representation of the geometry of the parts, the robot workcell environment and appropriate welding process information. Tolerances, surface finish information and other variations from the nominal geometry of the parts are collectively called variational data [Requicha and Chan 1986]. By this research, information such as weld attributes, heat treatment data and others are also classified as variational or secondary feature information. This data reflects an intended function in a mechanical part and should be specified by designers and are normally subjected to reasoning by human visualization as fully automatic systems do not exist to synthesize the ideal geometry information and variational information. Although existing solid modelers provide unambiguous means for representing nominal geometry, they have no means to support the representation of weld feature information for subsequent down stream automation.

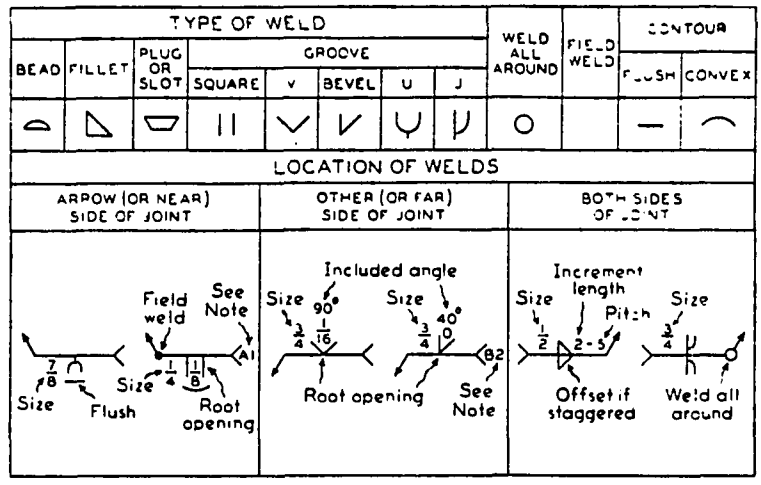
This chapter explains the effort to provide solid modelers with such information to help in the automatic programming of arc welding robots. The basic structure for representing weld feature information in a solid modeler is called the welding attribute graph (WAGRAPH) and the mechanism of incorporating this structure in

a solid modeler is discussed. The first Section provides existing representation of welds in drawings and lists the broad functional requirements of the system that will support weld features. The overall methodology of the weld feature information provision mechanism is explained in the next Section. The third Section briefly explains the hybrid solid modeling system to explain the complexity involved in the implementation. The representational issues involved are explained in Section 4 while the final Section summarizes this chapter. The implementation of the integrated CAD structure along with the welding knowledge mapping is explained in Chapter 5.

Theory of representation of welds

Engineering drawings have traditionally been used to document design and to communicate between design and other disciplines of manufacturing. In order to ensure that welded joints are interpreted unambiguously a standard methodology has been set-up to represent any weld, its type and properties [Welding Handbook 1984]. This Section will briefly introduce the required terminologies and for further details, the Welding Handbook [1984] can be referred to. The existing representation structure is shown in Figure 3.1. Figure 3.2 provides an example of a sample drawing that has a welded feature and its corresponding representation. There are two aspects to weld feature representation, one being for display purposes and the other for providing an ability to reason the geometry.

From a display standpoint, the standard provides the means of placing complete welding information on drawings. In this system, the joint is the basis for reference. The tail of the symbol is used for designating weld specifications, procedures or other supplementary information to be used in making of the weld. The notation



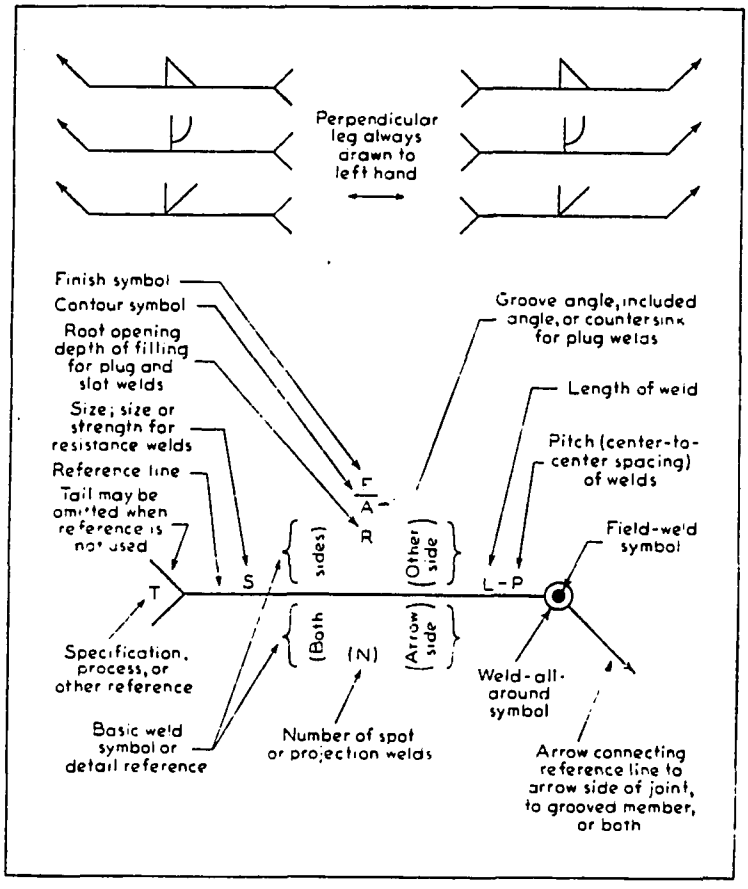


Figure 3.1: Existing representation structure for welds

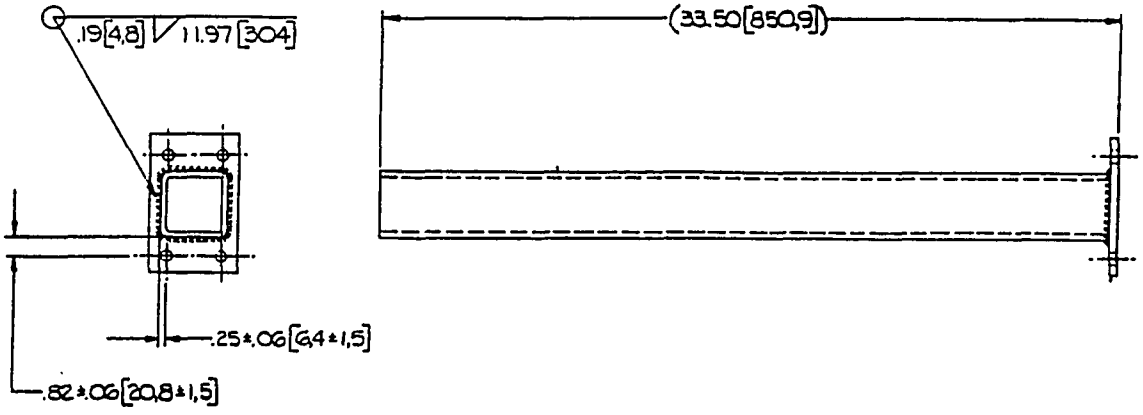


Figure 3.2: Sample drawing with weld feature

placed in the tail of the symbol indicates the process, the identification of the filler material that is to be used, whether peening or root chipping is required and others requirements to be established are left to the user. The assembled welding symbol consist of eight elements that convey the weld to a human eye. They include: (1) reference line (2) arrow (3) basic weld symbol (4) dimensions and other data (5) supplementary symbols (6) finish symbols (7) tail and (8) specifications, process or other references. However, this information must be provided as attributes of the corresponding feature in order for it to be used subsequently for welding procedure generation and reasoning.

The basic geometric entities are faces and edges and attributes can be added to these features. From the perspective of welding, the weld features happen to be those that lie on an object's boundary which typically is linear, or curvilinear in shape. The welding feature is the intersection of two surfaces that are created

by a modeling system. Representation schemes should therefore provide means for accommodating both simple and curved aggregate features. To be acquainted with the types of geometry that are encountered in welding, joint types and allowable welds need to be studied. The basic joints for arc welding include: (1) Butt (2) Corner (3) Tee (4) Lap and (5) Edge. The applicable welds and the corresponding joints are shown in Figure 3.3. The information that is relevant for a robot application is the location and orientation information that will be used by the kinematic module. A three plane coordinate system needs to be constructed with topological entities of the welding feature.

A need is seen for representing information (that was hitherto left to human visualization) via a systematic representation scheme to obtain geometry and datum surface information. To incorporate these, certain broad functional requirements can be listed:

1. **Capture Welding Semantics:** The methods employed to develop the datastructure must do more than just store welding data; the meaning of the weld must be formulated in a data structure that is essential for process parameter reasoning, orientation of gripper reasoning, sequence planning and others

2. **Support Standards:** The description rationale should be able to support all classes of standards as prescribed by the American Welding Society

3. **Adaptable:** The methodology should allow individual users to specify their own references and specifications

4. **Network With Geometric Entities:** It should be possible to interact with geometric entities that would allow for proper interpretation of geometry and

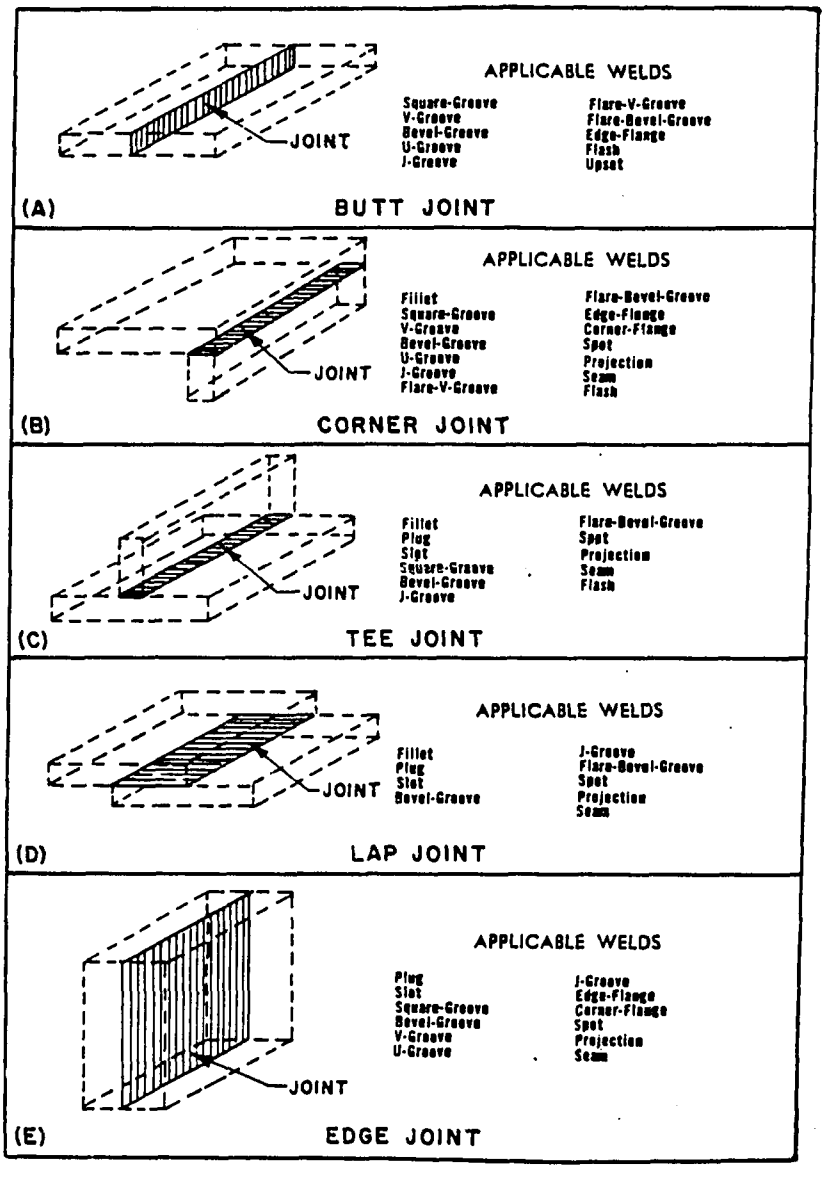


Figure 3.3: Types of weld joints and applicable welds

orientation

5. **Validity Checking of Welds With Geometric Entities:** The methodology should be able to interrogate and determine the validity of the weld with respect to the feature. No ambiguity should be allowed

6. **Legality Checking And Default Setting:** Rules of the Welding society standard should not be violated and should be explained to the user. Also if default settings need to be provided by the user, provision should be made

7. **Display weld Symbols:** The system should be able to display all weld symbols graphically as the current standards do and should be an exact replica of current engineering practice

Overall Methodology

The problem that is to be solved as part of the overall research is to identify weld features in a geometric model created by a solid modeling system and to attach weld property information to the identified node. Pure identification of the weld features from a CSG model is not directly available as it is not supported in a modeling system to start with. Further the non-uniqueness of a CSG model compounds the problem. A designer unless limited by a specially confined user interface can arbitrarily create two or more CSG trees to represent an identical object. Consider Figure 3.4 for example: It is a simple Tee joint with a fillet weld all across the length. It can be created in different ways. In the first case the weld is just the union of two blocks while in the second case it is the creation of two blocks A and B and then the regularized subtraction of C.

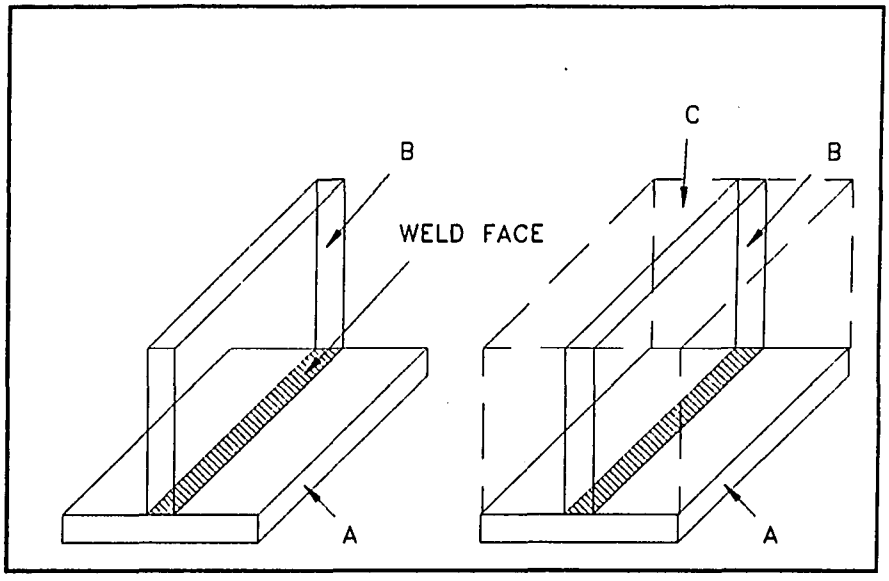


Figure 3.4: A simple Tee joint

To identify the weld automatically would be difficult as interacting volumes might occur in widely separated regions of the tree and a weld may have been created by an union or an intersection or a difference operation. If these weld features are identified then means can be devised to add attribute information and define coordinate systems. Therefore the solution to the problem should adopt one of the following means:

1. Explicitly define the welding feature when objects are created and attach property information or

2. From any given CSG tree, devise algorithms to isolate all the welds, unify the features and attach property information to those nodes as appropriate

The second approach is very difficult if not impossible and may not always guarantee correct results. Instead a more practical and viable approach would be

option 1. This general approach has been adopted and the following Sections will explain the details. As explained in Chapter 2, welding attributes have a lot in common with tolerance attributes, and since welding features have never before been represented in solid modeling systems, ideas have been drawn from the representation of tolerance attributes in solid modelers. The representation structure is applicable for any solid modeling system that uses CSG as the prime representation structure with other forms like B-Rep being provided as evaluated structures. IDEAS is used as the example solid modeling system.

In order to arrive at a scheme for providing information the type of information that is likely to be extracted at a later stage needs to be known. This will help in identifying types of parameters that are to be used. For a robot to weld a part, three types of information are needed which are classified as (1) know what (2) know why and (3) know how. These may be defined as follows:

- KNOW WHAT specifies the important geometric and feature information that needs to be contained in a computerized product model such that it can be used by the robot to know what is to be welded

- KNOW WHY models the process knowledge such that welding procedure solutions will be provided for the given problem and also a rationale behind the choice. This captures the knowledge that is traditionally applied by engineers.

- KNOW HOW specifies the means by which the robot can get to the weld and this know-how is obtained by modeling the kinematics and planning the trajectory of the welding robot and positioner. This is akin to the conventional algorithmic approach wherein numerical routines are written to perform a task.

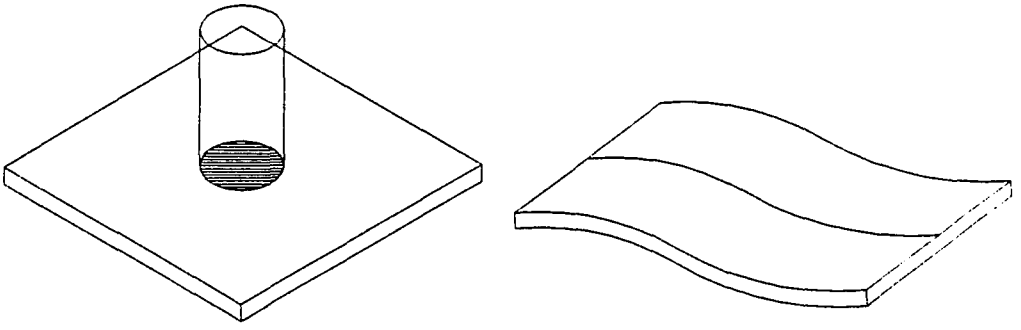


Figure 3.5: Sample welds showing complexity

Figure 3.5 gives a sample of the two classes of objects that can be encountered in arc welding and the need for the provision of the corresponding representation. The first case is a simple Tee joint and the interaction is between a block and a cylinder. The weld can be either fully circumferential or it can be a part of the circular trajectory. The other example is a weld between two free form surfaces (butt weld). The intersection of the two surfaces give the curve that needs to be welded and the feature information needs to be attached to the corresponding parent node.

The specific information that will be used by the application modules viz: knowledge processing and trajectory planning module can be identified as:

Type 1: Explicit feature information

Type 2: Implicit feature and attribute information

Type 3: Networked feature information

- TYPE 1 information: That which is available explicitly from the geometry database and which is that information which is independent of other geometric entities and can be obtained directly from answers to simple questions such as: "What are the two features in the weld, what is the joint type, what is the thickness of the weld?"

- TYPE 2 information: That which is not available explicitly in the feature definition but can be obtained either by computation or by accessing the additional structure of the feature for such information such as the weld size, the throat depth, the weld finish information.

- TYPE 3 information: That which is obtained in connection with more complex structures or with a combination of two or more features. The relationships provide typically the coordinate system information, or the reference work frame information. These are dependent on the type of model created.

The main research initiative here is therefore to design and implement a welding attribute graph (WAGRAPH) that can be associated with a CSG tree. The overall methodology is to provide a system that will help users to add the type of welding attribute information explained above to the leaves and nodes of a CSG tree and be subsequently used for the generation of welding process parameters and robot angles such that automatic programming of arc welding robots can be attempted. The following are the steps in providing such a representation structure:

1. Create the geometric primitives using a CSG-based modeling system

2. As primitives are created, identify the faces on which weld features occur. Check for consistency and attach weld feature information on those faces of the primitives. Provide a numbering and naming sequence for the faces of the primitives so that information retrieval is possible

3. Obtain the regularized intersection of the two faces to find the participating curves of the weld

4. If there are multiple weld edges participating in the same weld face, obtain the set union and arrange the attribute information in an order. Attach the weld attribute information (weld joint and type details) to the parent node of the two participating solids. Obtain also the measured entity information which defines the geometry

5. Define the work reference frame and the weld orientation at the start of the weld using pointers to the appropriate faces and edges

6. Devise schemes for easy access of the tree and retrieval of weld information. As the tree grows, provide a mechanism to acquire and pass information

7. Check for overall validity and consistency of the WAGRAPH

8. Implement a system and test sample drawings

Welding Attribute Graph (WAGRAPH)

The information asssociated with a welding graph can be categorized as primitive feature information, surface information, coordinate system information and edge information. In order to represent this a very simple mechanism has been devised

to incorporate the welding attributes in a CSG model. Although the structure is simple and straightforward it becomes complex due to implementation details. Figure 3.6 gives the semantic structure of the WAGRAPH. A pyramid like structure is provided with the actual entity of interest – the weld edge at the top of the pyramid structure and the general primitives where this topological entity belong (like a block) at the bottom of the pyramid. Such a structure could be uniformly applied to all secondary feature information such as heat treatment data, surface finish information and others. The main entities of this pyramid structure from bottom-up is given by the following:

Generic Primitives (GP)	:	User defined primitives such as a square tube
Honest Primitives (HP)	:	As provided by the system such as cylinder, block
Primitive Face (PFACE)	:	The nominal faces of each solid
Weld Faces (WFACE)	:	User defined faces of an object where the weld occurs
Union of Weld Face (UWFACE)	:	Groups of weld faces
Weld Edge (WEDGE)	:	The participating edge of the weld face for all types of weld other than butt weld
Union of Weld Edge (UWEDGE)	:	Groups of participating weld edges
Reference System (WFRAME)	:	To denote reference coordinate systems and weld coordinate systems (To be used by the kinematics module)

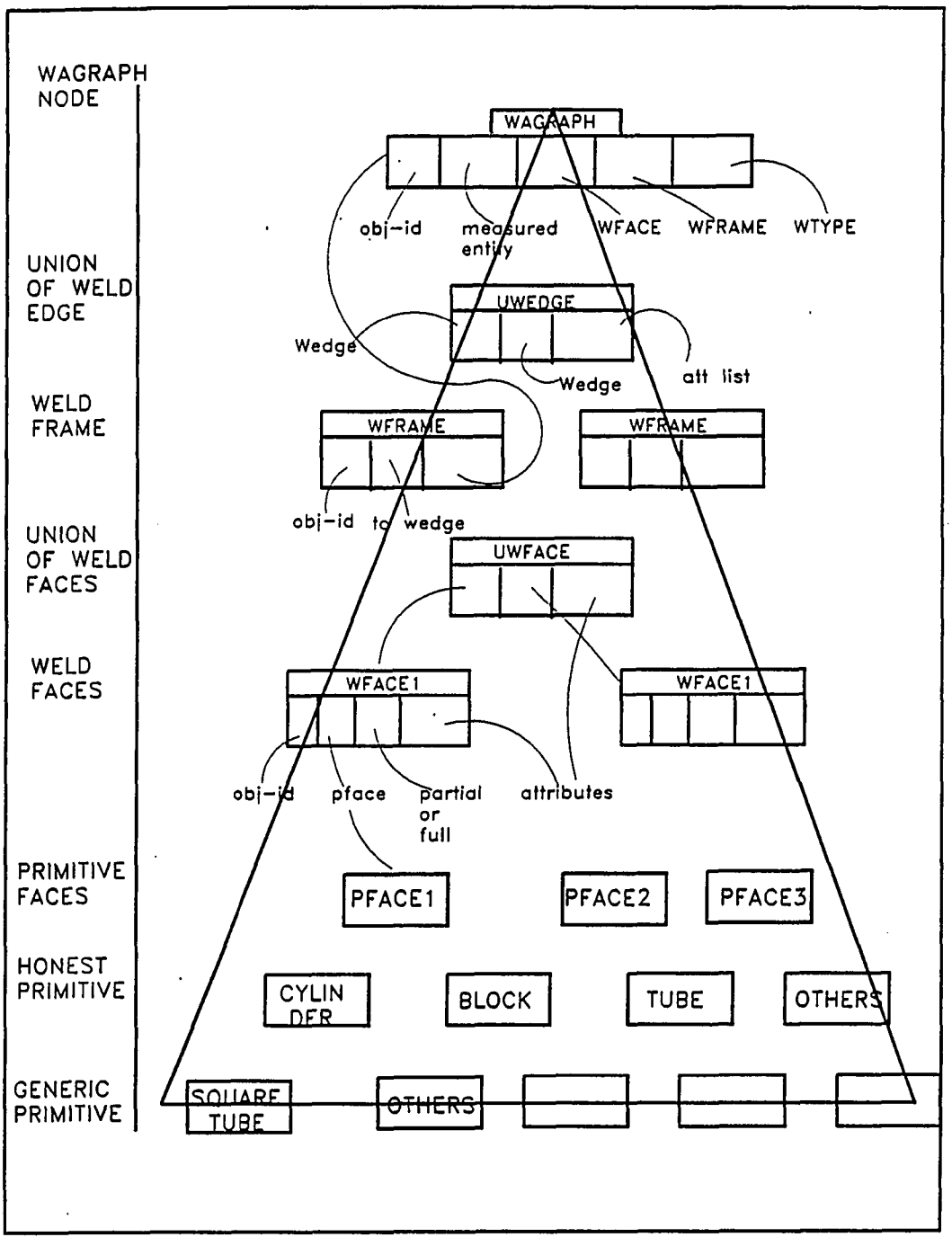


Figure 3.6: Semantic structure of the WAGRAPH

Two important questions involved in attaching welding attribute information are those involving the selection of the two interacting faces and the participating edges. The overall strategy is to attach the face information as pointers to the two participating primitives or solids or sub-solids (the leaves) and the edge information (geometric measured entities) along with weld attributes to the parent node of the two interacting solids. By this, the welds and their corresponding information in a CSG tree can be identified. This is explained by an example as in Figure 3.7. In this example, the two participating primitives (it could as well have been generics or sub-solids) are A and B. If there exists a weld in this tree, then the following is true: (1) it should be on one of the faces of A and (2) the other face that interacts with the face on A should be on B. The regularized intersection of these two faces will provide information about the weld face. This face then has participating edges that could have welds (exception – butt welds) and attributes needed for the weld edge can be associated with the parent node – C which is the union of A and B in this case.

The bottom level on the pyramid structure is the generic primitive (GP) – those combinations of honest primitives. A specific case that is commonly used in welding is a square tube (see Figure 3.2). This can be created by the difference (cut) of two blocks and can be specified by the length, outside square dimension and the thickness of the tube. In today's modelers this is provided as a 'Features' provision (IDEAS support this kind of primitives). The next entity is the honest primitive (HP). These are the primitives as provided by any solid modeling system such as block, cylinder, cone, torus and others. These together with 2-d profiles form the building block of any solid geometry modeling system.

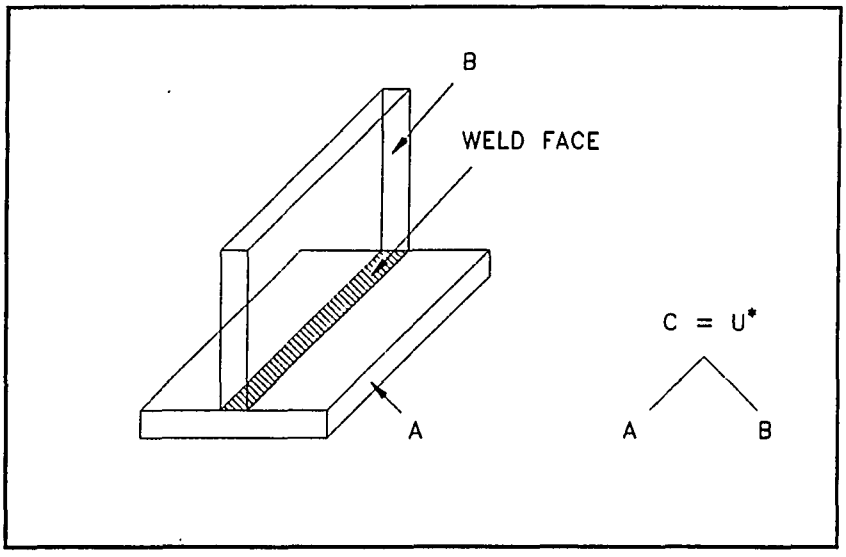


Figure 3.7: Attribute attachment in a small CSG tree

Next are the primitive face (PFACE) nodes corresponding to the nominal faces of an object. The weld can be in any one of these faces of the regular object. The particular face and the path in the overall CSG tree need to be identified for retrieval. Weld face (WFACE) nodes are the next in the hierarchy of the structure. Such a node represents the two faces that interact to produce the weld edge. This invariably will be a portion of one of the PFACES of a solid (or primitive) and a whole face of the other interacting solid. In some cases it is the interaction of two whole PFACES (as in butt weld). So a WFACE points to 4 entities: (1) an object-id of a primitive or sub-solid (2) a weld face (3) an operator to denote if it is a partial face or the whole face and (4) a node to denote the type of the weld which determines if the attribute list in node 4 needs to be null or not. The meaning of this structure can be better explained by a simple example. Consider again Figure 3.7. The face that needs to be welded is hatched. The participating WFACE is a partial PFACE on primitive A

and a whole PFACE in primitive B. So WFACE1 will have a pointer to object-id C1 – a block; to PFACE5 of primitive C1, it is necessary to declare it to be a partial face as far as that face is concerned, and the type of weld is determined from the user to be Tee joint. Since it is a Tee joint, node 4 of this structure is set to null. The other participating WFACE2 will have its nodes pointing to PFACE6 of solid B, full, Tee joint, and null. In general, the portion of the PFACE on any primitive that interacts with another whole PFACE results in a weld face.

UWFACE nodes are combinations of WFACE nodes. This node takes care of the possibility of having two welding faces on the same PFACE. The structure has three nodes: the first pointing to one of the WFACE, the second pointing to another WFACE, and the third to an attribute list.

The next node (WFRAME) conveys the robot tool orientation. This essentially represents the direction vector of three adjacent edges or the definition of one edge and a face (whose surface normal results in the Z direction vector). Two types of coordinate systems, one for the work reference frame and the other for the weld start frame are typically needed for the robot to calculate the inverse kinematics. The reference system is represented by pointing to either WFACES or WEDGES. The WFRAME structure when formed with edges has three elements as nodes. The first points to the compound object-id, the second node points to the first edge and the third node points to the other adjacent edge. The third direction vector is obtained by the cross product of the two. For the weld frame, the X direction is chosen along the direction of the weld, the Z direction being the surface normal of the participating PFACE and the Y direction completing the right handed convention by obtaining the cross product of Z and X. If the WFRAME structure is formed based on face

information, then the weld edge act as the X direction vector, and the participating face has a pointer to it. Subsequently, the surface normal of this face is calculated to obtain the direction vector in the Z direction. The Y direction vector is obtained by using the right hand convention.

The top node of the pyramid structure contains the edge information and is denoted by the WAGRAPH node. Semantically it denotes the intersection of two WFACES and contains the weld attribute information. The intersection can result in one or more participating edges and one or many of these edges can be a weld. The overall structure has base class information (details provided in Chapter 5) that provides the weld edge information and it points to the WFACE, WFRAME structures and a pointer to WTYPE – the welding attribute node. Other than pointers to other classes, the information stored in the WAGRAPH node are: (1) weld edge type – full or intermittent or none (2) weld curve type – the intersection information like a straight line weld, circular weld or as a NURBS curve (3) measured entity has 4 elements each of which points to a structure that contains the X, Y, Z location of (a) the start coordinate (b) the end coordinate of the weld (c) the direction vector of the X and (d) the direction vector of the Z axis.

The weld type node contains all information about the weld depending on the type of the joint. The structure of the various types of joint types and the weld themselves are explained in Figures 3.8 and 3.9. In the implementation Chapter are outlined the semantics and other pertinent details of the weld.

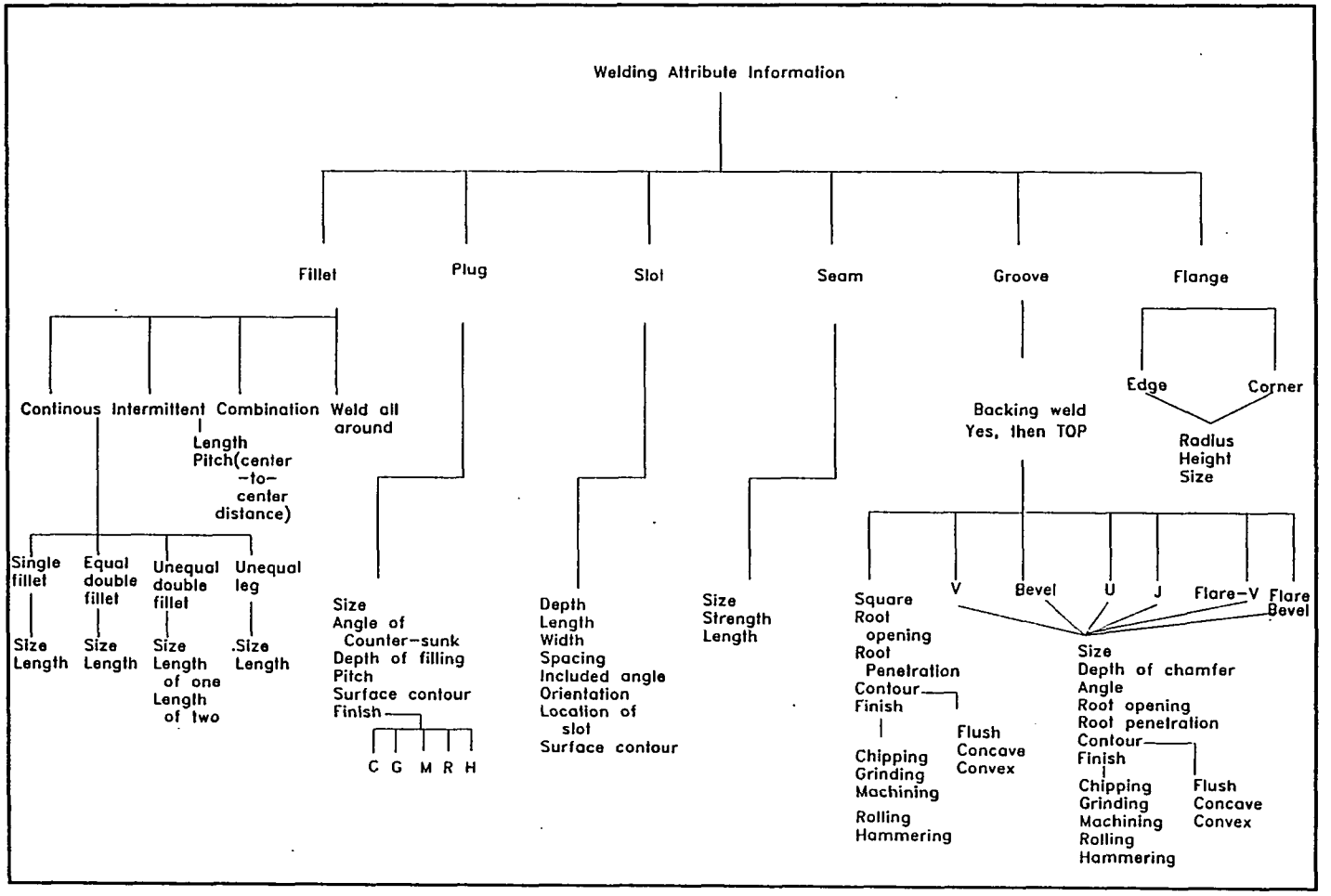


Figure 3.8: Attribute list of various weld types

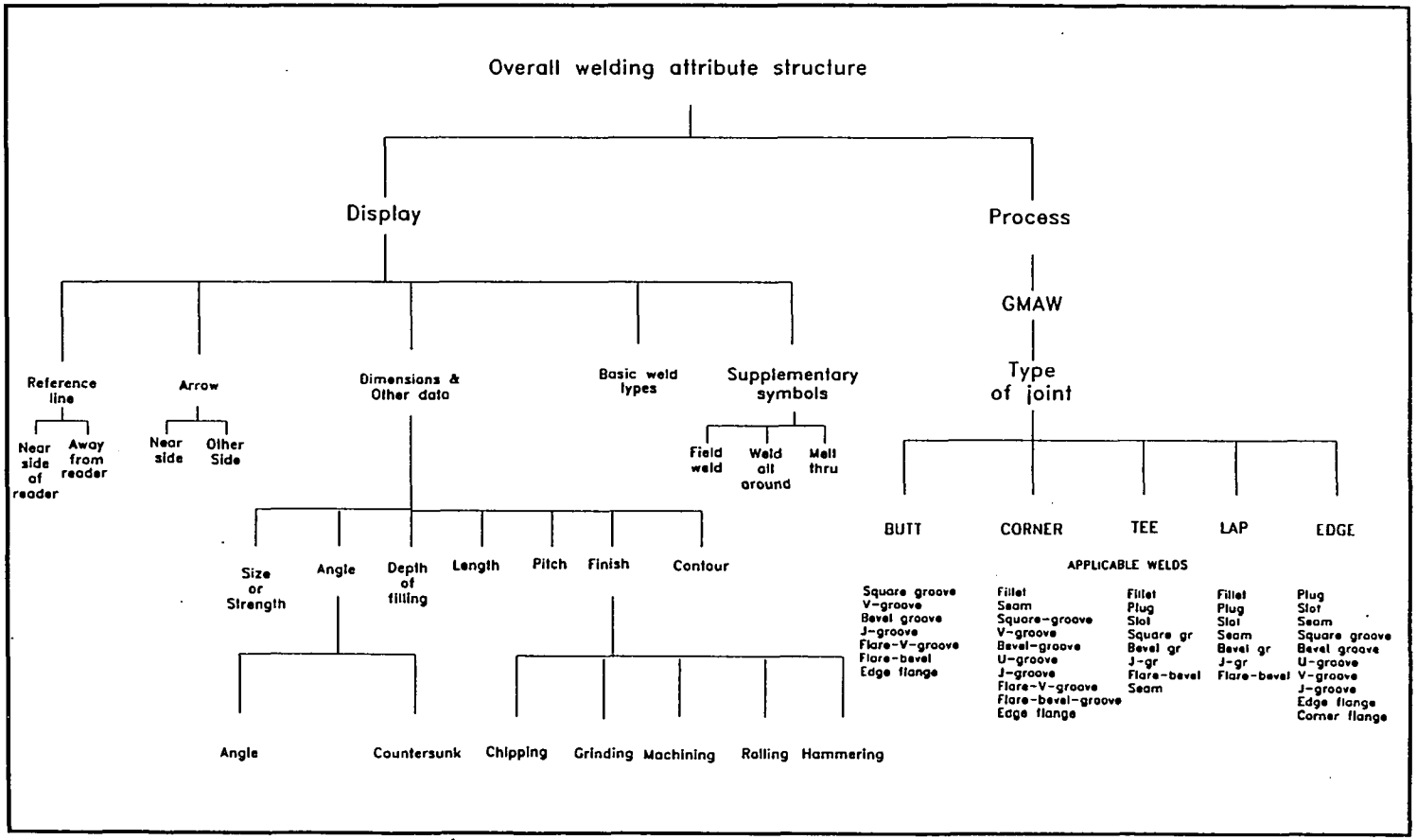


Figure 3.9: Overall welding attribute structure

Hybrid Solid Modeler

To represent the welding attribute information, an hybrid solid modeler (IDEAS) was chosen. The reasons for using such a solid modeler are:

1. It represents that set of multiple representation modelers where the primary representation structure is a constructive solid geometry model and also contains boundary information for archival purposes

2. A B-Rep model by itself would contain all information that is needed and any variational information can be simply attached to the faces and edges [Requicha and Chan 1986]. However, this implies that B-Rep information should be archived along with CSG representation and it is bulky. Further, the information that needs to be incorporated for welding is typically limited to a few faces and is not often needed. Therefore a primarily CSG-based system can allow the user either to use or discard information as and when necessary. The system thereby derives benefit from both modeling capabilities

3. The current trend towards feature-based design essentially is captured in CSG modelers where, at the outset the objects that can be used to create a model are implicit and combinations can result in any type of form feature required. These then act as generic features and for welding, apart from the regular features available, only a few more features (such as square tube) need to be created and this is relatively easy to attempt in a CSG-based system

4. It was the modeling system readily available for research

The solid modeling system IDEAS can be used in three different modes: (1) using primitives to position at particular locations and combining them by boolean operations; (2) using 2D constructors and profiles to create objects and (3) using generic features as created by the user which are combinations of primitives in the overall creation of the object. The collection of any or all of these solids and the corresponding information is called the current context.

The primary representation is a CSG tree. Although the exact implementation internals of the modeler is not known, from the creation and storage of information the storage is assumed to be a binary tree. For the example shown in Figure 3.10, the CSG tree will correspond to that shown by the side of it.

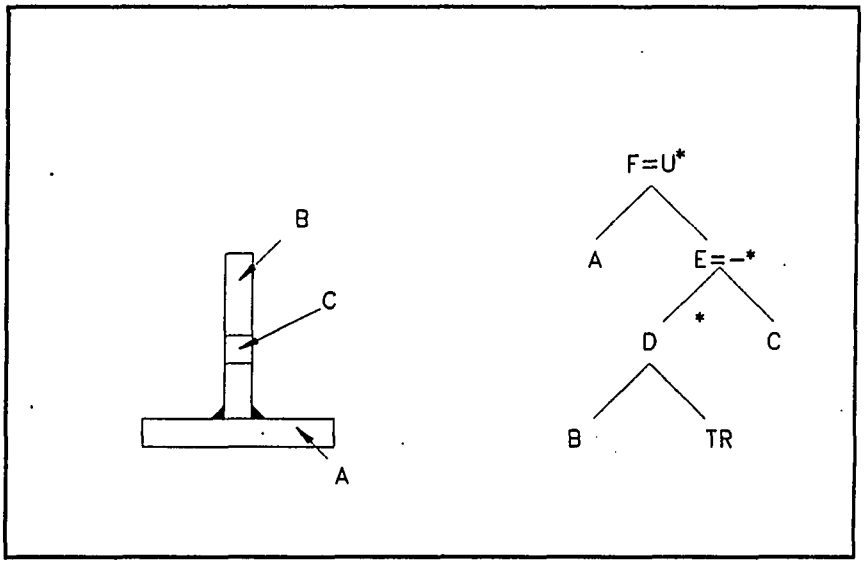


Figure 3.10: Primitives of a Tee weld and its associated graph

The figure explains the three modes of creating the solid. Invoking a primitive as in A; moving the primitive or sub-solid as in B and creating a user defined feature (form feature - hole) as in leaf C. However, in general the objects created

are more complex and this results in difficulties in the association of welding feature information. Typically IDEAS stores the object with the minimum information - the facet approximation, the objects shape and a set of untrimmed Non-Uniformed Rational B-Splines (NURBS) surfaces. However, it can also store information about the object in its database (PEARL) as a complete boundary representation. This forms the topological entities that describe the structure of the object. These include the entities: bodies, faces, loops, edges, and vertices. This briefly explains an environment in which many solids and sub-solids coexist and may potentially have associated welding information.

Representational Issues

The two main representational issues that need to be considered for incorporating weld feature information are: (1) Incremental construction of the WAGRAPH and (2) Validity and consistency of the WAGRAPH.

Incremental construction of the WAGRAPH

A weld feature can occur only when two solids or sub-solids interact. It may be necessary to attach weld feature information to the node of the two participating leaves. A WAGRAPH needs to be constructed only if there is a participating weld in that node and hence incremental construction is highly desirable. As objects are created, WAGRAPHS are added. Consider again the simple example of the two blocks in Figure 3.6. Figure 3.11 shows the construction of the WAGRAPH for this example. The primitives b1 and b2 are initially created as in any modeling system. At this point, the system does not know that these two objects may form a weld. These

two primitives are then combined to produce a compound object cc1. At this stage, the user indicates there exists a weld and the weld is in the combination of primitive b1 and primitive b2. Now the WAGRAPH structure is invoked. Two WFACE structures are defined by pointing to the appropriate object-id, the face numbers and an attribute to indicate whether the participating weld is a whole or partial face. By this method, the two participating faces are known and the intersection provides the weld curve and their corresponding edges. The other elements of the WAGRAPH - the WFRAME, WEDGE are also provided at this point. The user then proceeds to define other primitives and combinations. As the tree grows, the path from the current context (top of the tree) to the leaves are noted. The compound object structure is so arranged that it points to its two child objects. If the child is a primitive, it contains an ordered list of faces and since the exact participating face that contributes to the weld is already pointed to by the WFACE structure, the geometry of the weld can be obtained. If the current solid is a compound object, and the two participating solids are compound object themselves, then the corresponding path in the tree to reach the primitive face is noted for subsequent retrieval.

To obtain the actual weld intersection curve, an intersection of the participating two faces is required. It is known that faces of a new solid are the subset of the faces of each of the combining solids. Consider Figure 3.12. The shape consists of a union of a cylinder with another cylinder, a situation dealing with three-d objects with welds. Let us denote the cylinder 1 as π_1 and tube 2 as π_2. The cylinder is internally defined as the concatenation of three surfaces whose mutual curves of intersection delimit the face regions. The union of these face regions defines the boundary surface of the cylinder and the curves bound the active regions on the surfaces.

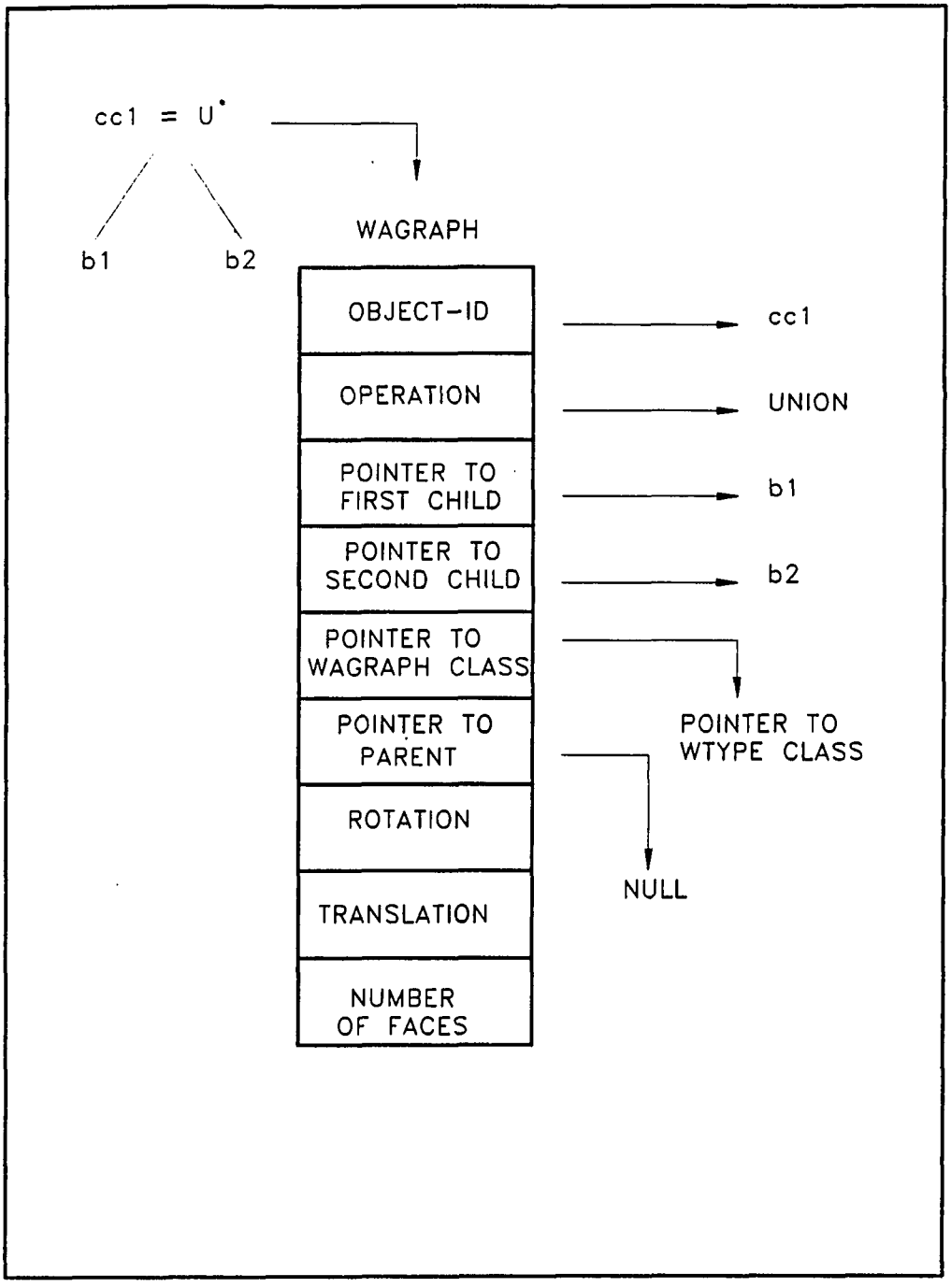


Figure 3.11: Stages in the definition of WAGRAPH

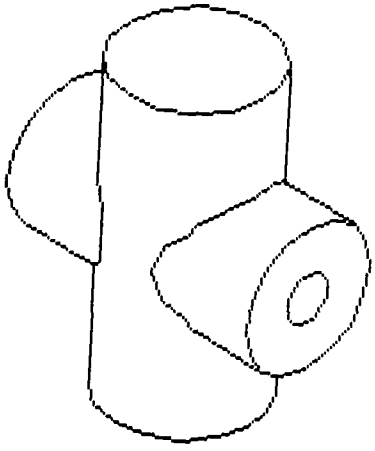


Figure 3.12: Three-dimensional curve information

The intersection of $b\pi_1$ and $b\pi_2$ is a closed space curve, and appears twice in the parametric space representation: one for π_2 and another for the surface on π_2. This spaced curve is a three-dimensional boundary representation and a portion or whole curve can have a weld. The attributes that have been described as the WAGRAPH structure can be associated with this node. Therefore the space curves resulting from the object combinations essentially represent the geometry information of the weld.

3.5.2 Validity and consistency checking

Any WAGRAPH that has been created needs to be a valid graph. Also, the editing of objects should result in the appropriate WAGRAPHS being removed from the overall CSG tree. To ensure the validity of a WAGRAPH representation the

folowing conditions need to be satisfied:

1. Any weld type that is created should be a valid weld for the two geometries involved. Example: A butt weld cannot be created when a compound object is created by the union of a cylinder placed on top of a block.

2. All the WFRAMES created of non-null objects should be valid (in that they must correspond to well defined edges or surfaces to form proper coordinate systems). This condition is not easy to check algorithmically.

3. Whenever objects are edited the corresponding face, edge, and frame information needs to be made invalid. Whenever the geometric meaning of one object that participates in the compound object is lost, then all interactions due to this object in the overall CSG tree need to be made invalid. Similarly provisions for changes to reference frame information should also be provided and their validity checked. All validity checking for weld features should be based on the compound object, since only a combination of two sub-solids or solids can produce a weld. Attaching and detaching a WAGRAPH to a compound object is the key validity check that needs to be attempted.

It should be noted that the actual implementation of these hybrid modelers may not be a binary tree and may be a graph, as in PADL-2 [Brown 1982]. It may be that the same primitive can be used as two instances: one as a geometric instance located at the origin and the other transformed to a different location. The translated instance will have the pointer to the same object with the translational operator alone. This results in certain issues that need to be considered with paths to the corresponding primitives and their faces since the child-parent relationship is

different. The same child can have two different parent nodes and the path name should be unambiguous in those instances. However, for our implementation the structure has been assumed to be a standard binary tree.

Another case that needs to be considered occurs when the same weld type is created and copied to be placed at different locations in the same solid. Consider the example of a block with four identical cylinders placed on one of the faces of the solid. It seems very reasonable for the weld information to be given once and for other cases to be copied automatically. If the WAGRAPH information needs to be copied, most of the structure information will be the same, but the weld coordinate information will need to be changed, and the participating faces for each of the weld will change so that the object combination curve information can be obtained from the intersection of the two appropriate faces. To attempt this automatically would be computationally complicated. To identify all the common weld features and to transfer them to the new instance and make only those changes needed for the new instance may not be the best option. Instead, at the current level of implementation it was decided to allow the user to be prompted with default weld values using the benefit of object-oriented programming and modify the problem for each case and then assign the information to the respective nodes.

Summary

This Chapter provides for the first time a representation structure for welds in geometry models based primarily on CSG with provisions for an evaluated model. The semantics of the structure has been explained and representational issues outlined. The implementation of this structure with the necessary validation along with

the knowledge mapping system and an outstanding application of welding procedure generation will be outlined in Chapter 5. It is shown that secondary feature information such as weld attributes, and surface finish can be added to a CSG tree and that this can be used in applications such as those which are likely to occur in the automatic programming of welding robots.

CHAPTER 4. KNOWLEDGE MAPPING SYSTEM FOR WELDING PROCEDURE GENERATION

The methods for acquiring and representing welding process knowledge is detailed in this chapter. The literature survey related to expert systems in welding suggests that welding procedure generation requires the use of an expert system methodology and that successful results can be obtained. The goal of this portion of the research was to consider seriously the problem once again, question the need for (and determine whether) systematic methodologies of knowledge acquisition may provide knowledge (both public and private) in solving an engineering problem and to develop a mapping system that would directly use the CAD description of weld features described in the previous chapter and generate weld process schedules. Certain surprising and important conclusions are drawn and their impact on the building of the system is outlined. The first Section provides a background review of the welding problem domain. The second Section outlines the overall methodology in developing a mapping system. The systematic procedure of knowledge acquisition from many experts is discussed in the third Section. The results of the survey/questionnaire and the welding knowledge acquired is categorized and explained in the same Section. The fourth Section provides a suitable methodology (based on the results of the previous Section) for representing the weld process knowledge. The implemen-

tation details in creating the welding process schedule system based on an expert system shell are outlined in this Section while the details of the integrated structure is explained in Chapter 5. Consistency checking in expert system shells, a necessary feature for knowledge base development/addition is also outlined in the Section five. The final Section summarizes the chapter.

Background

Having defined the product model and determined how weld information can be represented (in Chapter 3), the application modules can interact, extract, or infer data, or consult their knowledge-base and provide solutions. A tight coupling is required between the product and process models since parallel interactions are necessary. To understand the nature of the importance of process information in robot programming, consider a fillet joint from edge to edge of a plate. The appropriate procedure would be to:

- Cold weld with prescribed data
- Build up a weld at the edges without melting
- Fill the joint between the edges with a weaving motion

If this procedure is not adopted, it is more than likely that a melt-through will occur. It is very essential to study the process and build sets of rules and procedures from which welding data can be obtained and the weld process procedure generated. To understand the process and to collect knowledge a systematic methodology is required. The five stages in building a knowledge based system are (1) identification

(2) conceptualization (3) formulation (4) implementation and (5) verification. The characteristics of the given problem have already been outlined in Chapter 2. Steps 4, and 5 will be explained in Chapter 5. Conceptualization is the methodology of finding concepts to represent the problem-solving knowledge – in our case the Gas Metal Arc Welding procedure generation knowledge.

Any engineering problem is typically solved by managing cooperative knowledge sources. The multiple sources for welding come from different resources which are in all cases dependent on data and knowledge. Based on prior work carried out by other researchers in building expert systems for welding and in other manufacturing domains, the following preliminary conclusions were made:

1. Data comes from diverse sources
2. The traditional approach of selecting a pre-determined decision structure cannot be imposed
3. Sub tasks cannot be carried out sequentially
4. Results cannot be interfaced based on the decision structure

The problem solving paradigm that would be ideal to suit the above mentioned requirements should therefore be concurrent in collecting data and using the data for action. The three main elements that would accomplish this are: (1) the central repository (2) knowledge sources and (3) the operations that can be performed on them. The key here is the central repository. As the product representation of welds is developed, the data is recorded for use by all existing knowledge sources. When a particular source has sufficient information it can manipulate the representation

based on the operations that are possible. Action is then concurrent and cooperation is possible. To accomplish this the key challenges are: (1) how to build an environment that can support different knowledge sources and (2) how to develop the different knowledge sources. These are the two problems that are to be solved for any engineering problem requiring knowledge processing. To achieve this an integrated approach is required wherein data communication is not only performed, but exchange of knowledge is also attempted. To support this exchange, problem solving information needs to be represented explicitly and properly for each task.

The task of acquiring data and knowledge from experts was therefore undertaken (in addition to the traditional handbook knowledge). It was decided that since knowledge could come from various sources, different welding engineers should be able to provide varying and useful data. Since the goal was to obtain knowledge that can be used for robotic arc welding, it was decided as part of this task to explore possible differences between conventional gas metal arc welding and robotic welding. In addition the actual data for weld procedure generation was also sought. The "data search" was therefore approached in a systematic manner. The results obtained were very different from our initial understanding of the specific problem. Nevertheless the findings helped in the easier creation of the system and implementation subsequently became more algorithmic that heuristic.

Overall Methodology

The steps involved in this research in developing a knowledge mapping system for welding procedure generation were to:

1. Devise standard means to study (i) the differences between robotic gas metal arc welding and normal gas metal arc welding (ii) the order of welding procedure specification (iii) the expert opinion on the relationship between product and process variables (iv) the difference between the specification of the process parameters by academicians and practicing welding engineers

2. Use results of step 1 to arrive at meaningful conclusions, question if standard methodologies of knowledge acquisition can capture private (heuristic) knowledge and acquire welding process knowledge that needs to be mapped

3. Select a methodology based on results of step 1 and 2, use an appropriate shell/language and build the knowledge mapping system

4. Validate the results, outline shortcomings and provide steps to improve the mapping system

Knowledge Acquisition

Knowledge acquisition is the transfer and transformation of problem solving expertise from some knowledge source to a program. The objective was to tap both public and private knowledge involved in the specification of weld process parameters. To accomplish activities described in the first step of the overall methodology, a survey was designed using the standard design of experimental techniques. The survey was sent to nine welding engineers at plants of John Deere & Co., – manufacturers of farm equipment and users of robots for arc welding. To check the difference in the specification of process parameters for welding between practicing engineers and academicians the survey was answered by two professors who teach manufacturing

processes at the undergraduate level. Based on the results of the survey, the whole process of building a knowledge-based expert system for GMAW has been questioned.

Structure of Questionnaire/Survey

The objective of the questionnaire was not to obtain raw data but to obtain general guidelines and trends. These would in turn help in the specification of the problem domain. The answers to the questionnaire were solved using statistical methods but the replies were incomplete. The incompleteness came from the non-return of the questionnaire by the experts and non-answers to many important questions. This further led us to only abstract qualitative information. The details however assisted in the better understanding of the knowledge acquisition process in an engineering problem domain. The survey details sent to the experts is provided in Appendix C. The words expert, welding engineer, respondent are all interchangeably used.

Differences between manual GMAW and robotic GMAW To indicate the differences in robotic welding and manual arc welding, a set of questions was designed. Since it was decided that multiple experts would be used to provide knowledge, questions ranged from obtaining their level of expertise to listing the potential areas of heuristic information. In all, 29 questions were asked to elucidate information on the nature of the problem domain. The expertise of the interviewed welding engineers ranged from one year to sixteen years. The products that were welded using robots in all their plants were classified to be high volume – in that a robot was dedicated to weld a particular component. Flexibility of the product was not found as the interviewers belonged to companies that produced parts in large numbers and

this is representative of robot users. This confirmed our initial belief that robots are still used in high volume production and not for small-batch production. Research such as this will help in the robot's use in jobshops eventually.

The respondent's answers to questions concerning high level languages such as VAL II, RAIL varied from "just-heard-of" to "used-extensively". All the welding engineers had heard about graphical off-line programming systems but none of them had used one. They all agreed on the fact that even though higher level programming systems existed, a welding engineer is required to specify process parameters and plan the trajectory of the robot. The respondents uniformly felt that only 0-15% of the output from a higher level programming could be used and they attributed this to the differences between wire impingement point and the tool center point.

The significant result of this portion of the survey came from answers to the questions 8 through 10 (listed in Appendix C). The answers were divided in opinion about the differences in robotic GMAW and hard automated GMAW. However, they agreed that the specification of welding process parameters for robotic GMAW differed from manual GMAW. The difference however was perceived to be marginal. This is important from the standpoint of building a KBES. If the difference is indeed marginal, and if data are available in the form of a look-up table, then are the claims of researchers [Barborak et al. 1991] that KBES technology can be successfully employed for welding justifiable. Further, if the differences in welding process parameter specification are marginal and the optimal solution can be obtained by acquiring the private knowledge which compensates for the differences, then is it worth the effort in obtaining that "private knowledge"? The survey respondents felt that a KBES system would be a good start point and refinements to the model should be made by

the welding engineer. This leads to the creation of provisions in the KBES for such mechanisms that would help modify the existing rule base by the experts themselves and check for validity and consistency.

The subsequent questions of this part of the survey sought to determine whether a need for an expert system exists. It was uniformly agreed by the experts that knowledge constituted both public and private data. Public data for welding process specification came from the following sources: (1) American Welding Society (AWS)'s Welding handbooks (2) American Society of Metals (ASM)'s Handbook on welding and (3) Tables and charts of manufacturers such as HOBART, LINCOLN, L-TEC. The welding engineers felt that 10-30% of welding knowledge is heuristic in nature. The major contribution of a welding engineers' heuristics in the specification of welding process parameters is primarily in the areas of (1) torch angle vs. weld appearance (2) fine tuning of welding process parameters – current, voltage, and travel speed (3) the effect of process parameters on penetration required. Direct formulae are also used typically to calculate the weld bead volume, weld area, and for calculating the travel speed from these values. Yet another common and welcome answer in this part of the survey was that there are not many interacting parameters in the specification of welding process parameters. Finally it was agreed that facts and rules would adequately model the welding process specification problem and that the problem domain should break down into simple sub-tasks. These answers helped in visualizing the problem in a different perspective than that perceived originally. The structure of the system that needed to be built became simple due to the nature of the answers.

Product, process variables and their order In order to arrive at the list of product and process variables, an unordered list was provided to the experts as the second part of the questionnaire (see appendix C). The list of the product variables was found to be adequate but for one variable. The surface appearance (was not included in the original list), which the experts believed to help in the selection of process variables. The surface appearance was further classified into no rust, light to moderate rust, and rusty. The product variables that are needed in the selection of the process parameters are (1) material type (2) joint thickness (3) joint type (4) strength requirements (5) position of weld and (6) surface appearance. All these variables are fixed for a particular problem and obtained from the CAD database (provided there exist a structure for representing the weld feature). In this research, (as explained in the previous Chapter) since means have been provided to obtain these welding parameters there exists a direct input for the selection of the process parameters.

The results of the order of the process variables were not suitable for any statistical analysis. As suggested by Tonkay and Knott [1989] a method such as rank correlation would have been useful in an ideal case. However of the nine surveys sent to welding engineers, only five were returned. For this problem the experts were asked to order the parameters by assigning a number in ascending order and if there were to be a tie, provide the same number for all tied values. Two of the respondents had only two numbers (1 and 2) for the entire list. The remaining three classified this list into three sub-cases and this helped in the understanding of the problem better. This classification by the experts suggested the futility of any ordering scheme. The list is as follows:

Fixed parameters for a given machinery

- Electrode type
- Electrode size
- Gas mixture
- Gas flow rate
- Polarity - DCRP for GMAW

Operating Conditions (Process variables)

- Current
- Wirefeed rate
- Voltage
- Travel speed
- Robot-tool-to-material-gap (or electrode stickout or arc length)

Fixed parameters for the particular problem

- Torch angle
- Weave pattern
- Number of passes

Other parameters

- Dwell time

- Displacement from the joint for multi-pass welds

The welding engineers preferred this classification primarily due to the nature of parts they are normally used to in their plants. In high volume production, the first category of parameters are normally fixed for each robotic arc welding workcell since these act as dedicated machines. However, in a small volume production, depending on the nature of the job these parameters need to change. Hence during the construction of the knowledge-based system this order was adopted and rules were created that would be used in the selection of the first set of four process parameters also. The third category had very few rules governing the selection of the process parameters as will be described in the later stages. This order segmented the problem domain into clearly defined zones. The professors who were given the same questions ordered in more or less the same way without the four classification schemes. The results were satisfactory since the goal was to classify the problem rather than to find an exact solution.

Interaction of product and process variables To study the effects of the product variables on the process variables and the interaction of process variables, a carefully designed survey was developed. Since the number of respondents were to be only 9 and it would be time consuming for the welding engineer if the number of test cases were large it was decided to keep the number of questions to a minimum. This raises the question of the design of experiments. A good experimental design furnishes the required information with the minimum of experimental effort. A correct choice of method also needs to be chosen. The next important function is to provide a rational basis for providing the number of observation to be made.

Table 4.1: Factors and levels of experiment

Factors	Level I	Level II
Material Type (A)	1020 Steel	6061 Al
Thickness (B)	0.25 in	1.0 in
Position of weld (C)	Flat	Vertical
Geometry (D)	Plate to plate	Cylinder to tube

It was assumed that in problems such as welding, there would be different factors and the individual/combined effects would need to observed. This results in either constructing a factorial design or an incomplete factorial design. The number of factors chosen were the four product variables at two levels. The term factor is used to denote any feature of the experimental condition which may be deliberately varied from trial to trial. Once this is chosen, it is possible to determine not only the effect of each individual factor but also the way in which each effect depends on other factors (i.e. interactions). Inorder to avoid any bias, the trials are typically randomized. To increase the reliability of the experiment, replications are necessary. This was attempted in a manner similar to that described in the work of Tonkay and Knott [1989]. The idea here was to observe if the information provided would in any way help the construction of the expert system. The factors and levels are given in Table 4.1.

At four factors, and two levels there are 16 treatments. For reliability, 2 replications are usually made. This results in a total of 16 treatments repeated twice. In order to randomize the treatment, partial confounding called "Incomplete Random-

Table 4.2: Confounding factors

No.	Independent	Independent	General
I.	ABC	ABD	CD
II.	ABD	BCD	AC
III.	ACD	BCD	AB
IV.	ACD	ABC	BD

ized Block Design" has been attempted. For 2^4 factorial treatments, two blocks per replication is normally reasonable. However since there are four factors, more blocks per replication are preferred. The number of blocks per replication is of the order of 2^n. So the number of blocks chosen for this experiment is 2^2 i.e. 4 blocks. For a four block replication, 2^2-1 i.e. 3 interactions are confounded, of which two are independent and one is generalized. By partial confounding, the effect of the factor confounded is lost, but the main effect and other interactions can be studied with more precision. Further, the confounding interactions can be recovered from those replications in which they are not confounded.

The possible three factor interactions and their corresponding independent and general confounding are given in Table 4.2:

The general confounding is obtained by multiplying the letters of the independent interactions and omitting even powers. For example ABC X ABD = A^2B^2CD. Omitting A^2 and B^2, CD general confounding is obtained. Since two replications are being attempted, and partial confounding takes place, the effects of the confounded interaction can be recovered for one replication from the other replicate.

REPLICATE I

Confounded Interactions

- Independent: ABD BCD
- General: AC

REPLICATE II

Confounded Interactions

- Independent: ACD ABC
- General: BD

By this choice, the effects confounded in one on the other are obtained. The two replications are given in Figure 4.1.

Based on this and after blocking, the trials were assigned to the nine welding experts of John Deere & Co. Two of these were also provided to the professors to answer. At this point it was believed that a sound experiment was designed and that a good statistical analysis could be attempted from the results. Of the nine surveys sent, 5 replies were returned with answers to the welding parameter question. Of the five replies from the engineers due to the randomized design two questionnaires indicated cases corresponding to aluminum welding. Since the engineers were not used to any Aluminum welding, the questionnaire was returned empty. Of the three replies left, two questionnaires had vertical/overhead welding and since these engineers were not experienced in that area they opted not to answer. The remaining answers were insufficient even to attempt an analysis of experiments based on missing data.

Replication I		Replication II	
ABD, BCD, AC		ACD ABC BD	
Key Block		Key Block	
X1+X2+X4=0	0 0 0 0	X1+X3+X4=0	0 0 0 0
X2+X3+X4=0	1 0 1 1	X1+X2+X3=0	1 0 1 0
	0 1 0 1		1 1 0 1
	0 1 0 1		0 1 1 1
Block II		Block II	
X1+X2+X4=1	1 1 0 1	X1+X3+X4=1	
X2+X3+X4=0	1 0 0 0	X1+X2+X3=0	0 0 0 1
	1 1 0 1		0 1 1 0
	0 0 1 1		1 0 1 1
			1 1 0 0
Block III		Block III	
X1+X2+X4=0	1 0 0 1	X1+X3+X4=0	0 1 0 0
X2+X3+X4=1	0 1 1 1	X1+X2+X3=1	1 1 1 0
	0 0 1 0		0 0 1 1
	1 1 0 0		1 0 0 1
Block IV		Block IV	
X1+X2+X4=1	0 0 0 1	X1+X3+X4=1	
X2+X3+X4=1	0 1 0 0	X1+X2+X3=1	0 0 1 0
	1 0 1 0		0 1 0 1
	1 1 1 1		1 0 0 0
			1 1 1 1

Figure 4.1: Details of replication for the design of experiment

This went further to prove that knowledge acquisition is a complex process and that experts have a very narrow knowledge domain and standard methods prove at best to be good starting points. Also knowledge acquisition should begin with very specific problems. Nevertheless from the results obtained, a distinct match of the values specified by the experts with the welding handbooks could be seen (Refer Table 4.3 for the experts' answers and the handbook answer). The cases provided to experts were simplistic enough that a direct solution from the handbook could solve the problem. Even if a production part had been provided, the experts felt that they would still have provided answers comparable to those given in the welding handbook, since the fine tuning of the parameters comes into effect only with the fixtures and positioners in place. Also the surroundings in the arc welding workcell often dictated the exact value and answers provided initially were only an estimate and could be used merely as a start point. Further from a qualitative observation of the result, it could be seen that experts differ in the specification of the welding process parameters. Also, a bias appeared in the specification of process parameters, and this bias stemmed from the use of a particular manufacturer's product and these continue to specify the same sets of values for many years.

It is concluded that the design of experiments is suitable only for the understanding of the problem by the non-expert (the researcher or the knowledge engineer). It is not a suitable means for acquiring private knowledge. Although from the standpoint of statistical analysis the exercise was a failure, a valuable lesson was learned that could be applied to other engineering problems. It was agreed by experts that a highly specific problem domain where the expert is confident should be the basis of such an exercise and that means be provided by which subsequent knowledge representation

Table 4.3: Comparison of handbook specification and expert's specification

Parameter	Handbook	Expert
Electrode type	E70S-3	E70S-3
Electrode size	0.035 in	0.045 in
Gas mixture	CO_2 or C25	C25
Gas flow rate	40-50 cfh	35-40 cfh
Wire feed speed	375-400 ipm	440 ipm
Voltage	26-27	25-27
Travel speed	30-35 ipm	15-20 ipm
Arc length	0.75 in	0.75-1.0 in
Tool angle	10 degree push	10 - 15 degree push
Number of passes	1	1
Weave pattern	none	none

should be attempted by the expert. Nevertheless the ordering of the process parameters helped greatly to simplify the construction of the knowledge-based system and the answers helped in a better understanding of the whole problem of "knowledge for welding".

The next issue of concern was the dependence of the process variables and the interrelationship of process variables. Since the number of variables was too many, a factorial design was ruled out. Hence it was decided to obtain a qualitative observation on the interaction. The figures are given in Appendix C. From the five replies obtained general observations were arrived at. The experts varied widely in their choice of interactions. This can be attributed to: (1) the experts did not have the time to seriously look at the problem and (2) from the time they received the questionnaire to the time they returned it, there was no discussion. Therefore, dependencies

alone were picked and this formed a good basis for building the knowledge-based system. The point to be noted here is that these dependencies could very well have been obtained from the welding handbooks. The conclusions on the interactions include:

- Electrode type, electrode size, gas mixture, gas type primarily depend on the product parameters
- Current and voltage depended upon the process variables, electrode size and arc length. Current and wirefeed speed were dependent. Since electrode size was related to the product variables, current, and voltage in turn depended on them
- The other parameters were specified once for the given problem depending on the product variables and the other process parameter values

These automatically set the construction of the KBES to be in a particular order. (1) determine the variables that were classified as fixed parameters for the given machinery (2) obtain the set of operating conditions (called the process variables) based on the previous set of answers (3) finally determine the other parameters based on 1 and 2.

In all, the survey confirmed that process variables primarily depend on the product variables, and preliminary specification is based on simple considerations. The next step was to compile the actual welding knowledge (from the handbooks and the engineers) for the KBES. This is explained in the next Section.

GMAW knowledge for robotic welding

The procedure to acquire specific knowledge was divided into the same categories as explained in the previous Section. Knowledge primarily from the welding handbook and manufacturer's handbooks were compiled to form IF-THEN rules. The knowledge which went into the creation of the system is outlined below. The author has the program and the list of rules for those interested. Knowledge related to mild steel alone was encoded as it constitutes 90% of all welded fabrication [Welding Handbook 1984]. Salient details of the knowledge obtained from handbooks and welding engineers is described.

Knowledge for machinery specific parameters The machinery specific parameters are: (1) electrode type and size (2) gas mixture and flow rate.

Electrode Type and Size One of the important factors to consider in GMAW is the correct filler wire selection. The selection of this parameter decides the values of other parameters such as current, wire feed speed, travel speed. The proper selection of the electrode wire and the shielding gas also determines the physical and mechanical properties of the weld. The important criteria for selecting the choice of the filler wire are: (1) base metal composition (2) required mechanical properties (3) application specification requirements (4) weld joint design. The primary function of the filler metal is to control the deoxidation of the weld puddle and help determine the mechanical properties of the weld. The chief additions as deoxidizers are Silicon (0.4 to 1.0%), Manganese (1 to 2%), Aluminum, Titanium, and Zirconium ($< 0.2\%$). To improve the structural, mechanical, and/or corrosion properties Carbon, Nickel, Chromium, Molybdenum, are added. The most common electrodes used for mild

steel welding as per the American Welding Society Specification A 5.18 [1979] are: E70S-2, E70S-3, E70S-4, E70S-5, E70S-6, E70S-1B, and E70S7-S.

The rules that have been formed to select one of these electroes were based on the material type (killed, semi-killed, rimmed), surface appearance (rust condition) application domain (automobiles, farm equipment, home appliances, structural application), performance requirements, and position of the weld (flat, horizontal, or out-of-position).

The electrode size depends on the joint type and position, and the thickness of the base metal. Since the mode of metal transfer depends on the thickness and the voltage, this factor is also taken into account. Once a given electrode diameter is chosen, it limits the current carrying capacity of the electrode. A small electrode wire will produce deeper penetration than a large diameter wire at the same current settings. Hence the selection is dependent on the objective function (maximize production, maximize penetration and others). Larger diameter electrodes are normally used on thicker material to assure the presence of sufficient heat to melt the base metal and gain proper fusion. The large sizes are also selected to give the highest possible welding speed which relates to the objective function of maximizing production rate. The electrode diameter range varies from 0.023 in to 0.0781 in.

Shielding Gas Mixture and Flow Rate In order to prevent contamination of the molten weld puddle, the air in the weld zone has to be displaced. This is effected by a shielding gas. The contamination of the weld causes reduction in ductility and impact strength. It also causes porosity, inclusions, and underbead cracking in the weld metal. To avoid these problems, the gases that are used include argon, helium, CO_2, and very small amounts of O_2, N_2, and H_2. Argon, Helium, and CO_2 can

be used alone or in proper combinations as binary, ternary, or quarternary mixtures. The properties of shielding gases that affect the performance of the welding process include: (1) thermal properties (such as conductivity) at elevated temperatures (2) reaction of gases with the base metal (3) effect of the gas on the metal transfer mode. The thermal conductivity of the gas determines the voltage and the heat input to the weld. The base metal reaction determines arc stability and good fusion between the weld and the base metal. The shielding gas also determines the metal transfer mode and the depth the workpiece can be melted. Spray transfer is not obtained when using CO_2 is used. Also spatter tends to increase when mixtures are rich in CO_2. The performance criteria, the metal transfer modes, the cost factor, availability of gases, material thickness, position of weld, are checked before the KBES arrives at the choice of the shielding gas. For GMAW the most commonly used shielding gases are CO_2, 75% Argon-25% CO_2 (called C25), and Argon - 0-10%CO_2 - 1-3%O_2 (called STARGON - tradename Union Carbide).

Although CO_2 is not an inert gas, sound welds can be consistently achieved free from porosity and defects. Its popularity is due to the common availability, quality weld performance, low cost, and easy installation. However, the drawbacks are: (1) it will not spray transfer (2) high weld spatter levels. C25 shielding gas is known universally to be the gas for GMAW with short circuiting transfer on mild steel. The gas operates well in high current application, achieving good arc stability, puddle control, and good bead appearance. STARGON is used due to its versatility in welding carbon steel, low alloy steel, and stainless steel utilizing any metal transfer mode.

Since the main function of the shielding gas is to mechanically displace the

atmosphere to prevent contamination of the weld metal as it moves from the electrode, the flow of gas must be adjusted to ensure adequate protection and so as not to cause turbulence. Only a rough guide is provided in the literature and the number varies between 20 and 40 cubic feet per hour. The experts tend to specify the same gas flow rate regardless of the type of the weld. They concur that it is acceptable if it is within the specified range.

Knowledge for operating conditions After having chosen the electrode type and gas mixture, the operating conditions need to be chosen. The four important parameters are the welding current, voltage, arc length, and travel speed. These parameters affect the weld characteristics to a great extent.

Current In GMAW, when all other process variables are kept constant the welding current varies directly with the wire feed speed or melting rate in a non-linear relation. Further, when the wire diameter is increased at any wirefeed rate, the welding current increases. The burn-of curve has a linear increase in melt off as current increases and at higher currents with small diameter wires the burn-off curve becomes non-linear. Penetration also increases with increasing current for a particular wire diameter.

Based on the thickness of the metal to be welded, the objective function, and the position of the weld, a factor (ranging from 0.1 to 0.85 of the lower limit) is chosen and added to the lower range of the current specified in the handbooks. The current setting is related to the electrode choice since once a particular diameter wire is chosen, the current carrying capacity is limited.

Voltage The power supplies used for GMAW are typically Direct Current Constant Potential. The voltage indicated by power sources is considered the arc voltage

which in turn is a measure of the arc length considering the drops in the circuit. As the voltage is increased, the arc length increases which results in a spread of the heat energy input resulting in less penetration. Also, excessively increased voltage results in spatter, porosity and undercuts. Excessively low voltages cause porosity and overlap. The factors that decide the voltage are (1) metal thickness (2) type of joint (3) position of weld (4) electrode size and (5) gas composition. Since all the parameters listed are determined previously, voltage selection becomes straight forward. A range is typically specified and experts feel that normally trial welds are required to adjust the value to obtain the most favorable metal transfer and weld appearance. The values specified by the KBES provide a good starting point.

Travel Speed For a fixed value of current and voltage, there is normally a single travel speed that will result in a particular weld shape. Significant changes in travel speed require changes in current and voltage. The three criteria used for proper travel speeds are:

1. As the material thickness increases, the travel speed must be lowered
2. For a given material thickness and joint design, as the welding current is increases, travel speed is also increased. The converse is also true
3. Higher travel speeds can be attained using forehand welding techniques and longer arc length

The travel speed is calculated from the formula:

$$W_t = 0.8 * N * WFR * WB_a/WB_v$$

where W_t is the welding travel speed, N is the number of passes, WFR is the wire feed rate, WB_v is the weld bead volume and WB_a is the weld bead area.

Arc Length As arc length increases, the voltage increases resulting in smaller penetration. Arc length is primarily dependent on the material thickness and the choice of current and voltage, and ranges from 0.25 to 0.5 inches for short circuit transfer and from 0.5 to 1.0 inches for globular transfer.

Fixed parameters for the given weld The four operating parameters having been chosen, the other significant parameters are the torch angle and the number of passes. The torch angle helps in the adjustment of the travel speed. The typical angles are 45 degrees in the cross sectional plane for TEE weld and 5 to 10 degrees in the direction of travel. For butt welds the torch angle is held perpendicular in the cross sectional plane and 5 to 10 degrees leading in the longitudinal direction. By positioning the weld axis at 15 degrees to the horizontal and welding down hill, a weld reinforcement can be flattened and travel speeds increased upto 50%.

The number of passes is primarily dependent on the material thickness, the torch speed, current and voltage. A greater number of passes may result in distortion but also an increase in the impact strength. An attempt is always made to minimize the number of passes, maximize heat input and reduce distortions.

Overall Conclusions on Survey/Questionnaire on Robotic GMAW

The conclusions have been classified as general and specific and they are outlined based on the results of the questionnaire and the knowledge acquired. The conclusions were sent to the welding engineers as part of the validation process. Most of

the conclusions were accepted with minor modifications (see Appendix D for actual answers). The modified list is presented here. Validation of the system is explained as part of Chapter 5.

General Conclusions

1. Robotic Gas Metal Arc Welding is not significantly different from manual arc welding in the specification of welding process parameters.

2. The variations in the process schedule specification affect robotic arc welding more than manual arc welding. This is attributed to the adaptive on-line visual feedback control of the human hand that corrects for imperfections in the process specifications.

3. To solve a welding process specification problem, "experts"(welding engineers) tend to solve in different ways. Multiple experts do not help in providing a consistent set of data. Instead the data obtained is skewed and knowledge acquisition becomes complex. A single, reliable and knowledgeable expert would be useful for knowledge acquisition while a group of experts would help in preliminary discussions.

4. Expert knowledge is extensive for a particular problem domain but is limited for similar but slightly different problem domains (A case in point: An engineer used to specifying weld process schedules for mild steel knows very little about aluminum welding).

5. Experts agree on the nature of welding process knowledge viz., public and private (heuristic) knowledge together contribute most to overall welding infor-

mation and

- Public knowledge is primarily from handbooks (AWS, ASM, and manufacturers handbooks HOBART, L-TEC and others).
- Private knowledge derives primarily from experience in areas of (1) fine tuning of specified process parameters (2) welding torch angle vs appearance and (3) travel speed, current, voltage adjustments.

Specific Conclusions

1. Welding process parameters depend primarily (but not completely) on the product variables (material type, thickness, joint type, strengths, surface appearance).

2. Interrelationship between process variables (those described in Table 2.1) does not exist to a great extent. Even if it does exist, it is often not quantifiable by the experts and hence process optimization is achieved by instinct. This results in a simple construction for the knowledge-based system.

3. A systematic knowledge acquisition procedure was developed hoping that it would help in the acquisition of public and private knowledge.

4. The advantages acquired using this standard scheme were:

   - To arrive at all product and process variables
   - To find the order of process parameter selection
   - To find the interaction of product and process variables and obtain general trends

5. However, systematic standard knowledge acquisition methodologies did not provide expected results for the following reasons:

   - The answers were invariably incomplete due to: (1) experts being busy all the time and not having the time to reply (2) lack of knowledge in areas other than the experts problem domain (which is very narrow)
   - A complete statistical analysis of the results was not possible due to the lack of reply from respondents of the survey. This questions the need for an elaborate scheme using statistical design of experiment techniques for building the questonnaire
   - The knowledge that is transferred from a welding engineer by way of these acquisition procedures only provides the public knowledge (i.e) data from handbooks and not "so-called" private knowledge. This is again attributed to the type of questions asked. Specific instances alone elucidate answers from experts that use private knowledge. Arriving at the number of cases that should be provided for welding engineers is (a) difficult (b) does not ensure that all private knowledge can be encoded as certain specific cases may not have been encountered before by the experts and they may have to conduct experiments to obtain answers and this results in "answer-not-known" situation (c) the experts have their own time frame and may not be willing to spend too much time.
   - The previous point was further exemplified by the results of the survey. The same questionnaire was provided to two professors who teach welding at the undergraduate level. The response from them was similar to that

from experts and the professors used standard handbooks to arrive at the results. The results were very close to those given by the practicing experts. However, the welding experts performed better for specific instances of process specification. This is attributed to the fact that professors are not used to specifying process parameters on a day-to-day basis and further refrained from answering those questions. This highlights a very critical need for experts themselves to construct the expert system and build it over a period of time. These standard methodologies for knowledge acquisition will at best provide estimates of the problem domain.

6. Existing so called "knowledge-based systems" for welding (that are available commercially and as part of research prototypes) are primarily based on data and charts. However, the information is in a IF (clause) THEN (action) format. Hence it is suitable for the use of expert system shells rather than the use of traditional procedural languages.

7. The standard methodologies developed for the acquisition of welding process knowledge are inadequate in capturing private knowledge. But they help in providing general guidelines and may help in other problem domains as in diagnosis or planning problems.

Based on all the above mentioned conclusions and the knowledge obtained to map the weld process knowledge, a mapping system was developed to generate weld process schedules. This was implemented in two ways (1) using an existing knowledge-based system (PCPLUS) and (2) hard coding the knowledge in C++ language and coupling with the CAD structure (which was also written in C++) that was provided

for incorporating weld feature information in a solid model so that direct access to the design knowledge was possible (this implementation is explained in Chapter 5). The shell-based implementation is explained in the next Section.

Implementation Of The KBES Using A Shell

The intent of this Section is to demonstrate the ease of representing the weld process knowledge (once properly acquired) using an expert system shell. The details will be briefly outlined as conceptually there is nothing new, and is merely an application system written using PCPLUS [1987], a rule-based expert system shell. However, at the end of this Section, the requirements that are essential for future modifications of the rule base by a welding engineer (such that heuristic knowledge) can be updated (This is outlined since the conclusion in the previous Section pointed to such a need). PCPLUS was chosen as the shell for the following reasons: (1) the knowledge base can be continuously updated and a maximum of 1000 rules can be created (2) the inference mechanism uses a backward chaining approach that is typically needed for such a consultative system (3) the software runs on PC's so that the welding engineers can use them at the plant (4) graphics, external routines can be called (5) a trace facility is provided to determine how a consultation was arrived.

The knowledge that was acquired and described in the previous Section is to be represented. The fundamental building blocks that determine the mode of knowledge representation are the parameters, rules, and frames. PCPLUS stores attributes as parameters and the characteristics of the parameter are described by the semantics of the language, namely TYPE, PROMPT, TRANSLATION, and EXPECT values provided in the expert system. The representation scheme then characterizes this

information in the form of IF-THEN rules to produce the correct recommendation. The representation of the knowledge is a combination of data structures for storing information as well as the development of procedures for intelligent manipulation. The representation scheme that PCPLUS allows is by frames. A frame is a collection of information. The information of the problem domain is defined in the form of parameters and rules. Some information defines the structure and operation of the knowledge, and the data identifies the frame goal, initial data, and other essential components of the structure. Figure 4.2 explains the multi-frame concept adopted in developing this rule-based system.

The reason for using the multi-frame concept are: (1) the knowledge base can be segmented properly (2) addition of knowledge is easy (3) since there appears a sequence in the specification of the parameters, the successive chaining of frames is ideal and (4) consistency checking (if provided) is easy. The two concepts that are involved in the creation of the frames are the inheritance and instantiation. Inheritance is the order imposed in a parent and child frame. The value attributed to any parent frame can be accessed by a child frame and not vice versa. If another subframe is added then this frame has access to the root frame and the parent frame. However neither the root frame nor the parent frame has access to the new sub-frame's parameters. The other concept is that of instantiation. This refers to the process by which an expert system software activates or enters a frame during consultation. Instantiation is performed in a dynamic manner. In our problem the root frame consisted of the goal properties and the order in which each frame need to be instantiated.

The purpose of any KBES is to arrive at a conclusion. The conclusion in our case is the selection of all welding process parameters for the given problem. The frames,

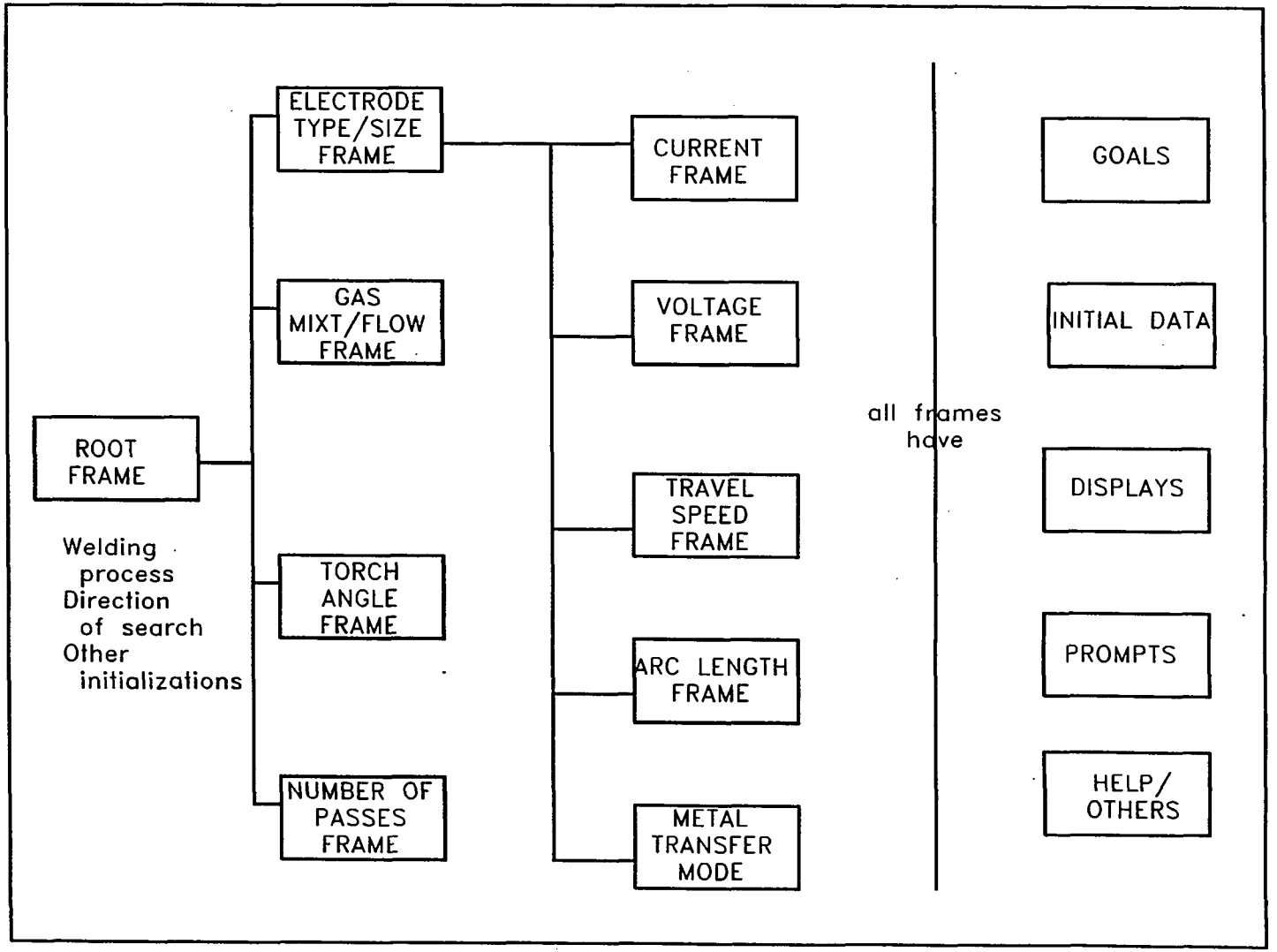


Figure 4.2: Representation scheme for the knowledge base

rules, and parameters work together to provide a conclusion for the consultation. The knowledge base is controlled by the backward chaining mechanism that has the following categories: (a) is goal driven (b) seeks only the data needed for the solution (c) instantiates subframes on its path to the goal.

The root frame contains the goal parameter which starts the consultation. The goal parameter RECOMMENDED is to be satisfied and the search for the goal drives the consultation. When PCPLUS finds a THEN part of the rule to match it instantiates the goal variable and the consultation ends. Before the first rule in the root frame is consulted to begin the backward chaining mechanism, the initial data of the root frame is invoked. This contains the various parameters that obtain the product data which include: (1) material type (2) material thickness (3) joint type (4) strength requirements (5) surface appearance (6) application specification. This information is provided by the user, since this shell-based system was developed as a stand alone KBES and does not have any interaction with the design module. However, the next Chapter will discuss the integrated environment. At this point, after the initial data has been obtained, all parameters are in the root frame. No sub-frame needs to be instantiated. The first rule has all the welding process parameters that need to be satisfied as part of the antecedent clause. This results in the knowledge base searching to find these parameters in the THEN part of other rules. Since each welding parameter value is obtained from each sub-frame the first sub-frame that is instantiated is the GET_ELECTRODE frame. The goal that has to be satisfied in this frame is the E_TYPE_RECOMMENDED. There are 16 rules in this sub-frame. Three rules are provided as a sample to understand the creation and execution of the program.

RULE 001

SUBJECT :: Electrode_type_rule

IF :: ((Material = Killed OR Semi_killed) AND
Metal_transfer = Short_circuit AND Out_of_position AND
Impact_strength <= 20 AND Tensile_strength <= 72000 AND
Yield_strength <= 60000)

THEN :: (E_type_recommended AND
Print "The electrode grade is E70S-2")

RULE 002

SUBJECT :: Electrode_type_rule

IF :: (! E_type_recommended)

THEN :: (E_type_recommended AND SHOW Get_electrode_grade
"The chosen electrode grade by the user is ")

RULE 003

SUBJECT :: Electrode_type_rule

IF :: (Position = vertical OR Position = overhead)

THEN :: (Out_of_position)

In order for the goal parameter to be satisfied, the parameters in the IF part need to be satisfied. Most of the IF part parameter value have already been obtained by the initial data portion of the root frame. The two parameters that need to be obtained are the Metal transfer and the weld position. For the weld position to be solved, the inference engine checks for the rule that has this parameter in the THEN

Table 4.4: Product data for sample weld

Material	:	Mild steel
Material Type	:	Semi-killed
Material thickness	:	0.25 inches
Application requirement	:	Farm implements
Strength requirement	:	Tensile: Not a criterion
		Impact: Not known
Objective	:	Not known
Surface appearance	:	Not Known
Joint Type	:	Tee
Weld type	:	Fillet

portion. This results in Rule 3 being fired. This in turn checks if the welding position is vertical or overhead and arrives at a conclusion and returns the value to the first rule. If all the parameters of the IF portion of the first rule are satisfied, the action clause is executed and the goal parameter of this sub-frame is satisfied. If it fails, it goes to satisfy the other rules in the frame. If all rules fail, Rule 2 is satisfied. The THEN clause of this rule is executed to obtain the user specified electrode type. This briefly shows how the knowledge base works on the basis of backward chaining. Once this sub-frame goal is returned true, the next sub-frame (using the CONSIDER-FRAME property in PCPLUS) Electrode_size is attempted. This continues till all the parameters are obtained. The total number of rules in the knowledge base is 137. The knowledge provides reasonably good weld schedules. A sample session for a problem is provided in Table 4.4 and 4.5.

This is the same problem presented in Table 4.2 where a comparison between the handbook and the expert was highlighted. The values depicted in these tables match very well excepting the travel speed. The reason for the value being different was that

Table 4.5: Expert process schedule for sample problem

Electrode type	:	E70S-3
Electrode size	:	0.035 or 0.045 in
Gas mixture	:	C25
Gas flow rate	:	35 cfh
Wire feed speed	:	400 ipm
Voltage	:	27
Travel speed	:	31.0 ipm
Arc length	:	0.75 in
Tool angle	:	45 degrees in the cross sectional plane 10 degrees push angle
Number of passes	:	1

the non-specification of the fillet size to the welding engineer and the engineer had made the remark that travel speed should be specified in accordance with the weld size. Validation of a knowledge-based system is very important and is approached in a different perspective and is outlined along with the implementation of the integrated system in the next Chapter. However, in order to attempt the goals set forth (together with – that of acquiring the private knowledge), certain additions need to be provided for shells such as PCPLUS which do not have any means of consistency checking. The types of consistency and validity checking that need to be provided (and a method proposed for implementation) are outlined in the next Section.

Consistency Checking In PCPLUS

In a typical robot workcell environment, welding engineers constantly learn new aspects of welding and programming and update their mental database. If the same knowledge is to be captured, the welding engineer should be able to encode that

information in the knowledge base. When this is attempted certain checks should be provided such that the welding engineer does not enter rules that could be of one of the following type: (a) redundant rules (b) conflicting rules (c) circular rule chains and (d) subsumed rules. Currently PCPLUS does not provide any facility for validity checking and this is a necessity for knowledge building. The definitions of these conditions are provided below:

Redundant rules Two rules are redundant if the IF parts of the two rules are equivalent and one or more conclusions are equivalent. The IF parts are equivalent if the IF parts have the same number of conditions and each condition in one rule is equivalent to a condition in another rule. An example of a redundant rule would be:

RULE 001

SUBJECT :: Electrode_size_rule

IF :: Material_thickness <= 0.025 in AND

Metal_transfer = short_circuit

THEN :: Electrodesize = 0.030 in

RULE 002

SUBJECT :: Electrode_size_rule

IF :: Metal_transfer = short_circuit AND

Material_thickness <= 0.025

THEN :: Electrodesize = 0.030 in

Conflicting rules Two rules are conflicting if they succeed in the same situation with different conclusions. An example of a conflicting rule would be:

RULE 001

SUBJECT :: Get_data

IF :: Position = vertical or Position = overhead

THEN :: Weld = Out-of-position

RULE 002

SUBJECT :: Get_data

IF :: Position = vertical or Position = overhead

THEN :: Weld != Out-of-position

Circular rule chains A set of rules is circular if the chaining in the set forms a cycle. Example: rule 1: IF A THEN B; and rule 2: IF B THEN A form a cyclic rule.
Subsumed rules Two rules or rule chains are in subsumption if they have the same action clause but one contains additional constraints in the antecedent part.

If a welding engineer needs to add new knowledge without the help of a knowledge engineer then means should be provided in PCPLUS to ensure validity. Since the internals of PCPLUS are not accessible and only access to the LISP version of the existing knowledge base is available, the following is proposed as a solution for consistency checking based on Nguyen [1987]. Once the rules are created (initial knowledge base) the file can be saved in different formats. The source format stores them as LISP lists that contain all information about the frames, rules, and parameters. This format is the most convenient to read and store in a manner that is needed so that consistency checking can be attempted. If a new rule is to be added to the system, then the program or interface between the user and PCPLUS should do the

following:

1. The already existing rule base should be converted into appropriate tables for comparison purposes

2. As the new rule is added, it should be parsed by the program and the contents – the IF and THEN portions should be separated

3. This should then be checked based on various algorithms with the tables created in step 1 to verify the consistency.

To accomplish this in PCPLUS, three entities need to be built: (1) a parser to read the LISP version; (2) tables of comparisons and (3) algorithms to check validity. This is not a trivial task and draws heavily on the fundamentals of compiler techniques and implementation is not trivial (and nor the goal). However the steps involved would be:

1. Convert the compound IF clauses into a disjunctive normal form such that the compound conditions are separated into simple conditions.

2. Check if two simple conditions are equal by the unification algorithm.

3. Create three two-dimensional tables where the row and column indices represent rules. The tables to be created are based on the comparisons of: (a) IF portions, called the IF-IF table; (b) THEN portions called the THEN-THEN table, and (c) IF and THEN portions called the THEN-IF table.

4. These can then be used by the various algorithms as described by Nguyen [1987] for checking circular rule chains, conflicting rules, subsumed rules, and redundant rules.

Summary

The important contribution of this Chapter is not in the implementation of knowledge using an expert system shell as this has been accomplished by other researchers and is reported by Barborak et al. [1991]. The initial belief was that existing KBES captures public data alone and private heuristic data can be captured by methodical means. With this a survey was designed that would elicit knowledge. The attempt was a failure in terms of capturing the private knowledge and a subsequent statistical analysis, but highlighted that private knowledge acquisition is complex. The best solution would be to create a system based on available public data that would be a good starting point and subsequent addition should be done by experts themselves. In order for this to happen the shell needs to be endowed with validity checking means. The problems that need to be solved by the consistency checker are outlined and a method is proposed. This will hopefully be a part of future shells and lead to an easier and better development environment. However, based on the work in this Chapter, classification of public knowledge was achieved. Also this work indicates that in order for intelligent reasoning to take place, a richer representation scheme of the geometry of the workpiece and surroundings is essential. The next Chapter will outline the implementation of an integrated representation scheme and provide data extraction for the generation of weld process schedules and input to the trajectory planning module.

CHAPTER 5. IMPLEMENTATION OF THE INTEGRATED WELDING REPRESENTATION AND PROCESS MAPPING SYSTEM

Implementation details of the integrated system based on the representation structure of welding features and attributes and the knowledge mapping system presented in Chapters 3 and 4 are described in this Chapter. The overall methodology with system architecture details is provided in Section 1. Section 2 outlines the object-oriented programming paradigm and its application to this research. WA-GRAPH attachment and implementation in a CSG tree along with details of creation and retrieval of information are explained in the third Section. The extraction of explicit and implicit information from the CAD-based representation for use by the welding procedure generation module is described in the fourth Section. Section five provides a sample session. Section six describes the validation process and the final Section summarizes and ties together the entire system.

Overall System Architecture

Considerable work has been performed in various fields of robotic system development in areas such as tact, vision, sensors, simulation, motion planners [Steiger-Garcao and Camarinha-Matos 1987]. At the same time parallel growth has taken place in the area of CAD systems and the ability to abstract information from a

higher level representation. The necessity for a convergence of these concepts is felt clearly and the integration of these results is important. Keeping this in mind, a subset of robotics, that of arc welding performed by robots, was chosen as the research topic in order to create a system based on the building blocks which have been described earlier. The key element identified was the specification of structural and functional information and their interconnection to arrive at a synergestic system. This system cooperates with and acts in conjunction with various other modules as explained in Figure 5.1. This may not be the ideal model and future redefinition may occur, but for the present the interaction of various physical elements has been identified and solutions have been proposed to overcome them. There are two aspects to robot programming: (1) off-line planning and (2) on-line feedback. It is understood that the real world is not always as predicted in off-line programming systems (anything can happen in off-line) and sensorial on-line data collection will be the key to closing the loop. However it was considered that even the first stage – that of off-line programming of welding has not been completed and that this attempt would therefore be justified.

The system architecture is shown in Figure 5.2. At any instant the user interface provides the user a menu of logical operations for the current mode. Menus appear showing default parameters with methods for specifying the geometry and weld attributes. The system is developed in an object-oriented modeling environment, with the solid modeling system IDEAS running as a child process. The software is structured in three modules: (1) the modeling mode (2) the process mode and (3) the robot kinematics/trajectory planning mode. The outcome of the interaction results in (a) CAD database with complete weld information, (b) welding schedules for the

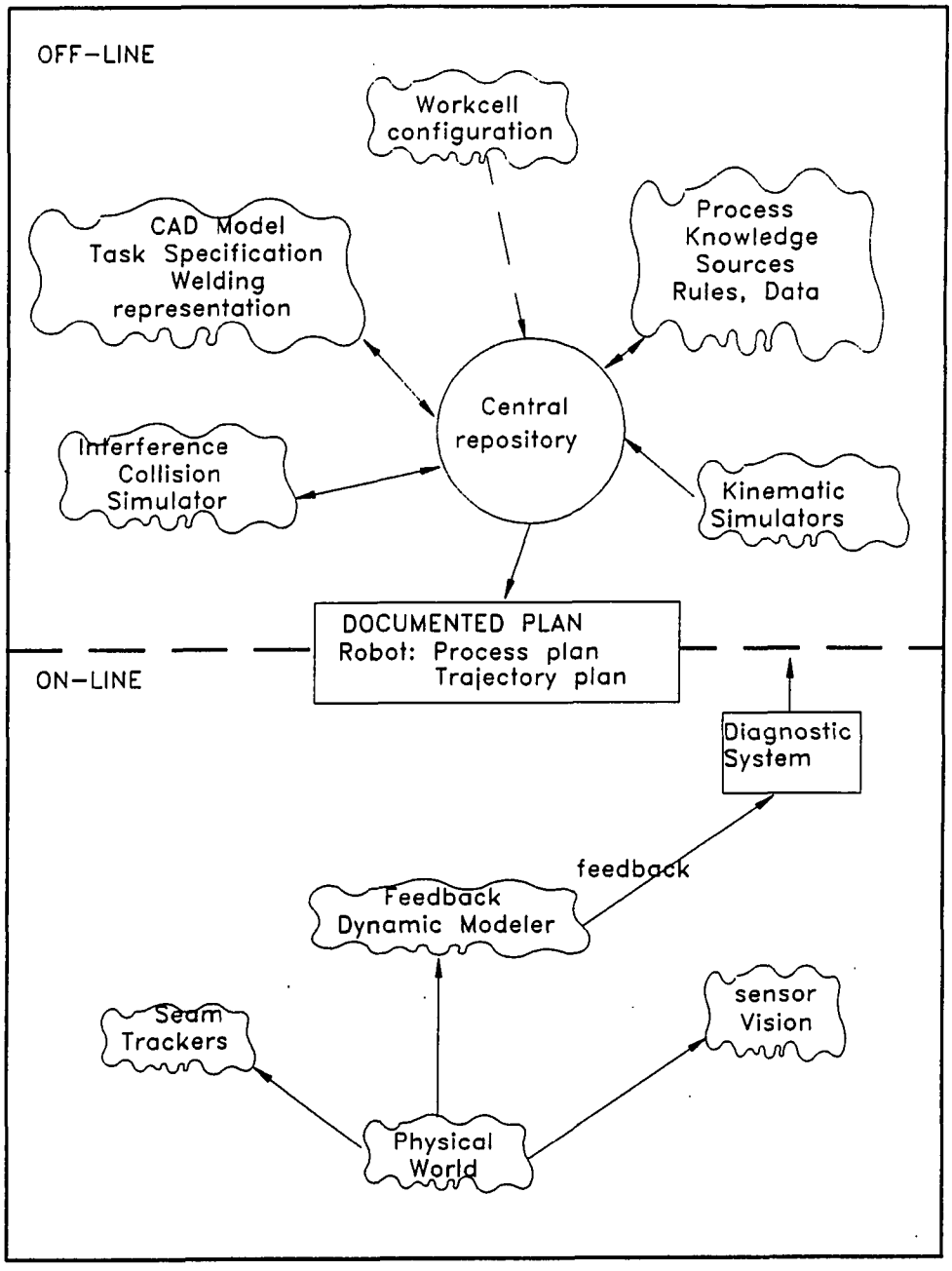


Figure 5.1: Overall interaction in an automatic robotic programming system

geometry described and, (c) robot and positioner orientation angles that may be used by the robot and positioner.

Object-Oriented Programming Paradigm

Object-oriented programming (OOP) is a technique – a paradigm for writing "good" programs for a set of problems and support object-oriented design [Stroustrup 1988]. The term object-oriented programming language means any language that has mechanisms to support the object-oriented style of programming very well. C++ is one such object-oriented programming language and is a superset of C.

Object-oriented design focuses on finding relevant abstractions and specifying them in such a way as to allow one to (a) solve the problem in hand (b) allow for future extensions (c) reuse the abstractions in solving similar problems. Hence object-oriented programming is the process of implementing such designs, which will make heavy use of the techniques of data abstraction and message passing. Such programming is most congenial in a programming language that allows:

- Encapsulation – all the functions that can access an object are in one place
- Data abstraction – hide the data structures and algorithms to implement the abstraction
- Inheritance – implement a data abstraction by stating the differences from other existing implementation
- Type Polymorphism – directly express the dynamic binding of message names to procedures

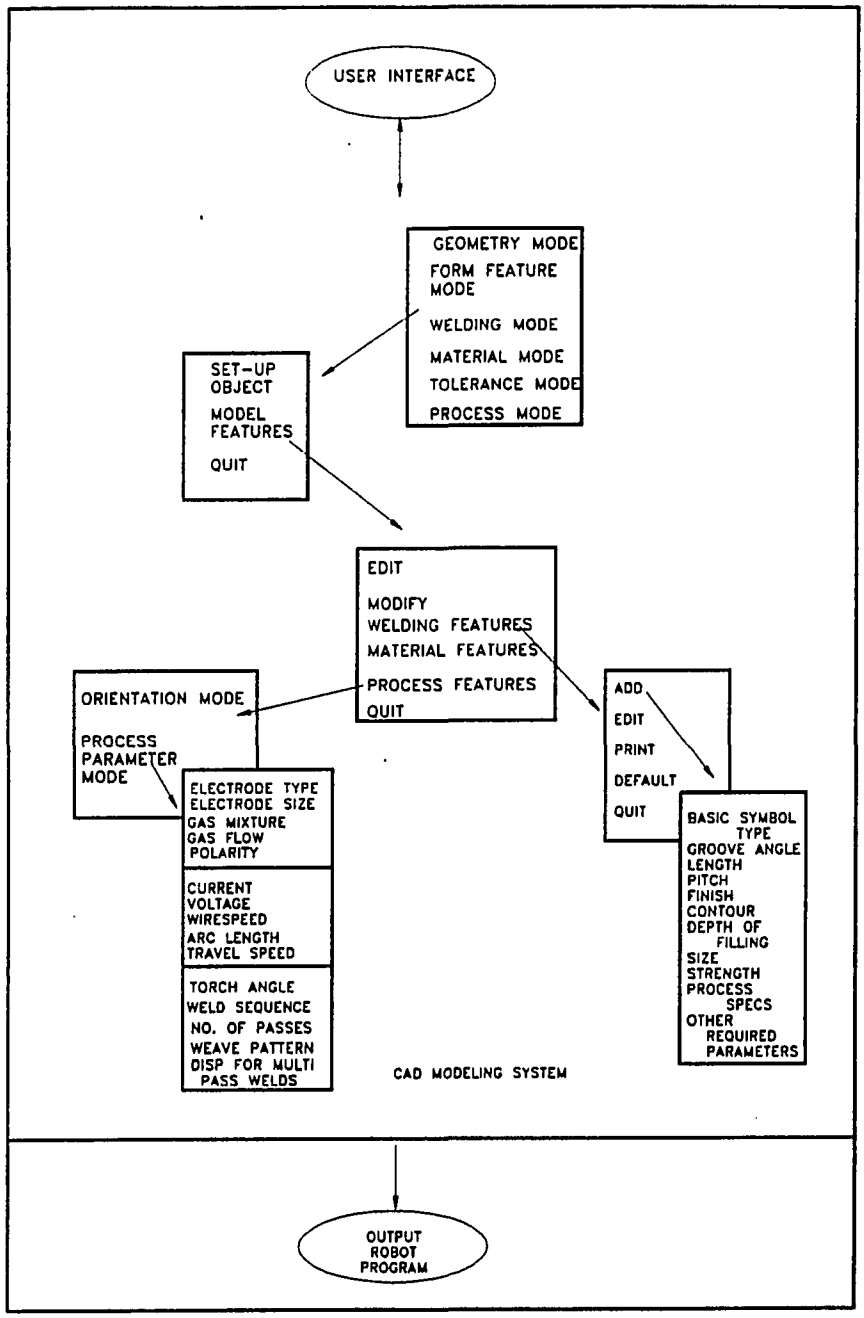


Figure 5.2: System architecture of the prototype system

C++ has all of these characteristics. Encapsulation means that all the functions that can access an object are in one place. C++ supports this by the **class** data type, which allows declaration of the functions that can access data within the body of its declaration. A class in C++ is similar to the structure in C and is the basis for much of the support of C++ for OOP. C++ allows to hide the representation of data in storage as well as restrict access to data. The class type allows data types that can be used without the knowledge of their representation in storage to be defined. All that is needed to be known are the appropriate user interfaces (functions) that interact and the corresponding argument types. C++ allows a class to be derived from another class. Inheritance allows the source code for a base class to be written and then the base class may be used to derive new classes with additional data or functions. Type polymorphism means that a pointer to an object can point to a variety of different types and an appropriate function can be selected at run time based on the type of object actually referenced. C++ implements this concept using dynamic binding, inheritance and type checking. These properties help in: (a) language extension (b) common interface (c) shorter development cycle (d) software reuse (e) easier software development and maintenance (f) cost effective and higher design/software productivity. Typical applications of OOP are in graphics, simulation, and large system development projects. The initial goals set forth during the start of the implementation were:

- Since access to the internals of the modeling system is not available, an interface between the user and the modeling system is needed. This translates to providing some of the facilities the modeler provides (such as creating, deleting, displaying simple and compound objects). This can use sharing of data

using classes and derived classes (facilities of inheritance), calling common functions for each primitive (example: a function display can be used to display a cylinder's property as well as a block's properties) – this requires features of polymorphism and each such class can have its own methods and data –this needs the property of encapsulation.

- There exists different types of weld and joint types. The weld information can be created once during the set up mode and any new weld types can be created and added to the system. This requires properties of data abstraction, software reuse and inheritance (implementing the difference alone for the new methods – example: between a butt and edge weld).

- The knowledge mapping system was initially decided to be created as a shell, so as to accomodate different knowledge bases (such as gas metal arc welding, submerged arc welding, tungsten inert gas welding and others), Object-oriented programming can provide the facilities for adding rules, deleting rules, displaying rules, or the consistency checking of rules. Each of these facilities can be a class in itself to perform a distinct operation and the system can be created and added in a modular manner. The user need not know the internals as long the interface is known. This relates to the concept of data abstraction and encapsulation.

C++ provides facilities for implementing all these requirements. It was chosen as the language of choice.

Welding Feature Representation Implementation

The representation structure is composed of generic objects and their associated methods as well as customizable objects and their methods. The customized objects are those that contain the process specific information while the generic methods hold the modeling object classes and their methods. Once these modeling objects are created, they are to be mapped with the application features – the welding procedure information. Using this information and an inbuilt inferring strategy, the system drives to output the process schedules in a data-driven approach.

The four main representation structures are: (1) simple object – for creation of the leaves of the CSG tree; (2) compound object – for creation of the nodes of the CSG tree; (3) WAGRAPH structure to associate welding information and (4) WTYPE structure to associate weld joint information. The first two are classified as generic objects while the last two are classified as customized objects.

Generic object

The main intention of this part of the implementation is the creation of the leaves and nodes of the CSG tree such that (a) the WAGRAPH can be attached (b) the information can be retrieved for application and (3) the necessary program for running IDEAS can be written.

Figure 5.3 explains the scheme of the class object. This class has sub-classes of simple_object and compound_object. The concept of inheritance is used extensively as there is much common data that needs to be created for the instance of a simple object or a compound object. This common data are the protected members of the base class object. By this arrangement, the sub-classes are allowed to access the

parent class data members while other classes cannot access these members and help in data hiding. The protected members for this base class are:

1. a pointer to the parent object
2. the rotation and translation of the object is of the class type three_d_coord. This provides the X,Y,Z float values which represent angles and distances respectively
3. the number of faces in the particular primitive or object so that numbering of primitive faces (for WFACE and PFACE data structures) can be attempted
4. an object-id. This object-id is useful for the associative array mapping. In the mapping the object-id is used as the key member and the contents of the object are used for later extraction

An instance of the object can be created using the constructor operation provided in the class. This initializes and allocates the memory for the object. The destructor allows memory to be compacted and reclaimed. Other public member functions that are created allow the derived classes to call these function and answer questions such as (a) what is the parent feature? (b) what are the child features? (c) what is the adjacent feature (d) what is the object-id. Other available functions allow extraction of information and output of required member data. The virtual operator allows the use of functions by the sub-classes and calls appropriate functions of the same name in different instances.

Simple objects: The simple objects that are created are the primitives that a modeling system supports. For the prototype the three primitives that were implemented

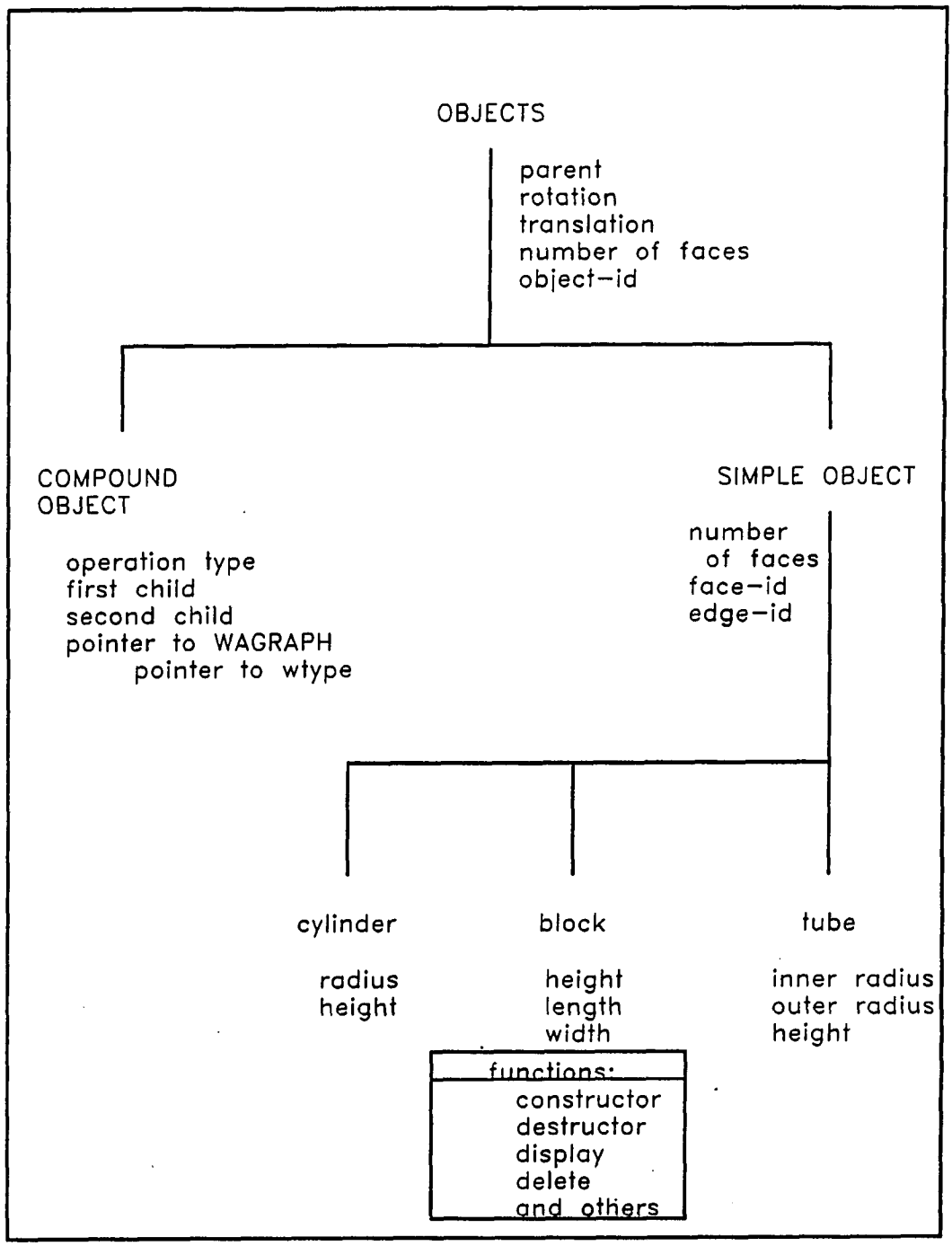


Figure 5.3: Schema of class object

are the block, cylinder, and the tube. The idea of stating the difference for future creation of the objects allows easy software building and when needed any number of additional features can be added. These honest primitives require the input of specific information. For example, the block requires the dimensions of length, width, and height. These are provided as input by the user. This input translates to being the private member of that class with the other public values being the protected data members of the simple object's parent – the object class. Like the class object, these simple objects have their own constructors and other appropriate member functions for display, delete and others. The simple object has the classes cylinder, block, tube as sub-classes. The common data member of the simple object are the number of faces and the object-id. The class has other functions to check and extract simple objects, and to check if the values entered are valid. Validity checking and consistency checking are carried out in the base class and specific functions are taken care of in the sub classes.

Compound Object: Compound objects are basically the combination of two simple primitives or the combination of two compound objects. They are the boolean (join, intersect, difference) of any two objects and this has to be essentially captured in a structure. The protected members are

- an operation type that is either a join, or an intersection, or a difference
- a pointer to the first child
- a pointer to the second child
- a pointer to the WAGRAPH class.

The WAGRAPH is set to null if there are no welds in the object combination and is set to the various attributes if a weld exists. When a compound object is deleted, then the meaning of the corresponding WAGRAPH (if any) is lost and hence it will also be deleted. This is part of the destructor function of the compound object. Other member functions are routine functions to check and verify various conditions.

Customizable objects

These classes are created in such a manner that to customize the program for a new process or weld type, the programmer must primarily concentrate on the process specific methods alone. The two main classes involved are WAGRAPH and WTYPE. The weld type has a sub-class called the joint_type. If a new joint type is to be added (since all the weld joint types were not implemented), then very few methods such as input, output, and process method need to be changed. Once the structure is known it is not very difficult to add the various types. These methods define the exact information that needs to be attached to the node such that it can be used for both display purposes and the generation of the process schedule when used by the knowledge mapping system.

WAGRAPH class: The WAGRAPH class is used primarily for the purpose of extracting the geometry information that is required for the trajectory planning of the weld, creates instances of them, and attaches them at nodes wherever there is a weld. Figure 5.4 provides the overall structure for the welding feature and attribute graph. Each WAGRAPH class has, as its private members, the evaluated entities: welding edge type, welding curve type, the measured entities (consisting of the start, end coordinate of the weld, the direction vector of the X and Z frame of the weld), the

WFRAME structure, the WFACE structure, and a pointer to WTYPE (refer Chapter 3). The pointer is to the sub-class WTYPE which has the sub-class joint type. A constructor allows the creation of this class and other functions (such as deleting, displaying), and is used for the various operations that need to be carried out on this graph. The next Section describes the details of how extraction of parameters take place.

WTYPE class: This class defines the various available types of weld joints. In order to ensure the validity of a weld, checks are carried out and only the appropriate joint type is instanced. This class is responsible for the creation of the joint type and this class is pointed to in the WAGRAPH structure. This along with the WAGRAPH class is associated with the node element so that retrieval is attempted during the knowledge mapping. For example when there is a Tee weld, and after the validity of the geometry is checked, the user selects the fillet_joint_type. This results in the creation of an instance of the class fillet_joint_type. This enquires from the user the fillet type, size, length, height, and pitch of the weld to store it as private members of the sub-class, joint type. It also inherits the weld type from its parent – the class WTYPE. Limited validity is performed to check the values provided. New classes of various joint types can be added depending on the usage. The facility of inheritance and modularity help in additional code being written. The prototype provides facility for Tee and butt weld with fillet joint and groove joint being programmed.

Knowledge Mapping System Implementation

During the initial stages of drawing up the functional requirements of the knowledge mapping system, it was thought that a need existed for an interpretive lan-

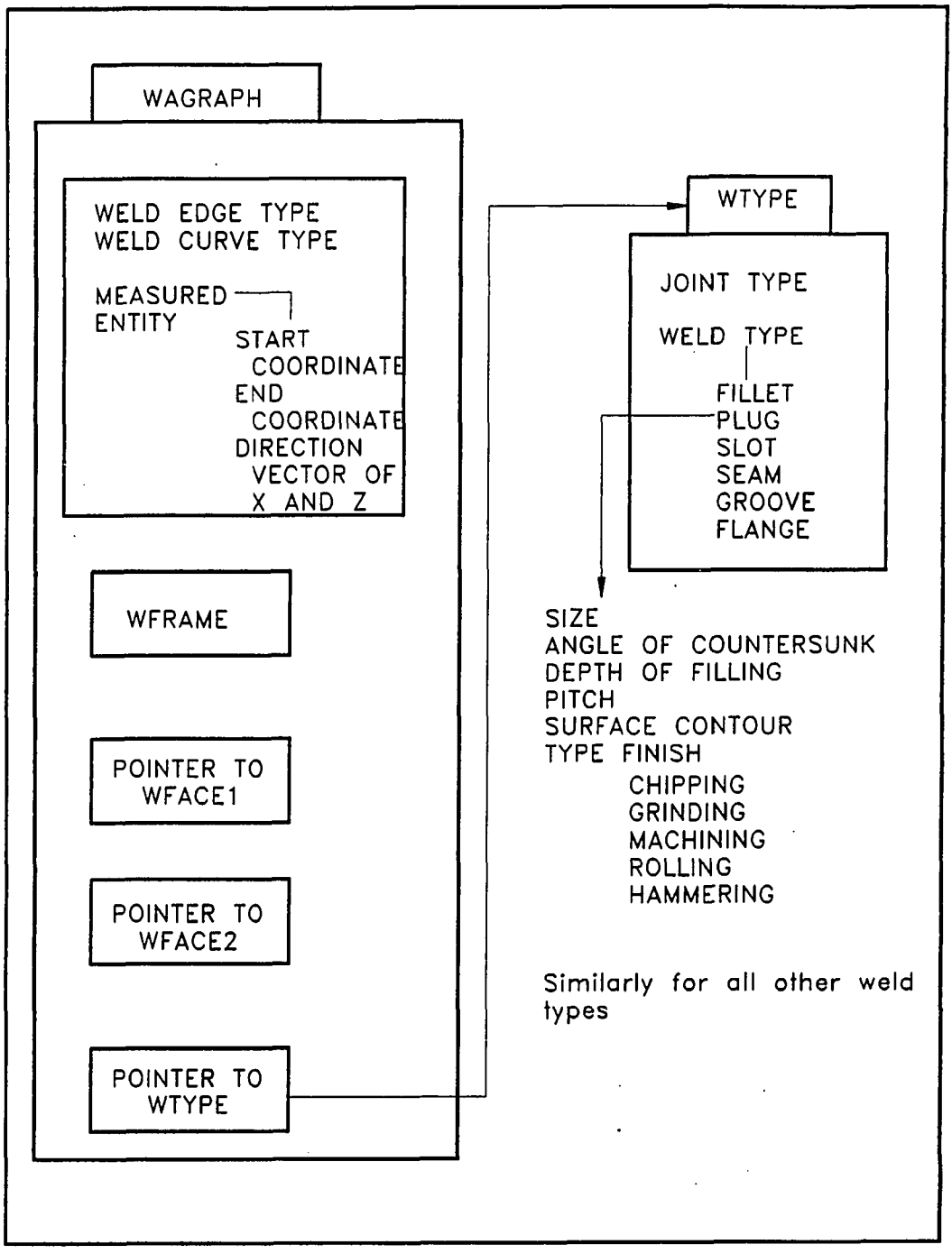


Figure 5.4: Schema of class WAGRAPH and WTYPE

guage or a compiled addition to the object-oriented programming environment to aid knowledge-based reasoning. Further, it was thought that there would be complex interactions between process parameters, that representation techniques may need to be provided as standard procedural techniques may not help. However, based on the previous findings (Chapter 4), it was learned that knowledge acquisition helped only to obtain public data. This data basically comprises sets of IF-THEN clauses. The reason why KBES shells are useful for representing this acquired welding knowledge is the easy interface the shell provides the user for editing and the inbuilt inferencing. The conclusion that the existing knowledge base is fixed and only future additions by welding engineers will lead to modifications in the knowledge base led to the building of the structure inside C++ although this does not explicitly support rule-based programming.

Framework

The ideal situation would be to combine AI programming and object programming so that logic/knowledge-based programming capabilities are added into the object-oriented framework [Su 1991]. The interpretive or the compiled language facility should allow the definition of variables, evaluation of expressions and condition, create the rules in the traditional rule-based language style (IF antecedent THEN action format) and be able to call appropriate functions to extract data, and input/output. Such a system could start with an object-oriented framework as provided by C++ and be able to add rules. The developed compiler can take an input of a C++ program containing the rules in a pre-defined format (as functions of a class with a special keyword say RULES, to indicate that it is followed by IF-THEN

clause). This input could be compiled to produce an equivalent C++ code that could be compiled by the standard C++ compiler to executable code. The rule information could be added in the pre-defined format and is essentially converted into functions in C++. The function name could be the name of the corresponding goal to be satisfied. The main function is the goal predicate. The sub-goals solutions are sought until the overall goal is unified. The key is to write a compiler (or a pre-processor) which can take an appropriate input in IF-THEN format (similar to PCPLUS or OPS5) and convert it to functions of C++. Since the goal of this research is not the writing of compilers (and is not also trivial), the easy approach was adopted – creating the functions directly and specifying control. Each rule is declared as a function. The overall goal is the main function. This in turn calls all subgoal functions and the solving of the goal results in the solving of sub-goals and this continues until the goal is resolved. This approach primarily uses the language features of C rather than C++ but was created as a C++ program (as it is after all a superset of C).

The important concept is that this program simulates PCPLUS to a very large extent. PCPLUS as it is provides primarily (1) a backward chaining inferencing mechanism (2) calling of appropriate sub-goals to satisfy the main goal (3) satisfying of parameters only once. The other facilities provided include (a) a convenient user-interface (b) multi-frame knowledge building (c) forward chaining (d) variable, expression evaluation (e) interface to external programs (f) tracing and explanations. It does not however provide any consistency and validity checking and is primitive in terms of error checking.

In a manner similar to that of the multi-frame concept provided in PCPLUS, rules for each one of the welding process parameters (such as electrode type, shielding

gas) are stored in a separate file and addition of knowledge for that particular parameter needs to be concerned with only that file – that is it is modularized. Figure 5.5 explains schematically the design of the program for each of the welding parameters. Since it is modular, and repeatable, the program is very similar for each of the welding process parameter with the knowledge being different in each case. Each rule is termed as an independent object and can be called by any other rule any number of times. A collection of all these rules essentially provides the solution tree for that particular parameter. The link for the tree is the call to another rule. In our case all rules are functions. The node, or rule, or object (all names interchangeably used) involves evaluation of relational conditions, expressions, calls to other rules (i.e. functions), extraction of information from the modeling classes (eg: material thickness), and formatting the information to the screen or to the file using the facilities provided by the programming system. Depending on the rule and the conditions to be satisfied, one rule might lead to the calling of another rule, which in turn to another, or the rule might fail. This then directs the pattern to search the next rule and upon resolving the goal or sub-goal the appropriate output concerning the welding process parameter is provided.

This search can be classified as performing three tasks: (1) invoking a function (2) filling values for variables and evaluating expressions (3) return action. The first step is just to invoke a call for one of the sub-goals to be satisfied. The second step consists of either extracting information from the geometry modeling binary tree with associated welding attribute information or asking the user to provide data. The return action takes place after the conditions are evaluated and takes it to the immediate higher level so that the search can continue. If the call is successful, a

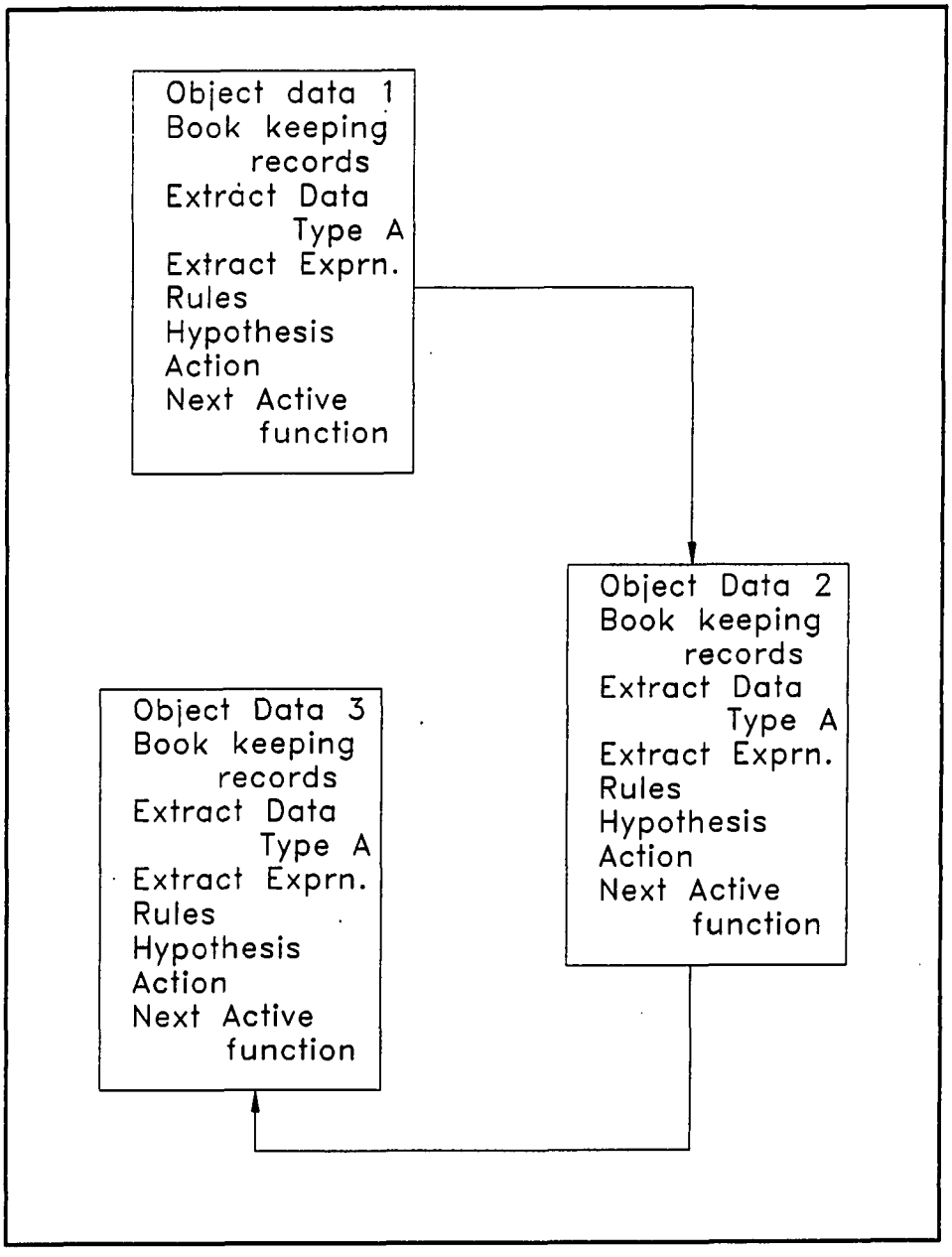


Figure 5.5: Schematic of the search mechanism

return value is returned. If all rules are tried and there are no more rules to be verified, the system will not be able to come to any conclusion and no data will be returned to the next level. At this instant the user is asked for information and this request is taken as the action of the rule. By implementing in this manner, the control logic is kept to a minimum and adding rules by the engineer is not a problem if the syntax is known. This system does not provide facilities for explanation (or how the system reached a particular conclusion). If a shell were to be developed using the OOP environment, then the explanation facility would be an essential feature. The structure of the rule is in such a format that to make modifications, the user needs to understand the meaning of C++ functions and the syntax.

Geometry parameter extraction

The overall objective is to satisfy the function gmaw_process(). This determines the welding schedules needed for the design already created. The binary tree that is available with the geometry and weld information is traversed from the top node to determine whether a weld exists. This is effected by a member function in the class compound called is_there_weld(). If there is no weld, the traversal of the tree is continued further and at each node the function to check whether there exist a weld is called. An affirmative on the return value of the function results in calling the main function which has all the welding parameters that need to be called as function routines. These rules are stored in separate files and the various files include: electrode_type, electrode_size, gas_mixture_type_flow, metal_transfer_mode, current, voltage, arc_length travel_speed, tool_angle. The results are provided as output to a file that can be processed for actual transfer to the robot.

The type of information that is typically extracted for the welding procedure generation is either Type 1, or Type 2 parameter (defined in Chapter 3). Type 1 parameters are already available as data members of the classes (the geometry or the welding attribute graphs). The extraction of these parameters are simple calls to data member functions of that particular instance. Type 2 parameters are evaluated entities based on the class data member values. These are calculated as soon as the geometry and the weld attribute values are assigned. When the parameter is required, it is retrieved for use. All values that are obtained are one of the following types: (a) structures (b) data members of class (c) functions. The analogy of slots within a frame (typical of frame-based representation) is obtained in the object-oriented programming environment by means of the private data members of the class and the constructor provides for default values. By this the property values of various weld types can be retrieved.

Type1 Parameters: The type 1 parameters that need to be extracted for the weld process module are: (1) material thickness (2) position (3) material type (4) joint type (5) weld type. The other type 1 parameters used in the kinematics module are: (1) start coordinate (2) end coordinate (3) participating features - the child leaves and the parent nodes.

To obtain the information, the product data needs to be organized and in a data structure most suitable for consultation. At run time, consulting the information that is available and resident in the memory, and access to it, is carried out through the member function of the appropriate classes. Consider the following as an example to determine the parent of a feature. The mapping program statements are:

```
pix j = oid_map.seek(obj_id1);
```

```
pix k = oid_map.seek(obj_id2);

c = oid_map.contents(j);
d = oid_map.contents(k);


if ((c -> parent_who() != null) || (d -> parent_who() != null)
{
   ....

}
```

and in the definition of the class object, there exists a function called parent_who. The partial code explaining this is given by:

```
class object
 {
  protected:
   object* par
   other data members


  public:
   object* parent_who() { return par };

   other member functions

 }
```

By this program fragment a few important concepts are highlighted and the usefulness of C++ can be seen. An associative array has very common use in programming just as a list, or stack is useful. A container class has already been written and is part of the library. What it allows are associative array operations (insertion, deletion, and membership of records based on an associated key). This requires the

specification of two types, the key type and the contents type. These are implemented in many ways but the map here has been implemented via a threaded AVL tree. However, the user does not need to know the implementation. The interface alone is important. The important concept of software reuse is clearly seen where a library that has been already created is being used. In this work a map is used with a string and the corresponding contents for that string's information. The string that was mapped was the object-id and the contents of the object-id are the data members of the object which is already created by the constructor function of the object. The data member of the object-id has a pointer to the parent object (see object* par in the program). The function that obtains the information is public to that class and returns the pointer to the parent object-id and this happens to be the parent of the current object. Without knowing the internals of the program the calling program has obtained the required data concerning who the parent of the current object is, and extraction of the value has been effected. As in the example of parent_who(), all other type 1 parameters – first_child, second_child, start_coordinate and end_coordinate of the weld, material_thickness and others are obtained. Each such member function is primarily a data member of the appropriate class. Any further data can be accessed whenever needed.

Type 2 parameters: Type 2 parameters required for the trajectory planning module are: (1) adjacent features (to detect access problems) (2) adjacent faces (to obtain the weld frame information) (3) normal of faces (again for weld frame information) and (4) weld curve information (to know the actual trajectory in a parametrized form). The various parameters that need to be extracted are primarily the geometry information needed for the weld trajectory planning module. This is attempted as

soon as the objects are created and the corresponding welding attribute graph created. The system evaluates all the type 2 entities and stores them in the appropriate class as a data member. This is essentially attached to the node of the binary tree corresponding to the particular object-id. Once this evaluated information has been stored, the traversal of the tree is the same as mentioned before. This avoids the need for parsing a text file if information were to be stored as a text file and later read during consultation. Consider for example the evaluation of information concerning the weld curve type by the function call weld_curve_type( ). This is accomplished by a series of steps with interaction to the solid modeling system, since there is no direct way of knowing the internal data structures of IDEAS. The steps include:

1. Based on the binary tree created by the interface, write the program file for IDEAS. Call up IDEAS to create the object.

2. Knowing that the two leaves and the compound object have been created based on the program input, the PEARL database is instructed to write the B-rep file.

3. This contains the topology and the dimension data. The information concerning the object combination surface is obtained.

4. Since IDEAS represent curves as a generalized NURBS curve (see appendix B), the latter is represented parametrically.

5. From the NURBS representation, the curve is identified.

Consider Figure 3.5 again. Assume the weld curve type needs to be found. The overall object is a simple block with a cylinder and has a weld edge on it. The steps

outlined previously result in the following:

1. This object is created using the welding representation/knowledge mapping program. When the appropriate menu (Write Program File for IDEAS), is selected it results in the binary tree information to be written as a program file. The program file consist of each of the command that IDEAS requires to create the drawing.

2. Since welds can be created only when object combinations occur, we are primarily interested in all those surfaces that have object combination. The union of the block and the cylinder resulted in 8 surfaces. The sixth surface has an object combination that can be a weld. The next step is to write the PEARL data file – which contains all the topology and dimension information. This file contains the needed information, which can be extracted.

3. IDEAS stores object combination surfaces with an id of zero. The file is searched for the surface-id with zero. The corresponding curve-id for this surface is obtained (in this case 26). This curve-id has a NURBS representation, as IDEAS stores them in that format. This curve information is accessed. The curve form has numbers ranging from 0 to 11, which provide if they are standard curves. In this example, the curve-form is given to be 2 which is a circle. This is the information we are interested in. The corresponding knot vectors, weights are obtained so that it can be transformed into a polynomial that can be used for the trajectory planning of the robot and positioner.

4. Now the next step is to convert this NURBS representation into a polynomial form. To accomplish this there exist different algorithms such as Boehm's

[1980] algorithm. These provide a method for evaluating a NURBS curve and producing a cubic polynomial without computing the basis functions. This allows the exact polynomial curve which can be differentiated and used in the trajectory planning problem.

Similar methods are provided for evaluated entities (type 2 and type 3 parameters) including: (1) surface normals (2) weld edge data (3) direction vector of X axis on the weld frame (4) direction vector of the Z axis on the weld frame and (5) face information details. Having extracted the various parameters that are needed for the knowledge base reasoning and trajectory planning module, an example is provided in the next sub-section to highlight the method.

Storage, retrieval, and traversal

Storage: Assuming that the WAGRAPH attachment has been already attempted, then there exists information concerning the weld and the geometry. This can be saved as a text file. However, retrieving this information for modifications, further building, and traversal will be difficult. Hence, it is stored in a dynamic data structure such that a binary tree data traversal can be performed to indicate the number of welds and its associated properties. The data structures for storage, retrieval and traversal is shown in Figures 5.6 and 5.7. This contains all object instances and the associated details of the weld attribute.

For the storage of information, the objects should first be resident in the memory. In this implementation, it is being stored as a map of type objectptr. This associative array has two types, the key type being a string which contains the object-id, and the contents type being a pointer to the corresponding object. As objects are

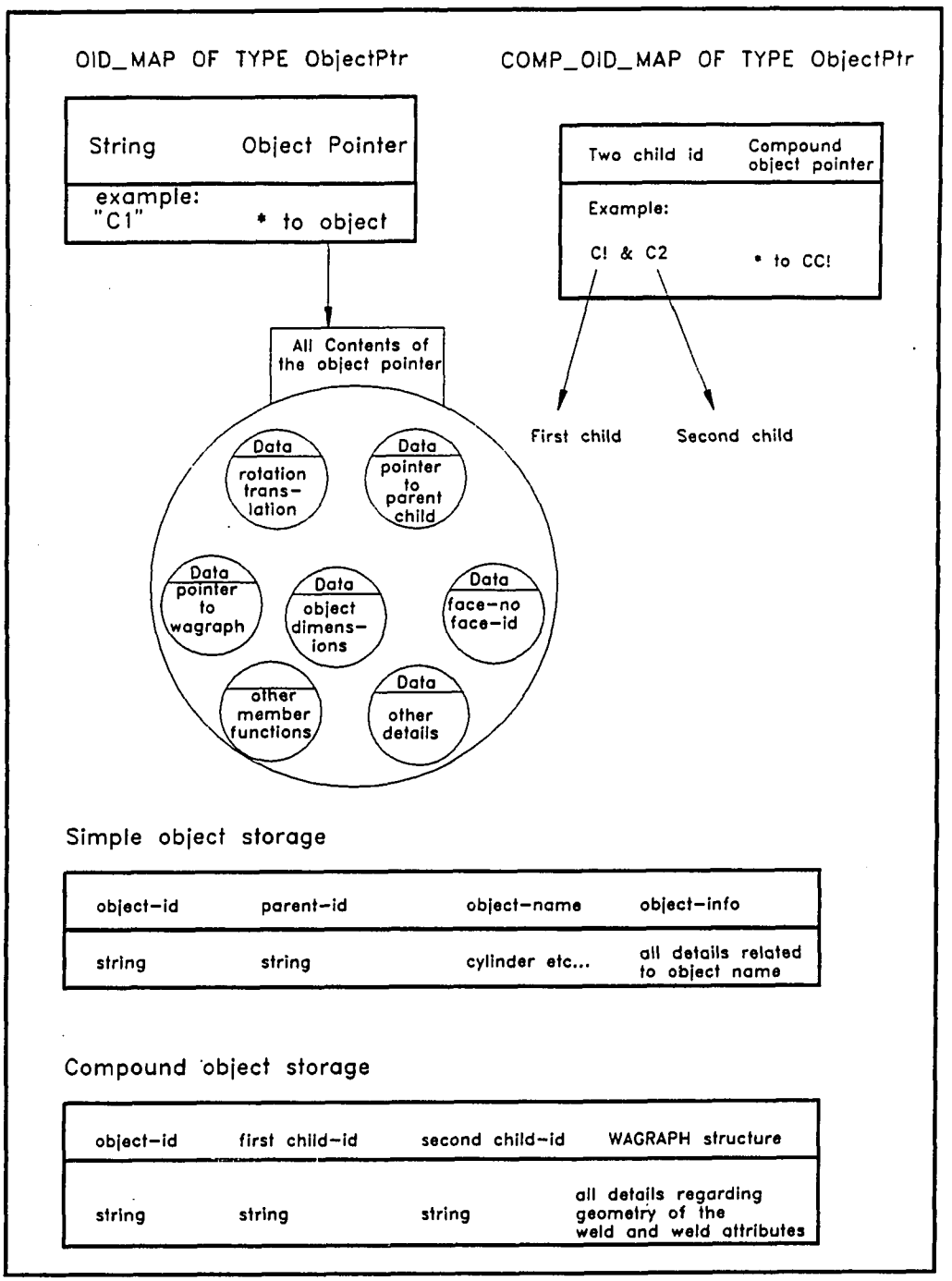


Figure 5.6: Data structures for storage

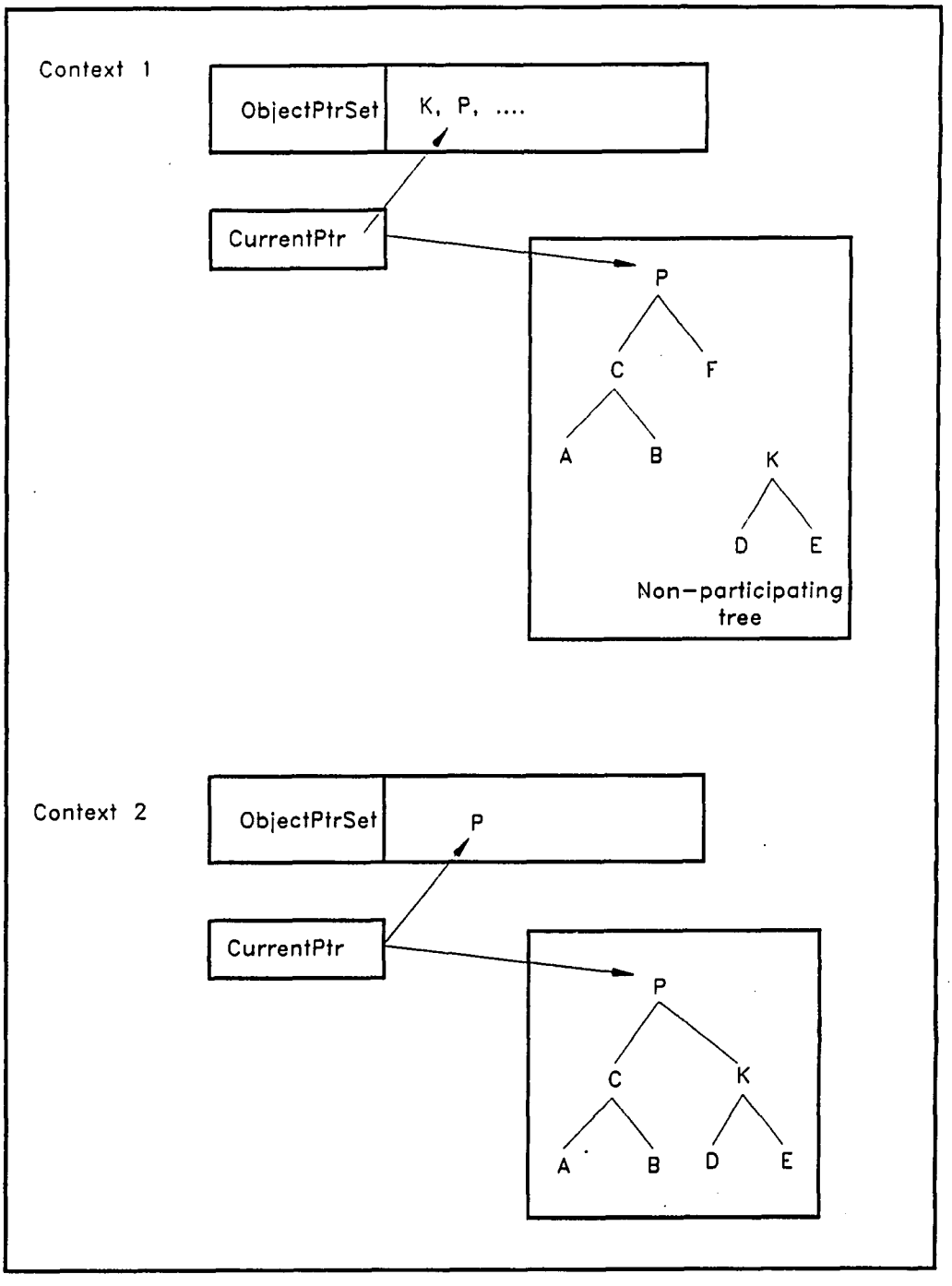


Figure 5.7: Data structures for retrieval and traversal

created, they get stored in the container class in a particular order determined by the map implementation. When type 1 parameters are extracted, this container class is accessed and corresponding to an object-id, the contents are retrieved. This map contains all the objects created - those connected to the binary tree of the current solid and those that were created but not used as a leaf or node. For example, at the middle of a session or when a session is completed, the user indicates that the information need to be stored for subsequent use. This data are stored in a binary format. All the records used in the files have been implemented as structs in C++.

The lower half of Figure 5.6 describes how a simple object and a compound object information is stored. The simple object contains the object-id as a string, the parent, the actual object name (such as a cylinder) and the corresponding information for the simple object. The idea of type polymorphism is again used here, by which whenever a store function is called, it invokes the appropriate function (simple.cylinder.store or simple.tube.store) and writes the exact amount of bytes in the file. The information that needs to be written on to the file is obtained from the map of the object-id and the corresponding pointer to its contents. For a compound object the comp.object.store writes the object-id, the first and second child object-id, and if a WAGRAPH exists, then the entire information is written as a structure. In addition to all this information, a password is set up at the top of the file to make sure the correct file is read. Now the file contains the necessary information that can be used for retrieval. It should be noted here, that neither in the associative map nor in the file storage, the CSG tree is explicitly stored as a binary tree. However, the binary tree is implicit, since the information concerning parent and child is stored and a parent can have only two children.

Retrieval: When a user needs to continue on an old session, or inquires about a weld, then the stored data need to be retrieved. When it is retrieved, it should also create the associative map as that allows data extraction of type 1 and type 2 parameters. In order for this to happen, the problem of whether the parent or the child should be created first needs to be resolved. It is typical to create the leaves first and then go on to create the nodes. However, we have a pointer to the children in a compound object, and the children point to its parent. This results in a cyclic structure. To circumvent this, another map of the same type objectptr is created with the name comp_obj_id. This contains the key as a string which is nothing but the two child object-id's separated by a special character (or space) and the other is a pointer to the corresponing compound object. The retrieval and loading on to run time memory is attempted in two phases. In the first pass, the objects are created without any contents along with the associative map for the compound object. In the second pass, corresponding to the compound object pointer the two child nodes based on the object-id of the compound_obj_id map is filled and corresponding parent information is also filled. By this the both the objects – simple and compound are created in the memory.

Traversal: Once the tree is created, the user may want to know if there exists a weld or extract some weld attribute values. To accomplish this, few validity checks need to be performed. There can be cases where simple objects, are created but do not participate in the overall object creation. In such a case, we need not extract details of them. Also, once it is included in the current context (the CSG tree) the corresponding details should be available for extraction. This is accomplished by having two structures: (1) a creation of an unordered set which contains all

the pointers of objects created called ObjectPtrSet; (2) another pointer called the CurrentPtr, which points always to the current solid. When a tree is created, it is immediately included in the ObjectPtrSet. Once these objects are associated with the overall CSG tree, the ObjectPtr is removed from the set and the CurrentPtr points to the current solid. When a tree traversal is to take place, the current solid is recursively searched to obtain the weld and other significant details.

Example rules

The equivalent code for the three example rules provided in Chapter 4 is outlined in C++ format to highlight the mechanism of obtaining the welding process parameters.

```
main()
{
   // declare variables
   // read data

        if (electrode_grade())
            cout << "The electrode grade has been selected \n ";
        else
            cout << "I was not able to come to any conclusion \n ";
}


electrode_grade()
{
        if ((mat_type == killed || mat_type == semi_killed) &&
          (mat == mild_steel) && (weld_position())) && (impact_strength <= 20)
           && (tensile_strength < 65000) && (metal_transfer()))
          { cout << "The electrode grade is E70S-3 \n ";
          return(true);}
        else
```

```
        electrode_grade2();
}


metal_transfer()
{
      if ((material_thickness < 0.125) && (weld_position()))
       return(short_circuit);
      else          return(0); }


weld_position()
{
      if (weld_pos_state == unknown) { weld_pos_state == known
       if ((pos1 == vertical || (pos1 == overhead))
         wpd = out_of_position; }
       else
         wpd = in_position;
} return wpd;
```

In this example, the goal rule that needs to be resolved is the type of electrode grade. In the main routine if the return value for the function electrode_grade() is true, the electrode grade is selected. In order for this to be satisfied, there are a series of conditions that have to be evaluated. Some are explicit type 1 information and some conditions call other rules. For example, the type of material is obtained based on the earlier CAD representation which contains this information as part of the class object's data member. Whether it is a type 1 parameter or type 2 parameter, each is obtained by a function call. The function call in turn returns the information that is needed. In this rule there is a function call called weld_position(). This function when called the first time has the state of the global variable (weld_pos_state) unknown since its slot has not yet been filled. It executes the necessary conditions in the function and returns the variable wpd. This contains the position of the weld (to be

either in position (flat or horizontal) or out of position (vertical or overhead). There exists another condition in the same rule that calls the function metal_transfer(). This in turn again calls weld_position() to determine the position of the variable. But it is wasteful to run the function again. This is taken care of by the variable weld_pos_state. Every time a function is run it is first checked with its own variable value. If it has already been determined the function is not run and returns its original value. If the function had not been evaluated, then the conditions are satisfied and the appropriate result is sent back to the next level. After all these calls have been successfully returned the control returns to one higher level, where it satisfies the overall goal and the system goes to the next parameter and establishes that goal. However, in the event the if portion of the first rule electrode_grade() failed, the else portion calls the next rule (not shown here) which is the function electrode_grade2(). This continues until all the rule base has been searched in that frame, and if it still does not resolve the goal rule, then input from the user is obtained to provide the electrode type. This is stored in a global variable for future use. This program simulates the main actions of PCPLUS. Although the system is primitive, it captures the intent of all that was sought at the start of the implementation stage.

A Sample Session

The system was created using GNU C++ [Tiemann 1990] on a DEC workstation with the IDEAS solid modeling system running as a child process. Although it is not a full fledged system working inside IDEAS (as it was not possible to work based on the internals of IDEAS), it does not limit the usefulness interms of attaching weld attributes to a CSG tree and extraction of the same. Further, the work exemplifies the

need for such a representation scheme and shows how mapping can be accomplished for an application purpose.

The WAGRAPH attachment involves the creation and location of geometry elements followed by the interactive attachment of the weld property values. The overall system is structured as a menu driven system and structured in levels of hierarchy (refer Figures 5.8 and 5.9). Creating the nominal geometry is the first step in the overall process. Once in the individual feature menu, the designer may create, delete, modify, or list the properties of the nominal feature. The system allows the access to the menu structure in any order as long as it makes valid sense. For example, a compound object cannot be created when there are no primitives (leaves) in the object collection or a cylinder cannot be on top of a cone for the purposes of welding. As the creation progresses, the variational information, or the WAGRAPH information can be added by invoking that part of the main menu. The WAGRAPH structure is invoked and the user specifies one of the solid nodes on the CSG tree as the current solid where the attribute information is to be added based on the display of the object list. The validity of the node is checked before attempting to attach the WAGRAPH.

The following are to be assigned: (a) the participating weld faces (b) the appropriate weld edges (c) the start of the weld (d) the end of the weld (e) the welding frames. The ray casting technique available in the solid modeler can be used. Cast-ray allows one to cast a ray along a given vector from an origin identified by the user. The program then reports back the surfaces that it intersects, and the XYZ coordinate point on the surface it intersects. Then from the provided list (if there are multiple surfaces) the appropriate surface can be picked. Likewise, the cursor is used to select entities and coordinates so that the information concerning the dimension

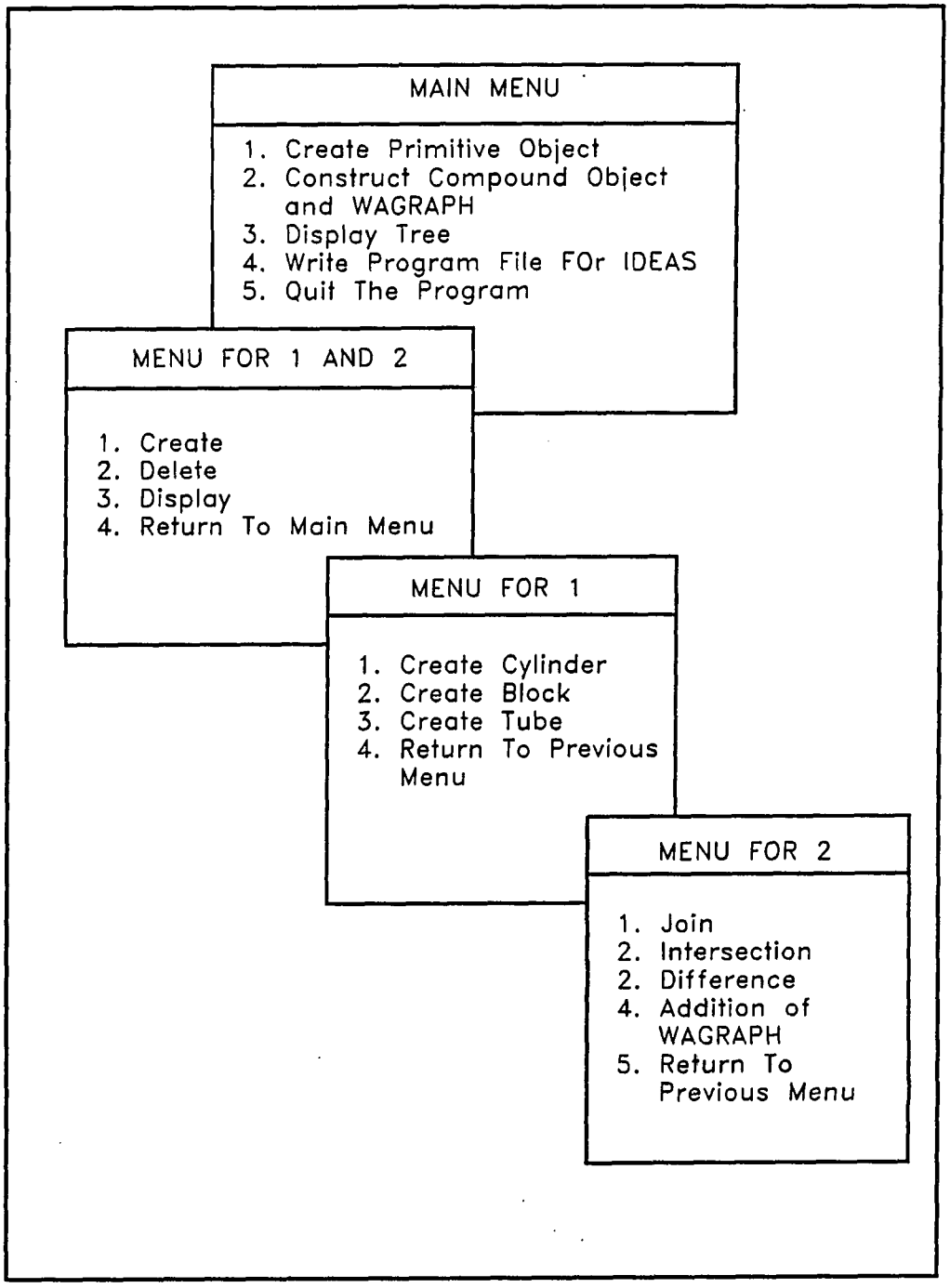


Figure 5.8: Menu structure for the system – Part 1

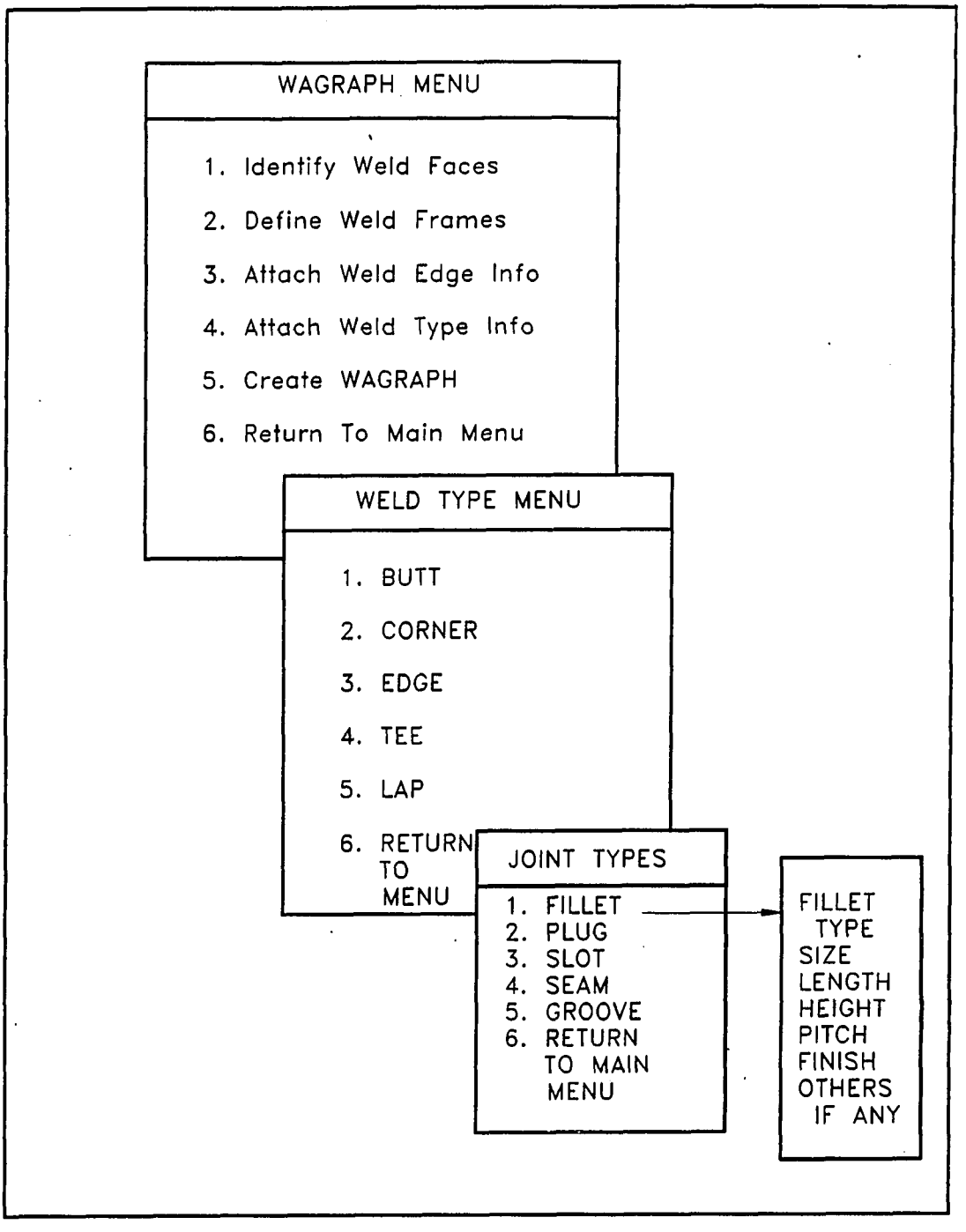


Figure 5.9: Menu structure for the system – Part 2

and topology can be obtained and necessary type 2 parameters evaluated. Since the goal was not the writing of a full-fledged graphics package, the user was made to interactively input the details (a drawback of the system). After selecting the appropriate face, edge, and frame information, the user through menu-based interaction assigns attributes of the weld to the parent node based on the WAGRAPH, WTYPE, WJOINT class structures explained in Section 3. As this has been created, the user is allowed to delete, modify, or display any of the values created.

When a feature is deleted, or attributes of the WAGRAPH are deleted, all corresponding pointers are removed. This means that if a child leaf needs to be deleted, then the compound object is deleted which in turn deletes the WAGRAPH that is attached. Also if the weld edge does not any longer exist the entire set of pointers to the WFACES and WFRAMES are deleted and made invalid. Deleting just a WAGRAPH results in that structure alone being termed invalid. The tree is then archived in a data structure format (as explained before). At any point, the user can continue to create objects and expand the tree. The addition of all this does not automatically invoke the WAGRAPH but does so only when the user replies to the question "Does a weld exist in the compound object just created". If the answer is YES, the user is allowed to enter values (with default values always provided to start with). Complicated editing that is possible is not taken care of. Sophisticated consistency and validity checkers need to be developed if this system is to be incorporated in a commercial system. The structure is saved as a binary tree with the WAGRAPH class instance being attached to the nodes. A text language type storing and a parser to back translate would be ideal.

Validation Of The System

Any system that is created has to be evaluated. The validation of typical knowledge-based systems is achieved by means of consultation of the expert system and comparing the results with the experts answers. The match between the two is seen as the indicator of good performance. The system is also given to the welding engineer, so that various test cases can be tried. The importance of such a validation is to make sure the knowledge engineer obtained the data properly and to report any bugs in the system created. This is similar to field tests of software. However, it was clear at the end of the industrial survey that only public knowledge was acquired and this was is based on tables, and handbooks. This resulted in the creation of a system based on available knowledge that fitted the IF-THEN rule format. The answers the system provided were therefore satisfactory. However, the intention in the present work was to validate the conclusions arrived at based on the experience in building the system. Hence at the end of this research, a set of recommendations were sent to the welding engineers who had cooperated in providing the data initially. The actual answers of the experts are provided in Appendix D. The recommendations sent to them were:

1. To have an on-line feedback based on a vision system to check the weld for spatter, porosity, lack of penetration, lack of fusion, undercutting, cracking, burnthrough etc. and correct the rule database as each experimentation is carried out. This has already been attempted by researchers independently but changing of rules in the knowledge base and subsequent correction is the central problem. Issues: Can this feedback be used for generalizations?

2. Conduct an extensive study on the "Effects of geometric shapes on welding process parameters" with the help of a welding engineer (full time), an integrated welding robot and positioner system; assigning weld parameters based on an existing knowledge-based system and monitor what the welding engineer does, to abstract private information. Issues: (a) will it provide results significantly different to warrant such an action? (b) will it be considered scientific? (c) what will it achieve in terms of generalization (d) when and where will it end?

3. Develop analytical models for each of the welding processes. Issues: Are we anywhere near developing welding process specification as a science-based tool

The experts agreed that something should be done to make the process specification process more science-based rather than experience-based. However, there was no consensus concerning the best direction of progress (Refer Appendix D for their answers). Many of the welding engineers were skeptical about the outcome of any of the three methods outlined but agreed on the need for a start at some point. This conclusion further indicates, that a greater effort is needed to study the entire problem of process specification of welding process parameters although theoretical work (this research) will doubtless help to solve some of the problems.

Summary

This Chapter has outlined the implementation of the integrated structure for welding feature and attribute representation and extraction. The novel use of an object-oriented programming paradigm for this application was highlighted and the system was developed using C++. The total number of lines of C++ code developed

for the system is around 4000. In addition, the knowledge-based system was also developed as a shell using PCPLUS. The representation of the weld attributes at the design stage allowed the automatic generation of robot weld process schedules. A sample session to highlight the steps and discuss shortcomings was described. It is believed that this system is a good starting point for the representation of secondary feature information in solid models. A richer and higher level representation (abstraction) of the elements involved in a robot workcell environment is required to attempt task level programming of robots. But this research presents itself as a good intermediate step between manual robot programming and totally automated task level robotic arc welding programming systems.

CHAPTER 6. TRAJECTORY PLANNING OF THE INTEGRATED ROBOT AND POSITIONER

Given a desired weld seam trajectory from the CAD database and the appropriate process knowledge, a systematic set of inverse solutions must be provided to instruct the robot to weld the trajectory. Although the robot is the principal member in a robotic workcell, other members need to be represented and used in the decision-making process. The major phases in any off-line robot programming system are:

1. Kinematic solver/trajectory planner
2. Interference detection/Collision avoidance
3. Real world feedback to correct for inaccuracies by sensor agents such as vision, seam tracking
4. Detect anamolies and correct the high order process plans

Different aspects of these problems are being attempted by various researchers. The first issue alone is addressed and a new approach is provided in this Chapter. It is stressed again that this dissertation conceives only the off-line development phase wherein a plan will be produced based on apriori information available with the aid of the consistent representation scheme. It is understood that the on-line phase is

an important factor in the total automation of robot programming but providing solutions for that phase of the research was not the intent of this work.

The need for an integrated welding robot and positioner in an arc welding workcell and methods currently adopted were highlighted in Chapter 2. The novel "fixed path modified continuation method" adopted to obtain the joint displacement and velocity histories continuously along the weld path is outlined in this Chapter. The methodology is implemented for a five axis GE P50 robot and a two axis positioner (similar to ESAB350). This Chapter is divided into six main Sections. The first Section describes the overall methodology. The modified continuation method is described in Section 2. The equations for the GE P50 robot and the positioner that describe the forward and inverse kinematics are outlined in Section 3. Section 4 provides the equations involved in the trajectory planning of the robot and the method whereby how the first order non-linear differential equations may be obtained. The fifth Section describes three examples that highlight the method developed and the final Section summarizes the Chapter.

Overall Methodology

A robot with five degrees of freedom and a two-axis positioner needs to be controlled simultaneously in real time. In commercial systems the robot controller controls both the robot and positioner. However, the tool center point is normally not related to the movement of the robot. This results in the necessity for teaching a number of intermediate points even for a simple straight line (see Figure 6.1) to obtain a coordinated motion of the robot and positioner.

This problem of coordinating the robot and the positioner by using the positioner

as a leader and the robot as a follower is now addressed. The traditional method of obtaining inverse kinematic solution at intermediate points and interpolating in between knot points is not attempted. The positioner is first oriented such that a weld gun oriented normal to the weld joint will be in a vertical line. Subsequently the robot trajectory is planned. The solution all along the trajectory is obtained knowing the start solution and the target system. The steps involved in determining the joint angles, velocities, and acceleration of the robot and positioner along the weld joint are:

1. Derive the kinematic equations for the positioner and the robot
2. Solve for the joint angles of the positioner such that the normal of the weld is in the global z direction using the continuation method. Obtain the position and orientation of all points on the weld in terms of the path parameter p, with respect to the base of the positioner
3. Transform this position and orientation to obtain the orientation of the weld joint with respect to the base frame of the robot
4. Obtain the inverse kinematic solution of the robot at the start point using the closed form solution derived in step 1
5. Knowing the target system as a function of the path parameter p (at p=1), track the path (from 0 to 1) using the continuation method to obtain joint angles, velocity, and acceleration
6. Verify the correctness of the solution by simulating the robot and positioner motion using a solid modeling system and a simulation software

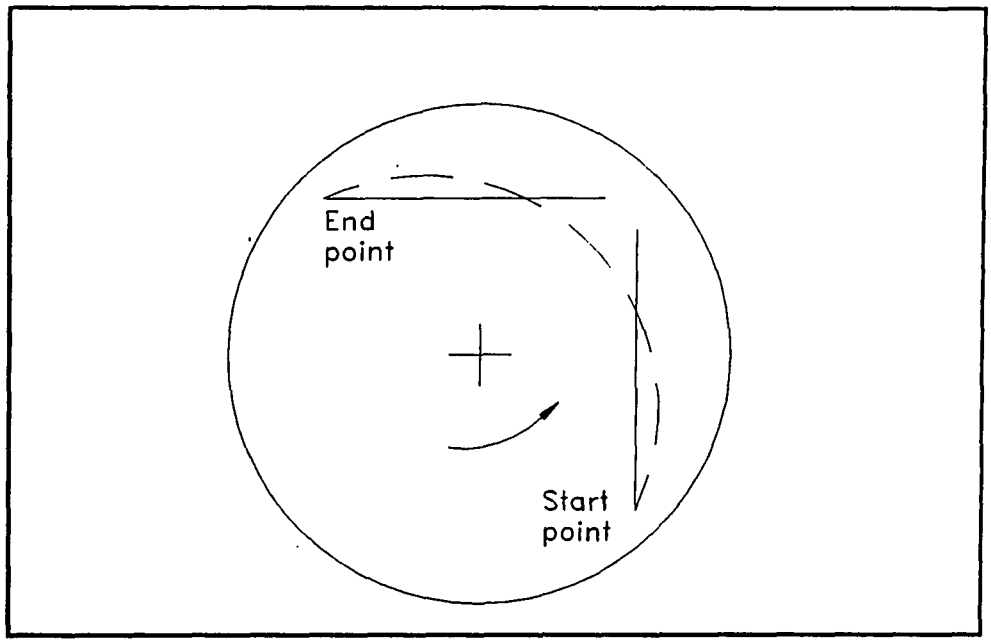


Figure 6.1: Example showing coordinated motion

The general problem of producing mechanical parts divides into two, according to whether a curve or surface is to be produced. Curves normally require operations such as flame cutting and continuous welding while surfaces are involved in machining. Since the primary goal is to use a robot to weld, the curve that represents the weld and its corresponding coordinates should be obtained. Three examples – a straight line weld, a circular weld, and a quarter circle in the XZ plane – are provided to highlight the method. These examples are considered to show one time movement of the positioner for the straight line and circular weld and a continuous, integrated movement of the robot and positioner for the quarter circle weld.

Fixed Path Modified Continuation Method

Continuation methods in the last few years have received considerable attention in applications to kinematic analysis and synthesis problems [Subbian and Flugrad 1991b]. Jo and Haug [1988] carried out the work space analysis of two degree-of-freedom robot arms and slider-crank mechanisms utilizing continuation methods. Tsai and Morgan [1985] and Wampler and Morgan [1989] succeeded in determining the inverse kinematic solutions for five and six degree-of-freedom manipulators using this technique. In this thesis, application of a modified continuation method to solve the trajectory planning problem of an integrated robot and positioner is proposed.

Continuation methods can be used to solve a system of n equations in either n or $(n+1)$ unknowns. To solve a system of n equations in n unknowns, say $\mathbf{f}(z) = 0$, a simple system is assumed to be a start system, say $\mathbf{g}(z) = 0$. This start system must be of the same degree as the original system and easy to solve. Homotopy functions can then be written by combining the two systems of equations as follows

$$\mathbf{h}(z,t) = \mathbf{f}(z)t + \mathbf{g}(z)(1-t) = 0$$

where t is the homotopy parameter and $\mathbf{h}$ the homotopy function. Depending on the choice of $\mathbf{g}(z)$ and the manner in which it is combined with $\mathbf{f}(z)$, the homotopy can be described as a coefficient homotopy, a parameter homotopy, or a secant homotopy. When $t = 0$, the homotopy function reduces to the start system; when $t = 1$ it becomes the original system of equations to be solved. Therefore, by increasing t from 0 to 1 and simultaneously tracking the values of the z variables, the original system $\mathbf{f}(z) = 0$ is solved. The z variables are tracked by integrating a set of first order ordinary differential equations with respect to the homotopy parameter t. The

solutions for the start system are used as initial conditions for the integration.

When solving a system of n equations in $(n+1)$ unknowns, solution curves are obtained, rather than a finite solution set. The procedure involved is a two step process. First, at least one point on each of the solution curves is determined. Let $\mathbf{e}(x) = 0$ be the system of n equations in $(n+1)$ unknowns. The procedure described by Morgan [1981] utilizes an extended Jacobian matrix of $\mathbf{e}(x) = 0$ to determine an $(n+1)$th equation. Hence, a set of $(n+1)$ equations in $(n+1)$ unknowns is solved using continuation to find the finite solution set of interest. Obtaining the $(n+1)$th equation analytically, can be difficult for a complicated system of equations; therefore an alternative approach is used here.

The system of equations under consideration has a trajectory path variable as the $(n+1)$th variable, which is 0 at the initial point and 1 at the terminal point. The path traversed by the robot between these two points is specified. The remaining n variables are the joint angles, whose values are to track the prescribed path. By setting $x_{n+1} = 0$ then, the inverse kinematic solutions for the robot at the initial point can be determined. These inverse kinematic solutions can either be calculated by closed-form solutions, or by using a continuation method [Tsai and Morgan, 1985, Wampler and Morgan, 1989].

The finite solutions obtained by the above procedures give points on the solution curves. The second step would be to trace the solution curves from the initial points, obtained above, by integrating a set of n first order ordinary differential equations of the n variables with respect to x_{n+1}. The differential equations are obtained from the extended Jacobian matrix of the given system of equations ($\mathbf{e}(x) = 0$). A detailed description of the procedures outlined here is provided by Morgan [1981, 1987].

Traditionally, the continuation method is used to solve a system of polynomial equations. The solutions represent the path taken by the system of equations where the start solution is assumed and the target system is known. This path can be tracked using various methods falling under the general umbrella of path-tracking numerical methods. However, in our case it is proposed that a different approach be used in which the path to be tracked is known, the start solution is known (beginning of path), and the target system is also known (end of path). The solution to the variables provide the joint displacements of the robot as it tracks the given path. Although we force the path upfront, we still have the problem of solving for a given set of variables knowing (1) a set of polynomial equations in cosines and sines of the joint angles, (2) the known inverse kinematic solution as the start solution and (3) the target system. The solution is required continuously along the path. In the strict sense of the continuation method the path cannot be fixed. Nevertheless, in the present case, the solution along the path is needed continuously and is obtained by this approach. Hence this method may be termed as the "Fixed Path Modified Continuation Method"

Development Of Equations

Process robots are usually revolute joint robots and typically have five degrees of freedom i.e., five moving links. The welding torch is attached to the wrist and link 6 is at the tip of the torch. A positioner in a welding workcell is typically two axis in nature providing the tilt and rotation of the worktable. Every two neighboring links are connected by a joint. Figure 6.2 is a generalized link $(i-1)$ paired at a joint i with another link, link i. Every joint is associated with a coordinate frame. The link

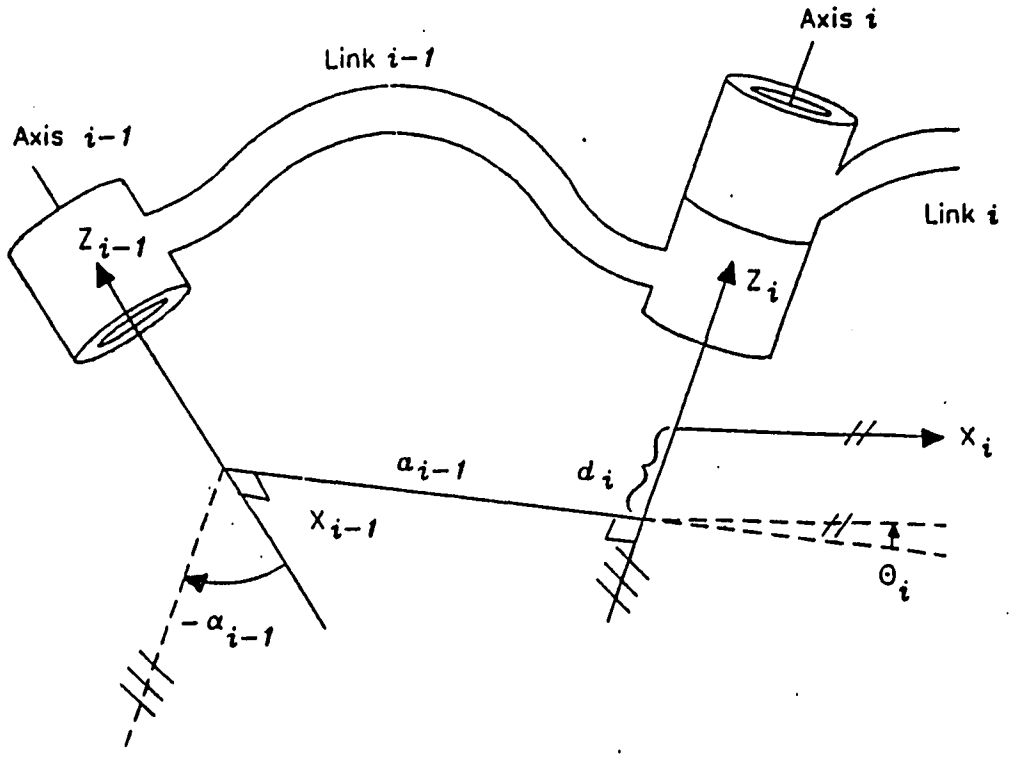


Figure 6.2: Definition of link parameters

parameters following Craig's [1986] convention are:

a_i: the distance from Z_i to Z_{i+1} measured along X_i

α_i: the angle between Z_i and Z_{i+1} measured about X_i

d_i: the distance from X_{i-1} to X_i measured along Z_i and

θ_i: the angle between X_{i-1} and X_i measured about Z_i

The homogeneous transformation matrix relating frames i and $i-1$ is given by:

$$ {}^{i-1}_{i}T = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.1) $$

where the notations C and S indicate *cos* and *sin*, respectively. Figure 6.3 provides the link parameters of the robot and the positioner.

Consider the nature of the subspace of the GE P50 robot. It is found that the subspace can be described by giving a constraint that the direction of the tool Z_T must lie in the plane of the arm. This is the plane containing axis 1 and the point where axes 4 and 5 intersect [Craig 1987]. This shows that all goal orientations are not possible with a five degree of freedom robot. However, in general this does not pose any problem.

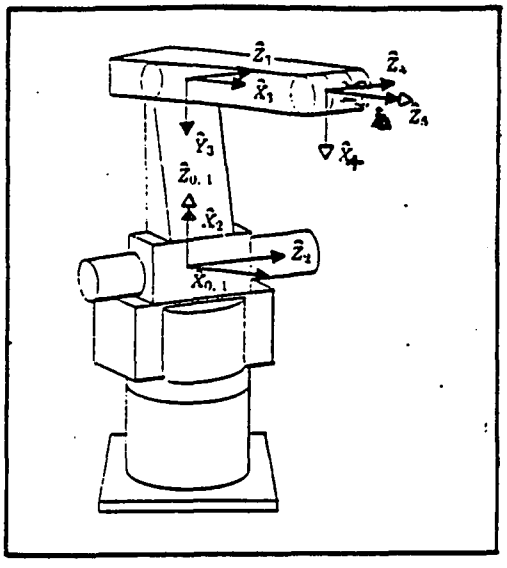
The GEP50 robot

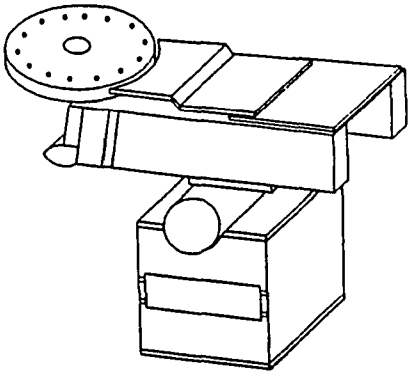
The kinematics of the manipulator can be evaluated by using equation 6.1 for each link, progressing from frame 0 to frame 5. Using the following transformation matrix from frame 5 to frame 6

$$ {}^{5}_{6}T = \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} & 0 \\ 0 & 1 & 0 & 0 \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} & 16.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2} $$

and by combining the results with Eq. 6.2 it is found that ${}^{0}_{6}T = {}^{0}_{1}T{}^{1}_{2}T{}^{2}_{3}T{}^{3}_{4}T{}^{4}_{5}T{}^{5}_{6}T$. This then describes any position in the final frame with respect to the robot's base. In the present case the final frame is the welding tip which is at a specified distance from the weld joint to account for the arc length as specified by the GMAW process. This position is given by:

$$ P_x = 16.0C\theta_1 S\theta_{234} + 23.5C\theta_1 C\theta_2 + 33.5C\theta_1 C\theta_{23} $$





GE P50 ROBOT

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	-90^o	0	0	θ_2
3	0	L2	0	θ_3
4	0	L3	0	θ_4
5	90^o	0	0	θ_5

ESAB 350 POSITIONER

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	90^o	24.0	-	θ_1
2	-90^o	0	0	θ_2

Figure 6.3: Configuration and link parameters

$$
\begin{aligned}
P_y &= 16.0 S\theta_1 S\theta_{234} + 23.5 S\theta_1 C\theta_2 + 33.5 S\theta_1 C\theta_{23} \\
P_z &= 16.0 C\theta_{234} - 23.5 S\theta_2 - 33.5 S\theta_{23}
\end{aligned} \tag{6.3}
$$

Inverse kinematics is the process of solving the problem of finding the required joint angles to place the tool frame (the welding torch tip) relative to the base frame given the position and orientation of the weld. The problem of solving the kinematic equations for any robot is non-linear in nature. However, closed form solutions exist for the GE P50 robot due to the nature of construction and the various joint angles are given by:

$$
\begin{aligned}
\theta_1 &= Atan2(P_y, P_x) \\
\theta_5 &= Atan2(l_y C\theta_1 - l_x S\theta_1, m_y C\theta_1 - m_x S\theta_1) \\
\theta_{234} &= Atan2(n_x C\theta_1 + n_y S\theta_1, n_z) \\
\theta_3 &= Atan2(\sqrt{1 - C^2\theta_3}, C\theta_3) \\
where, C\theta_3 &= (P_x^2 + P_y^2 + P_z^2 - l_2^2 - l_3^2)/2 l_2 l_3 \\
\theta_2 &= -Atan2(P_x, \sqrt{P_x^2 + p_y^2}) - Atan2(l_3 S\theta_3, l_2 + l_3 C\theta_3) \\
\theta_4 &= \theta_{234} - \theta_2 - \theta_3
\end{aligned} \tag{6.4}
$$

For the GEP50 manipulator a second solution would violate joint limits and so is not considered. In this case, let

$$
{}^0_5T = \begin{bmatrix} l_x & m_x & n_x & P_x \\ l_y & m_y & n_y & P_y \\ l_z & m_z & n_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5}
$$

which is typically given by knowing the location and orientation of the weld.

Two axis positioner

In general, a positioner used in an arc welding workcell is either two-axis or t hree-axis in nature. The main requirement of this positioner is to orient the weld joints in the object in any given direction. Given the position of the weld, it can be oriented in any given direction if there are three intersecting axis. But often it is not possible to orient in arbitrary directions, due to the interferences in the workpiece environment, obstacles, or other constraints dictated by the process. This will then normally lead to the use of some sort of search for an acceptable solution. The traditional Roll-Pitch-Yaw angle formulation ican be used to solve for the inverse kinematic problem – that of finding the angles for a given orientation. If the positioner is not of this type, the transformation for the 3-revolute case can be obtained by using equation 1. The overall transformation matrix is given by ${}^0_3T = {}^0_1T{}^1_2T{}^2_3T$. Since the final Z vector is required to be in the global Z direction, the third column of the overall transformation matrix is made equal to $[0,0,1]^T$. This results in the following three equations:

$$\begin{aligned}
n_x &= -\mu_2(-C\theta_1 S\theta_2 - S\theta_1 C\theta_2\lambda_1) + S\theta_1\mu_1\lambda_2 \\
n_y &= -\mu_2(-S\theta_1 S\theta_2\lambda_0 + C\theta_1 C\theta_2\lambda_0\lambda_1 - C\theta_2\mu_0\mu_1) \\
&\quad +\lambda_2(-C\theta_1\lambda_0\mu_1 - \mu_0\lambda_1) \\
n_z &= -\mu_2(-S\theta_1 S\theta_2\mu_0 + C\theta_1 C\theta_2\mu_0\lambda_1 + C\theta_2\lambda_0\mu_1) \\
&\quad +\lambda_2(-C\theta_1\mu_0\mu_1 + \lambda_0\lambda_1) \qquad (6.6)
\end{aligned}$$

where $\mu_i = Sin\alpha_i$ and $\lambda_i = Cos\alpha_i$. Further, n_x, n_y, n_z are 0, 0, 1 respectively. These three equations have three unknowns and can be solved using CONSOL or o

ther continuation routines as outlined in Tsai and Morgan [1985]. In the present case a two-intersecting axes positioner is used and the solution can be applied to other similar positioners also.

The positioner has two intersecting axes which provide tilt and rotation. The direct kinematic case–that of solving for the orientation and the position of the end frame on the positioner, given the position of the axis is given by:

$$ {}^{0}_{2}T = \begin{bmatrix} C\theta_1 C\theta_2 & -C\theta_1 S\theta_2 & -S\theta_1 & 24.0 \\ S\theta_2 & C\theta_2 & 0 & 0 \\ S\theta_1 C\theta_2 & -S\theta_1 S\theta_2 & C\theta_1 & 37.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.7} $$

The third frame is on the weld such that the Z axis points in the normal direction from the joint, the X axis along the weld joint and the Y axis completes the right hand rule and thus points in the weaving direction of the weld movement. Assuming the description of the goal frame is known i.e., the weld relative to the station frame, and is of the form:

$$ {}^{2}_{3}T = \begin{bmatrix} l_{xp} & m_{xp} & n_{xp} & P_{xp} \\ l_{yp} & m_{yp} & n_{yp} & P_{yp} \\ l_{zp} & m_{zp} & n_{zp} & P_{zp} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.8} $$

then, in order to solve for the condition such that the Z axis of the weld frame is in the global Z direction, the following has to be satisfied:

$$ \begin{aligned} C\theta_1 C\theta_2 n_{xp} - C\theta_1 S\theta_2 n_{yp} - S\theta_1 n_{zp} &= 0 \\ S\theta_2 n_{xp} + C\theta_2 n_{yp} &= 0 \\ S\theta_1 C\theta_2 n_{xp} - S\theta_1 S\theta_2 n_{yp} + C\theta_1 n_{zp} &= 1 \end{aligned} \tag{6.9} $$

The values of the angles required to obtain a particular orientation of the weld joint can therefore be calculated and this will be used in the trajectory planning.

Trajectory Planning Of The Integrated Robot And Positioner

Trajectory planning is the task of designing a path to move the manipulator from an initial position to some desired final position as determined by the weld seam in the part that needs to be welded. This motion involves a change in the orientation and position of the tool relative to the positioner station. Normally the motion is specified by assigning a sequence of points between the initial and final points which are termed knot points. Each of these knot points is a frame which specifies both the position and orientation of the tool relative to the station. Between the points the motion of the manipulator is defined to be a smooth function. To guarantee this, constraints on the spatial and temporal qualities of the path are specified between the knot points. Joint space schemes achieve the desired position and orientation at the knot points. In between the knot points the shape of the path is simple in joint space but complex when described in Cartesian paths [Craig 1986].

The trajectory planning outlined in this thesis is accomplished using the continuation method to solve a system of n equations in $(n + 1)$ unknowns. For such a system, the solution set will be a family of curves. The objective is to include a variable associated with the trajectory of the robot in Cartesian space in solving for the inverse kinematics. A normalized path variable, p, (where $p = 0$ at the start of the trajectory, and $p = 1$ at the end) was used.

In order to arrive at these sets of equations it is necessary to obtain the weld position information as a function of the path variable after the desired positioner

orientation has been achieved. Depending on the weld position and to ensure best possible welding conditions, the positioner has to be either oriented once, or rotated continuously. For this, the inverse solution of obtaining joint angles given the final location of the weld with respect to the center of the positioner table is to be attempted.

Using equation 6.9 the angles of the positioner can be obtained such that the normal of the weld is in the global Z direction. This gives:

$$\begin{aligned}
\theta_2 &= Atan2(-n_{yp}, n_{xp}) \\
or &= Atan2(n_{yp}, -n_{xp}) \\
\theta_1 &= Atan2(X, Y) \\
where, X &= (C\theta_2 n_{xp} - S\theta_2 n_{yp})/((C\theta_2 n_{xp} - S\theta_2 n_{yp})^2 + n_{zp}^2) \\
Y &= n_{zp}/(C\theta_2 n_{xp} - S\theta_2 n_{yp})((C\theta_2 n_{xp} - S\theta_2 n_{yp})^2 + n_{zp}^2) \quad (6.10)
\end{aligned}$$

Having found the joint angles, these values can be substituted in the forward transformation to obtain the position as a function of the path parameter, p and is given by:

$$\begin{aligned}
{}^0_3P_x &= C\theta_1 C\theta_2 P_{xp}(p) - C\theta_1 S\theta_2 P_{yp}(p) - S\theta_1 P_{zp}(p) + 24.0 \quad (6.11) \\
{}^0_3P_y &= S\theta_2 P_{xp}(p) + C\theta_2 P_{yp}(p) \quad (6.12) \\
{}^0_3P_z &= S\theta_1 C\theta_2 P_{xp}(p) - S\theta_1 S\theta_2 P_{yp}(p) + C\theta_1 P_{zp}(p) + 37.0 \quad (6.13)
\end{aligned}$$

This position is then transformed to the robot's base coordinate frame. It is necessary to arrive at the set of five non-linear equations that will be solved for planning the trajectory. For a five-revolute-joint manipulator, only the position of an axis in the hand can be freely specified. The axis of interest is the Z_6 axis. If the

sixth frame is located at the tip of the torch and the unit vector attached to the Z_6 axis then by definition:

$$P_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad and \quad U_6 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{6.14}$$

and since the position and orientation of the hand is already specified by A_{eq} it follows that

$$\begin{aligned} P_0 &= A_{eq}P_6 \\ U_0 &= A_{eq}P_6 \end{aligned} \tag{6.15}$$

The transformation of coordinates from P_6 to P_0 is given by:

$$\begin{aligned} P_0 &= {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T\,P_6 \\ U_0 &= {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T\,U_6 \end{aligned} \tag{6.16}$$

Premultiplying both sides of equation 6.16 by ${}^1_2T^{-1}\,{}^0_1T^{-1}$,

$$\begin{aligned} {}^1_2T^{-1}\,{}^0_1T^{-1}P_0 &= {}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T\,U_6 \\ {}^1_2T^{-1}\,{}^0_1T^{-1}U_0 &= {}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T\,U_6 \end{aligned} \tag{6.17}$$

This will result in two 3 x 1 matrix and equating the left hand side and the right hand side of the matrices the following six equations are obtained,

$$\begin{aligned} C\theta_1C\theta_2P_x + S\theta_1C\theta_2P_y - S\theta_2P_z &= l_4C\theta_3S\theta_4 + l_4S\theta_3C\theta_4 \\ &\quad + C\theta_3l_3 + l_2 \end{aligned} \tag{6.18}$$

$$-C\theta_1 S\theta_2 P_x - S\theta_1 S\theta_2 P_y - C\theta_2 P_z = l_4 S\theta_3 S\theta_4 - l_4 C\theta_3 C\theta_4 + S\theta_3 l_3 \quad (6.19)$$

$$-S\theta_1 P_x + C\theta_1 P_y = 0.0 \quad (6.20)$$

$$C\theta_1 C\theta_2 n_x + S\theta_1 C\theta_2 n_y - S\theta_2 n_z = (C\theta_3 C\theta_4 S\theta_5 + S\theta_3 S\theta_4 C\theta_5)/\sqrt{2} + (C\theta_3 S\theta_4 + S\theta_3 C\theta_4)/\sqrt{2} \quad (6.21)$$

$$-C\theta_1 S\theta_2 n_x - S\theta_1 S\theta_2 n_y - C\theta_2 n_z = (S\theta_3 C\theta_4 C\theta_5 + C\theta_3 S\theta_4 C\theta_5)/\sqrt{2} + (S\theta_3 S\theta_4 - C\theta_3 C\theta_4)/\sqrt{2} \quad (6.22)$$

$$-S\theta_1 n_x + C\theta_1 n_y = S\theta_5/\sqrt{2} \quad (6.23)$$

However, only two of the last three equations are independent, since they are related by the equation:

$$equation(21)^2 + equation(22)^2 + equation(23)^2 = 1 \quad (6.24)$$

Therefore there are five independent equations in five unknowns (θ_1 - θ_5). As stated earlier the solution can be obtained by continuation to obtain the inverse kinematic solution at the start point and this may be used to track the path. However, the closed form solution is used at the start point and then tracked using the modified continuation method to obtain joint angles all along the path. The trajectory is expressed as functions $P_x(p)$, $P_y(p)$ and $P_z(p)$ where, ($P_x(0)$, $P_y(0)$, $P_z(0)$) correspond to the starting point, and ($P_x(1)$, $P_y(1)$, $P_z(1)$) correspond to the end point of the trajectory.

To solve this system of five equations, the extended Jacobian matrix, DE, is formulated by evaluation of the partial derivatives of the functions with respect to the variables θ_1, θ_2, θ_3, θ_4, θ_5 and p. The resulting Jacobian is a 5 x 6 matrix and has the form:

$$DE = \begin{bmatrix} \frac{\partial E_1}{\partial \theta_1} & \frac{\partial E_1}{\partial \theta_2} & \frac{\partial E_1}{\partial \theta_3} & \frac{\partial E_1}{\partial \theta_4} & \frac{\partial E_1}{\partial \theta_5} & \frac{\partial E_1}{\partial p} \\ \frac{\partial E_2}{\partial \theta_1} & \frac{\partial E_2}{\partial \theta_2} & \frac{\partial E_2}{\partial \theta_3} & \frac{\partial E_2}{\partial \theta_4} & \frac{\partial E_2}{\partial \theta_5} & \frac{\partial E_2}{\partial p} \\ \frac{\partial E_3}{\partial \theta_1} & \frac{\partial E_3}{\partial \theta_2} & \frac{\partial E_3}{\partial \theta_3} & \frac{\partial E_3}{\partial \theta_4} & \frac{\partial E_3}{\partial \theta_5} & \frac{\partial E_3}{\partial p} \\ \frac{\partial E_4}{\partial \theta_1} & \frac{\partial E_4}{\partial \theta_2} & \frac{\partial E_4}{\partial \theta_3} & \frac{\partial E_4}{\partial \theta_4} & \frac{\partial E_4}{\partial \theta_5} & \frac{\partial E_4}{\partial p} \\ \frac{\partial E_5}{\partial \theta_1} & \frac{\partial E_5}{\partial \theta_2} & \frac{\partial E_5}{\partial \theta_3} & \frac{\partial E_5}{\partial \theta_4} & \frac{\partial E_5}{\partial \theta_5} & \frac{\partial E_5}{\partial p} \end{bmatrix} \tag{6.25}$$

This extended Jacobian matrix is used to determine the five first order differential equations of the the joint space variables with respect to the cartesian path variable. Thus,

$$d\theta_i/dp \;=\; (-1)^{i+1} det(DE_{[i]})/Den \tag{6.26}$$

where, $i = 1,2,3,4,5$, $DE_{[i]}$ is the Jacobian with the ith column deleted, and Den is the negative determinant of the Jacobian with the sixth column deleted. Cramer's rule is used to obtain the first order differential equations to show the concept. Other numerical routines can be used to speed calculation time. The first order differential equations thus obtained are integrated with the known solution at the starting point of the trajectory providing the initial conditions. The integration provides θ_1, θ_2, θ_3, θ_4, and θ_5 values as a function of the path variable p. The joint velocity histories are also obtained if the velocity of the manipulator tip along its trajectory, dp/dt, is specified. The formula $d\theta_i/dt = (d\theta_i/dp)(dp/dt)$ is used to determine the velocities of the joint variables.

Examples

Examples are provided to explain the method and how it works in the different modes that are normally encountered in robot-positioner arc welding workcells. Here it is assumed that the shape of the curve that needs to be welded is obtained from the CAD module which is explained elsewhere (Chapter 4). Also conditions required for welding are obtained from the knowledge-based module. These two modules provide the curve information and the welding tip velocity information. Based on this the kinematics module decide on the nature of the positioner movement and the subsequent orientation of the robot to reach the points. Typically a one time movement of the positioner or a continuous movement of the positioner is required in arc welding workcells. To illustrate these, three examples are provided. A straight line and a circular weld to show one time movement, and a quarter of a circle weld to show continuous and coordinated movement of the robot and positioner.

Example 1: A straight line weld

Trajectory planning was attempted for a straight line weld to highlight the method. The straight line weld under consideration was assumed to be in the XZ plane of the second coordinate frame of the positioner. As stated earlier, the third frame was placed at the start of the weld with the X axis being along the weld, the Z axis normal to the weld and the Y axis completing the right hand convention. The weld starts at (2,0,2) and ends at (2,0,4). In terms of the path parameter p, the P_x, P_y, P_z are 2, 0, 2 + 2p respectively. The ${}^{2}_{3}T$ matrix is also obtained which describes the position and orientation of the weld with respect to the center of the positioner. This matrix is given by:

$$ {}^{2}_{3}T = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 2+2p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.27} $$

For this problem, the positioner has to be oriented only once and using equation 6.10 θ_1 and θ_2 are obtained for the positioner such that the normal of the weld is in the global direction (-90^o and $+90^o$ respectively). Using equations 6.11, 6.12 and 6.13 ${}^{0}_{3}P_x, {}^{0}_{3}P_y, {}^{0}_{3}P_z$ become 26.0+2p, 2.0, 37.0 respectively where P ranges from 0 to 1.

The overall transformation matrix describing the position and orientation of the weld with respect to the positioner's base is given by:

$$ {}^{0}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 26.0+2p \\ 0 & 1 & 0 & 2.0 \\ 0 & 0 & 1 & 37.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.28} $$

Knowing that Z_6 should point down, and that X_6 should be in the direction of weld, and Y_6 to complete the right hand convention the sixth frame orientation becomes:

$$ Frame6 = \begin{bmatrix} -1 & 0 & 0 & 49.0-2p \\ 0 & 1 & 0 & -2.0 \\ 0 & 0 & -1 & 7.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.29} $$

In general any position not in the original plane of the arm can be reached only by turning the robot's base by an angle equal to $Atan2(P_y, P_x)$. This results in the defining an intermediate frame which is obtained by rotating about Z_6 by the same amount so that the given point is in the plane of the robot arm. After making the substitutions, ${}^0_6T$ for the robot becomes:

$$ {}^0_6T = \begin{bmatrix} C\theta_i C\theta_f & C\theta_i S\theta_f & S\theta_i & P_x \\ -S\theta_f & C\theta_f & 0 & P_y \\ -S\theta_i C\theta_f & -S\theta_i S\theta_f & C\theta_i & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.30} $$

where θ_i is the angle by which the tool's pointing direction and the global Z varies and θ_f is the angle given by $Atan2(P_y, P_x)$. Since the tool frame needs to be in the global -Z direction, θ_i is 180 degrees.

The initial solution at p=0 was calculated using the closed form inverse kinematic solution given by equation 6.4. Using this and the functions $P_x(p)$, $P_y(p)$ and $P_z(p)$ for the specified trajectory, the extended Jacobian was evaluated as outlined in the previous Section. A computer program was developed to generate the first order differential equations. This was then numerically integrated using the Adam-Moulton method and the GEAR method to determine the joint angles, velocities all along the trajectory.

Plots of the resultant joint angles and velocities all along the path parameter are given in Figures 6.4 and 6.5. The joint angle information of θ_5 is zero all along the path and hence has not been plotted. The joint velocities plotted were obtained assuming that the manipulator followed its trajectory at a constant unit velocity,

i.e., $dp/dt = 1$. In the actual situation, the knowledge-based system will specify the velocity of the welding tip and the actual velocities would be a scalar multiple of these values. On driving the servomotors attached to the joints using the calculated results the robot would trace a circular path with a constant velocity. The trajectory was checked by carrying out the forward kinematic analysis of the robot and using a simulation model.

Example 2: A circular weld

The circular weld was assumed to be inclined by 45 degrees to the horizontal plane of the positioner worktable. This would result in a one-time movement of the positioner to bring the weld trajectory to the horizontal plane. The circular trajectory had its center at 0,0,5 with respect to the table center and a radius of 5 units. The parametric representation of the curve is given by:

$$X = 5Cos(2\pi p)/\sqrt{2} - 5/\sqrt{2}$$

$$Y = 5Sin(2\pi p)$$

$$Z = 5Cos(2\pi p)/\sqrt{2}$$

where p ranges from 0 to 1 indicating the start and finish of the circle. The third frame is placed at the start of the weld with the Z direction being the normal to the weld, the X axis being the tangent to the weld and the Y axis completing the right hand convention. The transformation matrix that describes the weld with respect to the center of the positioner is given by:

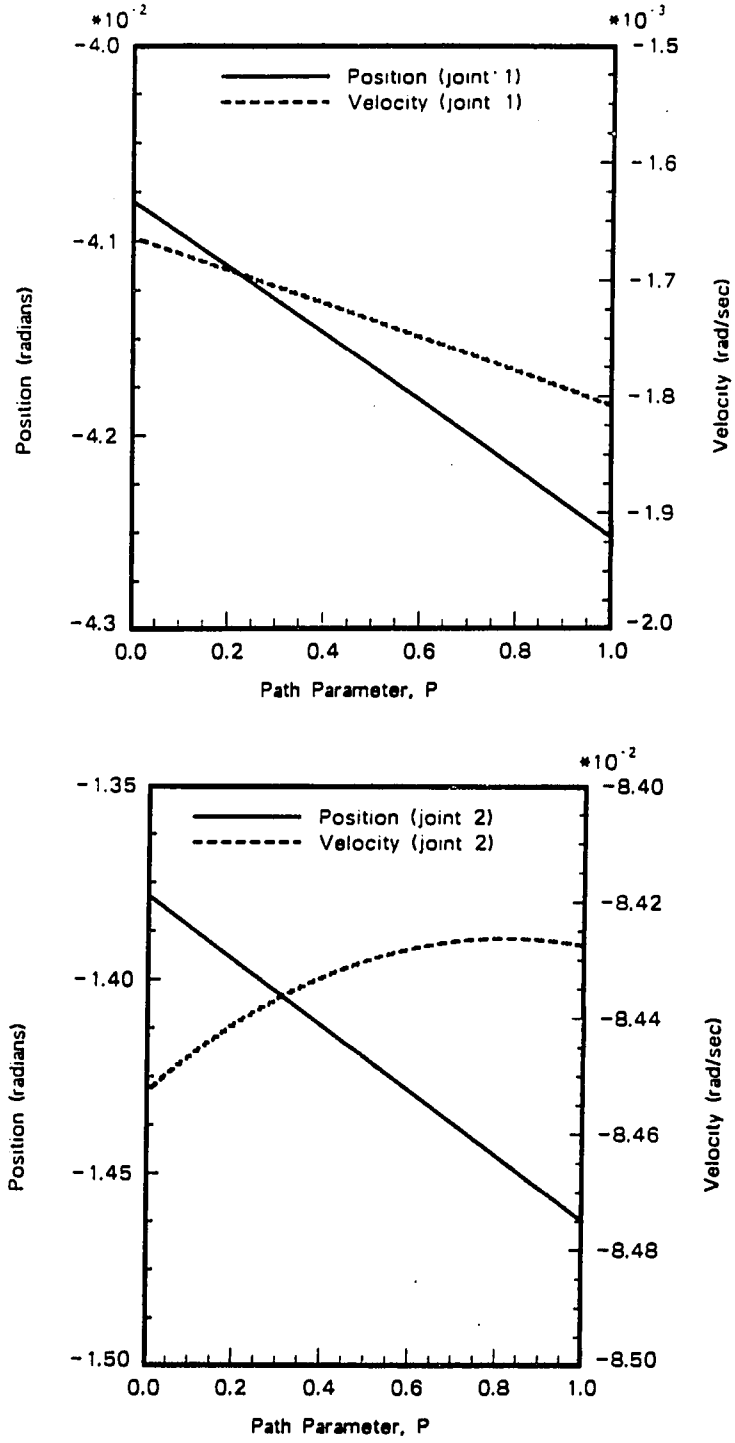


Figure 6.4: Example 1: Joint 1, 2 displacement and velocity history of robot

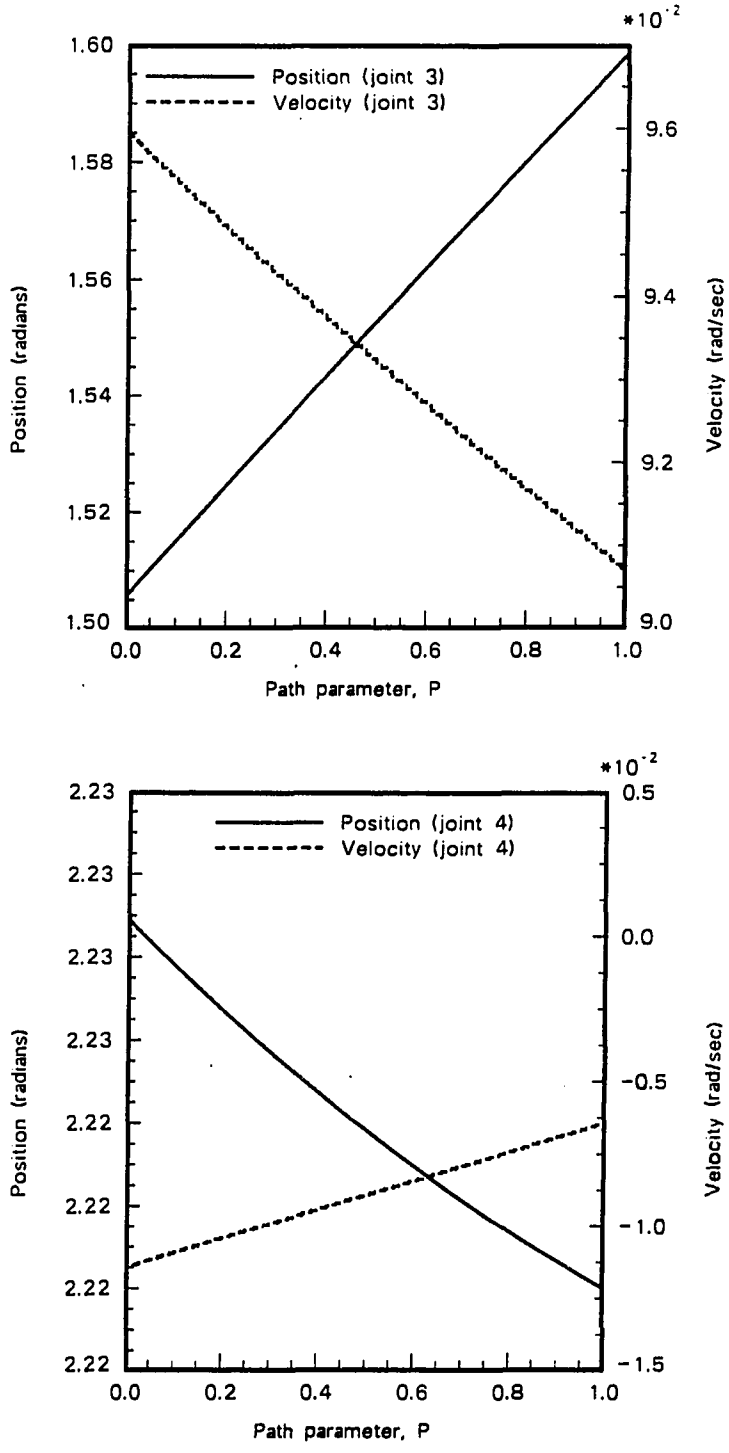


Figure 6.5: Example 1: Joint 3, 4 displacement and velocity history of robot

$$
{}^{2}_{3}T = \begin{bmatrix} Cos(2\pi p)/\sqrt{2} & -Sin(2\pi p)/\sqrt{2} & -1/\sqrt{2} & 5Cos(2\pi p)/\sqrt{2} - 5/\sqrt{2} \\ Sin(2\pi p) & Cos(2\pi p) & 0 & 5Sin(2\pi p) \\ Cos(2\pi p)/\sqrt{2} & -Sin(2\pi p)/\sqrt{2} & 1/\sqrt{2} & 5Cos(2\pi p)/\sqrt{2} + 5/\sqrt{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.31)
$$

In order for the normal to be in the global Z direction, the positioner has to be rotated only once. Using equation 6.10 θ_1 becomes -45 and θ_2 becomes 0 degrees. Substituting this result in equations 6.11 through 6.13 the position of the weld with respect to the positioners base frame is obtained. After proper transformations the orientation and position of the weld with respect to the robots base is given by:

$$
{}^{0}_{6}T = \begin{bmatrix} -Cos(2\pi p) & -Sin(2\pi p) & 0 & 75 - (5Cos(2\pi p) + 24) \\ -Sin(2\pi p) & Cos(2\pi p) & 0 & -5Sin(2\pi p) \\ 0 & 0 & -1 & 12.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.32)
$$

This matrix is used to obtain the start solution using the closed form inverse solutions as given by equation 6.4. These are then used by the continuation method program to obtain inverse kinematic solution all along the path. The joint displacement and velocity histories are shown in Figures 6.6 and 6.7.

Example 3: A quarter circle in the XZ plane

This example is used to highlight the coordinated movement of the robot and positioner. A quarter of a circle is used as the generating curve for the weld. Since the weld is assumed to be in the XZ plane, the normal of the curve as the weld torch

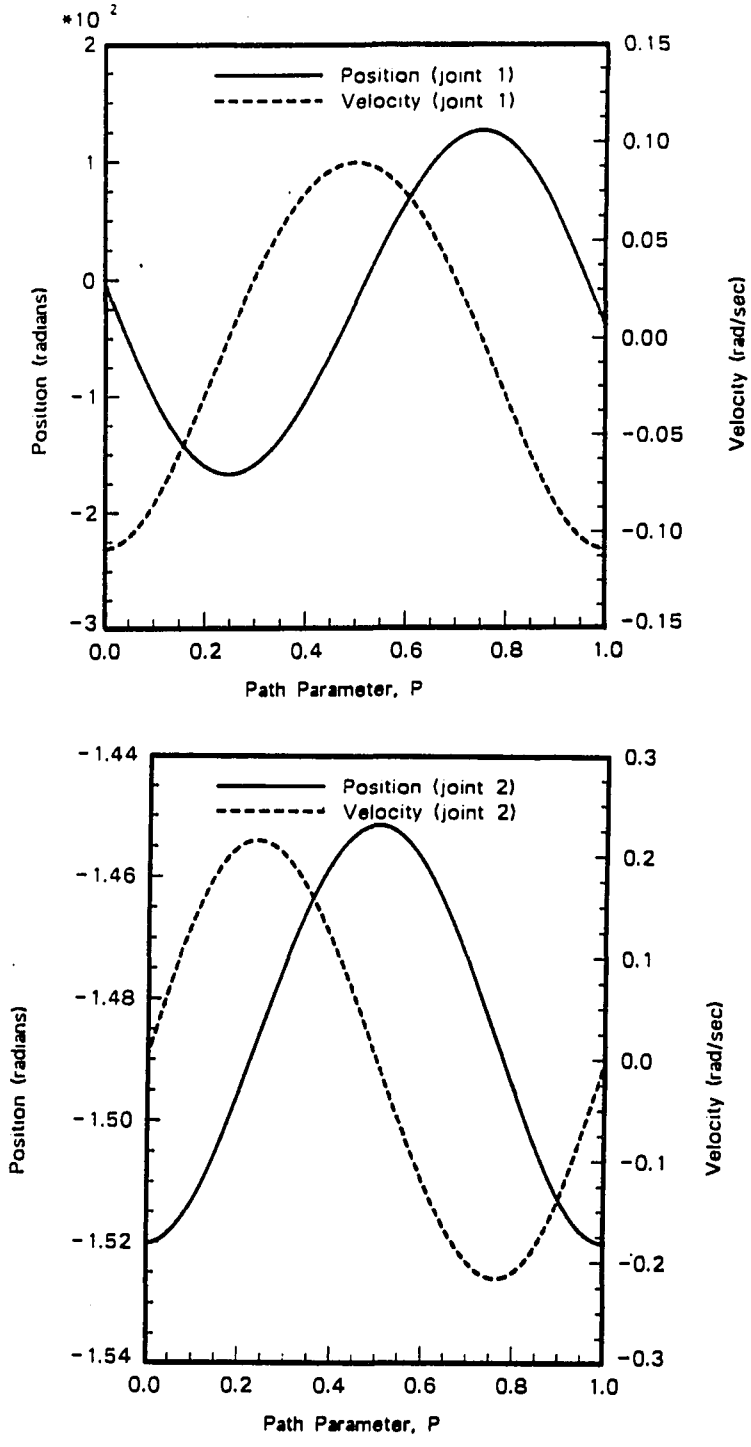


Figure 6.6: Example 2: Joint 1, 2 displacement and velocity history of robot

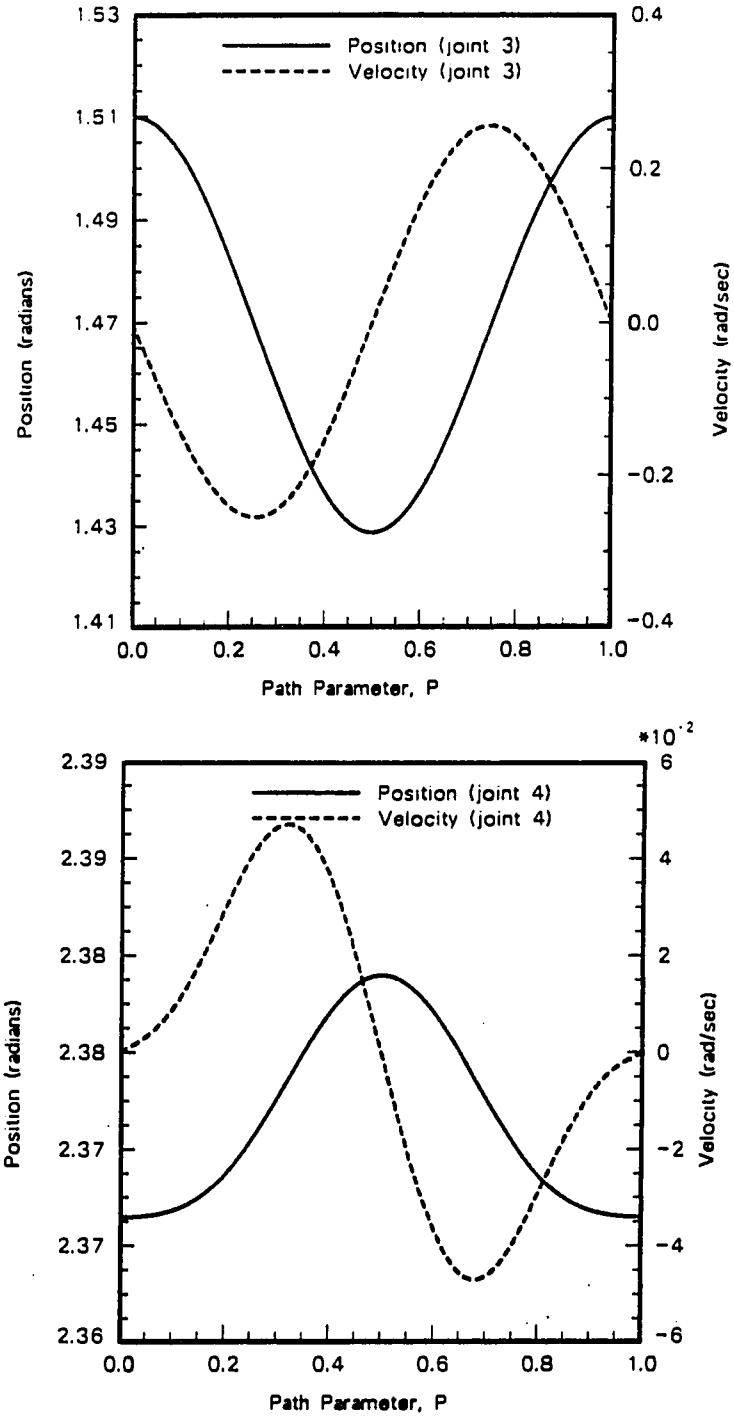


Figure 6.7: Example 2: Joint 3, 4 displacement and velocity history of robot

travels varies from a horizontal position at p=0, to a vertical position at p=1. This results in a need to orient the weld as the position changes. By traditional means, an exceedingly large number of points need to be taught to coordinate movement of the robot and positioner. By this method a continuous trajectory is obtained having a known start solution.

The quarter circle weld's parametric representation is given by:

$$X = 5Cos(\pi p/2)$$

$$Y = -5$$

$$Z = 5 - 5Sin(\pi p/2)$$

where p ranges from 0 to 1. Positioning the third frame on the weld,

$$\overset{2}{3}T = \begin{bmatrix} -Sin(\pi p/2) & 0 & -Cos(\pi p/2) & 5Cos(\pi p/2) \\ 0 & -1 & 0 & -5 \\ -Cos(\pi p/2) & 0 & Sin(\pi p/2) & 5 - 5Sin(\pi p/2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.33}$$

Again using equation 6.10, θ_2 becomes 0 while $S_1 = -Cos(\pi p/2)$ $C_1 = Sin(\pi p/2)$ giving $\theta_1 = Atan2(S_1, C_1)$.

It follows that after performing the necessary transformations the position and orientation of the weld with respect to the robot's base is given by:

$$
{}^{0}_{6}T = \begin{bmatrix} -(-C_1C_2S\theta + S_1C\theta) & C_1S_2 & -(-C_1C_2C\theta - S_1S\theta) & P_x \\ -s_2S\theta & -C_2 & S_2C\theta & P_y \\ -(-S_1C_2S\theta - C_1C\theta) & -S_1S_2 & -(-S_1C_2C\theta + C_1S\theta) & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.34}
$$

where $\theta = (\pi p/2)$ and

$$P_x = 75.0 + (5C_1C_2C\theta + 5C_1S_2 - S_1(5 - 5S\theta) + 24.0)$$

$$P_y = -(5S_2C\theta - 5C_2)$$

$$P_z = 7.5 + 5S_1C_2C\theta + 5S_1S_2 + c_1(5 - 5S\theta)$$

As before this matrix is used to obtain the start solution and the inverse kinematic solution all along the path. Figure 6.8 provides the displacement and velocity information of the positioner while Figures 6.9 and 6.10 give the joint displacement and velocities for all points.

Summary

These examples have shown the use of the novel continuation method for continuous trajectory planning. A need was established for an integrated positioner and robot environment for an arc welding workcell. The method based on the continuation method was outlined and implemented for a GE P50 robot and a 2-axis positioner. A traditional continuation method was used to obtain the inverse kinematics of a three revolute manipulator that is typical of arc welding positioners. A special case of a two-axis positioner was however used to position the weld such that

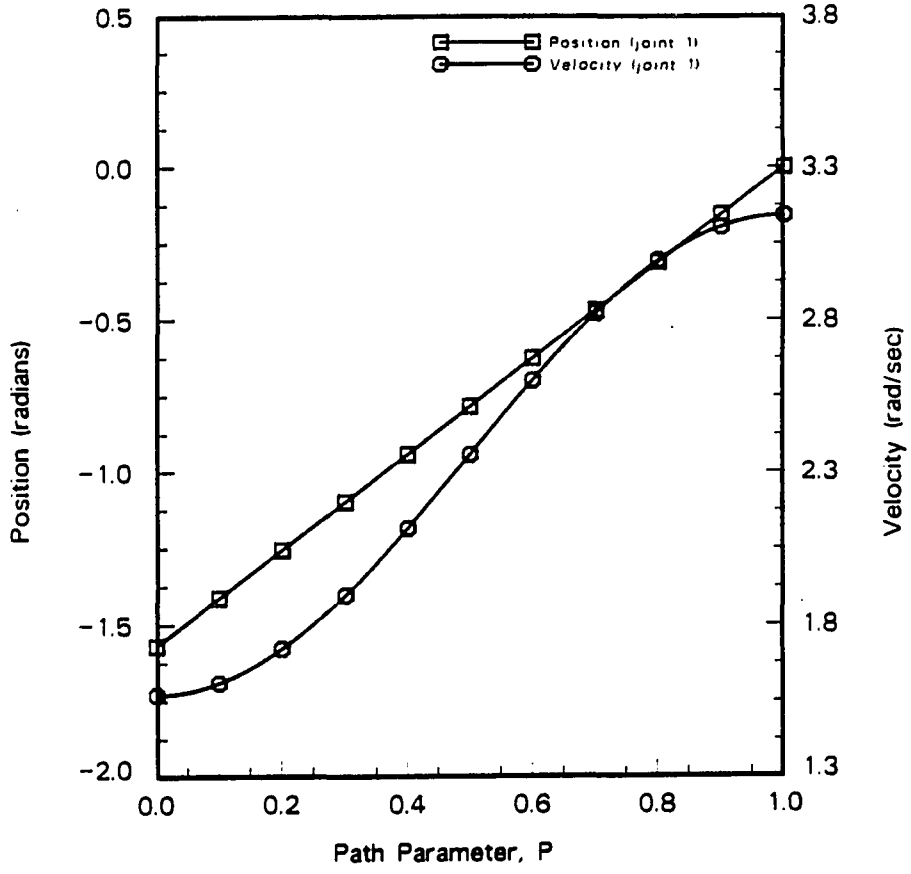


Figure 6.8: Example 3: Joint 1 displacement and velocity history of positioner

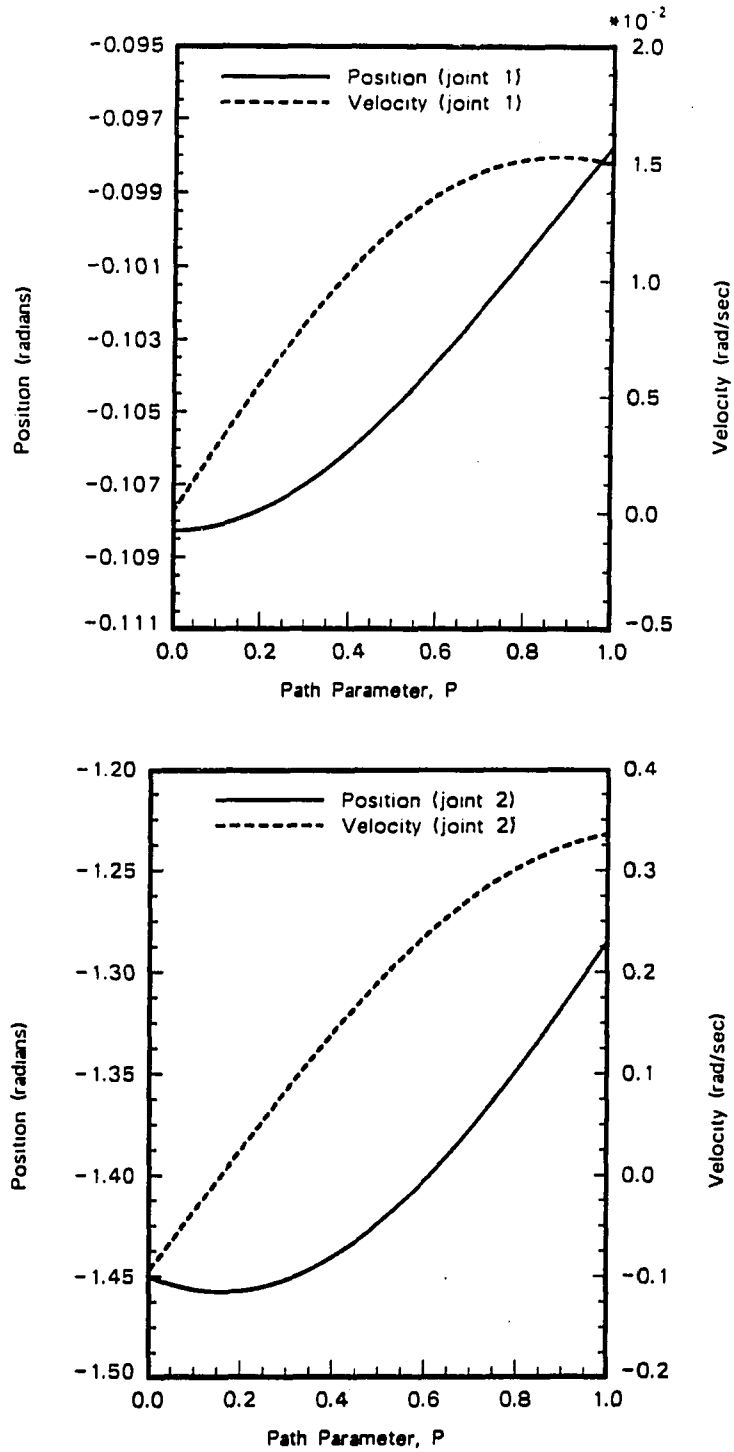


Figure 6.9: Example 3: Joint 1, 2 displacement and velocity history of robot

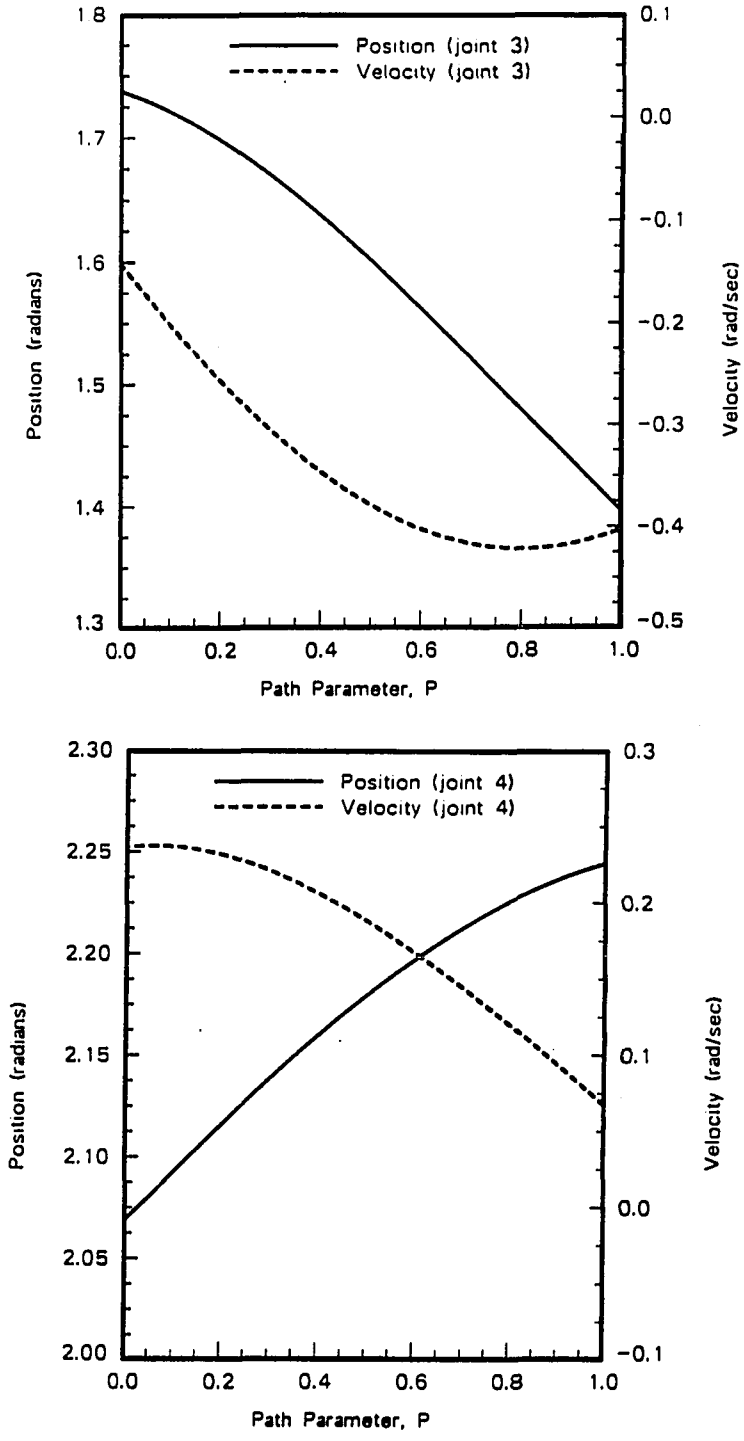


Figure 6.10: Example 3: Joint 3, 4 displacement and velocity history of robot

the normal was in the global Z direction. A different approach called the "Fixed Path Modified Continuation Method" was used to obtain the joint displacement and velocity histories of the GE P50 robot for the weld path. It was assumed however that the paths are collision free and that the workcell always allow the weld normal to be vertical. In the event this is not possible (which is often the case in fact), an optimization procedure can be developed to search the limited joint space using continuation methods. This method also holds promise for collision free trajectory planning. This robust mathematical technique has great potential for application in future off-line programming systems.

CHAPTER 7. CONCLUSIONS

The objectives of this research were to develop an integrated environment for the automatic programming of arc welding robots by means of a consistent representation structure for weld geometry features and weld process knowledge and generate robot and positioner joint angles, velocity histories, and process conditions for paths normally encountered in robotic arc welding. The purpose of this Chapter is to summarize the conclusions drawn from the development of such a system and propose recommendations for future research.

Summary

Automatic programming of arc welding robots is complete when there exists a system for off-line planning and simulation, and an on-line feedback system to adaptively control the welding process. This work concentrated on the first part – that of generating programs for a welding robot off-line based on a CAD description. Hence, in the strict sense the system developed is not a true automatic programming system. However, this research provides a framework on which other systems can be built to close the feedback loop and make it completely automatic. Important contributions by way of this research can be summarized as:

1. Current generation solid modelers do not support the automatic planning and generation of welding process parameters and robot programming because they do not contain secondary feature or variational information such as welding features and attributes. This research for the first time, has proposed a structure for capturing welding information by means of a welding attribute graph (WAGRAPH) that can be attached to the tree of a CSG-based modeler.

2. The welding feature information is implemented in such a manner that extraction of explicit and implicit information has been made possible. Geometry information is extracted for use by the trajectory planning module while geometry and attribute information are used for weld process selection. An integrated system has been developed that uses the geometry information and the mapped welding knowledge to automatically generate the process schedules required for gas metal arc welding robots.

3. The WAGRAPH implementation has utilized the benefits of object-oriented programming (prototype implemented in C++) and supports incremental construction of the graph along with appropriate editing, validity checking facilities.

4. To map weld process knowledge, a systematic methodology of knowledge acquisition using statistical techniques was designed. The intention was to capture the public and private knowledge of welding engineers. The procedures developed were insufficient to acquire the heuristic (private) information from the experts and seriously questioned the need of a knowledge-based system. However, general dependencies, order of the knowledge-based system development, and trends were captured by the knowledge acquisition procedures. This helped

in the easier construction of the system.

5. It was concluded that robotic arc welding is not significantly different from manual arc welding but more sensitive to the specification of weld process parameters as there does not exist an on-line feedback like that of a human hand or eyes in manual arc welding. Experts use heuristic information for "fine-tuning" weld process parameters and handbook data to provide the range. This results in the need for the creation of a primitive knowledge mapping system based on handbook knowledge and subsequent addition of private knowledge by the welding experts themselves. Such an addition would enrich the knowledge-based system to a great extent.

6. If welding engineers need to add knowledge, provisions should be made for validity and consistency checking of the additional information. A methodology was proposed by this research to check the validity and consistency of knowledge addition using an expert system shell. In addition, the knowledge mapping was performed in an integrated manner using the object-oriented programming paradigm.

7. A novel method was proposed and implemented to continuously plan the trajectory of welding profiles in an integrated robot and positioner environment.

8. The "fixed-path modified continuation method" is a robust mathematical technique to obtain joint angles, and velocity histories all along the path given the start point and the target system. Various paths normally encountered in arc welding – those which require one time movement or a continuous movement of a two-axis positioner were tested to verify the method's ease and power. The

method has great potential for application in off-line programming systems and on-line controllers.

9. It is recommended that for total automatic programming systems the following need to be developed: (a) the development of on-line feedback systems using vision, seam tracking and other methods; (b) further research on the development of welding as a science-based system rather than as an art and (c) a study of the effects of geometry on the specification of process in arc welding robots.

Recommendations For Future Work

Various aspects of the research that can be explored based on this work are suggested in this Section. These are based on the experience gained during the development of the integrated system. These additions to the system will help in the creation of a truly automatic programming system for robotic gas metal arc welding. This will eventually help in the use of robots where flexibility is very important as in jobshops or custom fabrication units.

Richer representation scheme

This research proposed a basic methodology to add welding features and attributes to a solid modeler. However, to support robotic arc welding programming and planning, a higher level representation scheme in addition to the existing structure should be developed to model the robot, positioner, and surroundings. The representation scheme should be able to specify the features of the work cell in such a manner that collision free paths and optimization of the weld trajectory in accordance with the process specifications can be attempted. This will help further

research in task level programming of robots.

Knowledge mapping shell

It is uniformly seen that representation of information in a CAD-based system is currently used only to drive an application system or help in the planning and programming of elements in the manufacturing domain. To attempt this, a great amount of knowledge needs to be mapped. Current modeling systems do not provide a means to model the process. Various systems have been created to use the facilities of a modeling system for application purposes and they exist as stand alone systems. Instead, if provision is created in CAD systems in the form of a shell such that it accomodates different knowledge representation schemes, then automatic planning and programming in different domains of manufacturing can be created easily. This requires extensive work by engineering researchers in the specification of functional requirements of application domains in the design and manufacture of mechanical parts and by computer science researchers in providing the representational framework.

Weld process specification improvement

Specification of welding process parameters depends heavily on experimentation and is based on trial and error. A better understanding of the arc welding process and the effects of parameters such as magnetic blow, current, travel speed and others on penetration and similar parameters that affect welding need further study. The effects of geometry on the specification of process parameters can be investigated further. A vision-based system that can identify and characterize the weld needs to

be integrated with the CAD-knowledge mapping system developed in this research. This will complete the loop in the automatic programming of arc welding robots. This should be able to adaptively correct the welding process parameters and change rules of the knowledge-based system accordingly.

Collision free planning of welding trajectories

The work on trajectory planning provided a method to obtain continuous information about the welding paths. It assumed simple workpiece geometry and no limitations other than robot physical limits in terms of reach and access. However, in the real world it is not always possible to reach the weld due to interferences and access problems. Further, due to the constraints of the process, the condition of the normal of the weld to be in the global Z direction may not also be possible. The formulation of the continuation method as an optimization problem and for the planning of collision free trajectories can be successfully attempted.

Visualization

The method of trajectory planning of an integrated system resulted in the histories of joint angles and velocity all along the welding path. This method can be successfully integrated with a visualization scheme in the off-line programming of arc welding robots. The joint angle and velocity information generated by the continuation formulation can be used in conjunction with a solid modeling system and simulation software to visualize the interference free trajectory of a coordinated robot and positioner system.

REFERENCES

Alberry, P. J., D. Taylor, and D. Yapp 1987. An Expert System For Welding Engineers. *Proceedings of 1st International Conference on Computer Technology in Welding*, Editor: W. Lucas, TWI, UK, 355-363.

Angeles, J. 1986. Iterative Kinematic Inversion of General Five-Axis Robot Manipulators. *The International Journal of Robotics Research*, vol.4, no.4:59-70.

ANSI Y14.5M 1982. Dimensioning and Tolerancing. *ANSI Standard Y14.5M-1982*, United States of America Standards Institute, New York.

Ardayfio, D. D. 1987. Fundamentals of Robotics. *Marcel Dekker Inc.*, New York.

Barborak, D.M., D.W. Dickinson, and R. B. Madigan 1991. PC-Based Expert Systems and Their Applications to Welding. *Welding Journal*, vol. 71, no.1:29-38.

Boehm, W. 1980. Inserting New Knots into B-Spline Curves. *Computer-Aided Design*, vol.12, no.4:199-201.

Bolmsjo, S., 1989. Programming Robot Systems For Arc Welding in Small Series Production. *Robotics and Computer Integrated Engineering*, vol.5, no.2/3: 199-205.

Brown, C. M., 1982. PADL-2: A Technical Summary. *IEEE Computer Graphics Applications*, vol.2, no.2:69-84, March.

Buchal, R. O., D. B. Cherchas, Sassani, F, and J. P. Duncan 1989. Simulated Off-Line Programming of Welding Robots. *International Journal of Robotics Research*, vol.8, no.3:31-43.

Cary, H. B. 1991. Summary of Computer Programs for Welding Engineering. *Welding Journal*, vol. 71, no.1:40-45.

Chan, S. F., R. H. Weston, and K. Case 1988. Robot Simulation and Off-Line Programming. *Computer-Aided Engineering*, vol.5, no.4:157-162.

Chand, S., and K. L. Doty. 1985. On-Line Polynomial Trajectories for Robot Manipulators. *The International Journal of Robotics Research*, vol.4, no.2:38-48.

Charny, J. 1984. Off-line Robot Programming. *Proceedings of ASME International Computers in Engineering Conference*, August.

Craig, J. J. 1986. Introduction to Robotics. *Addison-Wesley Publishing Company*, Reading, Massachusetts.

Craig, J. J. 1987. Coordinated Motion of Industrial Robots and 2-DOF Orienting Tables. *Proceedings of the 17th International Symposium on Industrial Robots*, Chicago, April.

Cunningham, J. J., and J. R. Dixon 1988. Designing With Features: The Origin of Features. *Proceedings of the ASME International Computers In Engineering Conference*, San Francisco, Calif., August, 237-243.

Devgun, M.S., and S. Padmanabhan. 1990. Feature Representation and Extraction in CAD Models for Manufacturing Automation: Methods and Issues. *Selected Papers of X International Conference on Production Research*, Taylor Francis Publications, August.

Dixon, J. R., E. C. Libardi, S. C. Luby, and M. Vaghul 1987. Expert Systems For Mechanical Design: Examples of Symbolic Representation of Design Geometries. *Engineering With Computers*, vol. 2, 1-10.

Dorn, L., and S. Majumder 1988. An Expert System For Welding Design. *Proceedings of 2nd International Conference on Computer Technology in Welding*, Paper 1, TWI, UK.

Fu, K. S., R. C. Gonzalez, and C. S. G. Lee. 1987. Robotics: Control, Sensing, Vision, and Intelligence. *McGraw-Hill Book Company*, New York, New York.

Gindy, N. N. Z. 1989. A Hierarchical Structure for Form Features. *International Journal of Production Research*, vol. 27, no,12:2089-2103.

Goldenberg, A. A., and D. L. Lawrence. 1986. End Effector Path Generation. *Jour-*

nal of Dynamic Systems, Measurement, and Control, vol.108, no,2:158-162.

Gruver, W.A., B.L. Saroka, J.J. Craig, and T.L. Turner, T.L. 1984. Evaluation of Commercially Available Robot Programming Languages. *Proceedings Robots IX*

Gupta, K. C., and K. Kazerounian. 1985. Improved Numerical Solutions of Inverse Kinematics of Robots. *IEEE International Conference on Robotics and Automation*, IEEE Computer Society Press, Maryland, U.S.A., pp. 743-748.

Hathaway, W. F., and G. A. Finn 1986. Microcomputer Expert Systems For Welded Construction. *ASCE Proceedings of Fourth International Conference on Computing in Civil Engineering.*

Hayes-Roth, F., D. L. Waterman, and D. B. Lenat. 1983. Building Expert Systems. *Addison Wesley Publishing Company*, New York, New York.

Hornick, M. L., and B. Ravani. 1986. Computer-Aided Off-Line Planning and Programming of Robot Motion. *The International Journal of Robotics Research*, vol.4, no.4:18-31.

Hummel, K. E., and S. L. Brooks 1986. Symbolic Representation of Manufacturing Features for an Automated Process Planning System. *Proceedings: Knowledge Based Expert Systems for Manufacturing*, editors: Lu, S. C-Y., and R. Komanduri, PED: vol. 24, ASME Winter Annual Meeting, Anaheim, Cal. U.S.A.

IDEAS 1990. Solid Modeling User's Guide, SDRC Corporation, Milford, OH, U.S.A.

James, E. B., and R. G. Baker 1987. The Application of Knowledge-Based Systems in Welding. *Proceedings 1st International Conference on Computer Technology in Welding*, Editors: W. Lucas, TWI, UK, 375-382.

Jo, D. -Y., and E. J. Haug. 1988. Workspace Analysis of Multi body Mechanical Systems using Continuation Methods. pp 447-456. *Trends and Developments in Mechanisms, Machines and Robotics.* The American Society of Mechanical Engineers, New York, New York.

Kerth, Jr. W. J., and R. J. Kerth 1984. Mobile and Stationary Adaptive Welding Systems. *Proceedings Robots 8 Conference*, MS84-384. Society of Manufacturing Engineers.

Kuhne, A. H., H. B. Cary, and F. B. Prinz 1987. An Expert System For Robotic Arc Welding. *Welding Journal*, vol. 67, no.11:21-25.

Kumara, S. R. T., S. Joshi, R. L. Kashyap, C. L. Moodie, T. C. Chang 1985. Expert Systems In Industrial Engineering. *International Journal of Production Research*, vol. 24, no.5:1107-1125.

Lane, J.D. 1987. Robotic Welding State Of The Art. *Robotic Welding*, edited by J.D. Lane, IFS Publications Ltd, UK.

Larson, P. G. 1985. The Weld Scheduler Expert System. *Research Report RDD: 86:3431-04-02:01*, Babcock and Wilcox.

L-TEC 1989. MIG Welding Handbook. L-TEC Welding & Cutting Systems, Florence, SC, U.S.A.

Lucas, W. 1987. Microcomputer Systems, Software, and Expert Systems for Welding Engineering. *Welding Journal*, vol. 67, no.4:19-30.

Miner, R. H. 1985. A Method For The Representation and Manipulation of Geometric Features in solid Model. *M.S. Thesis*, Department of Mechanical Engineering, Massachussets Institute of Technology.

Morgan, A. 1981. A Method for Computing all Solutions to Systems of Polynomial Equations. *GM Research Publication*, GMR 3651.

Morgan, A. 1987. Solving Polynomial Systems using Continuation for Engineering and Scientific Problems. *Prentice-Hall, INC.*, Englewood Cliffs, New Jersey.

Mortenson, M. E., 1985. Geometric Modeling. *John Wiley & Sons*, New York, NY, U.S.A.

Nguyen, T. A. 1987. Verifying Consistency of Production Systems. *Proceedings, IEEE Conference on Knowledge-Based Systems*, vol. 1:4-8.

Peigl, L., 1991. On NURBS: A survey. *IEEE Computer Graphics and Applications*, vol. 11, no.1:55-71.

Pratt, M. J., and P. R. Wilson 1985. Requirements for the Support of Form Features in a Solid Modeling System. *Report No. R-85-ASPP-01*, CAM-I Inc, Arlington,

Texas.

Reeves, R. E., T. D. Manley, A. Potter, and R. Ford 1988. Expert Systems Technology - An Avenue to Intelligent Weld Process Control System, *Welding Journal*, vol. 67, no.6:33-41.

Requicha, A.A.G. 1980. Representations for Rigid Solids: Theory, Methods, and Systems. *Computing Surveys*, vol.12, No.4:437-464.

Requicha, A. A. G., and S. C. Chan 1986. Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry. *IEEE Journal of Robotics and Automation*, vol. RA-2, no.3:156-166.

Requicha, A.A.G., and H.B. Voelcker. 1982. Solid Modeling: A Historical Summary and Contemporary Assessment. *IEEE Computer Graphics and Applications*, vol 2. no.2:9-24.

Ribiero, E. A., J. D. Turner, and R. A. Farrar 1987. Research in Welding Expert Systems at Southampton University. *Proceedings 1st International Conference on Computer Technology in Welding*, Editors: W. Lucas, TWI, UK, 413-418.

Shah, J. J., and A. S. Bhatnagar 1989. Group Technology Classification From Feature-Based Geometric Models. *Manufacturing Review*, vol. 2, no.3:204-213.

Shah, J. J., Hsiao, D., and R. Robinson 1990. A Framework for Manufacturablilty Evaluation in a Feature-Based CAD System. *Proceedings of NSF Design and Manufacturing Systems Conference*, Arizona State University, Tempe.

Shah, J. J., and D. Miller 1989. A Structure for Supporting Geometric Tolerances for Computer Integrated Manufacturing. *Transactions of the North American Manufacturing Research Institution of SME*, SME, Dearborn, MI, pp. 344-351.

Shah, J.J., and M. Rogers. 1988. Functional Requirements and Conceptual Design of the Feature-Based Modeling System. *Computer-Aided Engineering Journal*, vol.5, no.1:9-15.

Sicard, P., M. D. Levine 1988. An Approach to an Expert Robot Welding System. *IEEE Transactions on Systems Man and Cybernetics*, vol. 18, no.2.

Singh, V. K. and K. C. Gupta. 1989. A Manipulator Jacobian Based Modified

Newton-Raphson Algorithm (JMNR) for Robot Inverse Kinematics. *Advances in Design Automation - 1989: Mechanical Systems Analysis, Design and Simulation*, ASME DE vol.19, No.3:327-332.

Srinivasan, R., C. R. Liu, and K. S. Fu 1985. Extraction of Manufacturing Details From Geometric Models, *Computers in Industrial Engineering*, vol. 9, no.2:125-133.

Steiger-Garcao, A., and L. M. Camarinha-Matos. 1987. A Conceptual Structure for a Robot Station Programming System, *Robotics*, Elsevier Science Publishers, vol. 3:195-204.

Stroustrup, B., 1986. The C++ Programming Language. *Addison Wesley Publishing Company*, Reading, MA, U.S.A.

Stroustrup, B., 1988. What is Object-Oriented Programming. *IEEE Software*, vol. 5, no.3:10-20.

Subbian, T., D. R. Flugrad., and P. Kavanagh. 1991a. Robot Trajectory Planning By A Continuation Method. *Proceedings, Factory Automation and Information Management*, University of Limerick, Limerick, Ireland, March 13-15, pp:394-403.

Subbian, T., and D. R. Flugrad. 1991b. Use of Continuation Method For Kinematic Synthesis. *8th World Congress of IFTomm*, Prague, Czechoslovakia. To be published.

Taylor, A. 1989. An Investigation of Knowledge-Based Models for Automated Procedure Generation in the Arc Welding Domain. *International Journal of Production Research*, vol 27, no.11:1855-1862.

Taylor, R. H. 1979. Planning and Execution of Straight Line Manipulator Trajectories. *IBM Journal of Research and Development*, vol.23, No.4:424-436.

Thompson, D. R., A. Ray, S. Kumara 1988. A Hierarchically Structured Knowledge-Based System For Welding Automation and Control. *ASME Journal of Engineering For Industry*, vol. 110, 71-76.

Tiemann, M. D., 1990. The User's Guide to GNU C++, Version 1.37.1. *Free Software Foundation Inc.*, U.S.A.

Tonkay, G. R., and K. Knott 1989. An Expert System For Welding. *in Artificial Intelligence - Manufacturing Theory and Practice*, edited by Kumara, S. T., Kashyap,

R. L., and Soyster, A. L., Industrial Engineering and Management Press, Georgia, U.S.A., pp 647-686.

Tsai, L. -W., and A. P. Morgan. 1985. Solving the Kinematics of the most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods. *Journal of Mechanisms, Transmissions and Automation in Design*, vol.107:189-200.

Vaghul, M., J. R. Dixon, G. Zinsmeister, M. K. Simmons 1985. Expert Systems in a CAD Environment: Injection Moulding Part Design as an Example. *Proceedings of the ASME International Computers in Engineering Conference*, Boston, Mass.

Wampler, C., and A. Morgan. 1989. Solving the 6-R Inverse Position Problem Using a Generic-Case Solution Methodology. *GM Research Publication*, GMR 6702.

Wampler, C. W., A. P. Morgan, and A. J. Sommese. 1990b. Numerical Continuation Methods for Solving Polynomial Systems Arising in Kinematics. *Journal of Mechanical Design*, vol.112, No.1:59-68.

Wang, K. 1988. B-Splines Joint Trajectory Planning. *Computers in Industry*, vol.10, No.2:113-122.

Welding Handbook 1984. Welding Processes: Gas, Arc, and Resistance, in *Section 2 Welding Handbook*, Seventh Edition, American Welding Society, New York, New York.

American Welding Society Specification A 5.18., 1979. AWS, New York, New York.

Wolovich, W. A. 1987. Robotics: Basic Analysis and Design. *CBS College Publishing*, New York, New York.

Wu, S., 1991. Integrating Logic and Object-Oriented Programming. *OOPS Messenger, A Quarterly Publication of the Special Interest Group on Programming Languages*, ACM Press, vol. 2, No.1:28-37.

Yong, Y. F., J. A. Gleave, J. L. Green, and M. C. Bonney 1985. Off-Line Programming of Robots. in *Industrial Handbook on Robotics*, Wiley and Sons.

Zimmerman, R., 1990. Personal Communication With John Deere Welding Engineer.

APPENDIX A. GLOSSARY OF TERMS

Abbreviations:

AI	:	Artificial Intelligence
B-Rep	:	Boundary Representation
CAD	:	Computer-Aided Design
CAEE	:	Computer-Aided Engineering Environment
CAM	:	Computer-Aided Manufacturing
CSG	:	Constructive Solid Geometry
DCRP	:	Direct Current Reverse Polarity
GMAW	:	Gas Metal Arc Welding
KBES	:	Knowledge-Based Expert System
OOP	:	Object-Oriented Programming
WAGRAPH	:	Welding Attribute GRAPH

Terms:

Artificial Intelligence: It is the study of methods for solving tasks that require "human-like" intelligence. The goal of artificial intelligence is to develop computer systems that in some way think or solve problems in a way that would be considered intelligent if done by humans.

Cartesian Space: Generally a two or three-dimensional rectangular space defined by mutually perpendicular Cartesian axes, normally denoted as x and y, or x, y, and z. In robotics, a Cartesian space vector may also include appropriate orientation information.

Confounding: The device of reducing the block size by making one or more interaction contrasts identical with block contrasts, is known as confounding.

Constructive Solid Geometry (CSG): CSG representations of objects are ordered binary trees whose leaf or terminal nodes are either primitives or transformation data for rigid body motions. The nonterminal nodes are either regularized Boolean operators (union, difference, intersect) or rigid-body motions (translation and/or rotation) that operate on their two subnodes (or subsolids).

Expert Systems: An expert system is a program which has high quality specific knowledge for solving domain-specific problems. Expert systems handle real world problems using computers to reach the same results, as would a human expert, if faced with a comparable problem.

Factorial Design: A factorial design involves simulataneously more than one factor at two or more levels.

Forward and Backward Chaining: Forward chaining methods start with a set of initial conditions and reach the desired goal in a manner similar to the bottom-up strategy and hence are used for synthesis. Backward chaining methods start with a goal situation, decompose this goal to sub-goals and continue this process until all the sub-goals are solvable. This approach is similar to the top-down strategy and is often used for analysis.

Forward and Inverse Kinematics: Kinematics is the science of motion which treats motion without regard to the forces that cause it. The method of computing the position and orientation of the manipulator's end effector relative to the base of the manipulator as a function of the joint variables is called forward kinematics. The method of finding the required joint angles given the desired position and orientation of the tool relative to the station.

GMAW: Gas metal-arc welding also known as MIG welding is an arc-welding process wherein coalescence is produced by heating with an arc between a continuous filler metal (consumable) electrode and the work. Shielding is obtained entirely from an externally supplied gas or gas mixture.

Heuristic Rules: Techniques used by human experts for solving problems based on their experience and knowledge rather than results based on analytical or algorithmic procedures.

Inference engine: Inference engine is a part of an expert system with problem solving paradigms. It uses the forward chaining and/or backward chaining methods for problem solving.

Knowledge Base: It is a collection of data specific to the domain. In most of the literature, the set of rules for interpreting the data are also included in the knowledge base.

Learning: The ability of an artificial intelligence method to make changes to the knowledge base and some part of the inference engine (based on experience) such that the changes have a long-term effect on the performance of the method.

Link Space: An n-dimensional space defined by the various displacements of an n-link manipulator.

Object-Oriented Programming: Object-oriented programming rejects the traditional dichotomy between data and procedures and substitutes the concepts of objects and messages. Objects contain both data and procedures. It shifts the emphasis from algorithms, or how things get done to object declarations, or what needs to be manipulated.

Shell: Shells are special environments for the use of expert systems with a built in inference engine.

Trajectory: A spatial position/time curve that usually represents a desired manipulator motion in either link or Cartesian space. A path along with appropriate velocity and/or acceleration information.

Trajectory Planner: A program that produces a time varying signal representing the desired path or trajectory that a robot is to follow in either Cartesian or link space.

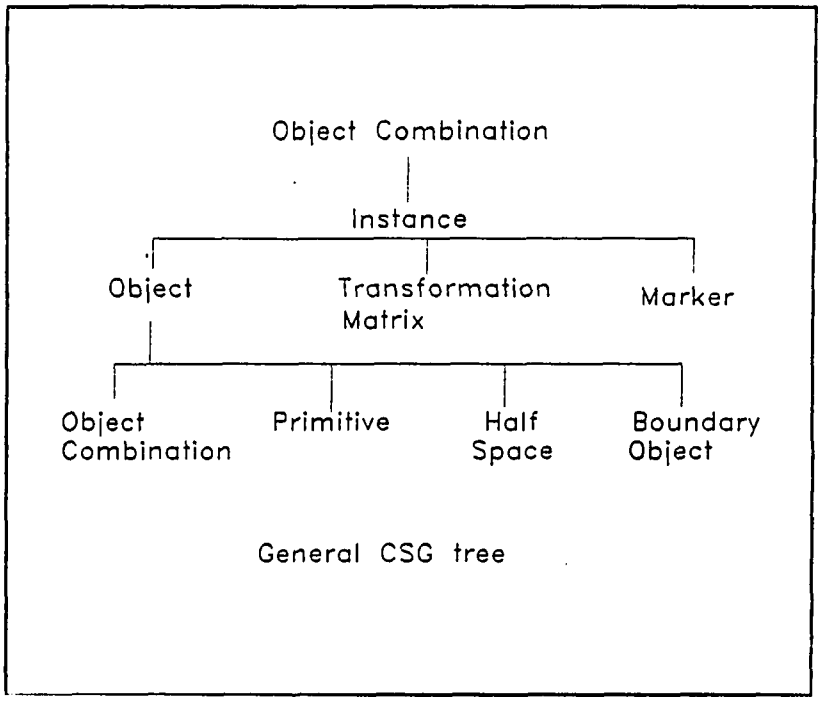
APPENDIX B. CSG, FEATURES, AND NURBS

The following is an abbreviated explanation of Constructive Solid Geometry representation scheme in modelers adapted from Mortenson [1985]. The definitions of features is from IDEAS solid modeler reference manual [IDEAS 1990]. The definition of NURBS is from Piegl's survey [1991].

CSG: Constructive Solid Geometry (CSG) is a term for modeling methods that define complex solids as composition of simpler solids (primitives). Boolean operators are used to execute the composition.

CSG representations of objects are ordered binary trees whose leaf or terminal nodes are either primitives or transformation data for rigid body motions. The non terminal nodes are either regularized Boolean operators (union difference and intersect) or rigid-body motions (translation and/or rotation) that operate on their two subnodes (or subsolids). Each subtree of a node (not a transformation leaf) represents a solid resulting from the combination and transformation operations indicated below it. The root represents the final object. Figure 10.1 provides a simple example of the CSG representation.

If the primitive elements of a modeling system are valid bounded solids defined by the system and the combining operators are regularized, then the resulting solid models are valid and bounded. The most common approach is to offer a finite set



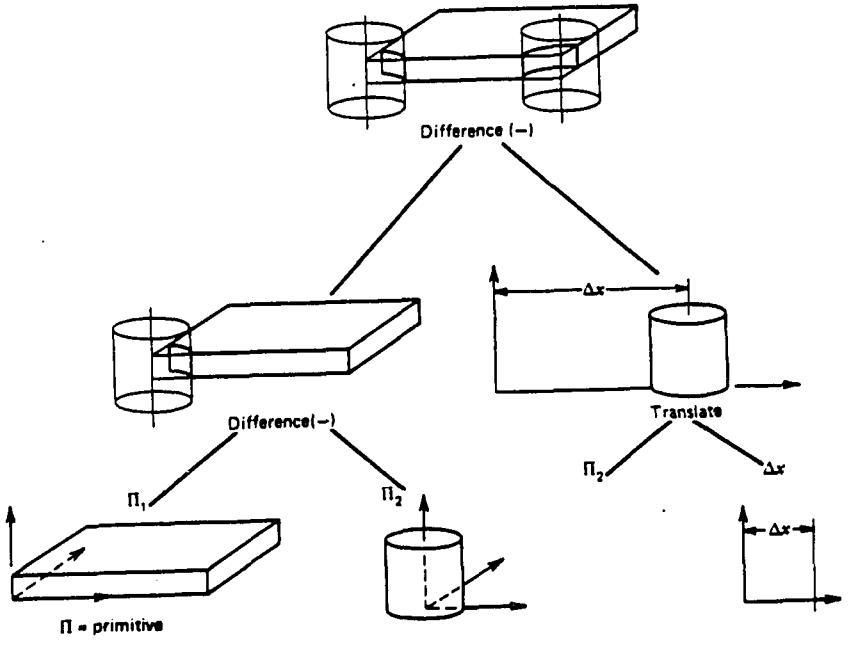


Figure B.1: Constructive solid geometry representation

of concise, compact primitives whose size, shape, position, and orientation are determined by a small set of user-specified parameters. For a block type primitive, the user specifies the length, width, height, and initial position and the modeling system checks the validity of the parameters. Other common parameters are the cylinder, sphere, cone, and torus. The primitives themselves are usually represented by the intersection of a set of curved or planar half-spaces. The primitive cylinder for example is represented by the intersection of a cylindrical half-space and two planar half-spaces.

Most hybrid modelers generate two representations of a solid. The first is a constructive representation exemplified by a binary tree data structure linking primitives and successive subsolids by using combining operators and transformations. The second is the boundary representation, which describes the faces, edges, and vertices of the boundary of the solid. This description contains both the topological representation of the connectivity of the boundary elements and numerical data describing the shape geometry and position of these elements. The boundary representation is computed from the constructive representation by a set of algorithms called the boundary evaluator. The boundary evaluator determines where component faces are truncated and new edges and vertices are created or deleted. Where boundary elements overlap or coincide, the evaluator merges them into a single element and thus maintains a consistent, non-redundant data structure representing a solid's boundary. The constructive form is often called the procedural or unevaluated model, and the boundary form is often called an evaluated model.

FEATURES: The feature definition task as provided in solid modeling systems allows one to create and manage user-defined features from objects. The use of

user-defined features is the development of an object in which the object has some variable dimensions, while other dimensions always relate to the variable dimensions. An example is a counter-sunk hole. User-defined features are also extremely helpful in post-testing modifications. By changing the dimensions that are variable, the dependent dimensions are automatically adjusted. The initial geometry of a user-defined feature is defined by any object created in the object modeling task.

The feature definition task is intended to be used to create features and load them from a feature library. An important application of feature definition is the sharing of the feature library with other users. By creating and using form features, improvements in desing and modeling productivity can be achieved, design modification is made easier and the design intent can be captured for others' use in the product development process. Typical manufacturing features created include: slot, hole, counter-sunk hole, bolt hole circle, tapped hole flanges, fillets.

NURBS: The two major ingredients of a Non-Uniform Rational B-Splines are rational and B-splines. A NURBS curve is a vector-valued piecewise rational poly nomial function of the form [Peigl 1991]:

$$C(u) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,p}(u)}{\sum_{i=0}^{n} w_i N_{i,p}(u)} \tag{B.1}$$

where the w_i are the so-called weights, the P_i are the control points (just as in the case of nonrational curves), and $N_{i,p}(u)$ are the normalized B-spline basis functions of degree p defined recursively as

$$
\begin{aligned}
N_{i,0}(u) &= 1 \ if u_i <= u_{i+1} \\
&\quad 0 \ otherwise \\
N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \\
&\quad \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)
\end{aligned}
\tag{B.2}
$$

where u_i are the so-called knots forming a knot vector. The degree, number of knots, and number of control points are related by the formula m = n + p + 1.

$$
U = u_O, u_1, ..., u_m \tag{B.3}
$$

A NURBS surface is the rational generalization of the tensor product nonrational B-spline surface and is defined as follows:

$$
S(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} P_{i,j} N_{i,p}(u) N_{j,q}(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j} N_{i,p}(u) N_{j,q}(v)} \tag{B.4}
$$

where $w_{i,j}$ are the weights, $P_{i,j}$ form a control net, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the normalized B-splines of degree p and q in the u and v directions, respectively, defined over the knot vectors

$$
\begin{aligned}
U &= [0,0,....,0,u_{p+1},....,u_{r-p-1},1,1,.....,1] \\
V &= [0,0,....,0,v_{q+1},....,u_{s-q-1},1,1,.....,1]
\end{aligned}
\tag{B.5}
$$

where the end knots are repeated with multiplicities p+1, and q+1, respectively, and r = n + p + 1 and s = m + q + 1. Although the surface was obtained by generalizating the tensor-product surface form, a NURBS surface is, in general, not a tensor-product surface.

APPENDIX C. KNOWLEDGE ACQUISITION SURVEY

The survey was designed in two main parts. The purpose of the first part of the survey (the enclosed part) was to direct the means for knowledge acquisition and prioritize the welding parameters involved. This part would also try to identify the problems in current robotic arc welding systems. The purpose of the second part of the survey was to conduct detailed interviews with a group of respondents to discuss the results of the first part and to further provide weld process knowledge needed for the automatic robot programming system. The overall goal of this survey was to study the following:

1. Problems in programming arc welding robots and the deficiencies in teach/high level languages and graphical off-line programming systems for arc welding robots
2. Understand the process of specifying a welding schedule
3. Order of welding process parameter selection and its classification
4. Relationships between product variables and process variables for robotic arc welding
5. Effect of the robot and the positioner on the welding process parameters

Questionnaire

Please place a cross mark on the sub-division you choose. If answers need to be provided, please write them in the space provided. If the space provided is not enough, please feel free to write on the back of the sheet. If you feel more than one answer is suitable, please place a cross mark on all of the answers you choose, and write any comments you feel to be necessary.

1. How long have you been a welding engineer (A) 1-3 years (B) 3-6 years (c) 6-10 years (D) over 10 years

2. How do you classify the parts welded in your plant and provide a number by the side of the chosen sub-division (A) large volume (B) medium volume (C) small batch production

3. What form of programming are you used to? Encircle the languages and systems used (A) Teach (B) High level textual languages (VAL II, RAIL, AML, AL, MCL, others) (C) High-level graphical programming systems (ROBOGRAPHIX, ROBOCAM, McAuto, ROBOTEACH, ROBOT-SIM, others)

4. In these high level programming languages do you still have to specify the following

   (a) Specification of weld coordinates by teach pendant (YES / NO)

   (b) Specification of weld process parameters (YES / NO)

   (c) Others, if any please mention

5. In these graphical off-line programming systems, do you have to specify the following

   (a) Specification of weld paths by lead-thru simulation

   (b) Specification of welding process parameters

   (c) Others, if any please mention

6. What percentage of the output program from a high level/graphical off-line programming system can be used directly? (A) 0-15% (B) 15-40% (C) 40-70% (D) 70-95% (E) 100%

7. What reasons do you attribute to the answer in question 6?

8. Do you see significant differences between hard automated GMAW and robotic arc GMAW (A) Yes (B) No

9. Due to the flexibility of the robot and the positioner does the process parameter specification change in robotic welding in comparison to manual GMAW (A) Yes (B) No

10. How would you classify the change in the values of the process parameters as (A) Drastic (B) Significant (C) Marginal

11. What is the source of your process specification knowledge. Place a crosss mark on all that you think and indicate the sources

    (a) Handbooks .....

    (b) Tables and charts .....

    (c) Private heuristic knowledge .....

12. What is the percentage of private heuristic knowledge that is used in the welding process parameter specification (A) less than 10% (B) 10-20% (C) 30-50% (D) Greater

13. Are there typical areas that you employ private knowledge. If so, please specify the areas:

14. Are there direct formulae that you apply to obtain welding process parameters (A) yes (B) no

15. If answer to 14 is YES, please indicate the source(s)

16. Is there a need for a logical reasoning in the selection of proper welding parameters (A) Yes (B) No

17. Is there a need for a knowledge-based system in the selection of proper welding parameters (A) Yes (B) No

18. If answer to question is No, then why not

19. if answer to 17 is YES, is the knowledge in a form that can be expressed as facts and relationships among these facts (A) Yes (B) No

20. If answer to question 19 is YES, can the relationships be described by IF-THEN rules (A) Yes (B) No

21. Do you feel that there are too many interacting features that play in the selection of welding process aprameters (A) Yes (B) No

22. Facts, constraints, rules: Would these adequately classify all the knowledge involved (A) Yes (B) No

23. If answer to question 22 is no, mention any other representation schemes you feel relevant

Welding Process Parameter Classification

1. Do you agree on the following classification of welding process parameters? Indicate any others you feel that has been omitted?

Product Variables

- Material Type
- Joint thickness
- Type of joint
- Position of joint
- Geometry of workpiece

Process Variables

- Current
- Voltage
- Wirefeed
- Travel speed
- Electrode type
- Electrode size
- Arc length
- Gas mixture
- Gas flow rate
- Robot tool angle
- Number of passes
- Weave pattern
- Polarity

- Displacement from the joint for multipass welds
- Dwell time at the start and end of weld

2. Please list the order of the process parameters selection to the right of the parameters listed above. If you feel there is a tie, provide the same number for all tied values. The first parameter selected would have a number 1.

The table below is to study the relationship between product and process variables. Listed below are the categories of dependency. If you feel there is any dependency of any of the variables indicate by a letter grade. The possible responses are:

A: Complete dependency
B: Strong dependency
C: Little dependency
D: No dependency
E: Not known

	Material	Thickness	Geometry (Joint type assumed)	Welding position
Current				
Voltage				
Travel Speed				
Electrode Type and Size				
Arc Length				
Gas mixture				
Gas flow rate				
Robot tool angle				
Robot tool-to-mtl. gap				
Slope of workpiece joint				
Dwell time				
Number of passes				
Weave pattern				

Figure C.1: Effect of product variables on process variables

The table below is to study the relationship between dependent process parameters. Listed below are the categories of dependency. If you feel there is a dependency of any of the variables indicate by a letter. The possible responses are: A: Complete dependency B: Strong dependency C: Little dependency D: No dependency E: Not known

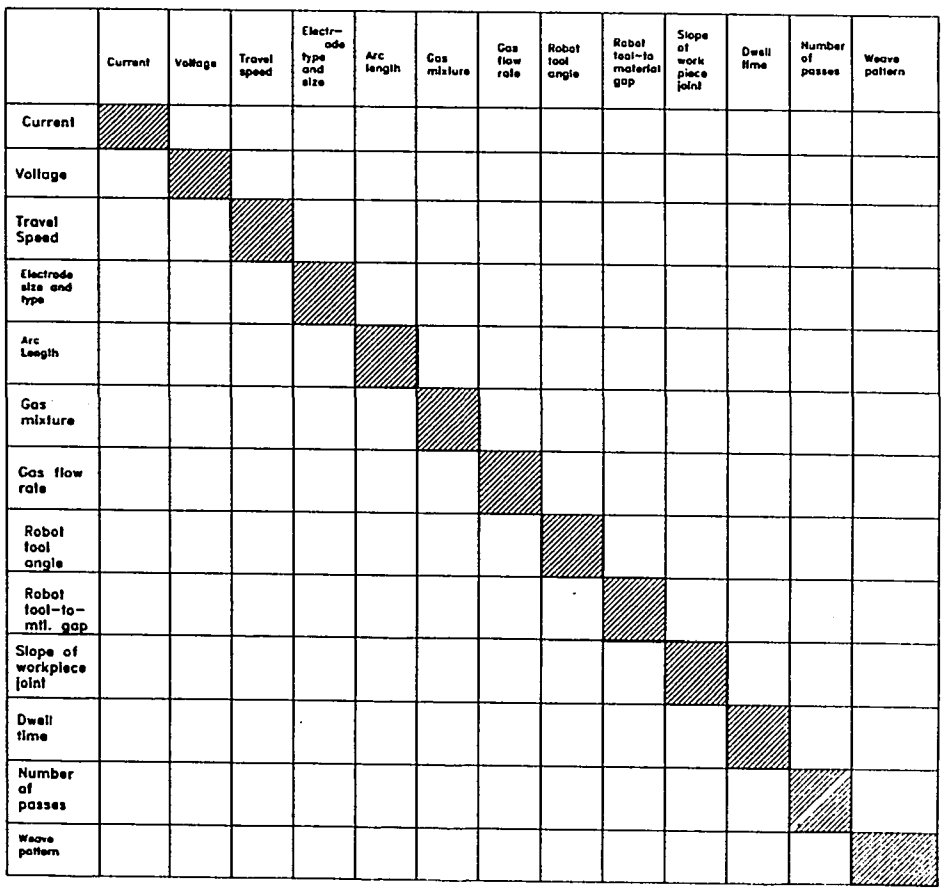


Figure C.2: Interdependency of process variables

APPENDIX D. EXPERTS' COMMENTS ON SURVEY RESULTS

The following are the actual comments from five different experts (welding engineers) on the conclusions sent to them as part of the validation process. Most of them have answered them point by point. Refer to Chapter 4 for the conclusions sent.

EXPERT 1

General Conclusions

1. OK.

2. Insert the word "visual" before adaptive. Human "adaptive controls" not only correct in imperfections in process specifications but also in the area of workpiece variability (i.e. fitup). Fitup variations (caused by component part variation and perhaps marginal fixturing) is a major problem with GMAW in the manufacturing or construction machinery and agricultural machinery.

3. Maybe. Not totally clear.

4. Not necessarily. I have changed my opinion in this area. Some of us involved in the welding of mild steel tend to become skilled at optimizing within a narrow range of materials. Other weld engineers might typically work in a firm which welds many types of materials such as nickel alloys, aluminum alloys,

copper alloys, stainless steels, titanium alloys. They would know how to set process specifications to weld many types of materials. They probably would not have the time to optimize for absolute maximum productivity however. Certain governing codes may not allow them much opportunity to optimize. Such engineers would normally be employed by a firm which does work in the aerospace, aircraft, defense, food processing and similar industries. I would guess that your survey did not include many of these engineers.

5. OK - may include other parameters as dictated by the situation.

Specific Conclusions

1. Specification of material type generally dictates strength (i.e. ASTM A572 Grade 50 HRS indicates a 50 ksi yield hot rolled steel plate). Does "surface appearance" mean surface preparation? For example, in the case of hot rolled steel does surface preparation mean cold rolled, hot rolled, hot rolled pickled and oiled, etc.?

2. Interrelationships of process variables do exist. In general, welding engineers have a gut feel of these relationships. Quantification of the relationships is a subject of some research at this point in time.

3. OK.

4. OK.

5. OK. It is disappointing you did not get responses from some engineers.

6. OK.

7. OK.

8. OK.

Recommendations

1. OK. It will be difficult to do much generalization without an enormous amount of research.

2. Not sure.

3. The general approach has not been very scientific. Those employed in welding mild steel tend to "wing it" and hope. Those involved with welding more exotic alloys normally have to take a more scientific approach to setting process parameters.

EXPERT 2

General Conclusions

1. I strongly agree Interrelationships do exist with process variables.

2. R&D is very important for process improvements but does not get to do it.

Recommendations

1. It has to be cost effective, proven and still there would be doubts among experts that it works as it is after all a computer.

2. I agree but it may not be possible

EXPERT 3

General Conclusions

1. True.

2. Robotic arc welding requires and takes advantage of a wider range of weld process variations compared to that of manual arc welding during a given weld cycle. Manual arc welding depends solely upon an operator experience in coordinating hand/eye motion.

3. True.

4. True

5. True.

Specific Conclusions

1. Not completely.

2. Not clear.

3. True

4. True

5. True

6. True

7. True

Recommendations

1. How can a vision system possibly check for all of the listed weld conditions. By the time the algorithm for all this was developed and integrated in some form of component, it would be more cost effective to use other methods. This statement is unjustified and impractical.

2. True. May never end and produce satisfactory results.

3. Welding is as much of an art as a science. Scientists are continually trying to provide definition to that which has previously considered abstract. Scientist create definition by either gathering information through long hours of empirical study or try to audit opinions from so called experts to gain some consensus. Each so called "expert" has empirically determined that there are too many total process variables to try to monitor and control. They have empirically determined what process variables are important and thus need to be used as trigger variables to control the majority of the effects of the experts' specific welding application. Each weld application varies from one another. Each experts approach varies from one another for a given application and so consensus development is extremely difficult.

EXPERT 4

Welding as well as all natural phenomenon obey laws. Therefore if we are clever enough we can find out what they are and produce a science based system to specify welding processes. However, there is a commonly held belief that welding is not a science, that it is an art to the exclusion of being a science. There are several reasons for this belief:

1. Welding parameters are inter-related. For example weld penetration is affected by the interaction of: wire feed speed, voltage, gun angle, gun to workpiece distance, wire diameter, travel speed, and shield gas mixture.

2. Little understood or difficult to visualize physical laws sometimes dramatically affect welding. An example of this is magnetic arc blow.

3. Some parameters have the opposite effect than expected. This is because the effects of an unexpected law is greater than the effect of the expected rule. An example is the effect voltage has on penetration. Many welders expect to see an increase in penetration because increasing voltage increases energy input. However, increased voltage results in increased arc length which spreads the heat energy over an increased area resulting in less penetration.

4. Weld equipment may not be repeatable enough.Robotic equipment is getting better all the time but welding power supplies for the most part use relatively old SCR technology. I suspect that SCR power supplies are not repeatable enough in some welding situations. This then appears to give a non repeatable weld. The new inverter style power supplies may improve on this situation.

5. An electric arc is violent and not necessarily steady state. Most models used to describe welding assume steady state conditions. I am not sure, but I suspect that the transients found in the arc may at times be significant.

6. The welding environment is hostile to feedback transducers which make it difficult to gather and validate data.

7. Feedback as to weld penetration pattern is difficult to obtain apart from actually cutting up the weldment. This is often prohibitively expensive so experimentation for the sake of knowledge is not done.

8. Welding robots are designed to be programmed by trial and error. As a result weld paths are programmed by guessing. This results in a robot weld often looking worse than a hand welders weld. The hand welder of course is constantly receiving feedback, and constantly changing weld gun angle and wire stickout. A 1 degree change in gun angle can be the difference between a good weld and a bad weld.

Even though the above mentioned problems make the goal of a science-based welding difficult, I do believe progress can be made and the benefit of such a system would be great. A relatively simple thing such as a robot with a weld path and gun angle that could be geometrically described and programmed would be of tremendous value.

APPENDIX E. WELDING CONDITIONS FOR MILD STEEL

The following are tables of process conditions for welding mild steel. These have been adapted from the Welding handbook and manufacturer's handbook.

Table E.1: Mild Steel and low alloy steel wires: Chemical composition requirements (Percent – balance iron

AWS CLASS.	L-TEC DESIG-NATION	CARBON	MANG.	SILICON	SULFUR	PHOS.	MOLYB.	OTHER
E70S-1	–	0.07-0.19	0.90-1.40	0.30-0.50	0.035	0.025	–	–
E70S-2	65	0.06	0.90-1.40	0.40-0.70	0.035	0.025	–	0.05-0.15 TITAN. 0.02-0.12 ZIRC. 0.05-0.15 ALUM.
E70S-3	82	0.06-0.15	0.90-1.40	0.45-0.70	0.035	0.025	–	–
E70S-4	85	0.07-0.15	0.90-1.40	0.65-0.85	0.035	0.025	–	–
E70S-5	–	0.07-0.19	0.90-1.40	0.30-0.60	0.035	0.025	–	0.50-0.90 ALUM.
E70S-6	86	0.07-0.15	1.40-1.85	0.80-1.15	0.035	0.025	–	–
E70S-1B	83	0.07-0.12	1.60-2.10	0.50-0.80	0.035	0.025	0.40-0.60	–
E70S-G	85A	NO CHEMICAL REQUIREMENTS						

Table E.2: Mild Steel and low alloy steel wires: Mechanical property requirements

AWS CLASS	L-TEC DESIG-NATION	SHIELDING GAS	CURRENT & POLARITY	TENSILE STRENGTH min. PSI ($kg\ mm^2$)	YIELD STRENGTH @ 0.2% offset min. PSI ($kg\ mm^2$)	ELONG. IN 2 IN. MIN. %	MINIMUM "V" NOTCH IMPACT
E70S-1	–	Ar 1-5% O_2[1]	DCRP	72.000 (50.6)	60.000 (42.2)	22	NOT SPEC.
E70S-2	65	Ar 1.5% O_2[1] CO_2	DCRP	72.000 (50.6)	60.000 (42.2)	22	20 FT LB @ -20^{o} F (2.8 KG-M) (-17.8^{o} C)
E70S-3	82	Ar 1-5% O_2[1] CO_2	DCRP	72.000 (50.6)	60.000 (42.2)	22	20 FT LB @ 0^{o} F (2.8 KG-M) (-17.8^{o} C)
E70S-4	85	CO_2[1]	DCRP	72.000 (50.6)	60.000 (42.2)	22	NOT SPEC.
E70S-5	–	CO_2[1]	DCRP	72.000 (50.6)	60.000 (42.2)	22	NOT SPEC.
E70S-6	86	CO_2[1]	DCRP	72.000 (50.6)	60.000 (42.2)	22	20 FT LB@ -20^{o} F (2.8 KG-M) (-28.9^{o} C)
E70S-1B	83	CO_2[1]	DCRP	72.000 (50.6)	60.000 (42.2)	17	20 FT LB@ -20^{o} F (2.8 KG-M) (-28.9^{o} C)
E70S-G	85A	NOT SPEC.	NOT SPEC.	72.000 (50.6)	60.000 (42.2)	22	NOT SPEC.

[1] Ar-CO_2 CAN BE USED FOR ALL WIRES.
*AWS A5.18-69

Table E.3: Welding conditions for mild steel short-circuit arc and E70S-3 wire

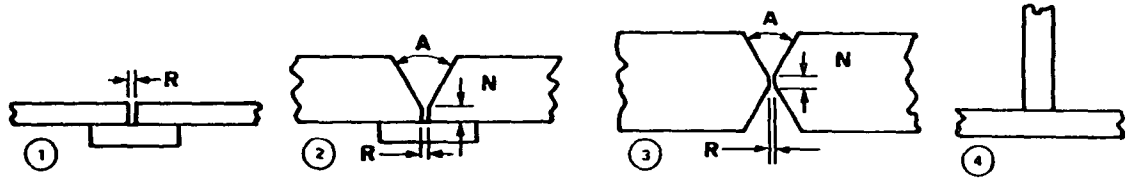


PLATE THICK.		WELD POSITION	JOINT DESIGN	(R) ROOT OPENING		(N) NOSE THICK		FILLER METAL DIA.		WIRE FEED SPEED		(4) VOLTAGE	AMPS DCRP	TRAVEL SPEED		NO OF PASSES
IN.	mm			IN.	mm	IN.	mm	IN	mm	IPM	mm/sec			IPM	mm/sec	
025	64	F,H,V[3],O	1 & 4	0				030	(76)	110 120	47 51	13 14	45 50	20 25	8 11	1
037	94	F,H,V[3],O	1 & 4	0				030	(76)	125 135	53 57	13 14	55 60	20 25	8 11	1
		F,H,V[3],O	1 & 4	0				035	(89)	110 120	47 51	15 16	70 75	30 35	13 15	1
		F	1	1/32	(79)			035	(89)	180 190	76 80	16 17	110 115	25 30	11 13	1
		H	1	1/32	(79)			035	(89)	170 180	72 76	16 17	105 110	25 30	11 13	1
0625	16		4					035	(89)	180 190	76 80	16 17	110 115	23 28	10 12	1
		V[3],O	1	1/32	(79)			035	(89)	140 150	59 63	15 16	85 90	13 18	5 8	1
			4					035	(89)	145 155	61 66	15 16	90 95	23 28	10 12	1
		F	1	1/32	(79)			035	(89)	265 275	112 116	18 20	150 155	14 19	6 8	1
			1	1/32	(79)			045	(1 1)	150 160	63 68	18 19	160 165	15 20	6 8	1
		H	1	1/32	(79)			035	(89)	220 230	93 97	17 18	130 135	13 18	5 8	1
125	32		4					035	(89)	270 280	114 118	18 20	155 160	23 28	10 12	1
			4					045	(1 1)	175 185	74 78	18 20	175 185	25 30	11 13	1
		V[3],O	1	1/32	(79)			035	(89)	220 230	93 97	17 18	130 135	13 18	5 8	1
			4					035	(89)	220 230	93 97	17 19	130 135	18 23	8 10	1
		F	1	3/16	(4 8)			045	(1 1)	220 230	93 97	19 20	210 215	15 20	6 10	1
			2	3/32	(2 4)	1/16	(1 6)	045	(1 1)	220 230	93 97	19 20	210 215	13 18	5 10	1
		H	4					045	(1 1)	210 225	89 95	19 21	210 215	15 20	6 8	1
1875	48		1	3/16	(4 8)			045	(1 1)	180 190	76 80	18 20	175 185	12 17	5 7	2
			2	3/32	(2 4)	1/16	(1 6)	045	(1 1)	180 190	76 80	18 20	175 185	15 20	6 8	1
		V[3],O	2	3/32	(2 4)	1/16	(1 6)	035	(89)	200 210	85 89	17 18	120125	10 15	4 6	2
			4					035	(89)	240 250	102 106	17 19	140 145	13 18	5 8	1

Table E.4: Welding conditions for mild steel short-circuit arc and E70S-3 wire continued

PLATE THICK.		WELD	JOINT DE-	(R) ROOT OPENING		(N) NOSE THICK.		FILLER METAL DIA.		WIRE FEED SPEED		(4)	AMPS	TRAVEL SPEED		NO OF
IN.	mm	POSITION	SIGN	IN	mm	IN.	mm	IN.	mm	IPM	mm/sec.	VOLTAGE	DCRP	IPM	mm/sec.	PASSES
		F	2	3/32	(2 4)	1/16	(1 6)	045	(1 1)	235 245	99 104	20 21	220 225	12 17	5 7	2
		H	2	3/32	(2 4)	1/16	(1 6)	045	(1 1)	180 190	76 80	18 20	175 185	8 13	3 5	2
250	6 4		4					045	(1 1)	235 245	99 104	20 21	220 225	8 13	3 5	1
		V,O	2	3/32	(2 4)	1/16	(1 6)	035	(89)	200 210	85 89	17 18	120 125	6 8	2 3	2
			4					035	(89)	240 250	102 106	18 19	140 145	11 16	5 7	2
		V	4					035	(89)	220 230	93 97	17 19	130 135	4 6	2 3	1
		H	2	3/32	(2 4)	1/16	(1 6)	045	(1 1)	180 190	76 80	18 20	175 185	12 17	5 7	4
			4					045	(1 1)	235 245	99 104	20 21	220 225	8 13	3 5	2
375	9 5	V	2	3/32	(2 4)	1/16	(1 6)	035	(89)	270 280	114 118	19 20	150 155	13 18	5 8	2
														5 7	2 3	1
			4					035	(89)	270 280	114 118	19 20	150 155	4 6	2 3	2
		O	4 & 2	3/32	(2 4)	1/16	(1 6)	035	(89)	290 300	123 127	19 21	165 175	9 14	4 6	3
		H	3	3/32	(2 4)	1/16	(1 6)	045	(1 1)	180 190	76 80	18 20	175 185	8 13	3 5	4
			4					045	(1 1)	235 245	99 104	20 21	220 225	11 16	5 7	4
500	12 7	V	3	3/32	(2 4)	1/16	(1 6)	035	(89)	270 280	114 118	19 20	150 155	8 10	3 4	4
			4					035	(89)	270 280	114 118	19 20	150 155	11 16	5 7	2
														4 6	2 3	2
		O	4 & 2	3/32	(2 4)	1/16	(1 6)	035	(89)	290 300	123 127	19 21	165 175	8 13	3 5	5

Note 1 Included angle dependent on torch accessibility to root

Note 2 For lap welds, increase speed 10%

Note 3 Vertical down travel

Note 4 Arc voltage measured between the feed rolls and workpiece

Note 5 The above conditions are also applicable for welding stainless steel with allowance for higher voltages due to the helium based A 1025 gas and for the increased viscosity of the weld

Note 6 Shielding Gas Flow 35 40 cfh (991 1133 l/hr)

Note 7 A 45° 60°

Table E.5: Welding conditions for mild steel spray arc and E70S-3 wire

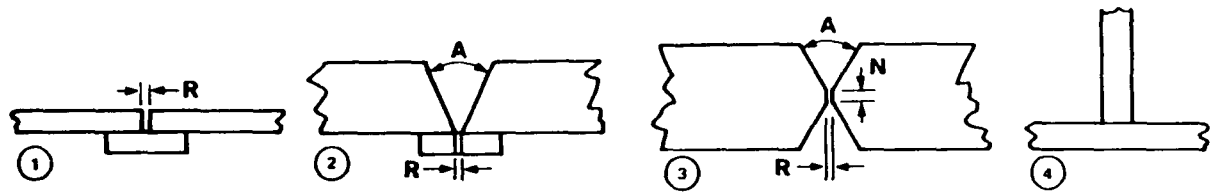


PLATE THICK.		JOINT DESIGN	(R) ROOT OPENING		(N) NOSE THICK.		FILLER METAL DIA.		WIRE FEED SPEED		(3) VOLT	AMPS DCRP	TRAVEL SPEED		NO OF PASSES
IN.	mm		IN.	mm	IN.	mm	IN.	mm	IPM	mm/sec.			IPM	mm/sec	
125	3.2	1	1/16	(1.6)			035	(.89)	350 375	148 159	26 27	190 200	20 25	8 11	1
		4					035	(.89)	375 400	159 169	26 27	200 210	30 35	13 15	1
250	6.4	1	3/16	(4.8)			1/16	(1.6)	185 195	78 82	26 27	310 320	8 13	3 5	1
		2	3/32	(2.4)			1/16	(1.6)	170 180	72 76	25 26	290 300	12 17	5 7	2
		2	3/32	(2.4)			045	(1.1)	400 425	169 180	29 31	320 330	17 22	7 9	2
		4					1/16	(1.6)	235 245	99 104	27 28	360 370	15 20	6 8	1
		4					045	(1.1)	425 450	180 190	30 32	330 340	14 19	6 8	1
375	9.5	2	3/32	(2.4)			1/16	(1.6)	215 225	91 95	26 27	340 350	11 16	5 7	2
		3	1/16	(1.6)	3/32	(2.4)	045	(1.1)	365 385	154 163	29 30	300 310	11 16	5 7	2
		3	1/16	(1.6)	3/32	(2.4)	1/16	(1.6)	170 180	72 76	25 26	290 300	10 15	4 6	2
		4					1/16	(1.6)	205 215	87 91	26 27	300 340	10 15	4 6	2
500	12.7	2					1/16	(1.6)	195 210	82 89	26 27	320 330	17 22	7 9	4
		3	1/16	(1.6)	3/32	(2.4)	1/16	(1.6)	185 195	78 82	26 27	310 320	17 22	7 9	4
		4					1/16	(1.6)	235 245	99 104	27 28	360 370	15 20	6 8	3
625	15.9	3	1/16	(1.6)	3/32	(2.4)	1/16	(1.6)	195 210	82 89	26 27	320 330	13 18	5 8	4
		4					1/16	(1.6)	215 225	91 95	27 28	340 350	13 18	5 8	4
750	19.1	3	1/16	(1.6)	3/32	(2.4)	1/16	(1.6)	195 210	82 89	26 27	320 330	11 16	5 7	4
		4					1/16	(1.6)	235 245	99 104	27 28	360 370	10 15	4 6	6

Note 1 Included angle dependent on torch accessibility to root

Note 2 For lap welds, increase speed 10%

Note 3 Arc voltage measured between the feed rolls and workpiece

Note 4 The above conditions are also applicable for welding stainless steel with an AR 1% O_2 shielding gas mixture

Note 5 Shielding Gas Flow 40 50 cfh (1133 1415 lph)

Note 6 A 45° 60°

APPENDIX F. KINEMATICS OF GE P50 ROBOT

The forward and inverse kinematics of the GE P50 robot is explained in this appendix.

Forward Kinematics

Based on the definitions in Chapter 6, the overall transformation matrix that describes the tool with respect to the base of the robot is given by:

$$ {}^{0}_{6}T = \begin{bmatrix} l_x & m_x & n_x & P_x \\ l_y & m_y & n_y & P_y \\ l_z & m_z & n_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.1} $$

where

$$ \begin{aligned} l_x &= (C\theta_1 C\theta_{234} C\theta_5 - S\theta_1 S\theta_5 - C\theta_1 S\theta_{234})/\sqrt{(2)} \\ l_y &= (S\theta_1 C\theta_{234} C\theta_5 + C\theta_1 S\theta_5 - S\theta_1 S\theta_{234})/\sqrt{(2)} \\ l_z &= (-S\theta_{234} C\theta_5 - C\theta_{234})/\sqrt{(2)} \end{aligned} \tag{F.2} $$

$$ m_x = -C\theta_1 C\theta_{234} S\theta_5 - S\theta_1 C\theta_5 $$

$$
\begin{aligned}
m_y &= -S\theta_1 C\theta_{234} S\theta_5 + C\theta_1 C\theta_5 \\
m_z &= S\theta_{234} S\theta_5
\end{aligned}
\tag{F.3}
$$

$$
\begin{aligned}
n_x &= (C\theta_1 C\theta_{234} C\theta_5 - S\theta_1 S\theta_5 + C\theta_1 S\theta_{234})/\sqrt{(2)} \\
n_y &= (S\theta_1 C\theta_{234} C\theta_5 + C\theta_1 S\theta_5 + S\theta_1 S\theta_{234})/\sqrt{(2)} \\
n_z &= (-S\theta_{234} C\theta_5 + C\theta_{234})/\sqrt{(2)}
\end{aligned}
\tag{F.4}
$$

$$
\begin{aligned}
P_x &= 16.0 C\theta_1 S\theta_{234} + 23.5 C\theta_1 C\theta_2 + 33.5 C\theta_1 C\theta_{23} \\
P_y &= 16.0 S\theta_1 S\theta_{234} + 23.5 S\theta_1 C\theta_2 + 33.5 S\theta_1 C\theta_{23} \\
P_z &= 16.0 C\theta_{234} - 23.5 S\theta_2 - 33.5 S\theta_{23}
\end{aligned}
\tag{F.5}
$$

This effectively describes the position and orientation of the tool frame from the joint vector.

Inverse Kinematics

The following is the method of obtaining the joint angles of the GE P50 robot given the position and orientation. Since the transformation matrix between the wrist frame and the tool frame is a constant, it is not considered in the calculation of the inverse kinematics.

$$
{}^0_5T = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T
\tag{F.6}
$$

If we let

$$ {}^{0}_{5}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & px \\ r_{21} & r_{22} & r_{23} & py \\ r_{31} & r_{32} & r_{33} & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.7} $$

and premulitply both sides by ${}^{1}_{0}T^{-1}$, we have

$$ {}^{0}_{1}T^{-1}{}^{0}_{5}T \;=\; {}^{1}_{2}T^{2}_{3}T^{3}_{4}T^{4}_{5}T \tag{F.8} $$

where the left hand side is

$$ \begin{bmatrix} C_1r_{11} + S_1r_{21} & C_1r_{12} + S_1r_{22} & C_1r_{13} + S_1r_23 & C_1px + S_1py \\ -S_1r_{11} + C_1r_{21} & -S_1r_{12} + C_1r_{22} & -S_1r_{13} + C_1r_{23} & -S_1px + C_1py \\ -r_{31} & -r_{32} & -r_{33} & -pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.9} $$

and the right hand side is given by

$$ {}^{0}_{5}T = \begin{bmatrix} * & * & S_{234} & * \\ & * & -C_{234} & * \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 01 & \end{bmatrix} \tag{F.10} $$

Equating the (3,4) elements, we get

$$ -S_1px + C_1py \;=\; 0 \tag{F.11} $$

which gives us

$$\theta_1 = Atan2(p_y, p_x) \tag{F.12}$$

Equating the (3,1) and (3,2) elements we get

$$\begin{aligned} S_5 &= -S_1 r_{11} + C_1 r_{21} \\ C_5 &= -S_1 r_{12} + C_1 r_{22} \end{aligned} \tag{F.13}$$

from which we calculate θ_5 as

$$\theta_5 = Atan2(r_{21}C_1 - r_{11}S_1, r_{22}C_1 - r_{12}S_1) \tag{F.14}$$

Equating the (2,3) and (1,2) elements we get

$$\begin{aligned} C_{234} &= r_{33} \\ S_{234} &= C_1 r_{13} + S_1 r_{23} \end{aligned} \tag{F.15}$$

which leads to

$$\theta_{234} = Atan2(r_{13}C_1 + r_{23}S_1, r_{33}) \tag{F.16}$$

To solve for the individual angles θ_2, θ_3, and θ_4, we will take a geometric approach. Figure 14.1 shows the plane of the arm with point A at joint axis 2, point B at joint axis 3, and point C at joint axis 4.

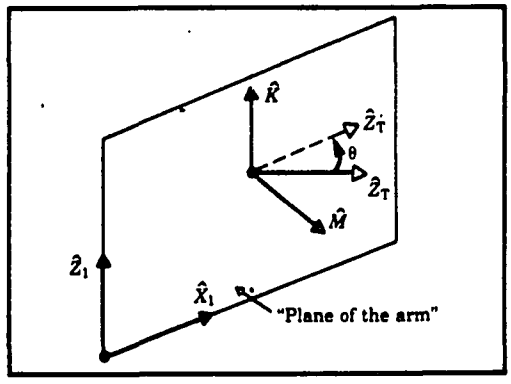


Figure F.1: The plane of the GE P50 robot

From the law of cosines applied to triangle ABC we have

$$Cos\theta_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2l_2l_3} \tag{F.17}$$

Then we have

$$\theta_3 = Atan2(\sqrt{(1 - C^2\theta_3)}, C\theta_3) \tag{F.18}$$

From the figure we see that

$$\begin{aligned} \theta_2 &= -Atan2(P_x, \sqrt{P_x^2 + p_y^2}) - Atan2(l_3S\theta_3, l_2 + l_3C\theta_3) \\ \theta_4 &= \theta_{234} - \theta_2 - \theta_3 \end{aligned} \tag{F.19}$$