

ROBOTIC AUTOMATION OF WELDING

A Thesis

by

ROMAN A. YODER

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Chair of Committee,	Prabhakar Pagilla
Co-Chair of Committee,	Swaroop Darbha
Committee Members,	Sivakumar Rathinam
	Behood Zoghi
Head of Department,	Bryan Rasmussen

December 2023

Major Subject: Mechanical Engineering

Copyright 2023 Roman A. Yoder

ABSTRACT

Welding can be defined as the joining of two or more materials by means of pressure or heat. In the last half-century significant interest has risen in the application of robots and collaborative robots (Cobots) for automated welding. The benefits of robotic welders are that they never tire, achieving a higher work efficiency than human welders. Further, they achieve more consistent results- impervious to distractions, operating at a constant speed with a higher precision. However, robotic solutions can not replicate the decision making process and adaptability of the human worker. Extensive use of logical operators, replicating the decision process, must be embedded in the solution.

This overall document introduces the efforts of a year long investigation into the implementation of an automated welding solution; utilizing a Miller 352 MPa and UR10e based system; achieving a throughput with a 0.524 cm/s average travel speed for 235.92 cm of weld length over two separated grids, in a prototype part. With a weld travel speed of 0.6 cm/s this implies that 87% of the time in a tour is spent executing the welds. This project presents a proof of concept, exploring the difficulties in automating a traditionally manual task. The results also made apparent the need for improved process control and robust performance of the system.

The outline of the document is as follows, starting with a introduction into the problem and relevant literature, followed by the methodology of the solution, a review of the current results, and a discussion about these results, possible improvements, and the future scope of the project.

DEDICATION

To my Parents, for the tools they gave me to follow my dreams.

ACKNOWLEDGMENTS

Special thanks to Exosent Engineering for the in depth look at their manufacturing process.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Prabhakar Pagilla, Darbha Swaroop and Sivakumar Rathinam of the Department of Mechanical Engineering and Professor Behbood Zoghi of the Department of Engineering Technology and Industrial Distribution. Thanks to PhD. student Sangam Chapagain who designed prototype parts and aided in running the experiments.

Funding Sources

Funding support for this project was provided by the RADAR project and James J. Cain Professorship II in the Department of Mechanical Engineering. The RADAR (Robotics and Automation Decision Framework for Agility and Resilience) project is a partnership between the Advanced Robotics Manufacturing (ARM) Institute and Texas A&M Engineering Experiment Station. RADARs funding is provided by the American Rescue Act and is part of a larger initiative by NIST (National Institute for Standards and Technology) to award high-impact projects for pandemic response research and development across eight manufacturing institutes in the Manufacturing USA network.

NOMENCLATURE

TCP	Tool Center Point
gui	Graphical User Interface
MIP	Mixed Integer Program
LP	Linear Program
Cobots	Collaborative Robots
HEZ	Heat Effective Zone
TSP	Travelling Salesman Problem
ICP	Iterative Closest Point
ipm	inches per minute
cfh	cubic feet per hour
RRT	Rapidly-exploring Random Trees
$\lceil x \rceil$	Round Function for x
T_v	specified Tool Vector for TCP frame
\hat{n}	normal, utilized for calculation of orientation and offset
O_v	orientation vector
Θ_d	n-dimensional desired joint value array
$V, V_{weldnodes}$	set of all nodes or weld nodes for a given graph
X	set of all possible path decision variables between every weld node
$X_{n,m}$	set of path decision variables between weld node n and m
E, E_w	set of graph edges

e_n	generic path or edges between two weld nodes, containing intermediate nodes
s_n	safety point associated with point n
D_w	set of all desired weld lines
d_w	given weld line
T_a	total angle change associated with a weld line or trajectory

TABLE OF CONTENTS

	Page
ABSTRACT	ii
DEDICATION.....	iii
ACKNOWLEDGMENTS	iv
CONTRIBUTORS AND FUNDING SOURCES	v
NOMENCLATURE	vi
TABLE OF CONTENTS	viii
LIST OF FIGURES	x
LIST OF TABLES.....	xiii
1. INTRODUCTION AND LITERATURE REVIEW	1
1.1 Introduction	1
1.2 Literature Review	5
1.2.1 Welding Overview	5
1.2.2 Path and Motion Planning	8
1.2.3 Current Solutions	10
2. SOLUTION METHODOLOGY FOR ROBOTIC AUTOMATION OF WELDING	11
2.1 Experimental Setup	11
2.1.1 External Control.....	13
2.1.2 Part Matching and Weld Selection	16
2.1.3 Prototype Test Part	20
2.2 Path Generation	21
2.2.1 Normal Generation.....	22
2.2.2 Multi-Movement Planar Point-to-Point Path Generation	23
2.2.3 Weld Line Paths	26
2.3 Robotic Trajectory Generation	29
2.3.1 Joint Limits	29
2.3.2 Inverse Kinematics Solver	35
2.3.3 Normals to Orientations	36
2.3.4 Sigmoid Curves, B-splines, and Polynomial Interpolation.....	38
2.3.4.1 B-Spline Application to Joint Angles	51

2.4	Point Ordering-TSP	52
2.4.1	Graph Generation	53
2.4.2	Tack TSP	54
2.4.3	Weld TSP	55
2.4.3.1	MIP Equations	61
2.4.3.2	On-line Solver	66
3.	RESULTS.....	68
3.1	Parameter Test.....	68
3.2	Grid Prototype Welds	71
4.	DISCUSSION AND CONCLUSION	74
4.1	Discussion	74
4.2	Future Scope of The Project/Improvements	75
4.2.1	Selected TCP Frame	75
4.2.2	Welding Process Control	76
4.2.3	Additional Weld Trajectories, Paths.....	77
4.2.4	Improving Cost Estimation MIP	78
4.2.5	Improving Hardware	79
4.3	Conclusion	79
	REFERENCES	80
	APPENDIX A. PRELIMINARY RESULTS AND REFERENCES.....	85
A.1	Reference Figures	85
A.2	Original Point-to-Point Path Generation	85
A.3	Original Part Results	89
A.4	Weld Tip Re-calibration	91
A.5	Polynomial Curve Fitting.....	92
	APPENDIX B. EQUIPMENT.....	95
B.1	Relevant Manual Pages	95
B.2	Component List	102
B.3	Additional Equipment Images.....	102

LIST OF FIGURES

FIGURE	Page
1.1 Arc Welding Types with MIG (a) and TIG (b) Reprinted from [1].....	6
2.1 UR10e with Tregaskiss BA1 MIG Welding Gun Attachment	12
2.2 Miller MIG System	12
2.3 UR10e with BA1 Weld Gun with Work-Piece	14
2.4 Arduino Communication Diagram	15
2.5 Miller Provided System Diagram	15
2.6 Measurement Hardware	17
2.7 Before and After Selection of Weld Lines	19
2.8 Before and After Selection of Part Points for Orientation Matching	19
2.9 Terminal User Input to Register Matching Points	19
2.10 Exosent Lattice Grid Model	20
2.11 Prototype Part: a minimalist representation of the Exosent part.....	21
2.12 PyVista Calculated Surface Normal for Point 0	22
2.13 Scaled Normal Unit Vector	22
2.14 Standard Point to Point Movements.....	24
2.15 Coordinate Frames of TCP and Base Link	25
2.16 Control Distance Defined from the TCP to the Weld Piece	26
2.17 Spline Pathway for Weld Line.....	29
2.18 Visual Simplification of Rotational Transforms	30
2.19 Wrist Zero Configuration ($q_5 = q_4 = q_3 = 0^\circ$)	32
2.20 End Effector Configurations	33

2.21	Sigmoid Function with b Varied from 1 to 32	39
2.22	Sigmoid Joint Angle Solutions with $b=7$ and $\dot{\theta} = \frac{\pi}{s}$	42
2.23	Sigmoid Joint Angle Solutions with $b=10$ and $\dot{\theta} = \frac{\pi}{s}$	43
2.24	Weld Angles Fit to Sigmoid Function with $b=3$	44
2.25	Polynomial Solutions for Timed Joint Angles	46
2.26	Generated Trajectory Spline Fit Drop Down to a Desired Weld Point	47
2.27	Generated Trajectory Spline Fit X-Y Plane Reorientation	48
2.28	Generated Trajectory Spline Fit Lift Up End Effector	49
2.29	Generated Weld Trajectory Spline Fit	50
2.30	LKH Tack TSP	53
2.31	Discrete Graph Representations: with node locations at desired points [29]	59
2.32	Discrete Graph Kamada-Kawai Representations [29]	60
2.33	Python MIP Weld Tour	61
2.34	Paths Between $(u, v) \in V_{weldnodes}$: required weld edges in red	62
3.1	Long Weld Parameter Test	70
3.2	Consecutive Weld Parameter Test	70
3.3	Prototype Part with Two Grids Welded	71
3.4	Grid One Welds	72
3.5	Corner to Corner Welds with Porosity	73
4.1	Crater Timing: available settings	77
A.1	Welding Types Reprinted from [1]: here DCEP/DCEN stands for (Direct Current Electrode Positive/Negative)	85
A.2	Part V.1 Relative to UR10e	86
A.3	Part V.1 Represented as a Complete Graph (not every connection is drawn for simplicity): red identifies the internal subsets	87

A.4	Tall and Short Path Scaling for Part V.1: here D3 is the diagonal distance inside a grid, P1 is the plat height and H1 is the height of the parts ridges. Red Point:Low Control Point, Black Point: High Control Point	87
A.5	Original Part Control Point Paths	88
A.6	Original TSP Tacking Solution.....	89
A.7	Original Mixed Integer Program Solution to Weld TSP	89
A.8	Warping of the Original $\frac{1}{8}$ Inch Thick Weld Grid	90
A.9	Misaligned Welds of the Original $\frac{1}{8}$ Inch Thick Weld Grid	91
A.10	Burn-through Welds of the Original $\frac{1}{8}$ Inch thick Weld Grid	91
A.11	Quoted BA1 Error	92
B.1	Provided Miller Connection Diagram.....	95
B.2	Provided Miller Connection Information	96
B.3	Provided TCP for BA1	97
B.4	Wire Feeder Common Problems	98
B.5	Recommended Shielding Gas	99
B.6	Common Welder Issues and Solutions	100
B.7	Miller Arc Control	101
B.8	Miller Invision 352 MPa.....	103
B.9	Miller S-74 MPa, with crater and start functions	104
B.10	Miller Insight Module	105
B.11	Assembled System, with wire feeder on the left, and the S-74, Insight Module and Invision under the table, on the table is the real-size grid model	105
B.12	Under-side of Weld Table with: Invision 352, Insight Module, S-74, and UR10e controller	106
B.13	Under-side of Weld Table: labeled	106

LIST OF TABLES

TABLE	Page
3.1 Table of Weld Parameters	68
3.2 Table of Robotic Hyper-Parameters.....	69
B.1 Component List	102

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Robotic Welding has been in development since the inception of industrial robotic automation. Recent scientific and social shifts have introduced a new paradigm in this technology. This includes applying collaborative or co-robotic (cobot) solutions to small manufacturers, mitigating skilled labor shortages and improving industrial productivity.

Robotics presents an advantage over traditional skilled labor because it is consistent and efficient. A robot can replicate trajectories at a given velocity with greater precision and accuracy. This results in consistent weld quality; reducing cost of material waste associated with human error. Full industrial robotic automation also removes the human element from the welding operation, reducing the risk of injury. However, some manufacturers that do not have the means to implement a high level of industrial automation may still utilize manual welding. Cobots are safer compared to industrial robots for hybridized manufacturers. These manufacturers still have a large amount of labor, but can benefit from the use of automation with cobots; being cheaper and safer than industrial robots. This is because cobots are designed to work closely and safely with and around human operators.

To implement robotic welding, it is necessary to identify its challenges and limits. Robots are not intuitive, and cannot reason like a human; they can only make decisions based on programmable logic and provided information. The decisions of the robot are based around a set of provided orientations and positions. The goal being that these positions and orientations map the robot to the location of the desired workpiece and welds. For ease of use, a geometric model is placed at a given transformation in the workspace. The desired welds are identified relative to the part origin and mapped to the robots workspace; with these welds selected by the user.

Registration of the work-piece is the act of taking the position data of the physical part in the robots workspace to estimate its relative position and orientation. This position data is often ac-

quired with use of the robotic linkage, through either a sensor or user input. Sensors or vision based systems are difficult to implement, driving up cost and complexity. Cobotic welding solutions make use of the weld gun as a pointer to a registered position. The user positions a dedicated weld tip to gather accurate and precise position information of the physical part.

This registered position data, regardless of acquisition means, has associated errors that must be confined by the part matching algorithm such that, the estimated transformation results in appropriate weld locations. This error is also related to the amount of registered points and the space in which they occupy, requiring intelligent selection.

To move from weld to weld, the robot must be provided with a path between these points, in the order they are to be performed. These paths must be sufficiently smooth, such that any robotic trajectory along the path results in a continuous motion with minimal jerk. Parametric splines or polynomials can be fit to a given set of control points and constrained to be continuous up to a given degree. These control points must contain both point and orientation information to guide the Tool Center Point (TCP) from weld to weld and along each weld, to correctly deposit the weld material. The required orientation of these welds should be determined via the part geometry and user identification of weld lines in the part geometry. Given the size and geometry of a weld piece and its transformation, it is imperative that the paths guide the robot along a collision-free trajectory. Significant work has been done on these path finding methods, but all require the ability to check and eliminate points for collision avoidance.

The set of collision-free paths allows the robot to map its set of joint angles to the position and orientation data at discrete path points, spaced out in time. The overall jerk of the robot depends on the instantaneous change in joint angle acceleration along these path points. It is desirable that the robot smoothly accelerates through a joint angle solution set of a given path. As there are multiple joint angle solutions to a given point and orientation it is important that the solutions along a given path belong to the same joint angle subset: that is not an altered geometric solution. The geometric solutions along the path must position each link to avoid self or part collision. Multiple joint angle solutions can exist for an identical geometric solution dependent on the acceptable joint

angle range and must be filtered to avoid instantaneous joint angle changes of 2π .

The order in which the welds are visited plays an important role in the performance advantage of the automated process over the manual; but the effect of this performance advantage may vary. Most robotic solutions employ some form of predetermined traveling salesman problem (TSP) or online decision process to order point and orientation targets given a set of goals. A part with more welds will gain more competitive advantage from solving the optimal order of the welds compared to a smaller part, given a limited computation time, as this represents a fixed cost to the travel time associated with completing the part. This also involves the decision whether a method of selecting the next weld sequentially after each completed weld out competes that of a predetermined order requiring an initial solution time, and under what circumstances.

Weld parameters are selected by manual laborers through experience and knowledge based on material, thickness and weld geometry in a test and validation method. If the robot is to perform quality welds it is important that the given weld parameters are appropriately selected for a given work piece. The most common of which are wire feed rate and voltage. These parameters should be remotely controlled, with the weld equipment matching the desired values. Additionally the remote welder must utilize appropriate filler material, filler wire diameter, and gas supply for the weld piece. This welder should be easily integrated into the overarching cobotic system and hardware.

Over the past year work has been done to facilitate the development of an experimental apparatus capable of performing robust welds and integrating with a remotely controlled UR10e cobot. This project presents a proof of concept, exploring the difficulties in automating a traditionally manual task. The goals being to optimize an ordered set of required tacking points and weld lines, and solve the trajectory generation, utilizing smooth continuous paths. The weld points have been ordered with a formulation of a custom mixed integer program and, online next nearest point or greedy algorithm. The welder utilized is an Invision 352 MPa remote welder with cobot interfacing from Miller and a Tregaskiss BA1 Weld Gun fixed to the end of the UR10e. The results to date show the successful implementation of a simplified welding program able to handle planar-linear

welds. This is contingent upon a given geometric model of the workpiece, proper identification of weld points and accurate registration of the part in the workspace via positioning of the TCP.

1.2 Literature Review

Necessary to propose a solution to the problem of automated welding is a review of the relevant information. This first covers an overview of welding related material science, what contributes to a quality weld, and an overview of path planning and industrial robotic welding.

1.2.1 Welding Overview

Welding can be classified by being either arc, spot, gas, friction, beam or etc.– with arc welding being the most prolific in the welding industry. The use of electricity being popular due to its rapid heating and processing that reduces the overall heat effective zone (HEZ) of the weld piece, with the size of HEZ inversely proportional to input power [1].

Arc welding utilizes an applied current to generate heat, liquefying a supplied metal deposition to join two or more metal surfaces. Historically, industrial welding robotics have mainly utilized spot welding, popular in the automotive industry [2] with the average automobile containing approximately 5000 spot welds [1]. Spot welding passes a current or arc between two electrodes, moving through the air and then the target material, joining the separate materials together. In recent years the use of robotic arc welding has risen drastically. The most common types of arc welding are known as Metal Inert Gas (MIG) and Tungsten Inert Gas (TIG) seen in figure 1.1. Additionally an overview of all electrode supplied weld types can be found in figure A.1.

The primary difference between these two is that MIG welding, unlike TIG, uses a consumable electrode as a filler metal. This makes MIG welding more amendable to robotic welding as the gun can be designed as a single self contained end effector, whereas TIG may require an additional apparatus, robotic arm, or even human–robot interactions. Another application of robotic welding utilizes a laser to join the metal surfaces. This has the advantage of being extremely precise with respect to the HEZ; unfortunately these systems require significant upfront capital investment making them infeasible for all but large economies of scale. MIG therefore demonstrates itself as the front runner for providing smaller manufacturers the ability to automate due to its cost effectiveness and simplicity.

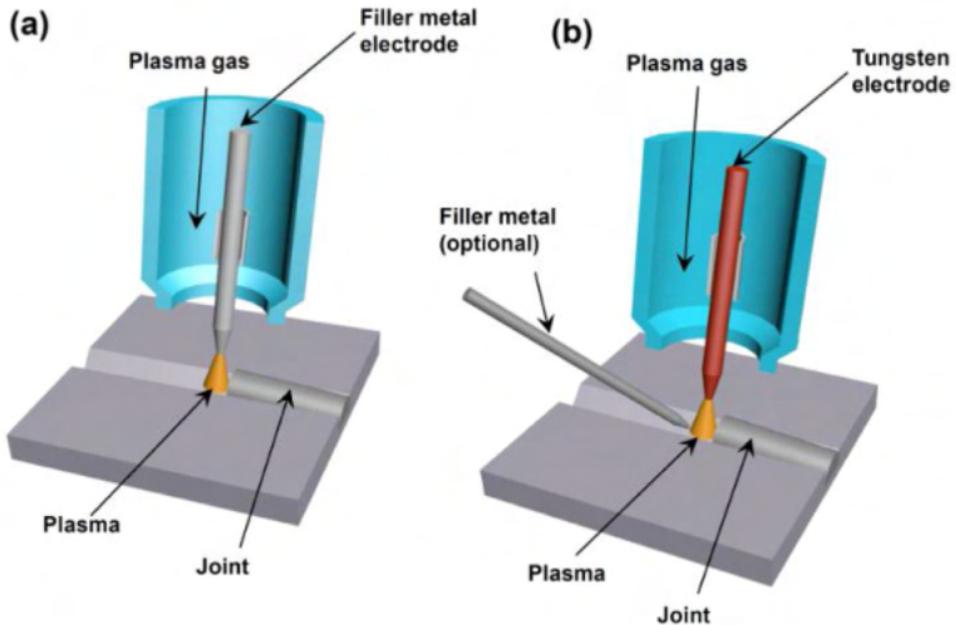


Figure 1.1: Arc Welding Types with MIG (a) and TIG (b) Reprinted from [1]

A multitude of parameters influence the weld quality, varying for each type of welding. The Tregaskiss manual for the BA1 reviews possible quality issues and related parameters. For example, one common problem is that of contact tip burnback— which is the occurrence of the MIG wire burning back to the exit tip faster than the wire can be fed. This occurs because the wire itself is arcing at the contact tip inside the nozzle, welding itself inside the tip [3]. An immediate solution to this is to increase the wire feed rate, to overcome the rate of burnback. This also depends on the tension setting of the MIG wire feeder— too high of a tension may create unnecessary strain on the motor, via the drive rollers and too low will result in improper feeding [4]. The magnitude of the supplied current affects the possibility of contact tip burnback. Given a higher current it is recommended to recess the contact tip further into the gun's nozzle creating more wire stick out, defined as the length from the tip of the weld gun to the weld. This moves the arc further away from the contact tip, reducing its heating and in turn extending its life as the consumables resistance increases proportional to temperature. Recessing the contact tip further also allows an improved gas

flow [5]. Less recess typically allows the welding of hard-to-reach joints and grooves, but also runs the risk of arcing the consumables, like the contact tip to the work piece and damaging them.

Other common problems in weld quality include excessive spatter, weld porosity, and improper weld penetration. Spatter is the scattering of molten metal droplets away from the desired weld pool, these droplets can cause injury or damage by landing on nearby humans, flammable material, or even damaging the consumables in a MIG gun. Weld porosity is unsightly cavities in the weld that result in lower strength. Welds that have a porosity defect must be redone immediately and are at risk of collapse [6]. Weld penetration is defined by the depth at which the weld has infiltrated the material. Too deep can result in significant warping of the material, but a shallow weld results in joint weakness and an improper bond [7]. Spatter formation can be impacted by the quality of materials used, if cheaper materials are used with weldable filler metals this will impact spatter. Spatter is also influenced by the voltage and current combination at a particular wire and gas combination. Parodying Ohms Law, if the voltage is too high or the current too low the resistance through the weld will be affected and the weld pool and wire won't be molten, stubbing the wire to create spatter. Spatter is additionally influenced by parameters: weld angle—geometry affects gas coverage and heat transfer, arc length—too long of an arc length can cause the metal droplets to impact the weld pool and create a splash, or loss of shielding gas protecting the weld pool— which also effects porosity [6]. Porosity is created by the presence of nitrogen, oxygen and hydrogen around the weld pool; upon cooling these separate and create cavitation, which can cause spattering. Thus proper shielding gas flow rate, and mixture is important to weld quality and the avoidance of porosity. Too low of a flow rate will result in improper pool coverage, too high will create turbulence in the flow inducing mixing of atmospheric air and creating impurities in the shielding gas. It is important to always check and maintain gas diffusers and lines, to insure the proper supply. Penetration is directly effected by the heat supplied to the weld pool— depending on the current and voltage. Tuning all of the parameters to achieve optimal weld performance is an involved process. Typically one starts with a recommended specification for a type of weld and then further adjusts based on observation. The optimallity of these parameters will further

change between individual welds as each quality control issue is affected by gun travel speed, weld parameter, surface cleanliness, temperature, and more.

1.2.2 Path and Motion Planning

The Travelling Salesman Problem (TSP)

The Traveling Salesman Problem or TSP has been one of the most important combinatorial optimization problems in the last century. The problem can be simply stated as: given a set of cities or points, find the best tour to visit every point once and return to the start, minimizing travel distance. This corresponds to finding the minimum length element from the set of Hamiltonian circuits. The TSP has found relevant applications in everything from package delivery, distribution, network planning, chip design, and robotics [8].

This problem is of special relevance as given the set of weld points, or weld lines; it is desirable to execute these in the optimal order with consideration to distance or some other cost. Unfortunately this problem has proven to be NP-hard by Karp in 1972, meaning that a TSP problem can not be solved in polynomial time [8]. What this implies is that a brute force methodology applied to the TSP will scale exponentially with the cardinality of the input set. This brute force technique may be obvious and efficient for small sets of points, but the possible permutations of tours will scale by a factorial of the number of inputs.

Another naive, but feasible, approach to this problem is to employ a greedy algorithm- also known as the nearest neighbor heuristic. The concept being to take the next point, as that with the lowest cost metric from the set of remaining points. However, the cost associated with this solution does not scale by a constant factor with respect to the cardinality [9].

Path Planning

Path planning refers to the identification of collision-free paths between the desired position, orientation, or joint goals. This can be done by first identifying a set of safe collision-free points connecting the current point to the desired. Points may be sampled randomly from the environment, with some beneficial bias, and then repeatedly checked given a conditional specifying if the

point falls outside of a part geometry; like the RRT* algorithm [10]. Points that fail to exist outside the bounds of a given part geometry may be eliminated or the algorithm may be constricted to an acceptable area.

These points may also be found by geometrically processing the work piece. Delaney's algorithm, that constructs triangles from a set of points such that no non-shared point of another triangle is inside the circle drawn around any triangle's vertices, is a possible solution. Random sampling can be replaced with sampling from a discrete set of safe positions and orientations, constructing a graph that can be processed by Dijkstra's or A* [11]. In some cases path planning is performed by user input, in which a human guides the robot to safe points in the workspace. This process can be cumbersome and inflexible to variations in work piece parts or environment.

A set of collision-free points is not adequate information to achieve a desired point and orientation. The inverse kinematics (IK) problem must be solved, specifying the required joint angles to achieve the desired transformation. It is well-known that there exists more than one set of joint angles to a given position and orientation; for 6 degrees of freedom robotic manipulators, as the mapping is not injective. Along the collision-free path with specified orientations, care must be taken to select the corresponding joint angles that provide a smooth motion for each joint. If a joint has a range of 4π , angles 2π , 0, and -2π are radially identical. Instantaneous change from these geometrically identical solutions will result in an infinite desired jerk, along a joint angle trajectory. The result of such is an erratic and unpredictable motion. This is a result of the Newton-Raphson method utilized to solve the inverse kinematics problem, which will find the a local minimum in a continuous joint angle space [12]. The found solution must then be post processed to ensure that the desired joint angles along the trajectory connect, or belong to specific set of joint limits. Given a set of connecting joint angles, a smooth function must be found that accelerates the robot from its current position, towards the desired position and orientation, and decelerates settling into the desired configuration. Usually a piecewise continuous function, or regression in the form of splines and polynomials are fit to a desired joint angle trajectory, with constraints that the initial and final acceleration and velocity are zero.

1.2.3 Current Solutions

Although co-robotic welding is an emerging industry, there are current companies developing new implementations, even utilizing the same Universal Robot family. One such example of this is Hirebotics, who developed an integrated UR10e welding system [13]. A client, Vortex Companies, reported that the use of Hirebotic's cobot welder reduced weld times for parts that previously took an hour by hand, to around 12 minutes; a significant increase in throughput. This system however, does not have an ability to interpret parts or CAD files. Instead the robot is trained by personnel, to reuse registered welds taken by using two buttons on the welding end effector; one to move the robot freely and another to log points. Vortex Companies responded well to the simplicity of the system, being intuitive and programmable from a smart phone app. The system falls short in its reliance on repeatability of weld programs, they rely on jigs or clamps in the workspace to repeat identical welds for a trained program. This implementation, utilized the UR10e and a nearly identical Tregasskiss' weld gun.

Another example, of robotic welding is the optional solutions from Path Robotics [14]. Developing multiple innovative robotic welding solutions capable of intelligent path planning centered around provided part geometries. The AF1 employs three independent robotic manipulators. Two Universal Robots' manipulators position weld parts, picked from nearby bins, that are then welded with a Yaskawa manipulator fit with a Miller Auto Continuum 500 to weld. Capable of handling disorganized parts from the bin with variable tolerances, the solution reads and adapts its weld parameters based on sensor input. A laser based system embedded in the Yaskawa's weld gun aligns welds on-live for accurate weld positioning and parameter selection. The solution also makes use of a rotating fixture table to position the desired work piece adding an additional degree of freedom to the system. In-fact, Path Robotics have made use of gyroscopic fixture tables in welding solutions [15]. Rotating a table to assist the welder around a circumference of a curved surface. The implementation being a premium is sold as a subscription.

2. SOLUTION METHODOLOGY FOR ROBOTIC AUTOMATION OF WELDING

The Solution Methodology aims to outline the techniques and equipment utilized in the implementation. First the experimental set up is outlined, followed by the safe path generation around a given part geometry, the generation of joint angle trajectories for these paths, and the ordering of these paths corresponding with a TSP solution.

2.1 Experimental Setup

The Experimental setup of the robotic welder included a UR10e Cobot pictured in figure 2.1 and a Miller remote welding system based around Miller's Invision 352 MPa MIG welder, responsible for managing the power supply, and the Insight Core Module 14-Pin responsible for communicating with the central computer. These integrate with the 74 Series MPa Drive Assembly and the S-74 MPa Control Box that facilitate feeding the required welding wire and gas to Tregaskiss' BA1 Cobot Air-Cooled MIG Gun.

This setup utilized 0.035 inch diameter mild steel wire as its filler material and electrode; with the flexibility to weld material with variable thickness from 1.2 mm to 12.7 mm [16]. The welder made use of two weld modes, pulsed and standard spray MIG, the difference between these being that spray has a constant current supply, creating a stream of molten filler metal, and pulsed MIG modulates its current to deposit the filler metal in individual droplets. Pulsed MIG shows improvements in the HEZ and weld quality becoming the default operation mode for the project. The acquired welder also offers control over wire feed rate, voltage or arc length and lead in/out ramps to gradually increase/decrease voltage, current or feed rate at the start/end of a triggered weld.



Figure 2.1: UR10e with Tregaskiss BA1 MIG Welding Gun Attachment



Figure 2.2: Miller MIG System

2.1.1 External Control

The Automated Welding system provided by Miller is designed to be controlled directly from the UR10e's control box, wiring the Insight Module's inputs and outputs directly to the robot. This allows the programming via the pendant; this however is less flexible than external control via the Linux based ROS sudo-operating system used. Although ROS allows easy integration of robotic systems such as sensors, motors, drivers and higher level planners; it is reliant on the functionality from previous developers, occasionally being depreciated. Because of this it was convenient to control the welder from a custom built Arduino circuit that operates as a independent ROS node.

The design of this control circuit was selected by probing the Insight Core Module, otherwise known as the control module. An image of the provided connection information can be seen in figure B.2. Note the 'JOG SOURCE' and 'JOG COMMAND', shorting these wires resulted in a forward jog of the wire feeder, extruding the wire out of the tip of the weld gun. This functionality was then supplemented with an Arduino controlled relay, powered by an independent 12 volt supply. The 'PURGE COMMAND', 'MODE COMMAND' and 'WELD ENABLE' were identically controlled with the purge command opening the gas line to the welding tip and 'WELD ENABLE' turning on the welder, which includes enabling voltage, current, gas flow, and feeding the wire. Then control was implemented for the voltage and wire feeder commands; shorting the wires for each of these resulted in a maxing out of the parameter, being Wire Feed Rate, or Voltage/Arc Length depending on the welding mode. Rather than utilizing a digitally controlled potentiometer, a variable voltage supply was controlled by the Arduino to surpass the provided voltage with a bias and achieve direct control.

With this in place the Arduino could then be programmed as an external ROS node subscribing to topics such as 'enable-weld', 'voltage', 'purge', etc. published via a Python script. The welder also outputs additional feedback parameters with the potential to be used for live parameter adjustments of the welder. These signals ranging from 0-10 Volts are not compatible with the 5 volt limit of the Arduino but could be adjusted with a voltage divider circuit. This was not implemented, being unnecessary for the initial proof of concept and parameter prototyping. A simplified electric



Figure 2.3: UR10e with BA1 Weld Gun with Work-Piece

communication diagram can be found for the Arduino controller in figure 2.4.

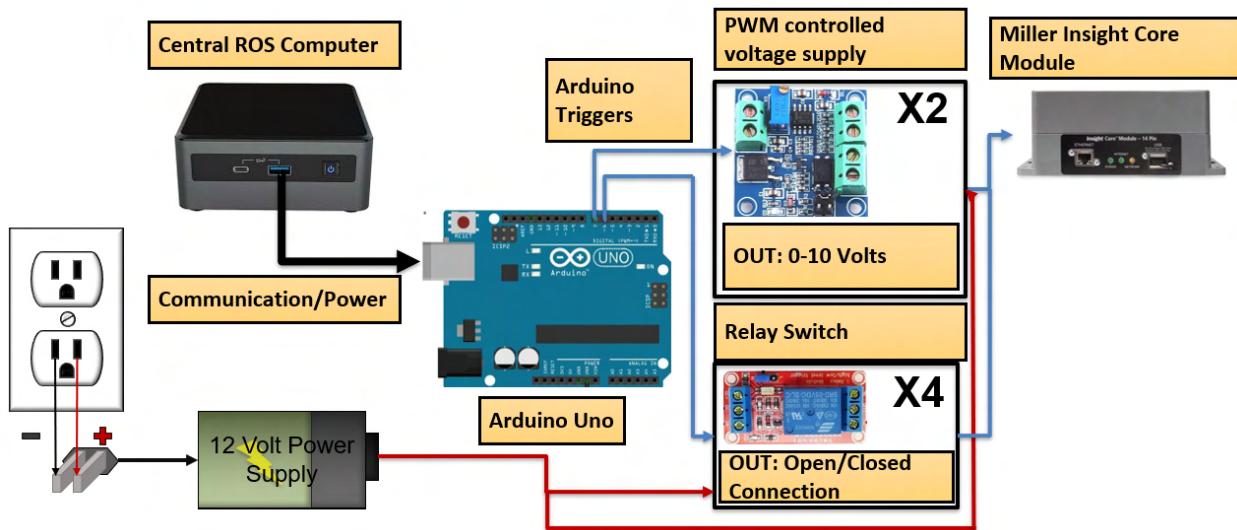


Figure 2.4: Arduino Communication Diagram

3-6. Equipment Connection Diagram With Cobot

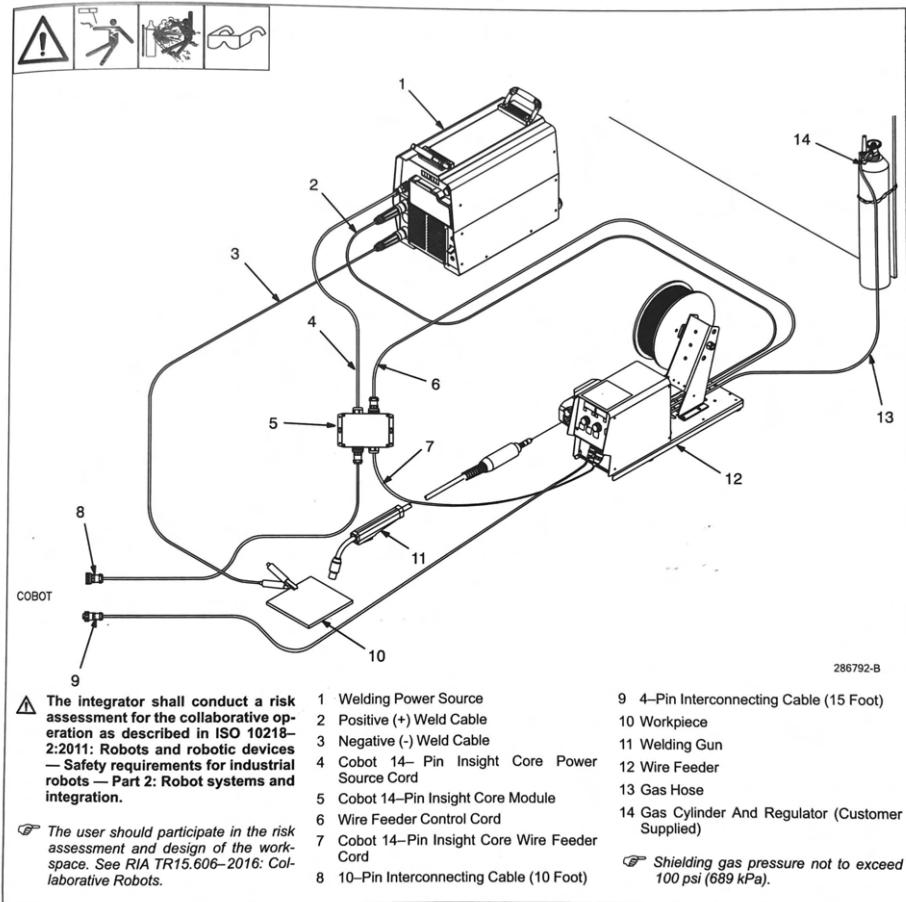


Figure 2.5: Miller Provided System Diagram

2.1.2 Part Matching and Weld Selection

Part Matching is necessary for the estimation of the work piece location to communicate the desired location of each weld point with the registration of user selected points on a geometric model of the part. Both the desired points to weld or tack and the points to match the model are selected by user input with the PyVista module, as in figures 2.7 and 2.8.

Given a set of selected points to match with in the digital twin the UR10e is switched into free drive mode. This allows the user to move the end effector by hand positioning the TCP in the digital points corresponding location in the work place. At each of these points an input from the computer terminal, as in figure 2.9, is used to trigger collection of the coordinate data of each point and input into an Iterative Closest Point (ICP) algorithm responsible for making the best possible estimation of the parts physical location [17].

Currently there exists no extended intelligence of the program to deterministically pick points that will lock the part into place; it has been demonstrated that the order the measured points are read in affects the accuracy of the ICP's results. For example, given a group of desired matching points selected from the digital part that all lie in the same plane; if these points are selected in a clockwise order and the read points from the physical workspace are sampled in a counter clockwise order the resulting transformation will be flipped upside down 180° from the desired transformation about the x or y-axis of the part frame.

To remedy this the user must understand to either read the points in the order matching the selection of points on the digital twin, or avoid rotational symmetry in the selection forcing the orientation to be correct given a read vertical point outside the plane of the other points—i.e. the set of vectors from all the possible combinations of the matching points must map to three dimensions.

The Registration of the points is done with the welding end effector equipped with a custom tip that facilities accurate point measurement with a tapered cone shown in figure 2.6.

Because this tool is a different length when compared to the nominal welding tip, this must be adjusted for within the program via a transformation along the tips orientation vector.

In fact, the entire program is heavily reliant on the precise estimation of the Tool Center Point



(a) Measurement tip



(b) Measurement tip in the BA1

Figure 2.6: Measurement Hardware

(TCP) that facilitates the point matching or welding. As witnessed in the development of the experimental apparatus, small errors in the TCP's estimation can result in carried error throughout the entire tacking and welding process. This was only evident due to the repeatability of these errors; being discovered by taking an array of points at variable orientations and revisiting them in a repeated orientation. If this repeated orientation was near the orientation of the read point the relative error from the physical TCP to the point was minimized; as these orientations diverged the error became a combination of the axis' displacement between the physical TCP and the estimated TCP in the digital model.

The culprit of this error being the differences between the quoted dimensions of the BA1 Weld Gun and the actual. This was resolved utilizing the onboard TCP estimation feature of the UR10e in which the user sets the tool tip in a constant location, a drilled hole, while varying the orientation of the robot. Each orientation is then noted and the pendent utilizes this to calculate a best estimation for the TCP. This is likely done through a form of triangulation, linking the read points of the current TSP point together and finding its average to estimate the novel.

During the point registration process, the robot is operated in free drive with the available button on the BA1 that has been programmed to enable the human operator to freely guide the tool tip to the desired point for registration. The allowable movement of the free drive can be restricted in a number of ways via programming with the UR10e Pendent and in this case the frame of the end effector is constrained such that it is always parallel with the negative of the global z-axis; this corresponds to the weld gun pointing directly downwards to weld on the surface of planar parts. Although this alignment with the z-axis may not be desirable in every general weld case, most parts can be welded adequately with this decrease in orientation freedom. The original TCP can be found in figure B.3. The offset from the flange was provided as (0, 0, 350) mm, the measured TCP was (-2.34, -5.5, 341.70) mm; representing a total error magnitude of 10.2282 mm, or a $\frac{10.2282}{350} = 2.92\%$ error. The visually observed error can be seen in figure A.11.

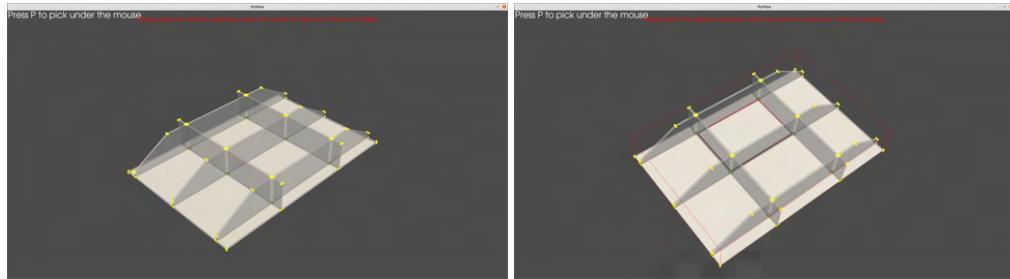


Figure 2.7: Before and After Selection of Weld Lines



Figure 2.8: Before and After Selection of Part Points for Orientation Matching

```
rylab@rylab-NUC8i7HVK: ~/ur10e_wp_2/src/weld_project/scripts 80x11
[ INFO] [1695940061.237642828]: Ready to take commands for planning group manipulator.
Initialization Arduino Communication:
Press Enter to Take a Point:
Read Point:
[ 0.47006797 -0.06497409 -0.01024709]
was that point correct[y/N]?:y
Press Enter to Take a Point:
Read Point:
[ 0.47006797 -0.06497409 -0.01024709]
was that point correct[y/N]?:
```

Figure 2.9: Terminal User Input to Register Matching Points

2.1.3 Prototype Test Part

The part in figure 2.11 represents the minimal interpretation of the part shown in figure 2.10, which was a mock part of a common bottle neck for a local gas transport trailer manufacturer. The prototype part is a significant reduction in material cost.

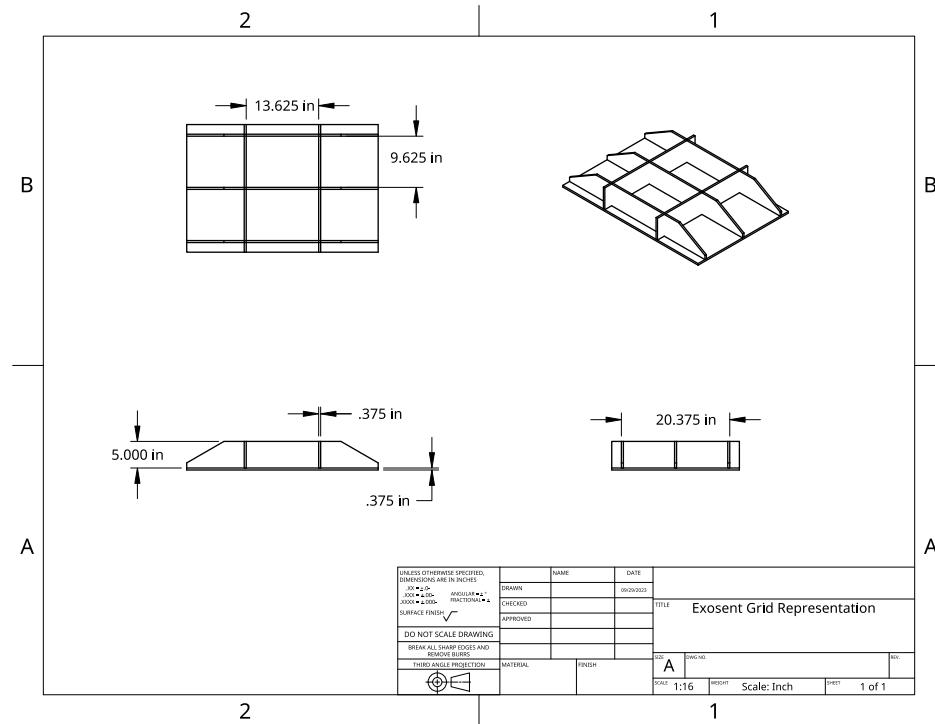


Figure 2.10: Exosent Lattice Grid Model

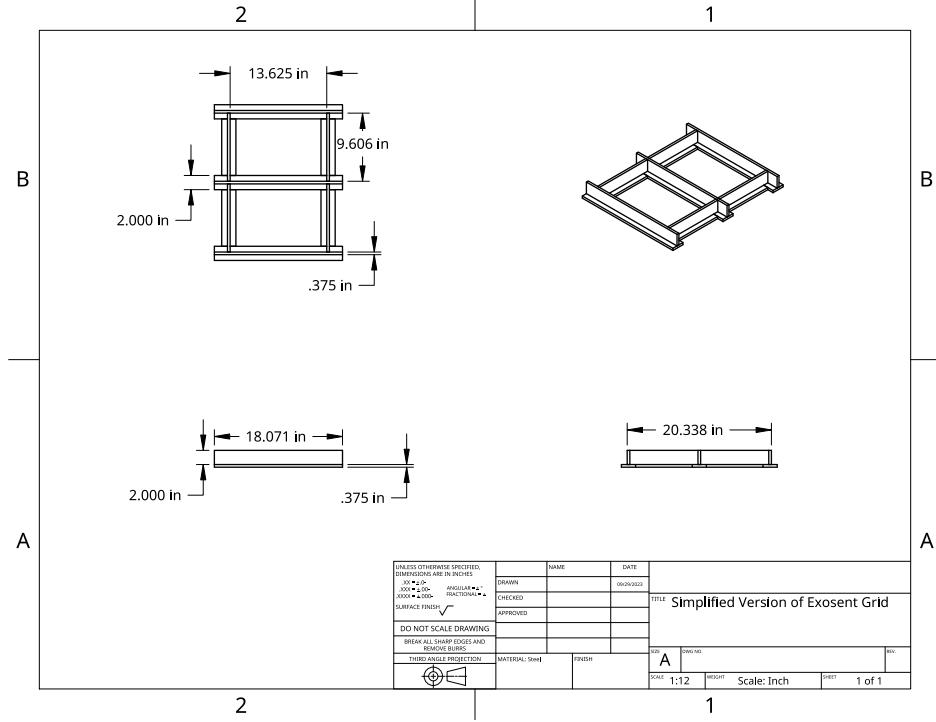


Figure 2.11: Prototype Part: a minimalist representation of the Exosent part

2.2 Path Generation

Path generation relies on the user to properly identify the desired weld and tack points. These are passed through the transformation provided by the ICP algorithm to represent them in the robots workspace. Then PyVista, a mesh handling Python module, is utilized to calculate surface normals on the mesh relative to the pre-transformed weld and tack points [18]. These surface normals were originally utilized to generate obstacle avoiding paths from point-to-point, but also correspond to the desired orientations. Path generation is done with the assumption that there is a universal height that offers clearance over the part as a whole. This clearance height is taken from reading the mesh dimensions via PyVista and multiplying by a safety factor. A comprehensive overview of the current method is given below.

2.2.1 Normal Generation

The normal generation is done at each desired point by summing the coincident plane's surface normals.

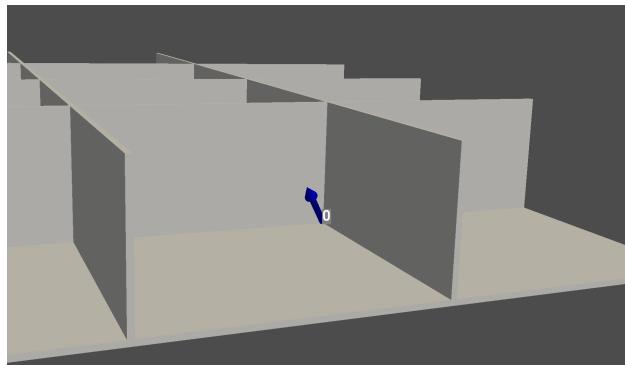


Figure 2.12: PyVista Calculated Surface Normal for Point 0

This normal calculation often returned a negative value in the vertical (z-axis) direction. This was manually overridden by setting the z component to be positive, then the vector was normalized by equally scaling the x-y and z components and dividing by the norm. Resulting in the vector shown in figure 2.13.

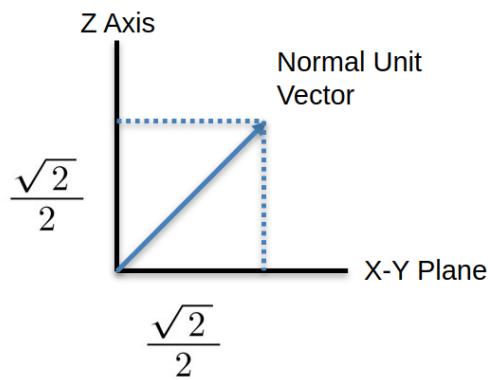


Figure 2.13: Scaled Normal Unit Vector

This normal, calculated via the point in the original mesh, may also be passed through the 3x3 rotation matrix in the calculated ICP transform. This normal is stored with the associated desired point, utilizing both to calculate a point above the vertical height of the part for safe transition or the desired orientation as discussed in section 2.3.3.

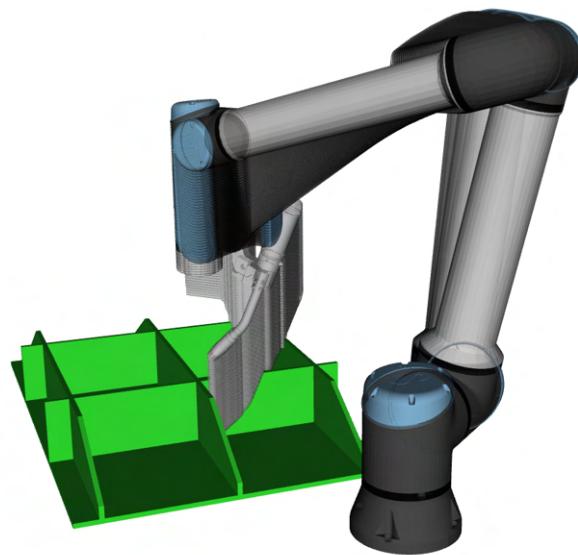
Smooth Paths were generated with splines implemented from the SciPy Python module [19]. These relied on an array of control points between both points in the Point-to-Point motion, weights designating the relative importance of passing through each point in the array, and a smoothness coefficient controlling the degree to which the spline differs from control points to achieve continuous low magnitude derivatives. To achieve precision and enforce continuity between consecutive paths, the weights of the start and end points are scaled several magnitudes larger than the intermediary; forcing this spline equation to closely approximate these points.

2.2.2 Multi-Movement Planar Point-to-Point Path Generation

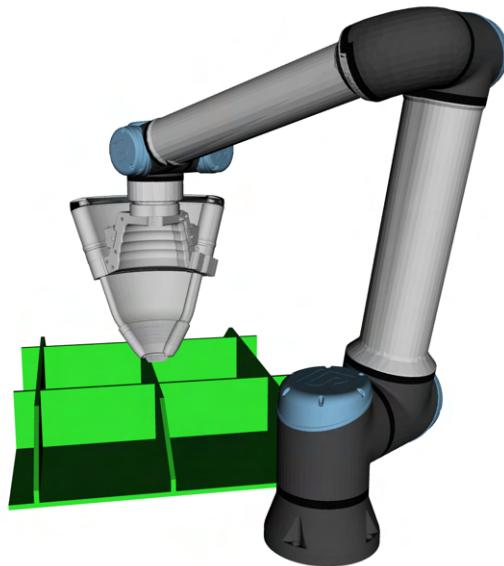
The weld program utilized straight line paths and a single height scaling plane directly above each desired tack or weld point. The robot moves between desired points by first lifting directly upwards and then translating directly above the next point in the raised x-y safety plane; followed by directly dropping down to this point in the orientation corresponding to the normal of that point. Robot Trajectories following these paths are shown in figure 2.18.

Additionally the trajectories must be generated for the associated weld paths of the part and are reliant on an python object that stores the necessary weld information associated with desired weld turns, oscillations etc.. This path is made relative to the identified weld line points, dependent on a control distance, safety distance, and normals at the start and finish of the weld.

Along the weld line the end effector or tip of the weld gun is maintained at a constant distance from the weld on the work piece itself; regardless of the orientations orthogonality to the weld line itself. This is to ensure that the influence from change of gas flow, arch length and other distance-influenced weld parameters are diminished. The variation of the orientation from start to finish is calculated via the normals of each point on the mesh and is so that the robot turns to get into the tight corners of common parts. As the robot turns the shortest distance from the tip itself to the



(a) Vertical trail



(b) Horizontal reorientation trail

Figure 2.14: Standard Point to Point Movements

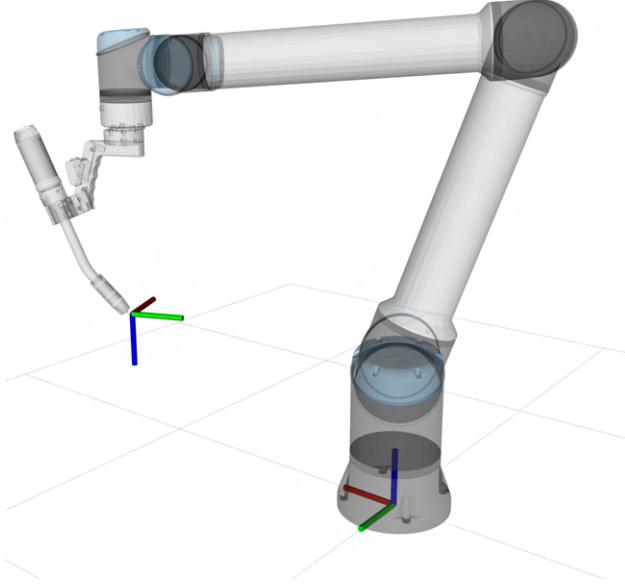


Figure 2.15: Coordinate Frames of TCP and Base Link

plane of welded wall reduces; being that length along the wire feeds vector is held constant. Figure 2.16 shows how this control distance relates to both the height and length of the TCP relative to the points the weld line is based off. Figure 2.15 also represents how the TCP frame is related to the base frame of the robot.

This can also be represented in equation form relative to the global coordinates with:

$$p_1 = [R_0^1] \hat{n}(CD) + p_0.$$

Here p_1 is the resulting TCP calculated by adding the normal vector scaled by the control distance and transformed by the representation of the part orientation in the workspace to the location of the part point p_0 in global coordinates.

The safety distance, not to be confused with control distance, influences the spacing over which the end effector rotates into a nominal orientation based on the direction of travel of the weld gun. All of the desired rotations apart from the nominal are linearly spaced out in a start and end section along the weld path with a length equal to the safety distance. This is to ensure the arm rotates into

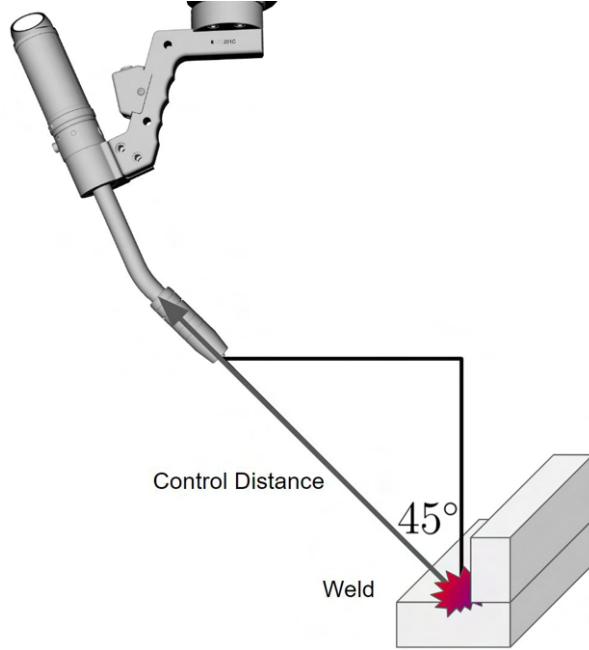


Figure 2.16: Control Distance Defined from the TCP to the Weld Piece

place before entering the corner of the desired weld. Further, given a longer weld, this allows the gun to travel in an optimized orientation for the quality of the weld. This may however result in trajectory issues down the road dependent on the overall timing of the robot along these paths.

2.2.3 Weld Line Paths

The planning for the weld line paths is handled by an individual Python object, storing and processing the associated data required to direct the robot along a desired weld path. This begins with the normals at the start and end of the trajectory,

$$\hat{n}_1, \hat{n}_2 \in \mathbb{R}^3$$

and each desired weld point.

$$p_1, p_2 \in \mathbb{R}^3$$

The line vector is then:

$$\vec{v}_l = p_2 - p_1.$$

Given a safety distance from the corners d_s and a interpolation resolution ρ , take the number of rotating points, rp_n , as a rounded integer based on the length of the line vector and safety distance:

$$rp_n = \lceil \rho \frac{d_s}{|\vec{v}_l|} \rceil, rp_n \in \mathbb{Z}^+.$$

Then take the sign S_z ,

$$[k_1, k_2, k_3] = \hat{n}_1 \times \hat{n}_2,$$

$$S_z = sgn(k_3).$$

The weld angles can then be spaced out by the interpolation resolution ρ and rp_n , taking three arrays to linearly space the corner to corner trajectory. This consists of the initial turn out of the weld corner, the portion of travel such that the y-axis of the TCP frame is perpendicular to the line vector, and then the turn into the corner. This is represented to the robot with linearly spaced points along each array, the total points in each array is proportional to the ratio of its length over the total weld length; with the first rotation having $rp_0 = rp_n$, the perpendicular portion having $lp_1 = \rho - 2 * rp_n$, and the last rotation having $rp_2 = rp_n$ indexes in the array. With the weld points spaced out as:

$$\Omega_v = [\omega_0, \omega_1, \omega_2]$$

s.t.

$$\omega_0 := [p_1, \dots, p_1 + \vec{v}_1(\frac{rp_0}{\rho}) : \forall n \in \{rp_0\}],$$

$$\omega_1 := [p_1 + \vec{v}_1(\frac{rp_0 + 1}{\rho}), \dots, p_1 + \vec{v}_1(\frac{rp_0 + lp_1}{\rho}) : \forall n \in \{lp_1\}],$$

$$\omega_2 := [p_1 + \vec{v}_1(\frac{rp_0 + lp_1}{\rho}, \dots, p_2 : \forall n \in \{rp_2\}].$$

Each of these position arrays are then coupled with a desired angle-orientation array, utilizing the sign S_z , corresponding to the direction of angle change, related to the weld direction of travel-

not the actual joint angle ¹:

$$B = [\beta_0, \beta_1, \beta_2]$$

$$\beta_0 = [0, \dots, (S_z)\frac{\pi}{4}, \forall n \in \{rp_0\}]$$

$$\beta_1 = [(S_z)\frac{\pi}{4}, \dots, (S_z)\frac{\pi}{4} : \forall n \in \{lp_1\}]$$

$$\beta_2 = [(S_z)\frac{\pi}{4}, \dots, S_z)\frac{\pi}{2} : \forall n \in \{rp_2\}]$$

Orientation vectors, and points adjusted by the control distance (CD) are specified for every angle and point pair at a shared index.

$$R_y(\theta) := \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The normal associated with each desired orientation is then defined as:

$$\hat{n}_d = R_y(\theta)\hat{n}_1$$

and the desired point for the TCP is:

$$p_d = p_n + (CD)\hat{n}_d$$

$$\forall \theta \in B, \omega \in \Omega_v.$$

Each normal \hat{n}_d and point p_d is then utilized in solving the IK problem storing the associated weld angles along the path.

¹This weld angle implementation is built around the assumption of an orthogonal grids normals

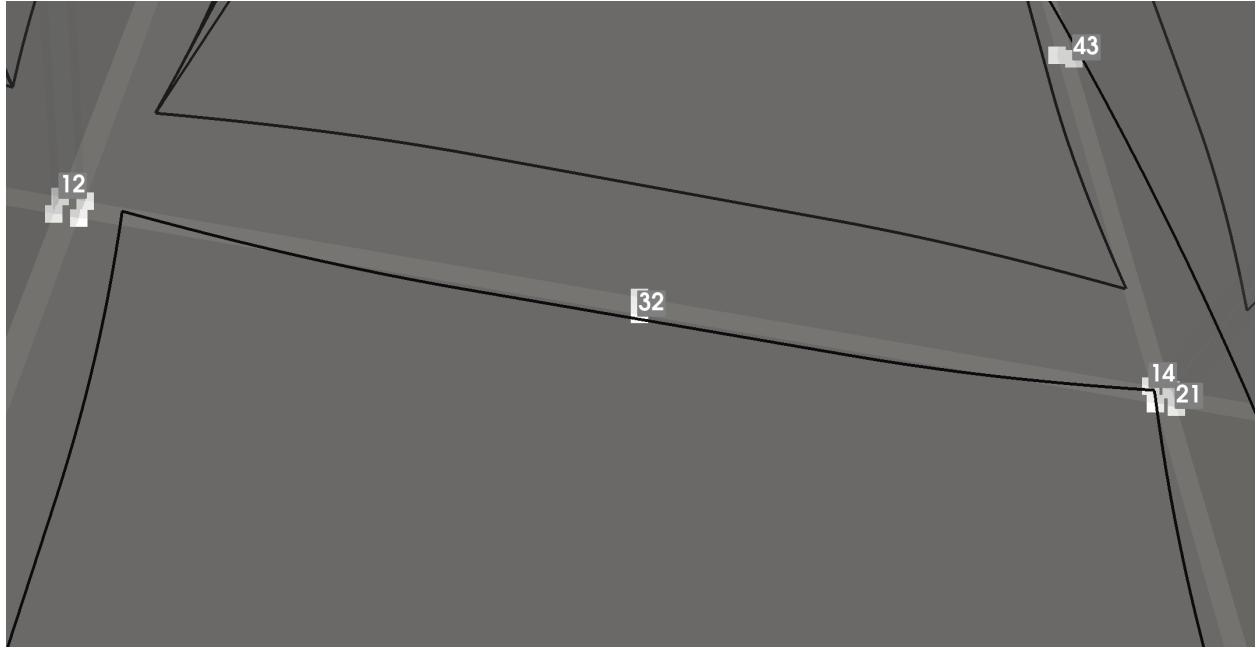


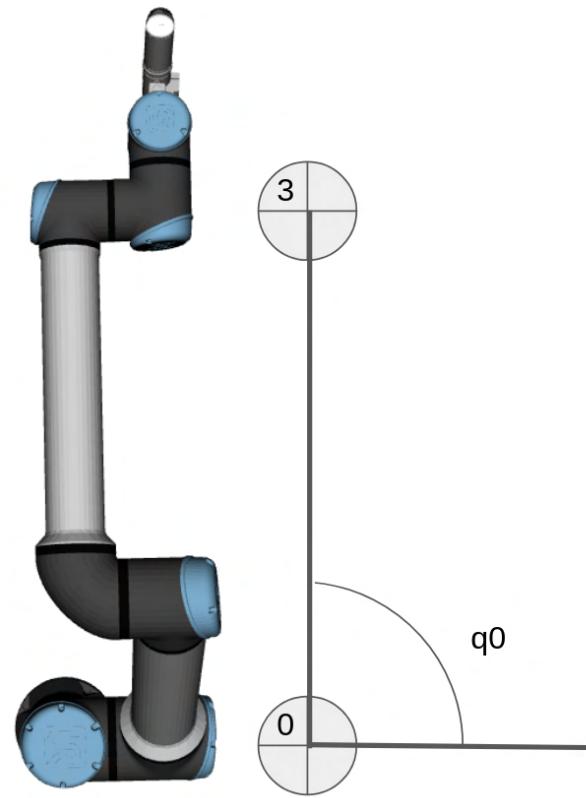
Figure 2.17: Spline Pathway for Weld Line

2.3 Robotic Trajectory Generation

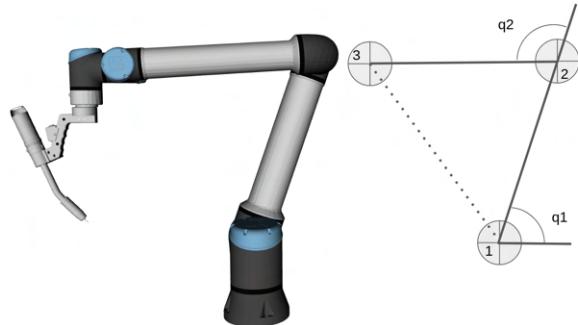
To follow desired paths a robotic trajectory must be computed and timed along each path. This involves solving the Inverse Kinematics (IK) that provide joint angles required to achieve a given point and orientation. Further these solutions must smoothly connect with each other along the path, and not break the required joint limits of the robot. This is done by specifying the joint limits to the IK Solver ensuring the joint angle solution exists within a certain subset of the joint angle space which can be referred to as a configuration type. Staying in this subset reduces or completely avoids instantaneous jumps in joint angle values along the path.

2.3.1 Joint Limits

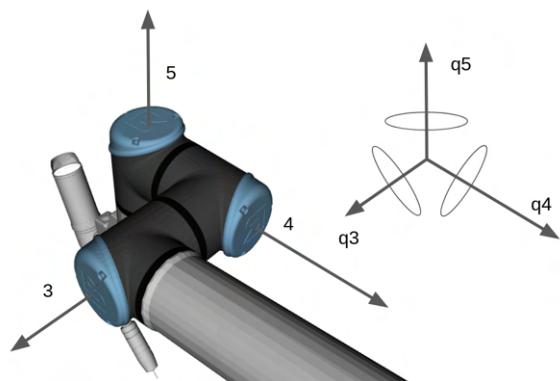
Multiple geometric solutions can produce the same end effector position and orientation. In graphic (b) of figure 2.18 an identical vector from 1 to 3 can be replicated by reflecting the vectors along the joint links across the axis of the vector; the q3 angle will then identically level the end effector. Further, ignoring the position, a final orientation assuming previous joint angles can be



(a) Base link rotation



(b) Planar positioning



(c) End effector orientation

Figure 2.18: Visual Simplification of Rotational Transforms

accomplished with multiple combinations of the yaw, pitch and roll represented in graphic (c). Given how this orientation is constructed it constrains the possible solutions of the horizontal vector which locates the vertical plane that adjusts the height and length of the end effector as in (b). Additionally to these multiple geometric solutions, there exist duplicates of each due to the standard joint limits of the UR10e being $+/-360^\circ$, creating 3 values for an absolute angle of 0° and 2 for every other. It would be hazardous if the connected IK solutions of the desired paths were to jump from either geometric class or any of the multiple values that create an identical absolute angle.

For the convenience of this solution the robot was constrained sequentially along each joint:

$$q_0 = \{\theta_0 : 2\pi \leq \theta_0 \leq \pi\}.$$

joint 0 is allowed its full range of 720° . Although IK solutions may have multiple solutions this is handled with rounding by integer values of π . The work piece to be welded is set upon a work table, because of this it is desirable that the 2nd joint, q_1 , does not dip below the plane of the table, the joint angle should be constrained such that this occurs. This is done relative to the initial angle of the joint, set between 0° and -180° . To avoid dropping below the coincident plane with the shoulder link the joint limits are set as:

$$q_1 = \{\theta_1 : -\pi \leq \theta_1 \leq 0\}.$$

A value of 0° for q_2 corresponds with the link being aligned with the previous link after the shoulder lift joint, q_1 . As a solution in the positive quadrant of q_2 can be replicated with a 180° rotation about q_0 and converting q_1 by subtracting its value from -180° ($q_1 = -\pi - \theta_1$), and a negation of the sign for q_2 ($q_2 = -\theta_2$). To avoid this quadrant flipping, in which the robot will reconfigure itself for a geometrically identical solution, q_2 is limited based on its initial value,

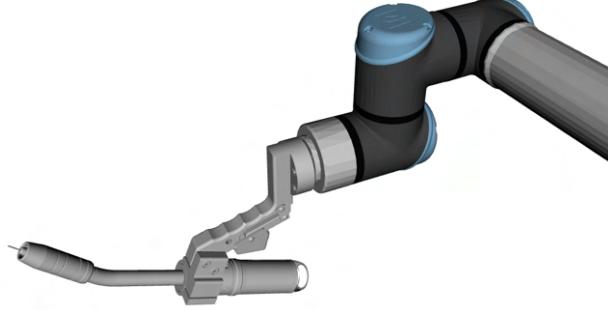


Figure 2.19: Wrist Zero Configuration ($q_5 = q_4 = q_3 = 0^\circ$)

which lies between positive and negative 180° given the assembly of the current UR10e.

$$q_2 = \{\theta_2 : 0 \leq \theta_2 \leq \pi\}, \quad \forall q_{2i} \leq 0$$

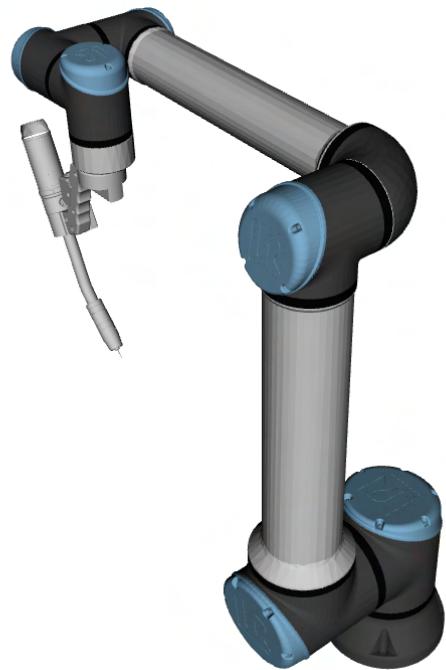
$$q_2 = \{\theta_2 : -\pi \leq \theta_2 \leq 0\}, \quad \forall q_{2i} > 0$$

This can be referred as either a positive or negative elbow joint. Figure 2.19 shows the final three joints of the wrists in the zero configuration where each joint angle is set to zero.

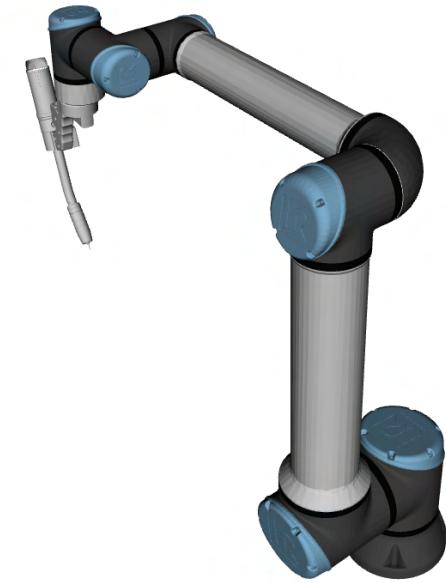
Figure 2.20 represents what are classified as the long and short configuration types. Assuming as was outlined in the point registration that the z-axis of the end-effector or TCP frame always remains parallel to the global z-axis; q_4 will always remain integer values of $\frac{\pi}{2}$ and q_3 , based on the configuration type, countering the induced orientation created by the previous joint angles to level and align the frame with the z-axis. In fact, a precise geometric relation could be written out for this angle. q_3 is then limited to a range of 180° based on its initial value, corresponding to the configuration type. First, the initial value of q_3 is divided by π and rounded both up and down to create upper and lower bounds θ_u and θ_l where:

$$\theta_u, \theta_l \in \{\Theta : \Theta = \pi \times \mathbf{Z}\},$$

$$q_3 = \{\theta_3 : \theta_l \leq \theta_3 \leq \theta_u\}.$$



(a) Short Configuration



(b) Long Configuration

Figure 2.20: End Effector Configurations

In initial program design both the long and short configuration types were tested to see if the IK solver could find solutions for each at closest and farthest point of the desired tack/weld points. It was then designed to opt for its current configuration (of which most times was physically long) and if a solution was found for both points remained in this configuration. If this solution could not be found it tried to find both solutions for a short configuration— which upon failing would quit the program. If found the IK solver would only find solutions for this configuration and the robot would have to transition to this on its own accord. However, the long configuration is superior in that there is a reduced risk of self collision and it can reach further distances, allowing the robot to operate further from the singularity at its base. Because of this the robot defaults to whatever configuration it reads from the joint angles of the physical robot when the program is run. Finally the end effector’s limits are implemented for q_5 . This was not initially done as early trials showed it possible for wrist 3 to travel its whole 720° without failure— later tests showed that the cord leading to the BA1 weld gun could occasionally be caught on the wrists of the robot and create a resisting torque that would cause the robot to fail. Wrist 3 of the robot was then centered about 0° and rotated in both directions until contacted was achieved between the wire feeding cable and the wrists; resulting in q_5 joint limits of:

$$q_5 = \{\theta_5 : -226.62^\circ \leq \theta_5 \leq 237.65^\circ\}.$$

With a total allowable wrist range of 464.27° . Fortunately this range is nearly centered around 0° and has an additional 104° of rotation over 360° allowing approximately 52° additional degrees of overlap in either direction. This is of significant concern to the initial design of the program being able to handle these novel joint limits; less than a full rotation of travel would force limitations on the robot’s ability to do weld lines in either one pass without resetting its configuration, depending on the severity of these restrictions it may impose that the robot has to flip from a positive elbow joint to a negative— an act that requires a significant amount of time/space. Performance not only depends on these joint limits but how these joint limits relate to the orientation and position of the

work piece. Rotation of this piece may cause some desired weld lines to be infeasible as the joint angle solutions are varied possibly violating the joint limits and requiring a reset. Fortunately, this was not an issue in the overall experimentation.

2.3.2 Inverse Kinematics Solver

The IK solver was not implemented by the research group itself; this is because the UR10e is a well known geometry for which *moveit* and other robotics libraries have developed optimized solvers [20]. Utilized for this was the *TRAC-IK* solver from TracLabs [21]. TracLabs mentions that this is an expansion of Orocoss' KDL (Kinematics and Dynamics Library) employing Newton's method to find the solution. Mentioned is that Newton's method can't strictly find solutions within a set of required joint limits. So instead, TracLabs quotes[22]:

TRAC-IK concurrently runs two IK implementations. One is a simple extension to KDLs Newton-based convergence algorithm that detects and mitigates local minima due to joint limits by random jumps. The second is an SQP (Sequential Quadratic Programming) nonlinear optimization approach which uses quasi-Newton methods that better handle joint limits. By default, the IK search returns immediately when either of these algorithms converges to an answer.

In implementation TRAC-IK operates as a class that accepts a desired orientation represented as a quaternion, a position vector, and an initial joint angle guess seed, where the algorithm will begin the search.

Throughout the program this seed state is fed as either the robot's current joint angles, or if computing along a given path the solution to the last point and orientation's joint angles. Regardless of this, some errors do still occur in the sequencing of these joint angle solutions as it may cause joint angles to leap from one extreme to another of a joint limit, another reason additional clearance on the end effector's joint limits is so beneficial. Typically, although it depends on the type of movement, the wrists at the end effector are more at risk of this as they are directly responsible for major angle changes that produce various orientations; while the previous angles tend to move

slowly through joint angle space as each acts through an extended lever arm.

2.3.3 Normals to Orientations

The desired orientation is calculated at each desired weld, or tack point, with the normal at that point determining the orientation and the robot interpolating along paths between these orientations, either linearly or as described in the weld path. This computation involves converting the desired rotation matrix to a quaternion— the rotational input. A given normal consists of both a vertical and horizontal component, as it determines location of the 45° BA1 weld gun to align the exiting wire. However, only the horizontal component is utilized in calculation of the desired orientation of these planar welds. This is done with taking the cross product between the designated tool vector, and the desired tool vector. Given a normal in the described orientation:

$$\hat{n} = [n_x, n_y, n_z],$$

and a aligned tool vector of:

$$T_v = [0, 1, 0].$$

The desired orientation vector equals:

$$O_v = [O_x, O_y, 0],$$

where:

$$O_x = -n_x, O_y = -n_y.$$

Then:

$$k = T_v \times O_v = [0, 0, -O_x]$$

$$\theta = \cos\left(\frac{T_v \cdot O_v}{|T_v||O_v|}\right)$$

with:

$$R_1 = R_y(\pi)$$

and:

$$\forall \hat{k} \neq [0, 0, 0],$$

$$\hat{k} = \frac{k}{|k|},$$

with the vector, angle representation of a rotation matrix known as the Rodrigues formula [12]:

$$Rot(\hat{k}, \theta) = e^{[\hat{k}]\theta} = I + \sin(\theta)[\hat{k}] + (1 - \cos(\theta))[\hat{k}]^2.$$

Then:

$$R_O = Rot(\hat{k}, \theta)R_1.$$

This defines the orientation of the tool relative to the global coordinate system by first rotating to align the \hat{y} tool vector with the given orientation and then spinning about this vector by π . This transformation expresses points in the TCP frame relative to the base frame:

$$p_b = R_O p_{TCP}.$$

Transforming a given vector in the tool frame to the base frame. However, cases arise when \hat{k} is a zero vector as this corresponds to either the tool vector aligning with the base y-axis or being rotated 180° from it. This is mitigated as follows:

if,

$$\hat{k} = [0, 0, 0]$$

and,

$$|\theta| \approx 180^\circ$$

then, it is assumed that the robot is operating in a x-y plane with the z-axis of the TCP frame orientated along the negative z-axis of the part transformation. This rotation can be taken with:

$$\hat{k} = [0, 0, +/- 1],$$

as this vector is rotating about the global z-axis by 180° the sign on this vector is superfluous. Then, R_O is identical as the previous description. If however:

$$|\theta| \approx 0^\circ$$

then R_O is simply taken as:

$$R_O = R_1.$$

2.3.4 Sigmoid Curves, B-splines, and Polynomial Interpolation

Sigmoid curves were utilized to time the trajectory, this was done by specifying a desired time span, joint velocity, or Cartesian velocity. The utilized function was of the form:

$$f(t) = \frac{1}{1 + e^{-b(2t-c)}},$$

with $c = 1$:

$$f(t) = \frac{1}{1 + e^{-b(2t-1)}}.$$

The corresponding function for a varied b can be seen in figure 2.21.

As b increases the curve remains more level within the 0 to 1 interval it is centered over. This is more desirable for timing a robotic trajectory as it implies the robot begins with a near zero velocity and acceleration, that smoothly accelerates across the timing of the trajectory. However, this infinitesimal curve does not necessarily map to the inclusive range 0 to 1 across its 0 to 1 domain. Thus, a methodology of weighted averages is used to interpolate along the discrete joint angle solutions and scale these with respect to the curve and desired time span. Given a desired time span Δt and a change in joint angles defined by the difference ² from the starting and end

²As described in section 2.4, this difference or cost can be scaled relative to each joint angle change with $\Delta\theta_i := \|(\vec{\theta}_{i+1} - \vec{\theta}_i) \cdot \vec{S}_j\|$

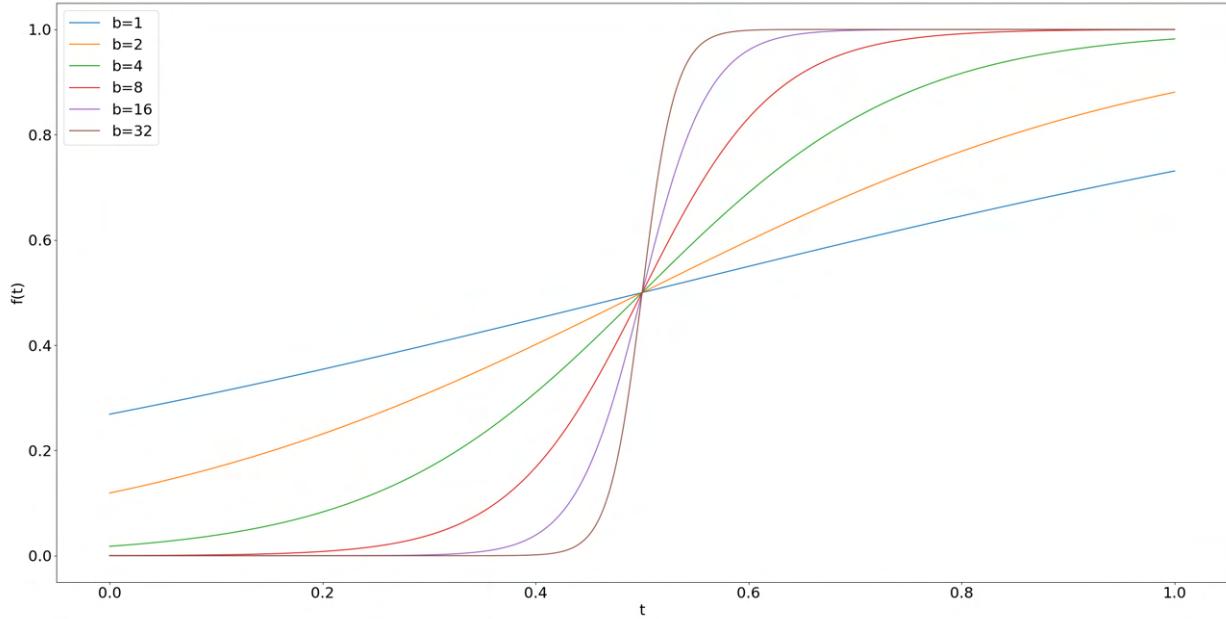


Figure 2.21: Sigmoid Function with b Varied from 1 to 32

point (this is also summed between every point in a trajectory for more complex paths):

$$\Delta(\theta) = \sum_{i=0}^{i=n-1} \sqrt{(\theta_{0i+1} - \theta_{0i})^2 + \dots + (\theta_{5i+1} - \theta_{5i})^2}$$

or with $i = 0$

$$\Delta(\theta) = \sqrt{(\theta_{0n} - \theta_{0i})^2 + \dots + (\theta_{0n} - \theta_{5i})^2},$$

where n is the number of joint angle points in the trajectory. Then the average joint velocity magnitude can be calculated as:

$$\dot{\theta} = \frac{\Delta(\theta)}{\Delta t}.$$

Alternatively a time duration can be calculated with a desired joint velocity magnitude:

$$\Delta t = \frac{\Delta(\theta)}{\dot{\theta}}.$$

Or if taking the Cartesian distance:

$$\Delta(d) = \sum_{i=0}^{i=n-1} \sqrt{(d_{xi+1} - d_{xi})^2 + (d_{yi+1} - d_{yi})^2 + (d_{zi+1} - d_{zi})^2}$$

or with $i = 0$

$$\Delta(d) = \sqrt{(d_{xn} - d_{xi})^2 + (d_{zn} - d_{zi})^2 + (d_{zn} - d_{zi})^2}.$$

Then given a desired Cartesian velocity magnitude, v , the desired time duration can be calculated as:

$$\Delta t = \frac{\Delta(d)}{v}.$$

Each of these alternatives are accessible in the program. But currently re-orientation movements to get above desired weld and tack points are performed with a desired average joint angle velocity—weld lines being critical that the speed matches a selected travel speed are performed with a Cartesian velocity and are integrated to accurately capture the path length.

For most trajectories that are not important to time in a Cartesian respect, it is preferred to use the joint timing method as the Cartesian method can return a path distance of zero for a pure reorientation, forcing the robot to jump instantaneously to these joint angles. Changes in both re-orientation and position are always captured in joint space.

With the desired time span either calculated or known the joint angles are timed with a Sigmoid curve for a selected b value. First the total change in height is calculated for the curve across the 0 to 1 domain:

$$\Delta(f(t)) = f(t = 1) - f(t = 0).$$

Then this is utilized with the desired time span to create weighted averages for each of the n joint arrays. First an array of points is linearly spaced from 0 to 1:

$$t = [0, \dots, 1],$$

at each of these points a ratio is calculated with respect to the total height of the Sigmoid curve:

$$\alpha = \frac{f(t) - f(0)}{\Delta(f(t))}.$$

To find the indexes for the weighted average:

$$i_{average} = \alpha n.$$

This value is then rounded up and down to produce an interval, call these new indexes upper i_u and lower i_l . The associated joint angles at each of these are pulled via the index to produce $\theta_{i_u} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$ and $\theta_{i_l} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$. Then the change in these joint angles along this interval is calculated as:

$$\Delta\theta = \theta_{i_u} - \theta_{i_l}.$$

Then the joint angles at $i_{average}$ are:

$$\theta(i_{average}) = \theta_{i_l} + \Delta\theta \left(\frac{i_{average} - i_l}{i_u - i_l} \right).$$

The index $i_{average}$, at which this occurs is then scaled by the desired time span:

$$t = i_{average} \Delta t.$$

A plot of these joint angles against the scaled time is shown in 2.22.

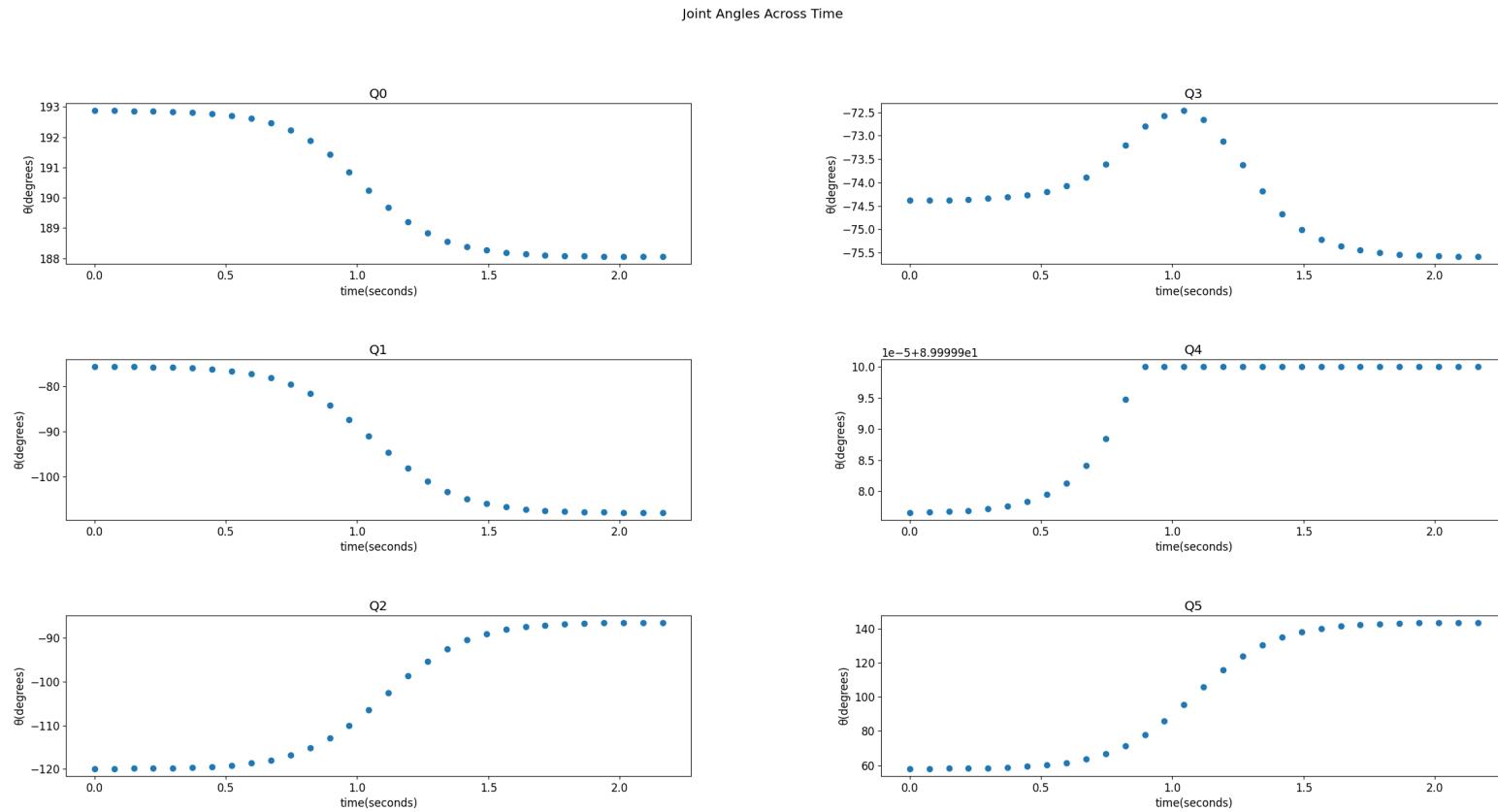


Figure 2.22: Sigmoid Joint Angle Solutions with $b=7$ and $\dot{\theta} = \frac{\pi}{4}s$

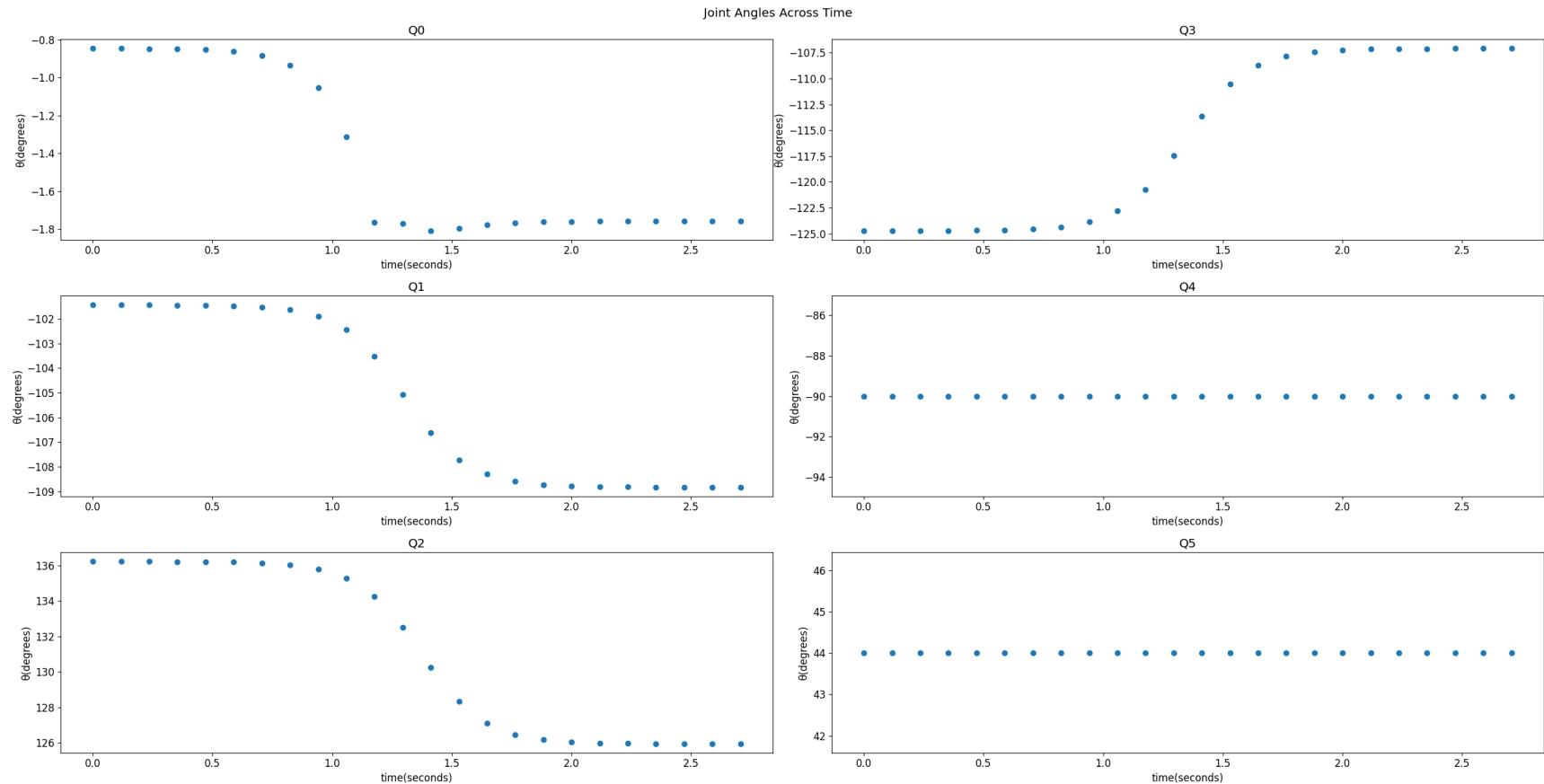


Figure 2.23: Sigmoid Joint Angle Solutions with $b=10$ and $\dot{\theta} = \frac{\pi}{5}$ s

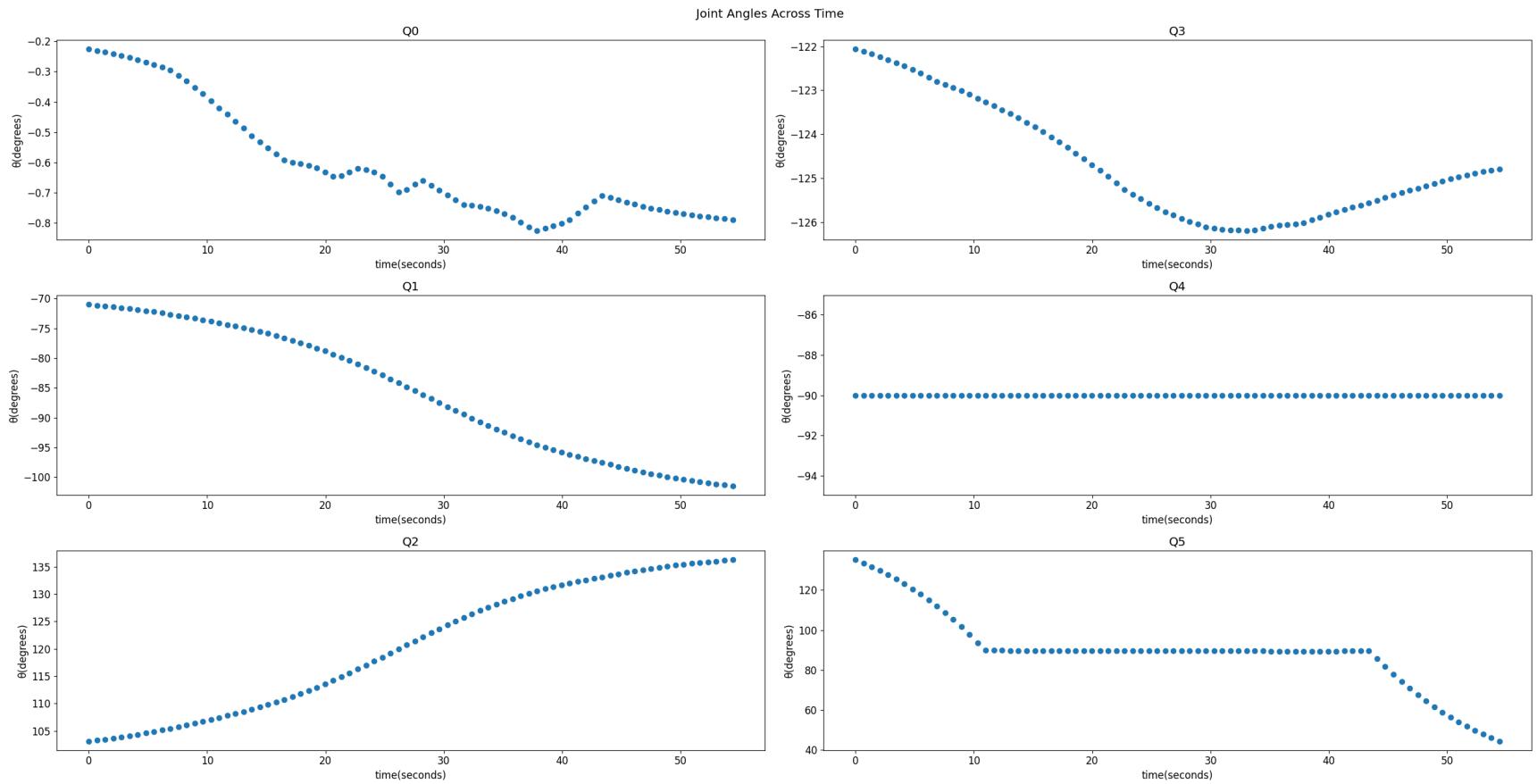


Figure 2.24: Weld Angles Fit to Sigmoid Function with $b=3$

Once the joint angles have been scaled with time, it is possible to directly pass this joint trajectory to the robot; but a useful procedure is to fit a set of polynomials to the sigmoid time and joint angle array's, as described in A.5. Results to this are shown in 2.25.

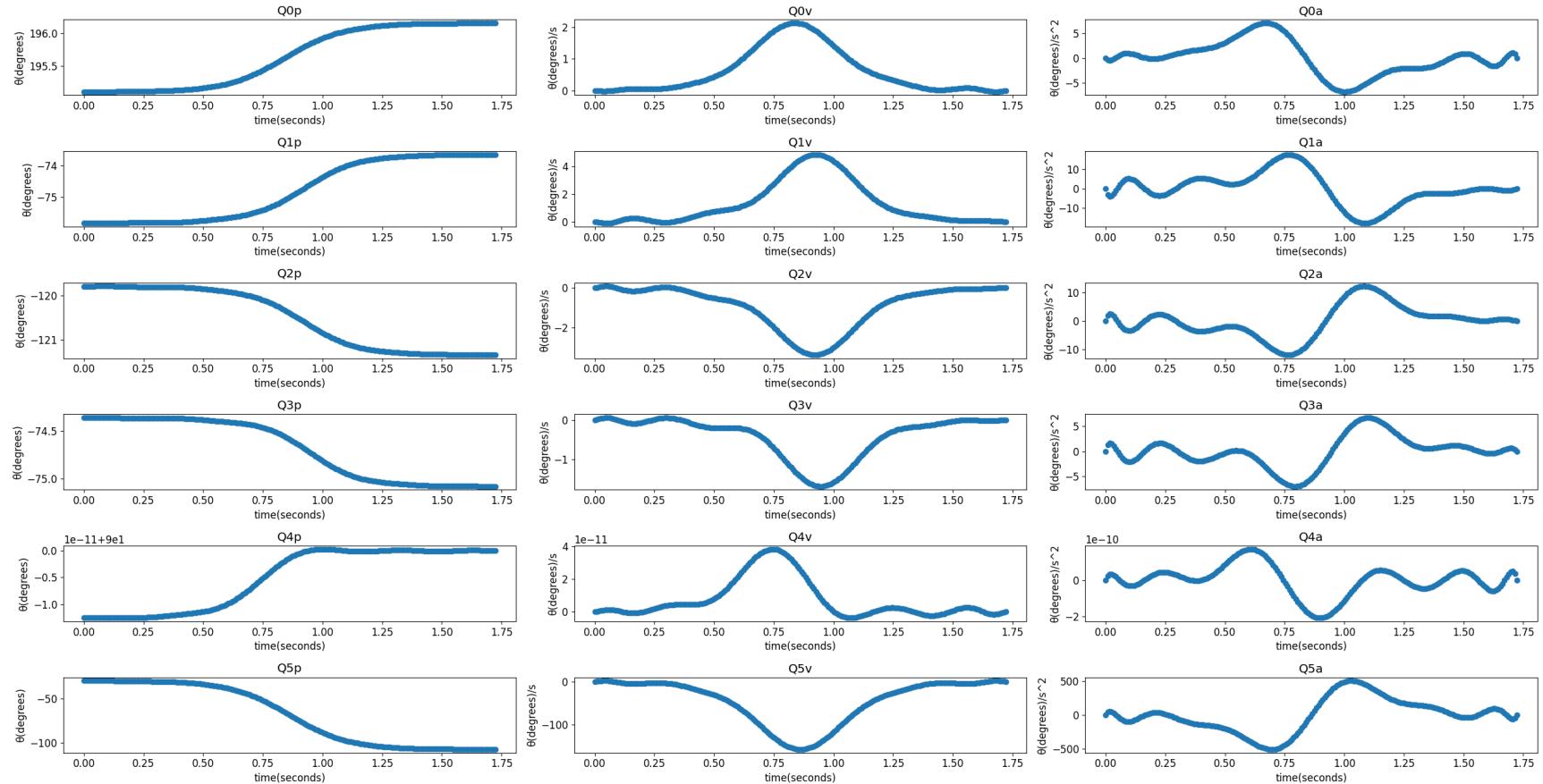


Figure 2.25: Polynomial Solutions for Timed Joint Angles

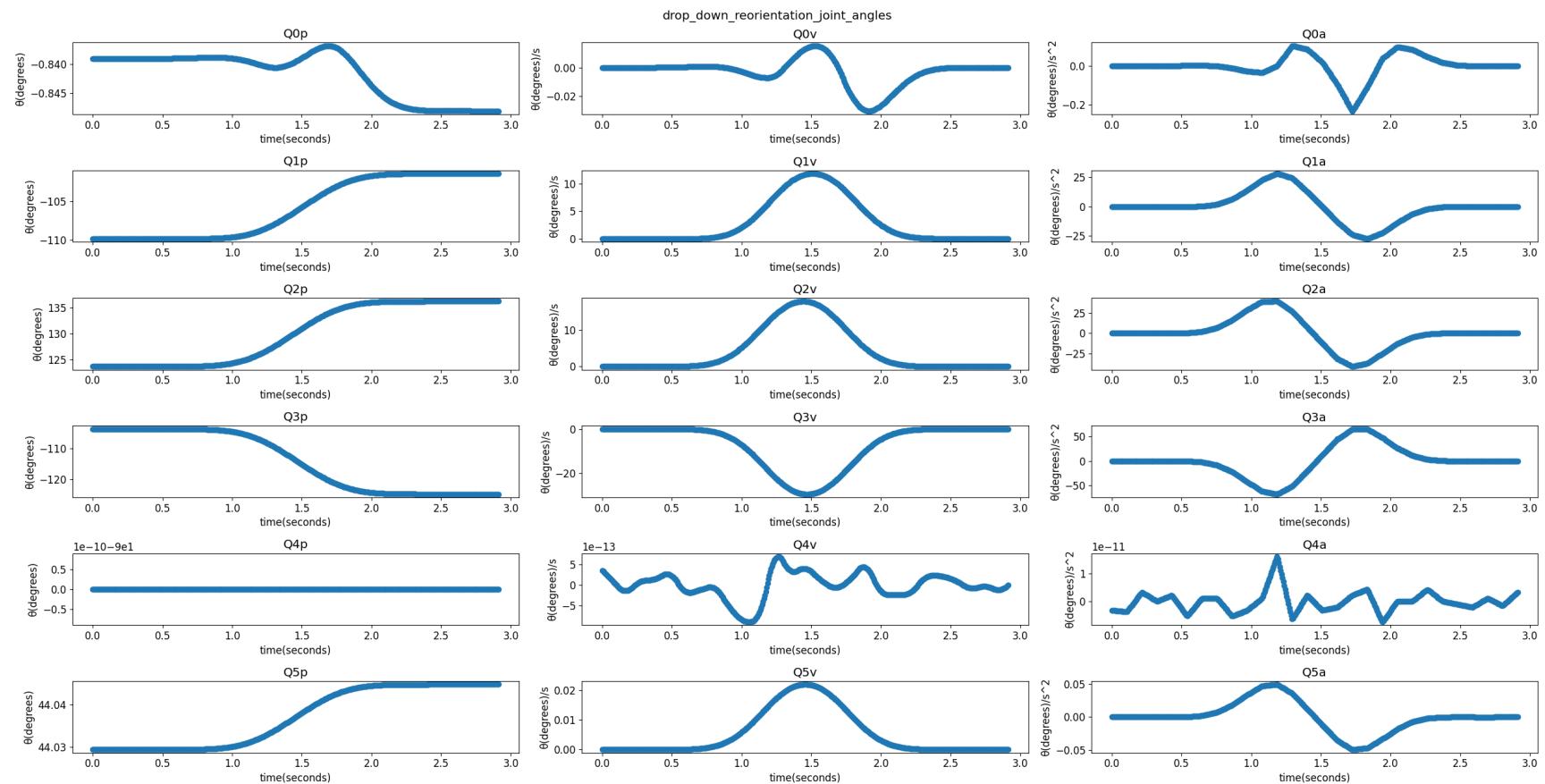


Figure 2.26: Generated Trajectory Spline Fit Drop Down to a Desired Weld Point

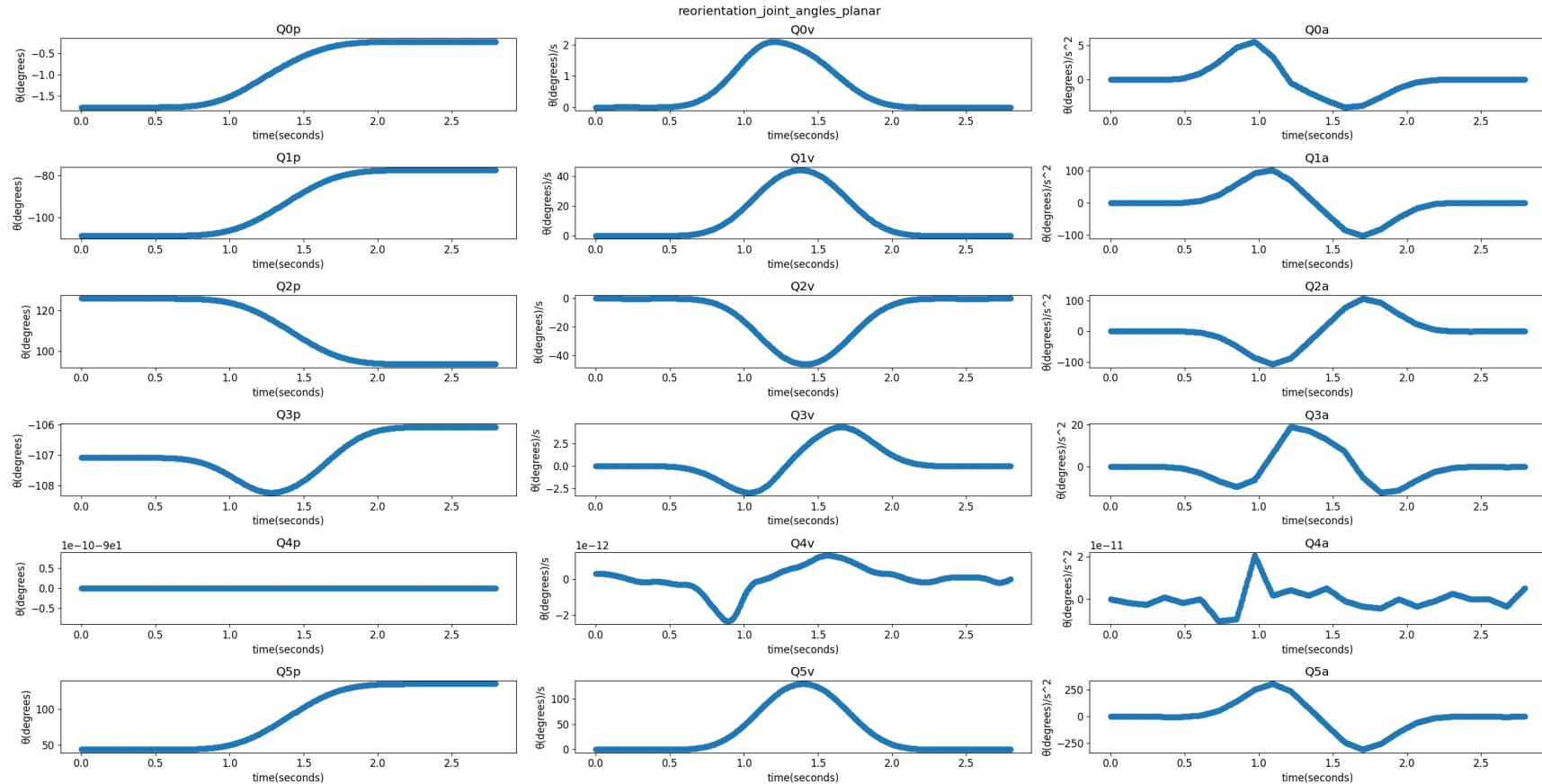


Figure 2.27: Generated Trajectory Spline Fit X-Y Plane Reorientation

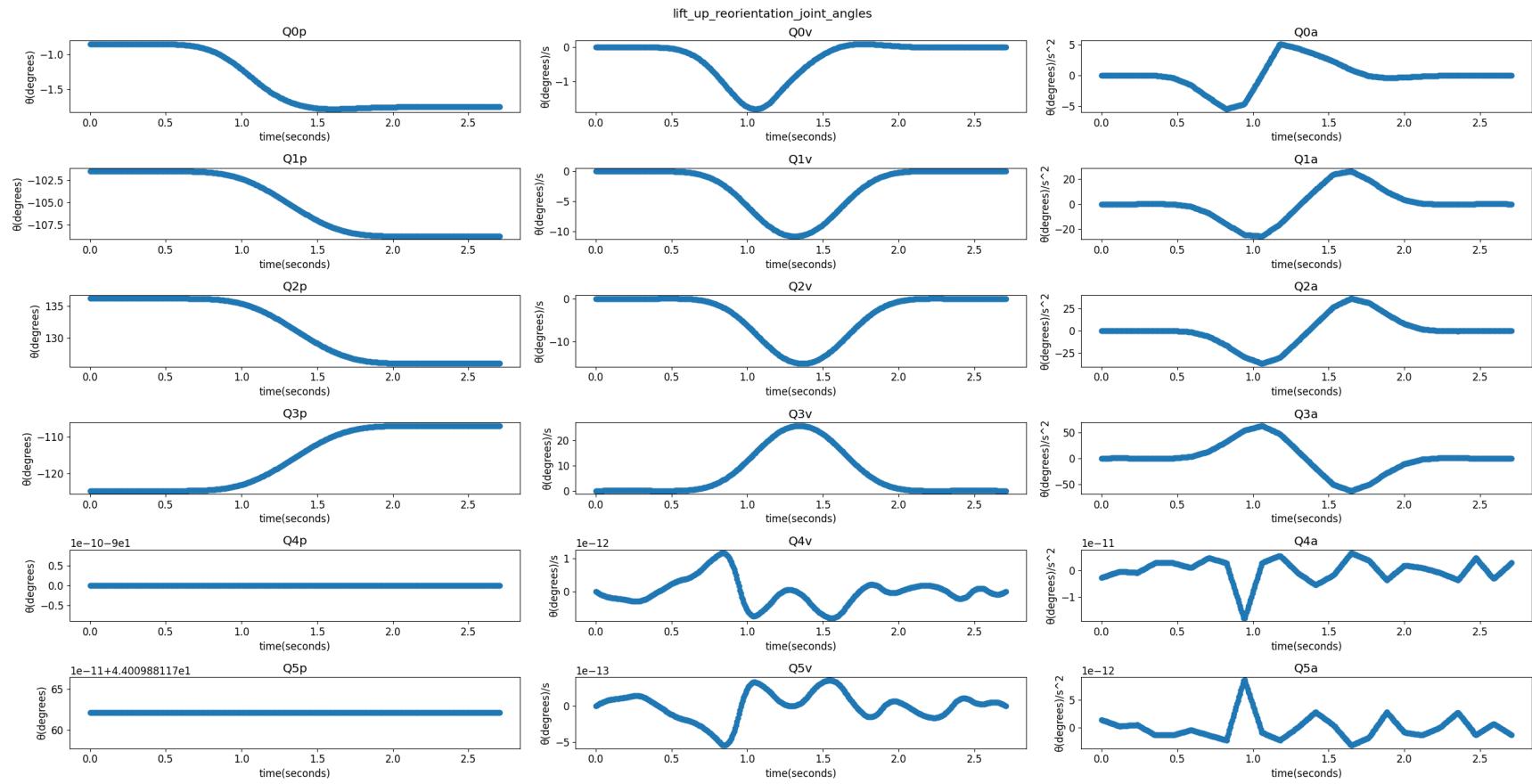


Figure 2.28: Generated Trajectory Spline Fit Lift Up End Effector

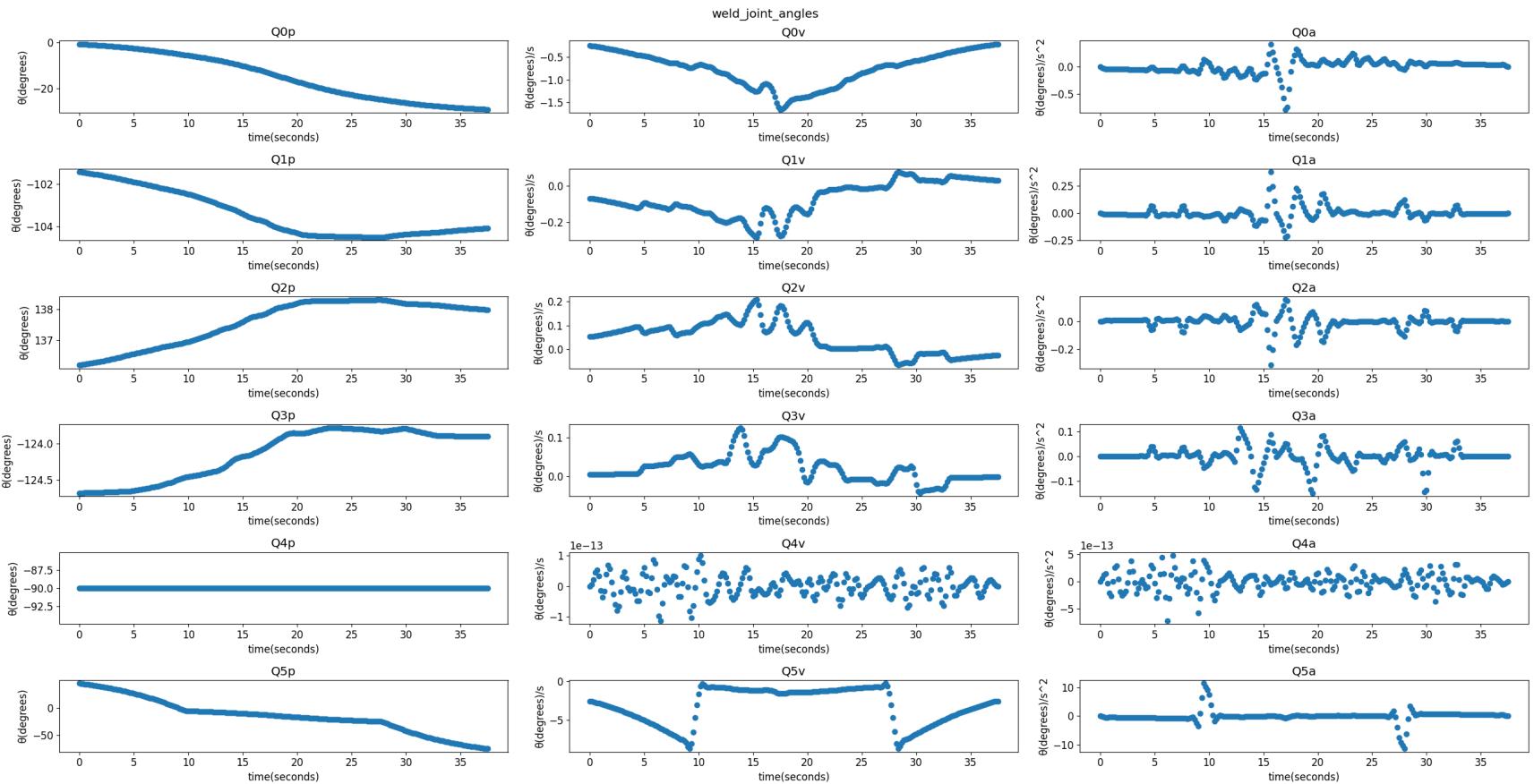


Figure 2.29: Generated Weld Trajectory Spline Fit

2.3.4.1 B-Spline Application to Joint Angles

Given the final set of desired joint angles along the given trajectory ($w_d \in W_d$), a set of parametric B-Splines, provided by Scipy are fit to each sub-coordinate array [19].

With,

$$\theta_i = [\theta_{0,i}, \theta_{1,i}, \theta_{2,i}, \theta_{3,i}, \theta_{4,i}, \theta_{5,i}]$$

for each desired joint angle solution in the given trajectory

$$\Theta_d = [\theta_i, \theta_{i+1}, \dots, \theta_n].$$

Separating out sub-arrays for each joint:

$$\Theta_0 = [\theta_{i,0}, \theta_{i+1,0}, \dots, \theta_{n,0}],$$

$$\Theta_1 = [\theta_{i,1}, \theta_{i+1,1}, \dots, \theta_{n,1}],$$

$$\Theta_2 = [\theta_{i,2}, \theta_{i+1,2}, \dots, \theta_{n,2}],$$

$$\Theta_3 = [\theta_{i,3}, \theta_{i+1,3}, \dots, \theta_{n,3}],$$

$$\Theta_4 = [\theta_{i,4}, \theta_{i+1,4}, \dots, \theta_{n,4}],$$

$$\Theta_5 = [\theta_{i,5}, \theta_{i+1,5}, \dots, \theta_{n,5}].$$

A time array is then specified for each point in Θ_d ,

$$t = [t_0, t_1, \dots, t_n] \quad \forall \theta_i \in \Theta_d$$

s.t.

$$t_0 = 0, t_n = 1.$$

The weights for the spline are specified as,

$$C = [c_0, c_1, \dots, c_n] \quad \forall \theta_i \in \Theta_d.$$

In practice the weights at the start and finish of the spline are set as a maximum, multiple magnitudes larger than the next largest weights— forcing it to exactly equal the start and end points. For each of the Θ_i sub arrays a spline is fit such that:

$$\theta_i(t) = s(t)$$

with,

$$s(t) = f(t_i, \Theta_i, C_i, \lambda)$$

where lambda is the smoothing coefficient. More information on these B-Splines can be found in *A Fortran package for generalized, cross-validatory spline smoothing and differentiation and Spline Models for Observational Data*[23][24].

2.4 Point Ordering-TSP

With the ability to move between any given set of desired tack or weld lines, one must select the order at which to execute these tacks or welds. The current implementation has available both an 'on-line' next nearest point algorithm for tack and weld, and a constrained or modified TSP solution. The tack TSP utilizes a LKH implementation, while the Weld TSP utilizes a branch and cut solver provided from Python MIP.

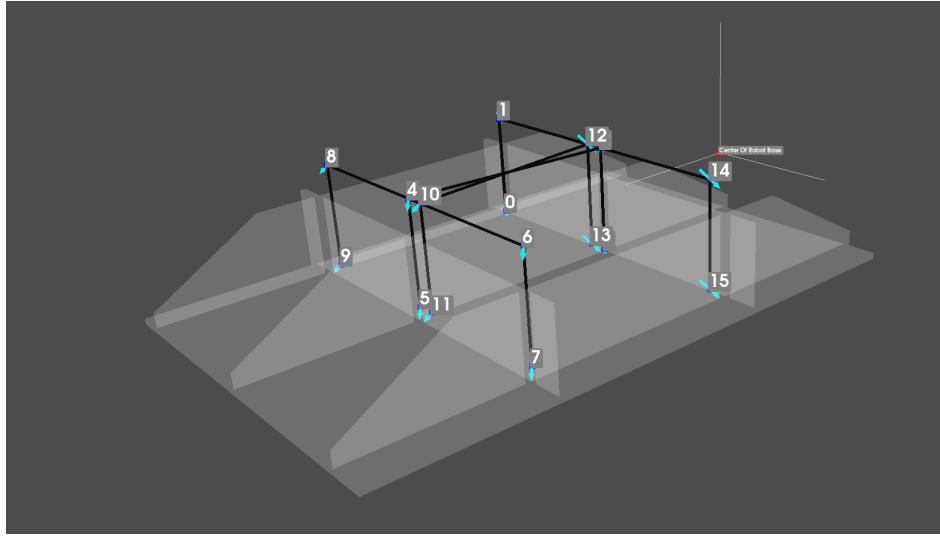


Figure 2.30: LKH Tack TSP

2.4.1 Graph Generation

Both the Weld and Tack TSP are reliant on a discrete graph representation of the weld and tack points. Each node in the graph is defined by its point and desired normal:

$$V = \{v : (\vec{p}, \hat{n})\},$$

$$\forall \vec{p}, \hat{n} \in \mathbb{R}^3.$$

Each edge can be defined as a combination of four desired configurations: two tack or desired weld points and two safety points specified in 2.2. These edges may either be represented as a single edge containing the associated travel points; or as the addition of four new nodes— with both being implemented in the software. Taking each edge as the subset of nodes:

$$E = \{(v_1, v_2) : v_1, s_1, s_2, v_2\}$$

$$\forall v_1, v_2 \in V$$

with,

$$s_n = v_n + \eta [0, 0, \vec{z}].$$

Where \vec{z} is determined via the normal, and η is the distance to the safety plane. The cost of each edge can be defined as the summation of euclidean distance over each associated set of points.

$$c_{euclidean} = \sum_{i=0}^3 \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}, \forall i \in \{v_1, s_1, s_2, v_2\}$$

Or a desired joint angle cost function, associated with the joint angles preluding the constraints of the desired joint angle trajectory. With Joint angle solutions defined for each point.

$$\vec{\theta}_i = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5], \forall i \in \{v_1, s_1, s_2, v_2\}$$

$$\Delta\theta_i := \|(\vec{\theta}_{i+1} - \vec{\theta}_i) \cdot \vec{S}_j\|$$

Here $\Delta\theta_i$ is taken as the difference between each consecutive IK solution with the inner product of \vec{S}_j , a scaling array which in the simplest case is $\vec{S}_j = [1, 1, 1, 1, 1, 1]$. Then,

$$c_{joint} = \sum_{i=0}^3 \Delta\theta_i.$$

For the joint angle cost associated with the tack TSP it was taken as the next nearest set of absolute joint angles, that achieves the desired change in \mathbb{R}^6 . Given a joint angle starting point $\vec{\theta}_i$, the next solution $\vec{\theta}_{i+1}$, is rounded down to the nearest π and then taken as a euclidean norm.

2.4.2 Tack TSP

The tack TSP is then solved by feeding the LKH solver the adjacency matrix of the generated graph [25][26][27].

$$A(a_{i,j}) = \begin{cases} C_f(i, j) & \forall i \neq j \\ 0 & \forall i = j \end{cases} \quad (2.1)$$

Where C_f is the joint or euclidean cost function between any two nodes i and j .

2.4.3 Weld TSP

The weld TSP utilizing Python MIP, starts with a graph generated as in 2.4.2 [28]. This graph is generated with two weld points and normals, associated with every desired weld, as the graphs initial vertices and edges. The set of desired weld lines D_w contains each weld points starting and ending configuration, \vec{w}_0 and \vec{w}_1 .

$$D_w = \{d_w : (\vec{w}_0, \vec{w}_1)\}$$

s.t.

$$\vec{w}_n = [p_n, \hat{n}_n].$$

Each start or finish point, \vec{w}_n , contains the associated point p_n and normal \hat{n}_n . For each weld line a node is added with its point and normal the average of the corresponding sub set.

$$V_{weld\ nodes} := \{\vec{v} : (p_w, \vec{n}_w), (p_w, \vec{n}_w) = \frac{\vec{w}_0 + \vec{w}_1}{2}, \forall \vec{w}_{0,1} \in d_w \in D_w\}$$

Associated edges are then added for each weld and connecting points.

$$E_w = \{\{(\vec{v}, \vec{w}_n) : \forall w \in \vec{v}\} \forall \vec{v} \in V_{weldnodes}\}$$

The cost for these new edges is calculated as in the tack TSP for either euclidean or joint angle cost with modifications for the joint angle cost as described.

Weld Joint Cost Function

The weld joint cost function is a modified version of the tack joint cost function. For two given weld vertices:

$$\forall \vec{u}, \vec{v} \in V_{weldnodes}$$

$$\exists\{(\gamma_0, z), (z, \gamma_1)\} \subset E_w, \forall n \in \{u, v\}.$$

Here, z is the connecting point(s) between two given weld vertices. For each associated weld node pairs u and v , there exists two, possibly non-unique, points γ_0 and γ_1 , that the weld node shares its only edges with, redefining these as:

$$\vec{u}_\gamma := (u_{\gamma 0}, u_{\gamma 1})$$

$$\vec{v}_\gamma := (v_{\gamma 0}, v_{\gamma 1})$$

or,

$$\vec{n}_w := (n_{\gamma 0}, n_{\gamma 1}).$$

Then the cost between any weld point $\vec{v} \in V_{weldnodes}$ and its connected $\vec{\gamma}_0$ or $\vec{\gamma}_1$ point is as defined in the tack cost formulation. However, the cost is altered from any point, $p_0 \in \vec{u}_\gamma$, to any other point, $p_1 \in \vec{v}_\gamma$. Call any pair of these points:

$$p_0, p_1.$$

If,

$$p_0 = p_1$$

then the cost C_{joint} is defined as zero:

$$p_0 = p_1 \Rightarrow C_{joint} = 0.$$

If $p_0 \neq p_1$ then the safety points from the subset of tack edges, must be included in the motion from one weld to the next as the connecting points are not coincident.

$$(p_0, p_1) \in E \Rightarrow (v1, s1, s2, v2)$$

The cost from $v1$ to $s1$ and $s2$ to $v2$ remain as the difference between the closest satisfying IK solutions. However, the cost from $s1$ to $s2$ is altered, take both safety point's joint angles.

$$\vec{\theta}_{s_1} = [\theta_{10}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{15}]$$

$$\vec{\theta}_{s_2} = [\theta_{20}, \theta_{22}, \theta_{23}, \theta_{24}, \theta_{25}]$$

Define the non-scaled $\Delta\theta$ as,

$$\Delta\theta_i := \|(\vec{\theta}_{s_2} - \vec{\theta}_{s_1})\| = [(\theta_{20} - \theta_{10}), (\theta_{21} - \theta_{11}), (\theta_{23} - \theta_{13}), (\theta_{24} - \theta_{14}), (\theta_{25} - \theta_{15})].$$

Then as in the tack solution, the first 5 changes in joint angle are taken as the difference of the nearest absolute angles. This holds for all but $\Delta\theta(\theta_{25} - \theta_{15})$ with the last change in joint angle 5, q_5 , taken as,

$$\Delta\theta(\theta_{25} - \theta_{15}) := (\theta_{25} + (-\pi) \lceil \frac{|\theta_{25} - \theta_{15}|}{\pi} \rceil sgn(\theta_{25} - \theta_{15})) - (\theta_{15}).$$

Here $\lceil x \rceil$ symbolizes rounding x to the nearest integer value and sgn represents the sign function. For $\Delta\theta(\theta_{25} - \theta_{15})$, the difference is calculated as follows. First the total angle change is calculated,

$$T_a = (\theta_{25} - \theta_{15}).$$

Given the joint limits on the end effector, $q_5 = \{\theta_5 : \theta_l \leq \theta_5 \leq \theta_u\}$.

If $T_a = 0$,

$$\Delta\theta(\theta_{25} - \theta_{15}) = 0.$$

If $T_a > 0$ and $(\theta_{25} - 2\pi) \geq \theta_l$,

$$\Delta\theta(\theta_{25} - \theta_{15}) = ((\theta_{25} - 2\pi) - \theta_{15}).$$

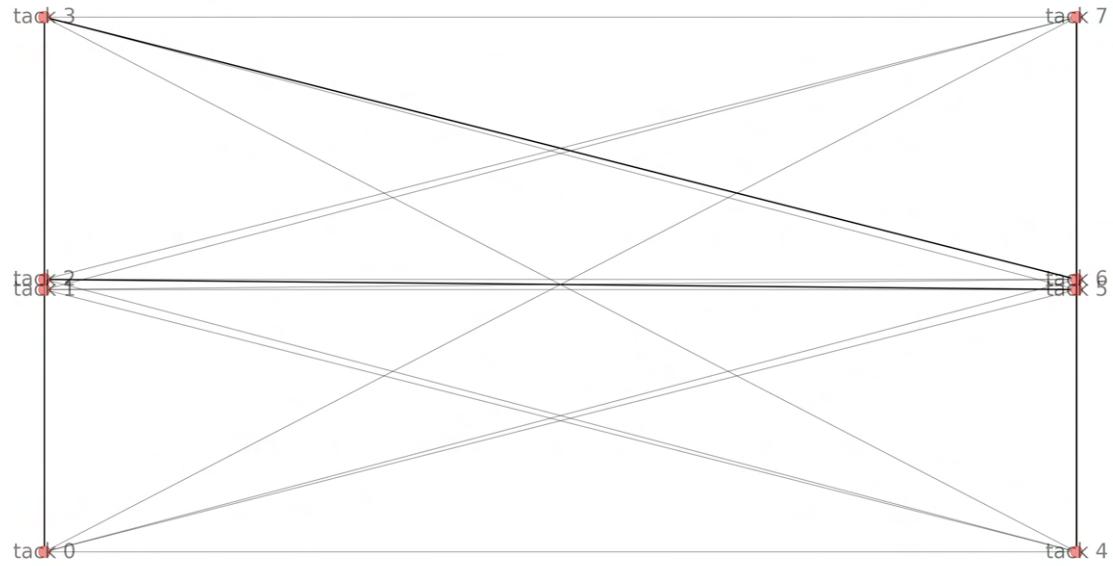
If $T_a < 0$ and $(\theta_{25} + 2\pi) \leq \theta_u$,

$$\Delta\theta(\theta_{25} - \theta_{15}) = ((\theta_{25} + 2\pi) - \theta_{15}).$$

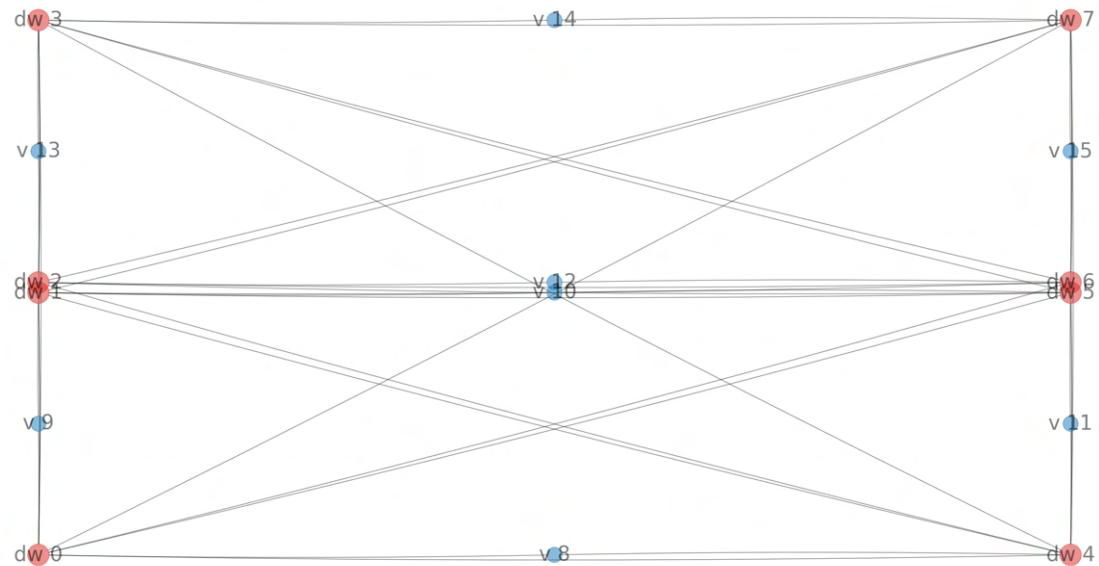
Otherwise the total angle change is taken as,

$$\Delta\theta(\theta_{25} - \theta_{15}) = T_a.$$

As before this total cost vector $\Delta\theta_i$ may still be multiplied with a scaling vector \vec{S}_j . The purpose of this modification in cost for the end effector's joint angle change is to more accurately capture the cost of the movement executed. During the course of a required weld tour, if the weld gun has to pick up to reach the next weld edge it looks ahead at the total orientation change in the next edge and pre-rotates the end effector's joint angle, q_5 , in the opposite direction that it will have to turn throughout the welding process. Although, the required rotation of the next weld edge is undetermined until the tour is solved; the reasoning is that the cost of maximizing or minimizing $|\theta_{25}|$ approximates its cost. Further, the cost associated with $T_a = 0$ encourages welds in the order of equivalent end effector angles q_5 .

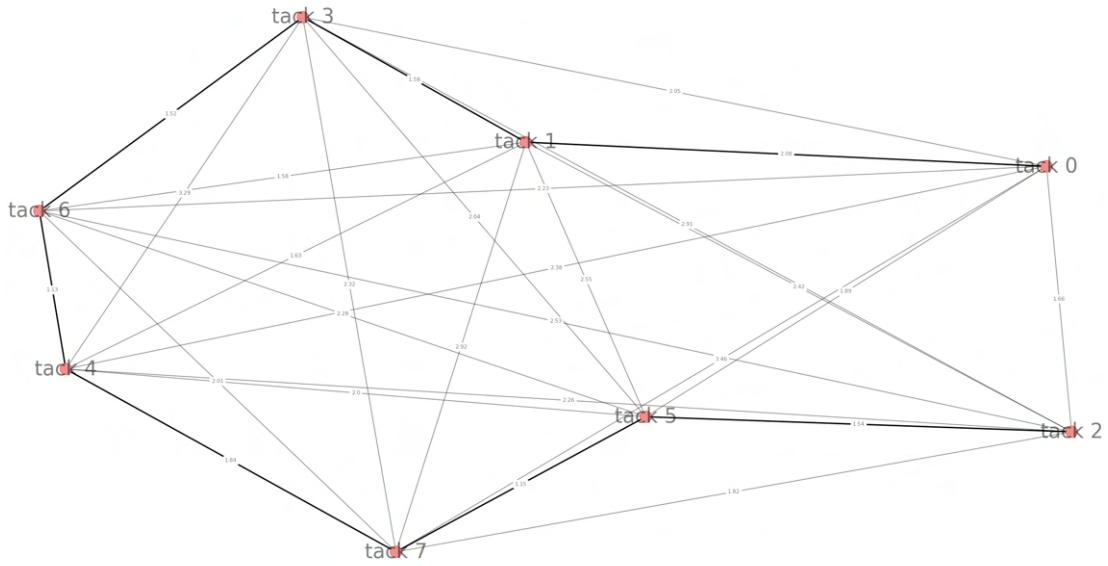


(a) Tack Graph Representation

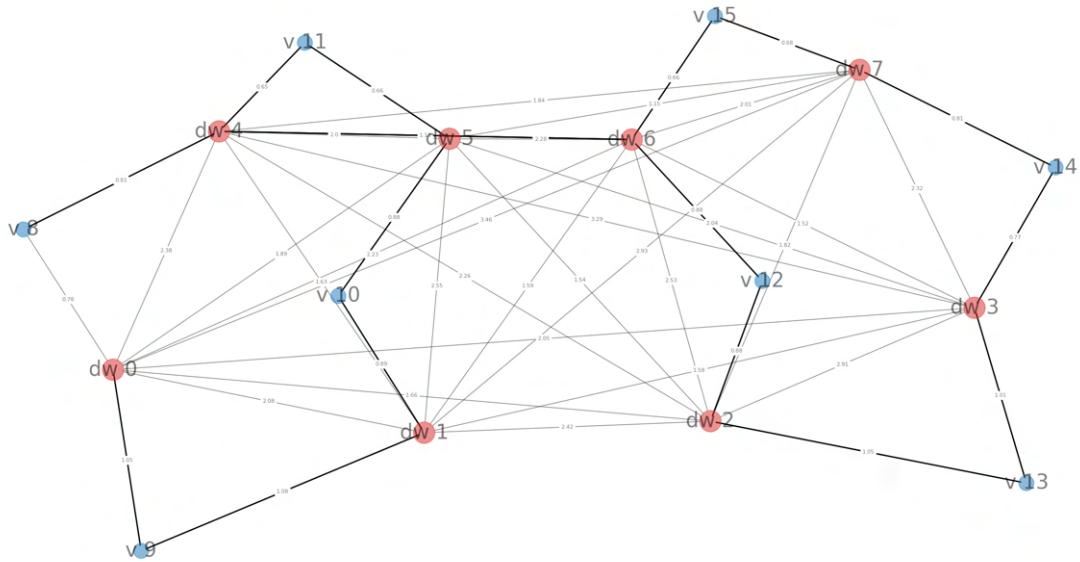


(b) Weld Graph Representation

Figure 2.31: Discrete Graph Representations: with node locations at desired points [29]



(a) Tack Graph Kamada-Kawai Representation, Solution in Black



(b) Weld Graph Kamada-Kawai Representation, Solution in Black

Figure 2.32: Discrete Graph Kamada-Kawai Representations [29]

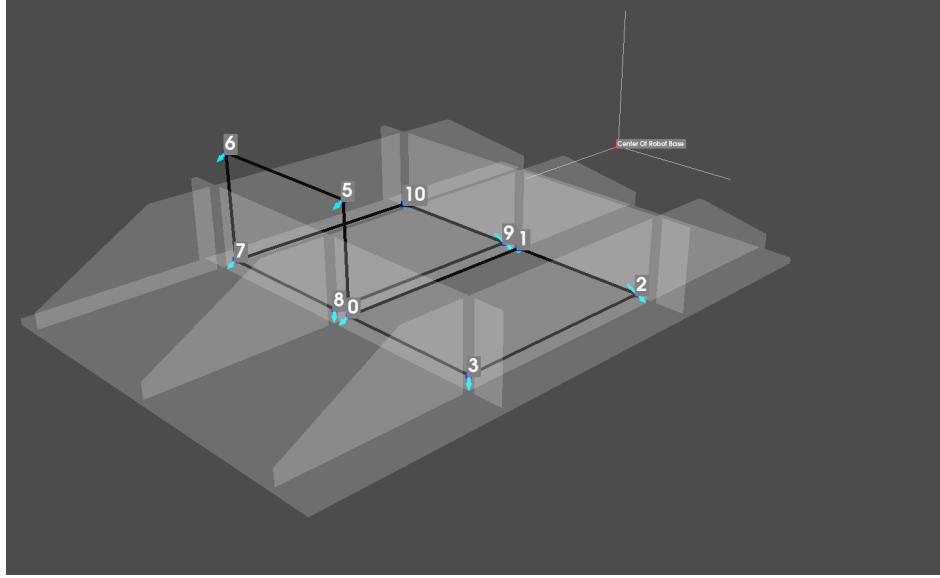


Figure 2.33: Python MIP Weld Tour

2.4.3.1 MIP Equations

With the weld graph and edge cost defined, it is possible to formulate the MIP for the welding TSP. First define a decision set X .

$$X := \{X_{n,m} : n, m \in \mathbb{Z}^+\}$$

$$X_{n,m} := \{(x_0, x_1, x_2, x_3) : x \in \{0, 1\}\}$$

$$\forall (u, v) \in V_{weldnodes}, \exists X_{n,m} \in X$$

$X_{n,m}$ represents four decision variables between two weld nodes. Take figure 2.34, for the weld to be completed across both u and v , each node must have both edges used in successive order as in, $(3, V) \Rightarrow (V, 1)$ and $(2, U) \Rightarrow (U, 4)$, or reversed. With edges defined for weld nodes (u, v) in 2.34, take four edges from v to u .

$$e_0 \Rightarrow v \rightarrow 1 \rightarrow 4 \rightarrow u$$

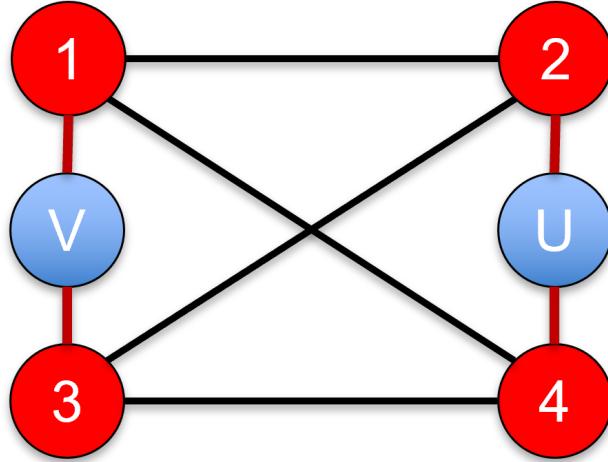


Figure 2.34: Paths Between $(u, v) \in V_{weldnodes}$: required weld edges in red

$$e_1 \Rightarrow v \rightarrow 1 \rightarrow 2 \rightarrow u$$

$$e_2 \Rightarrow v \rightarrow 3 \rightarrow 2 \rightarrow u$$

$$e_3 \Rightarrow v \rightarrow 3 \rightarrow 4 \rightarrow u$$

Take four edges from u to v .

$$e_4 \Rightarrow u \rightarrow 4 \rightarrow 1 \rightarrow v$$

$$e_5 \Rightarrow u \rightarrow 2 \rightarrow 1 \rightarrow v$$

$$e_6 \Rightarrow u \rightarrow 2 \rightarrow 3 \rightarrow v$$

$$e_7 \Rightarrow u \rightarrow 4 \rightarrow 3 \rightarrow v$$

Decision array $\{\{x_0, x_1, x_2, x_3\}, \{x_4, x_5, x_6, x_7\}\} \subset X$ then represents the corresponding edges, $\{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ that are used between any two weld nodes u and v . For the given nodes, the following in-equality constraints guarantee both weld nodes edges are selected, for a tour that visits every weld node.

$$I_{eq} := \left\{ \begin{array}{l} \{(x_0 + x_4 \leq 1), (x_0 + x_7 \leq 1), (x_3 + x_4 \leq 1), (x_3 + x_7 \leq 1)\} \\ \{(x_1 + x_5 \leq 1), (x_1 + x_6 \leq 1), (x_2 + x_5 \leq 1), (x_2 + x_6 \leq 1)\} \end{array} \right\}$$

For a tour over the subset $\{u, v\}$ this leaves only two feasible tours.

$$v \rightarrow 1 \rightarrow 2 \rightarrow u \rightarrow 4 \rightarrow 3 \rightarrow v$$

and,

$$v \rightarrow 1 \rightarrow 4 \rightarrow u \rightarrow 2 \rightarrow 3 \rightarrow v.$$

These inequalities are then applied across all pairs in $X_{m,n}$,

$$I\{X_{n,m}, X_{m,n}\} := \left\{ \begin{array}{l} \{(x_0 + x_4 \leq 1), (x_0 + x_7 \leq 1), (x_3 + x_4 \leq 1), (x_3 + x_7 \leq 1)\} \\ \{(x_1 + x_5 \leq 1), (x_1 + x_6 \leq 1), (x_2 + x_5 \leq 1), (x_2 + x_6 \leq 1)\} \end{array} \right\}$$

$$\forall n, m \in V_{weldnodes}.$$

Though it was implemented as,

$$\forall i \in V_{weldnodes}$$

and,

$$\forall j \in V_{weldnodes} \setminus \{i\}$$

and,

$$\forall k \in V_{weldnodes} \setminus \{i, k\}$$

$$\exists (i, j), (j, k) \in E_w.$$

Where E_w is the set of all weld edges. Then given the four paths associated with edge (i, j) and

the four paths associated with (j, k) .

$$E_w(i, j) = \{e_0, e_1, e_2, e_3\}$$

$$E_w(j, k) = \{e_4, e_5, e_6, e_7\}$$

each e_n for general indexes (u, v) can be defined as,

$$e_n := u \rightarrow \vec{w} \rightarrow v.$$

With $\vec{w} \in \mathbb{Z}, \mathbb{Z}^4$ being the set of connecting nodes; if one point in the first weld is repeated in the second weld there will be a total of 1 nodes between the desired weld nodes u and v , along the shortest possible path. Taking $\vec{w}_{i,j}$ as a list of indexes $\forall \vec{w} \in e_n \in E_w(i, j)$ and the same for $\vec{w}_{j,k}$. Where the list may be of any length, using the definitions:

$$\vec{w}_{u,v} := \{z_0, z_1, \dots, z_n\}$$

$$w_{u,v_f} := z_n$$

and,

$$w_{u,v_0} := z_0.$$

Then,

$$\forall x_n \in \{x_0, x_1, x_2, x_3\} \in X_{i,j} \in X$$

and,

$$\forall x'_n \in \{x'_0, x'_1, x'_2, x'_3\} \in X_{j,k} \in X$$

if,

$$w_{i,j_f} = w_{j,k_0} \Rightarrow x_n + x'_n <= 1$$

or if,

$$w_{i,j_0} = w_{j,k_f} \Rightarrow x_n + x'_n \leq 1.$$

With the selected weld lines constrained the MIP can be laid out. First the objective of the MIP is to minimize:

$$[C(m, n, l)]X.$$

Where C is the cost function between weld node m and n using path l .

This can also be represented as,

$$\sum_{i \in V_w, j \in V_w, k \in X_{i,j}} c_{i,j,k} x_{i,j,k}$$

with,

$$V_w := V_{welds}.$$

With $c_{i,j,k}$ being a summation of the cost of an array of points representing the path from weld node to weld node, and $x_{i,j,k}$ a binary integer representing its inclusion in the solution.

Then define a decision set Y ,

$$Y := \{\{y_i : y_i \geq 0\}\} : \forall i \in V_w\}.$$

The final constraints on X specify that each desired weld node is only entered and exited once, and on Y to eliminate possible subtours, adapted from Python MIP's solution for a standard TSP [28]:

$$\sum_{j \in V_w \setminus \{i\}, k \in X_{i,j}} x_{i,j} = 1 \quad \forall i \in V_w$$

$$\sum_{i \in V_w \setminus \{j\}, k \in X_{j,i}} x_{i,j} = 1 \quad \forall j \in V_w$$

and,

$$y_i - (n(V_w) + 1)(\sum_{k \in X_{i,j}} x_{i,j,k}) \geq y_j - n(V_w) \quad \forall i \in V_w \setminus \{0\}, \quad \forall j \in V_w \setminus \{0, i\}.$$

With $n(v_w)$ being the cardinality of the weld node set. Thus constraining the weld tour for the MIP.

2.4.3.2 On-line Solver

As mentioned in previous sections, the on-line solver is capable of executing either a greedy solution or executing from the list of TSP tour points for either a weld or tack tour. This greedy algorithm is described as below for both the tack and weld.

On-line Tack Pseudo Code The steps of the algorithm are as follows:

1. Calculate all the distances from the end effector's current position to an array containing the set of desired tack points $[p_i, p_{i+1}, \dots]$ and locate the point with the smallest distance from the current point, p_m .
2. If the current position of the end effector is below the safety plane move vertically to the plane.
3. Re-orientate to above p_m , drop down to the point and tack.
4. Remove p_m from the list of desired tack points $[p_i, p_{i+1}, \dots]$
5. If there are no more tack points, lift up, otherwise return to 1.

On-line Weld Pseudo Code The steps of the algorithm are as follows:

1. Calculate all the distances from the end effector's current position to an array containing the set of desired weld points $[p_i, p_{i+1}, \dots]$ acquired from every weld line $[w_0, w_1]$ in D_w and locate the point with the smallest distance from the current point, $p_m \in \{w_0, w_1\}$.
2. If the current position of the end effector is below the safety plane move vertically to the plane.

3. Re-orientate to above p_m , rotating the end effector as far in the opposite direction of the edges required angle change, and drop down.
4. Remove p_m and its associated point in the set of weld lines from the list of desired weld lines points $[dw_0, \dots, dw_n]$, and append it to the list of current weld lines to execute.
 - (a) Repeat step 1 for p_{m+1} .
 - (b) If, the next closest edge is coincident with the end point of the current edge and it is possible to append the joint angle solutions for a smooth function within joint limits, then: add the edge to the list of current welds to execute, remove it from the list of desired welds and return to 4.b, otherwise: go to 5.
5. Execute list of current weld lines and reset the current weld lines array.
6. If there are more weld lines return to 1. Otherwise, lift up to the vertical safety plane.

On-line Angle Pre-Rotation-Weld The pre-rotation of the end effector as it transitions through the safety plane from desired weld point to weld point works identical to that described in 2.4.3. With the modification that variable T_a or total angle change, correspond to the total required angle change for the next weld edge. With the weld edge being determined by the on-line next nearest point algorithm, or provided by the TSP weld tour.

$$Ta = (\theta_{w_2,5} - \theta_{w_1,5})$$

This is done to give the welder a greater chance of connecting coincident welds. By turning q_5 opposite to the required angle change of the next weld, the angle after the weld is completed will lie closer to the center of the joint limits, possibly allowing the welder to continue. If the welder can not continue to this coincident weld due to joint angle restraints, it rises vertically and resets q_5 for T_a belonging to the weld.

3. RESULTS

3.1 Parameter Test

Table 3.1: Table of Weld Parameters

Parameter(unit)	Value
Weld Mode(n/a)	Pulsed MIG
Wire Feed Rate(ipm)	504
Arc Length(ratio/100)	50
Gas Flow Rate(cfh)	25
Weld Tip Velocity(cm/s)	0.6
Weld Tip Control Distance(cm)	2

The higher wire feed rate in 3.1 helps to deposit more material into the weld. These parameters were found to be a quality operating point throughout all parameter tests. Welds with both a higher velocity and feed rate have occasionally shown good quality, with results deteriorating as velocity increases past 2 cm/s. A higher velocity at the same feed rate also results in less deposition, with feed rate being limited to a maximum of 780 ipm as in B.2. The total weld length in figure 3.1 is approximately 50 centimeters, at 0.6 cm/s this implies a weld duration of 83 seconds. For the two inner grids of the prototype in figure 3.3, there are approximately 7.74 feet or 235.92 cm for the 8 interior grid welds. If the robot spent 100% of its travel time welding, it could accomplish these in 6.55 min. The arc length ratio in table 3.1, is a value from 0 to 100, with its effect illustrated in B.7.

The scaling \vec{S}_j in table 3.2 is loosely based on the length of each joint's connected link length, over the total summed link length of the manipulator, with values adjusted by trial and error. The 'b' parameter for the welder is relatively low at 3. This is because this parameter is proportional to the time it takes the trajectory to reach its steady-state velocity, a lower 'b' value runs the risk of a higher observed jerk. Because the weld speed is so low at 0.6 cm/s this jerk due to a low

Table 3.2: Table of Robotic Hyper-Parameters

Parameter(unit)	Value
Weld Sigmoid ‘b’ parameter (n/a)	3
Weld Sigmoid ‘b’ parameter (n/a)	7-10
Reorientation Angular Rate (rads/s)	$\frac{\pi}{5}$
Sigmoid Timing and TSP Joint Cost Vector, \vec{S}_j (n/a)	[10, 10, 10, 5, 2.5, 1]
Weld Joint Angle Trajectory Spline Smoothness Coefficient (n/a)	2
Reorientation Joint Angle Trajectory Spline Smoothness Coefficient (n/a)	5
Weld Path Spline Smoothness Coefficient (n/a)	5

‘b’ is unnoticeable, and a more consistent weld velocity is achieved across the weld line. The smoothness coefficient for the weld line is lower than that of the reorientation; because as the smoothness coefficient increases, the accuracy of the trajectory will decrease.

In 3.1 the diameter of the weld decreases towards the middle of the line, this is likely a result of the end effector accelerating and decelerating into its trajectory, where the wire feeder quickly reaches its steady state of 500 ipm.

Previous tests had issues with the BA1, leaving the wire attached to the work-piece at the end of a weld. As the manipulator began to lift it would pull against the end effector causing the program to fail. This is the reason the crater and start functions were utilized on the welder’s options; both of these alter the starting and ending parameters of a weld when the enable function is triggered, reducing the risk of the filler wire sticking to the end of the weld and leading and into a quality arc at the start of the weld. The timing, wire feed rate, duration and magnitude of both of these lead in and out options are all adjustable but the defaults were utilized for the results. Figure 3.2 was conducted as a test to confirm the wire would no longer stick, with the gun lifting and resetting after each weld.



Figure 3.1: Long Weld Parameter Test



Figure 3.2: Consecutive Weld Parameter Test

3.2 Grid Prototype Welds

The welded part in figure 3.3 is a minimal representation of the industrial grid part (figure B.11). This minimal prototype required significantly less material, being a more suitable cost for successive part tests.

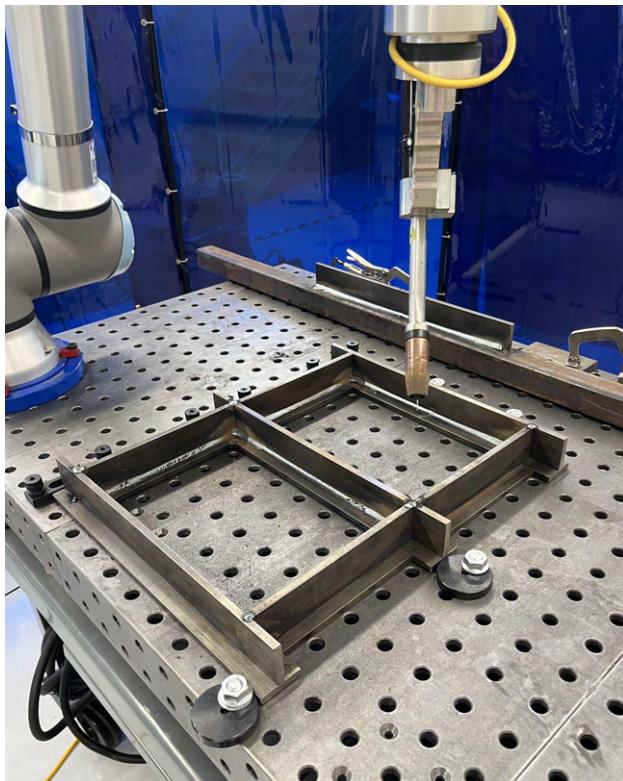


Figure 3.3: Prototype Part with Two Grids Welded

The total time of the two grid weld tour was approximately 7.5 minutes, for a total weld length of 7.74 feet or 235.92 cm. This gives the total weld length per time ratio of $\frac{234.92}{(60(7.5))} = 0.524 \text{ cm/s}$ which is 87% of the 0.6 cm/s weld speed– meaning 87% of the time the manipulator spent moving it is welding



(a) Successful Corner to Corner Weld



(b) Successful Continuous Square Weld

Figure 3.4: Grid One Welds



Figure 3.5: Corner to Corner Welds with Porosity

4. DISCUSSION AND CONCLUSION

The following is a reflection based on the current results and their relation to the developed methodology, followed by a overview of the possible scope of this project moving forward; with possible improvements summarized.

4.1 Discussion

The results represent a proof of concept for an automated robotic welding system; but does not demonstrate a fully automated and robust system. Poor weld quality although infrequent is present, as in the porosity shown in the results. This presents as a welding process control problem. Although the software has been fully packaged for easy integration with a user gui; this interface has not been implemented. The solution does not allow easy operation from technicians; requiring significant steps performed by developers. However, the project thus far has reached several milestones that facilitate the development of such a system; branching out from the current solution.

Both polynomial and B-spline functions where presented in the Solution Methodology; but only the splines were used outside of simulation. This is because the polynomial fit was an exact function, not a best-fit as in the splines. This made small instantaneous jumps in the desired weld angles create polynomial solutions with oscillations to large values outside of the joint angle limits. In-fact, the polynomial solution being an exact fit had higher oscillations when compared to a best-fit of the spline-which is a best fit of polynomial based functions. This is also referred to as over-fitting, or capturing small changes in data due to error.

Issues with process control are also present in the arc from the filler metals occasional failure to spark at the start of a weld, bending the filler metal into a coil. However, when the solution does work it shows increases in throughput and performance over a typical human welder. The mean quality of the welds performed likely surpasses that of a average human welder.

4.2 Future Scope of The Project/Improvements

This sections outlines recommended program improvements and advancements for the future scope of the project.

4.2.1 Selected TCP Frame

Although the current TCP frame approach works for and introduces a simplicity to the solution, it has numerous short comings. First, the computational requirements of implementing this although low, could be reduced. An alteration of the TCP frame's relative orientation at the tip to the part itself could reduce the computational load. This reduction is with respect to the control distance calculations that specify the point and the conversion of the normals utilized, in calculating this position, to the normals in calculating the orientation. One possible solution is instead defining the three dimensional TCP frame as follows. First, the tool vector could remain as the y-axis of the TCP frame but could be re-orientated such that it is parallel with the wire fed out of the weld gun but in the opposite direction. This means that surface normals that define the desired feed-in angle of the tool would match that of the tool itself, directly providing this orientation vector. This defines a plane from which the remaining orthogonal orientation vectors must lie. Given the single degree of freedom corresponding to the rotation of these orthogonal vectors, one can imagine the gun tip itself rotating about this axis, with the wire aligned along the specified normal. If the new z-vector is specified such that it is in the same plane as this orientation vector and the original z-axis of the tool, while remaining orthogonal; then the last vector, the x-axis, must point either directly to the left or right of the gun depending on the positive or negative value of the new z-axis in relation to the previous. Thus, if the x-axis is set as the current tangent of the path or the overall travel direction of the weld, then the gun would be orientated to travel along this path or in the specified travel vector. But this could invert the desired orientation if the corresponding direction of the z-axis for this new orientation does not produce a orthogonal basis vector such that the tool is orientated above the work piece rather than along it. Meaning link 6 of the robot lies parallel to the horizontal plane rather than vertical at any given weld. This unintended effect would

be hazardous as this is a larger horizontal area with a smaller clearance for vertical plates. This was realized into the creation of the framework for new tool frame. One would have to define that a given z or x direction is selected at this ambiguity and could be defined with concepts like working planes for the robot. If this normal vector and these working planes were specified correctly, the robot could also be adapted for welds above itself or on the bottom of a part, as well as many more variations.

However, this was outside the current scope as the implementation was utilized for quick proof of concept and can handle any planar welds. Further, the current relative frame of the TCP could be adapted to various non-planar weld types (by altering R_1) as well, but requires significantly more trigonometric processing.

4.2.2 Welding Process Control

Thus far the welding parameters have been set throughout the entirety of the weld tour; it would be an improvement if these parameters could be adjusted for different weld types through the course of welding. Also there is additional state information for the measured current and voltage during the welding process, for which a custom closed loop system could be developed. Additionally, the parameters for the lead in/out, crater and start options were taken as the default. Figure 4.1 provided by Miller shows the available parameters at phases of the weld.

SECTION 7 – SETTING SEQUENCE PARAMETERS

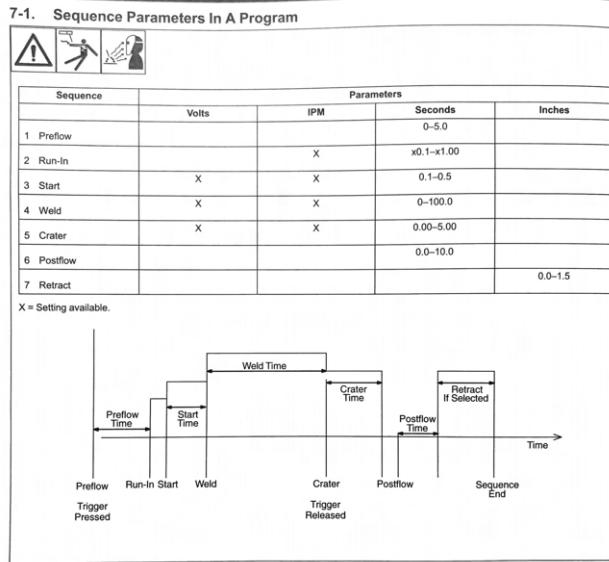


Figure 4.1: Crater Timing: available settings

4.2.3 Additional Weld Trajectories, Paths

The welder currently is only tuned to execute planar horizontal welds, although the necessary movement for vertical welds is implemented in the lift-up trajectory, executed when traversing the part. It would be useful to extend the current planar program to include vertical welds which would have to be tuned to avoid gravitational effects. This would first include vertical welds but in time could be extended to welds in a variety of orientations.

This could be done by grouping desired weld vectors into subsets occupying coincident planes. The arm could be configured into an orientation that allows the execution of each subset, extending the general solution to include the cost for transition between each operation plane.

In future implementations it would be useful to construct a more general solution, capable of generating the paths with the surface normals, either by correcting registered collisions or finding the path with a RRT* like algorithm from user selected points. These could introduce smoother trajectories that could be executed in a single joint trajectory.

The inclusion of more robust geometric processing algorithms may be useful. Currently in the

solution the desired points that are selected through PyVista, have to be embedded in the part file. It would be useful to operators if the solution could help to identify possible weld lines in the part.

The timing of the weld trajectories could also be improved, consideration was not given to the generated change over points in the joint trajectory. These can be identified by the three sub-domains of the joint angle trajectory for q_5 of a weld trajectory. These sub-domains correspond to the three identified by the weld equation; these are a result of the turn in and out of the corner in which the horizontal movement of the wire across the weld line is supplemented by the rotation of the end effector turning the TCP frame's y-axis to be perpendicular and in the plane of the weld line. Instantaneously this rotation ceases as the turn is completed; this is a result of utilizing a single Sigmoid or timing function across multiple change over points in the domain of the joint trajectory.

Past weld trajectory implementations utilized a constant weld normal turn between the weld start and finish. Experiments have also been pursued in oscillating weld paths with various quality affects and various nominal weld normals along the straight line section of the weld line. Another improvement would be the ability to execute general curved line welds. This would likely be done by aligning the desired weld normal as some offset from the path tangent at any point. This could be remedied with a method of either pre-determining the change over points or identifying them in a trajectory, and spacing individual Sigmoid timing function across each sub-domain to smoothly accelerate in and out of these regions.

Sensor integration could also lead to throughput improvements, the use of a laser guided or similar system may help to estimate the necessary weld parameters to a scanned weld line. Sensor input could also be adapted to speed up the part registration process.

4.2.4 Improving Cost Estimation MIP

Currently the decision variables in the Weld MIP TSP are not enough for the total cost of the solution to exactly equal the cost to run the solution. In other words the decision variables in the MIP do not accurately reflect the exact cost to transverse joint space in the solution. This is because only the cost or change in joint angles along each sub-path, from start point to end point,

is considered in the cost. This does not account for the accumulation of joint angles along each successive weld line, the cost to reset its end effector due to joint limits, or how the end effector winds up from reading the next weld line in the solution for the live execution.

4.2.5 Improving Hardware

The Arduino circuit represents a failure point– the cheap electronic system is likely to fail and could further be improved upon by integration into the available UR10e controller’s input and output terminals. The corresponding logic required to run the welder would have to be embedded into ROS.

4.3 Conclusion

Overall the project has advanced the pursuit of am automated robotic welding system with current results giving insight to a growing industry and guiding the future scope of the project. The next step in the implementation of the solution is to immediately work on an integrated user interface or gui, and replace the Arduino control system with an integration into the UR10e controller, making parameter options and program inputs available to operators.

The results of the current project, when without failures, showed an improvement in throughput and quality but a need for improvement in process control and robust operation procedures. Future experiments will aim to quantify the improvement in performance with acquired data.

REFERENCES

- [1] M. Biesuz, T. Saunders, D. Ke, M. J. Reece, C. Hu, and S. Grasso, “A review of electro-magnetic processing of materials (EPM): Heating, sintering, joining and forming,” *Journal of Materials Science & Technology*, vol. 69, pp. 239–272, Apr. 10, 2021, ISSN: 1005-0302. DOI: 10.1016/j.jmst.2020.06.049. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1005030220307106> (visited on 02/03/2023).
- [2] R. Goldiez. “Welding robots: Types, advantages, and limitations,” hirebotics.com. (Sep. 29, 2021), [Online]. Available: <https://blog.hirebotics.com/guide-to-welding-robots> (visited on 02/03/2023).
- [3] “5 causes of contact tip burnback & how to solve them,” American Torch Tip. (Aug. 22, 2022), [Online]. Available: <https://americantorchtip.com/blog/5-causes-of-contact-tip-burnback/> (visited on 02/15/2023).
- [4] . “How to adjust mig welder wire feed roller tension,” The Welders Warehouse. (Mar. 18, 2020), [Online]. Available: <https://www.thewelderswarehouse.com/blog/mig-welder-feed-roller-tension/> (visited on 02/15/2023).
- [5] S. Woods. “Based on a tip,” Techgen media. (Feb. 7, 2018), [Online]. Available: <https://fsmdirect.com/based-on-a-tip/> (visited on 02/15/2023).
- [6] “Porosity in welding - defects / imperfections in welds,” The Welding Institute. (), [Online]. Available: <https://www.twi-global.com/technical-knowledge/job-knowledge/defects-imperfections-in-welds-porosity-042.aspx> (visited on 02/15/2023).
- [7] “What is penetration in welding? - weldingtech.net,” Welding Tech. (Oct. 28, 2019), [Online]. Available: <https://weldingtech.net/penetration/> (visited on 02/15/2023).

- [8] M. Jünger, G. Reinelt, and G. Rinaldi, “Chapter 4 the traveling salesman problem,” in *Handbooks in Operations Research and Management Science*, ser. Network Models, vol. 7, Elsevier, Jan. 1, 1995, pp. 225–330. DOI: 10.1016/S0927-0507(05)80121-5. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0927050705801215> (visited on 02/16/2023).
- [9] B. Korte and J. Vygen, “The traveling salesman problem,” in *Combinatorial Optimization: Theory and Algorithms*, ser. Algorithms and Combinatorics, Berlin, Heidelberg: Springer, 2008, pp. 527–562, ISBN: 978-3-540-71844-4. DOI: 10.1007/978-3-540-71844-4_21. [Online]. Available: https://doi.org/10.1007/978-3-540-71844-4_21 (visited on 02/16/2023).
- [10] I. Noreen, A. Khan, and Z. Habib, “Optimal path planning using RRT* based approaches: A survey and future directions,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016, ISSN: 21565570, 2158107X. DOI: 10.14569/IJACSA.2016.071114. [Online]. Available: <http://thesai.org/Publications/ViewPaper?Volume=7&Issue=11&Code=ijacsa&SerialNo=14> (visited on 09/29/2023).
- [11] F. Ducho, A. Babinec, M. Kajan, *et al.*, “Path planning with modified a star algorithm for a mobile robot,” *Procedia Engineering*, Modelling of Mechanical and Mechatronic Systems, vol. 96, pp. 59–69, Jan. 1, 2014, ISSN: 1877-7058. DOI: 10.1016/j.proeng.2014.12.098. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S187770581403149X> (visited on 09/29/2023).
- [12] K. M. Lynch and F. C. Park, *Modern robotics: mechanics, planning, and control*. Cambridge, UK: Cambridge University Press, 2017, 528 pp., OCLC: ocn983881868, ISBN: 978-1-107-15630-2 978-1-316-60984-2.

- [13] “Cobot welder | made for welders | easy programming with your phone,” Hirebotics. (2023), [Online]. Available: <https://www.hirebotics.com/cobot-welder> (visited on 09/29/2023).
- [14] “Truly autonomous robots powered by AI technology,” Path Robotics. (), [Online]. Available: <https://www.path-robotics.com/> (visited on 09/29/2023).
- [15] “Truly autonomous welding - zero programming required - YouTube,” YouTube. (Mar. 3, 2022), [Online]. Available: <https://www.youtube.com/watch?v=uJcOWnwec6U&t=57s> (visited on 09/30/2023).
- [16] J. Grill. “MIG & flux core welding wire types & specification (with chart),” Weld Guru. Section: Flux-Cored Arc Welding. (Jun. 9, 2021), [Online]. Available: <https://weldguru.com/mig-welding-wire-types/> (visited on 09/06/2023).
- [17] C. Flannigan, M. Bazik, and J. Dinius, *Icp*, original-date: 2016-01-11T02:12:29Z, Sep. 15, 2023. [Online]. Available: <https://github.com/ClayFlannigan/icp> (visited on 09/28/2023).
- [18] C. B. Sullivan and A. A. Kaszynski, “PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK),” *Journal of Open Source Software*, vol. 4, no. 37, p. 1450, May 19, 2019, ISSN: 2475-9066. DOI: [10.21105/joss.01450](https://doi.org/10.21105/joss.01450). [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.01450> (visited on 09/06/2023).
- [19] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in python,” *Nature Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2, 2020, ISSN: 1548-7091, 1548-7105. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2). [Online]. Available: <https://www.nature.com/articles/s41592-019-0686-2> (visited on 09/06/2023).

- [20] D. Coleman, I. Sucan, S. Chitta, and N. Correll, *Reducing the barrier to entry of complex robotic software: A MoveIt! case study*, Apr. 14, 2014. arXiv: 1404 . 3785 [cs]. [Online]. Available: <http://arxiv.org/abs/1404.3785> (visited on 09/06/2023).
- [21] P. Beeson and B. Ames, “TRAC-IK: An open-source library for improved solving of generic inverse kinematics,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Seoul, South Korea: IEEE, Nov. 2015, pp. 928–935, ISBN: 978-1-4799-6885-5. DOI: 10 . 1109 / HUMANOIDS . 2015 . 7363472. [Online]. Available: <http://ieeexplore.ieee.org/document/7363472/> (visited on 09/06/2023).
- [22] “TRAC-IK,” TRACLabs. (), [Online]. Available: <https://traclabs.com/projects/trac-ik/> (visited on 09/06/2023).
- [23] H. J. Woltring, “A fortran package for generalized, cross-validatory spline smoothing and differentiation,” *Advances in Engineering Software* (1978), vol. 8, no. 2, pp. 104–113, Apr. 1986, ISSN: 01411195. DOI: 10 . 1016 / 0141 - 1195 (86) 90098 - 7. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/0141119586900987> (visited on 09/30/2023).
- [24] G. Wahba, *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Jan. 1990, ISBN: 978-0-89871-244-5 978-1-61197-012-8. DOI: 10 . 1137 / 1 . 9781611970128. [Online]. Available: <http://pubs.siam.org/doi/book/10.1137/1.9781611970128> (visited on 09/30/2023).
- [25] K. Helsgaun, “An effective implementation of the linkernighan traveling salesman heuristic,” *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, Oct. 2000, ISSN: 03772217. DOI: 10 . 1016 / S0377 - 2217 (99) 00284 - 2. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0377221799002842> (visited on 09/23/2023).
- [26] R. Tinós, K. Helsgaun, and D. Whitley, “Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic,” in *Parallel Problem Solving from Nature PPSN XV*, A. Auger,

C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds., vol. 11101, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 95–107, ISBN: 978-3-319-99252-5 978-3-319-99253-2. DOI: 10.1007/978-3-319-99253-2_8. [Online]. Available: http://link.springer.com/10.1007/978-3-319-99253-2_8 (visited on 09/23/2023).

- [27] K. Helsgaun, “General k-opt submoves for the linkernighan TSP heuristic,” *Mathematical Programming Computation*, vol. 1, no. 2, pp. 119–163, Oct. 2009, ISSN: 1867-2949, 1867-2957. DOI: 10.1007/s12532-009-0004-6. [Online]. Available: <http://link.springer.com/10.1007/s12532-009-0004-6> (visited on 09/23/2023).
- [28] T. A. M. Toffolo and H. G. Santos, *Python-MIP*. [Online]. Available: <https://www.python-mip.com/> (visited on 09/24/2023).
- [29] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using NetworkX,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.
- [30] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, First. JOHN WILEY & SONS, INC, 1989. [Online]. Available: <https://documentcloud.adobe.com/spodintegration/index.html?locale=en-us> (visited on 09/06/2023).
- [31] C. R. Harris, K. J. Millman, S. J. Van Der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 17, 2020, ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: <https://www.nature.com/articles/s41586-020-2649-2> (visited on 09/06/2023).

APPENDIX A

PRELIMINARY RESULTS AND REFERENCES

A.1 Reference Figures

Technique	Electrode	Filler metal	Shielding	Plasma gas	Indicative Electric arc data	Polarization
Shielded Metal Arc Welding	Consumable: metal core with flux coating	From the consumable electrode	Molten flux, gasses developed from the flux decomposition	Air + gasses developed from flux decomposition	25 – 500 A 15 – 35 V	DCEP, DCEN, AC
Flux-Cored Arc Welding	Consumable: tubular electrode containing powder of metal, flux, slag forming and deoxidizing agents	From the consumable electrode	Molten flux, gasses developed from the flux decomposition or inert shielding gas (Ar or CO ₂)	Air + gasses developed from flux decomposition or inert shielding gas (Ar or CO ₂)	100 – 600 A 20 – 30 V	DCEP, DCEN
Carbon Arc Welding	Carbon or graphite "partially" consumable	With (filler rod place close to the melted pool) or without	CO developed from the electrode oxidation	CO + atmospheric air	15 – 700 A	DCEN or AC
Gas Metal Arc Welding (also known as MIG)	Consumable	From the consumable electrode	Inert gas (usually Ar or mixtures with He, CO ₂ , H ₂ or small quantities of O ₂)	Same of the shielding gas	100 – 400 A 20 – 70 V	Usually DCEP
Gas Tungsten Arc Welding (also known as TIG)	Non consumable (Tungsten)	With (filler rod place close to the melted pool) or without	Inert gas (usually Ar, also He or mixtures are used)	Same of the shielding gas	300 – 500 A 10 – 15 V	Usually DCEN, less common AC and DCEP
Plasma Arc Welding (PAW)	Non consumable (Tungsten)	With (filler rod place close to the melted pool) or without	Inert gas (usually Ar, also Ar/He and Ar/H ₂ mixtures are used)	Ar (in this case two gasses are fluxed: one for shielding and one for sustaining the electric arc)	15 – 500 A (micro-welding systems are also available using 0.5–10 A) 20 – 40 V 40 – 200 A 20 – 40 V	Usually DCEN, less common AC
Plasma-MIG Welding	Consumable	With (filler rod place close to the melted pool) or without	Inert gas (usually Ar, or mixtures)	Ar (in this case two gasses are fluxed: one for shielding and one for sustaining the electric arc)	150 – 1000 A 20 – 40 V	Almost always DCEP
Submerged Arc Welding	Consumable	From the consumable electrode	Granular flux provided by and hopper moving close to the plasma torch	The discharge takes place within the molten fluxes	150 – 1000 A 20 – 40 V	DCEP, DCEN, AC
Stud Arc Welding	One electrode is a stud the other a metal component. When the arc creates a melt the stud is pushed on the component and welds	Without	Usually no shielding, for joining Al components an inert gas flow is used	Air	150 – 200 A	DCEP, DCEN
Capacitor Discharge Stud Arc Welding	One electrode is a stud the other a metal component. When the arc creates a melt the stud is pushed on the component and welds	Without	Usually no shielding, for joining Al components an inert gas flow is used	Air	2000 – 10,000 A	Capacitor discharge

Figure A.1: Welding Types Reprinted from [1]: here DCEP/DCEN stands for (Direct Current Electrode Positive/Negative)

A.2 Original Point-to-Point Path Generation

For the original grid part and implementation of the solution, two possible scaling's were given for each point, based on the movement to an adjacent inner point or the required movement over a vertical section of the part. As mentioned two height scalings were originally chosen and selected based on the distance between any two selected points:

As shown in figure A.3, in the original part there exists subsets of four points that don't have to cross a vertical ridge, as such when generating the path between any two random desired points

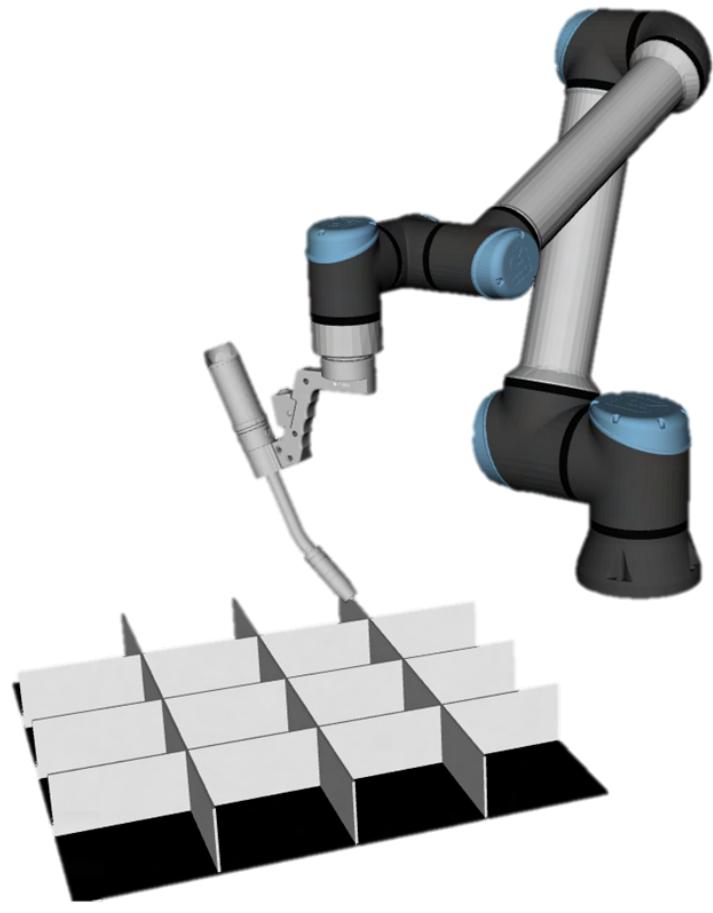


Figure A.2: Part V.1 Relative to UR10e

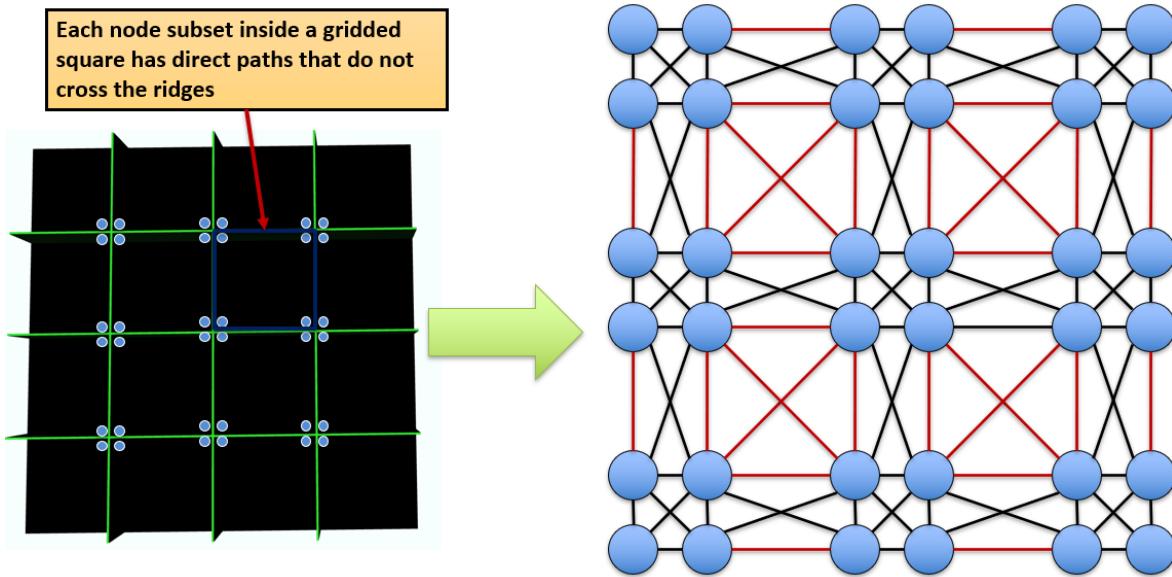


Figure A.3: Part V.1 Represented as a Complete Graph (not every connection is drawn for simplicity): red identifies the internal subsets

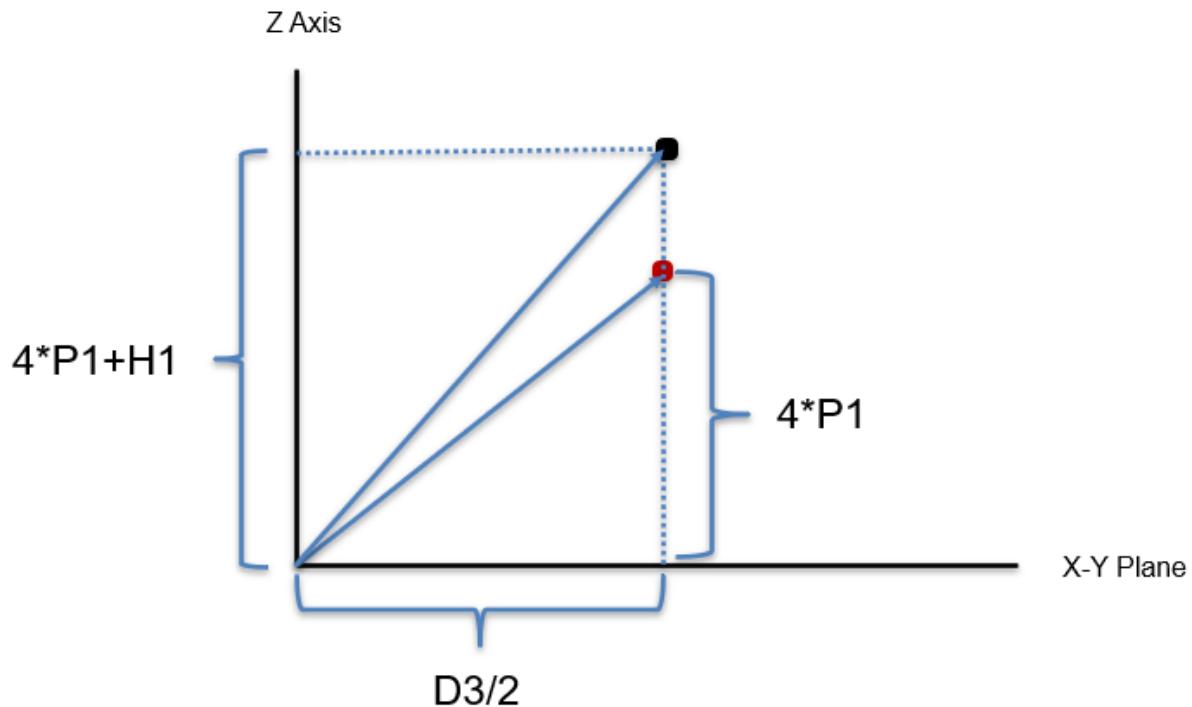


Figure A.4: Tall and Short Path Scaling for Part V.1: here D_3 is the diagonal distance inside a grid, P_1 is the plat height and H_1 is the height of the parts ridges. Red Point: Low Control Point, Black Point: High Control Point

a distance computation is utilized to classify these points. If the euclidean distance between these points is equal to the diagonal, or side length of the square grid, it is immediately classified as a interior edge, otherwise being a path that must clear the ridges. These points are fed into a B-Spline implementation in SciPy. Then these splines are interpolated at some pre-selected resolution scaled by the length of the spline. Examples of these generated splines can be found in figure A.5.

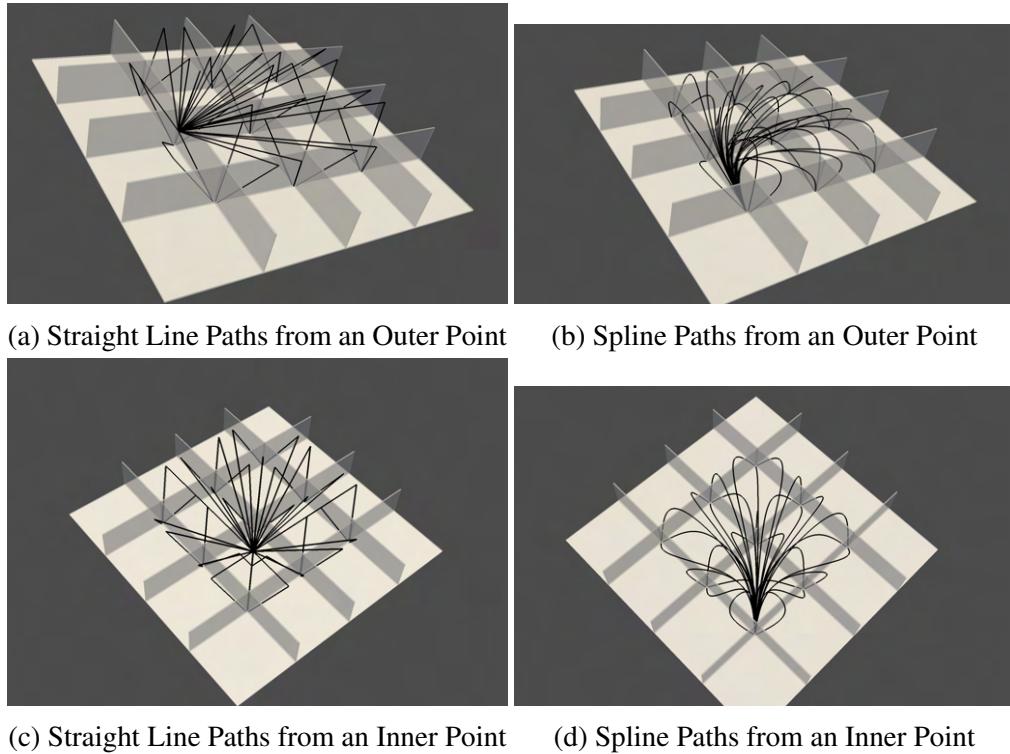


Figure A.5: Original Part Control Point Paths

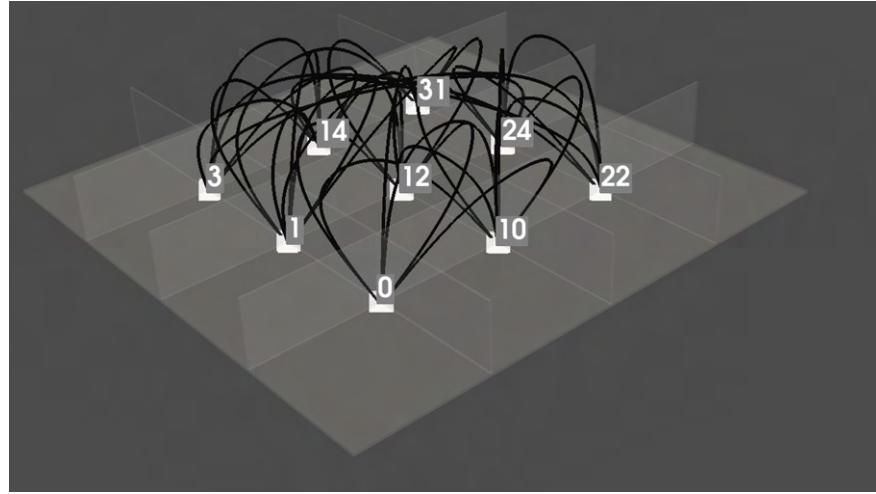


Figure A.6: Original TSP Tacking Solution

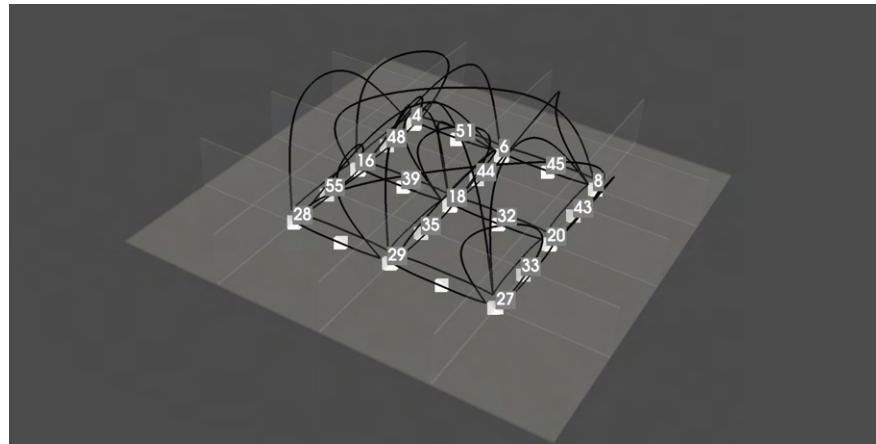


Figure A.7: Original Mixed Integer Program Solution to Weld TSP

A.3 Original Part Results

Figures in this section show the massive warping of the original part, misaligned welds and burn-through characteristic of the first implementation. The thickness of the early prototype part was below the recommended minimum for welding with 0.035 inch wire, this was a reason for the weld burning through the material and the significant deformation of the part due to poor heat

dissipation. The misaligned welds were because of a misaligned TCP utilized in the program compared to the physical tool. This is what led to the redesign of the part present in results.



Figure A.8: Warping of the Original $\frac{1}{8}$ Inch Thick Weld Grid



Figure A.9: Misaligned Welds of the Original $\frac{1}{8}$ Inch Thick Weld Grid



Figure A.10: Burn-through Welds of the Original $\frac{1}{8}$ Inch thick Weld Grid

A.4 Weld Tip Re-calibration

The observed error of the provided TCP point.



(a) TCP Error: metal tip versus white dot

(b) TCP Error Measured Distance

Figure A.11: Quoted BA1 Error

A.5 Polynomial Curve Fitting

The polynomial fitting was done with a procedure outlined in [30]. This is done with 5 parametric functions for each desired joint's trajectory. For each angle in the joint trajectory at a specified time, a constant must be added to the polynomial to facilitate this constraint. Given n angles at n points in time:

$$p(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_{n-1} t^{n-1}.$$

Additionally for each velocity and acceleration (or higher derivative) constraint k an additional degree must be added to the polynomial:

$$p(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_{n-1} t^{n-1} + \dots + a_{(n-1)+k} t^{(n-1)+k}.$$

These constraints can be laid out as:

$$\theta(t) = p(t)$$

for each specified angle at a specified time. If there is a given velocity or derivative constraint these can be expressed as:

$$\frac{d\theta(t)}{dt} = \frac{dp(t)}{dt},$$

$$\frac{d^2\theta(t)}{dt^2} = \frac{d^2p(t)}{dt^2},$$

or:

$$\frac{d^n\theta(t)}{dt^n} = \frac{d^n p(t)}{dt^n}.$$

For each time $t = t_0, \dots, t_f$ with $((n - 1) + k) = p$:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & \dots & t_0^p \\ 0 & 1 & 2t_0 & \dots & pt_0^{(p-1)} \\ 0 & 0 & 2 & \dots & p(p-1)t_0^{(p-2)} \\ \vdots & & & & \\ 1 & t_f & t_f^2 & \dots & t_f^p \\ 0 & 1 & 2t_f & \dots & pt_f^{(p-1)} \\ 0 & 0 & 2 & \dots & p(p-1)t_f^{(p-2)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \theta(t_0) \\ \frac{d\theta(t_0)}{dt} \\ \frac{d^2\theta(t_0)}{dt^2} \\ \vdots \\ \theta(t_f) \\ \frac{d\theta(t_f)}{dt} \\ \frac{d^2\theta(t_f)}{dt^2} \end{bmatrix}$$

as:

$$T \vec{a} = \vec{\Theta}$$

and can be solved utilizing the NumPy Python Library [31]:

$$\vec{a} = T^{-1} \vec{\Theta}.$$

As T is guaranteed to be non-singular. Further the components of T can be expressed as:

$$\forall (i, d) \in \{i \geq d\} \Rightarrow T_{(i,j)} = \left(\frac{i!}{(i-d)!}\right)(t_j)^{(i-d)},$$

$$\forall (i, d) \in \{i < d\} \Rightarrow T_{(i,j)} = 0.$$

Here d represents the given derivative at row j where:

$$\forall d \neq 0 \Leftrightarrow t_j = t_{j-1}.$$

An example of this polynomial fit is shown in 2.25. The constraints utilized are setting initial and final joint velocities to zero, and setting the desired joint angles at each time interval provided by the Sigmoid curve.

APPENDIX B

EQUIPMENT

B.1 Relevant Manual Pages

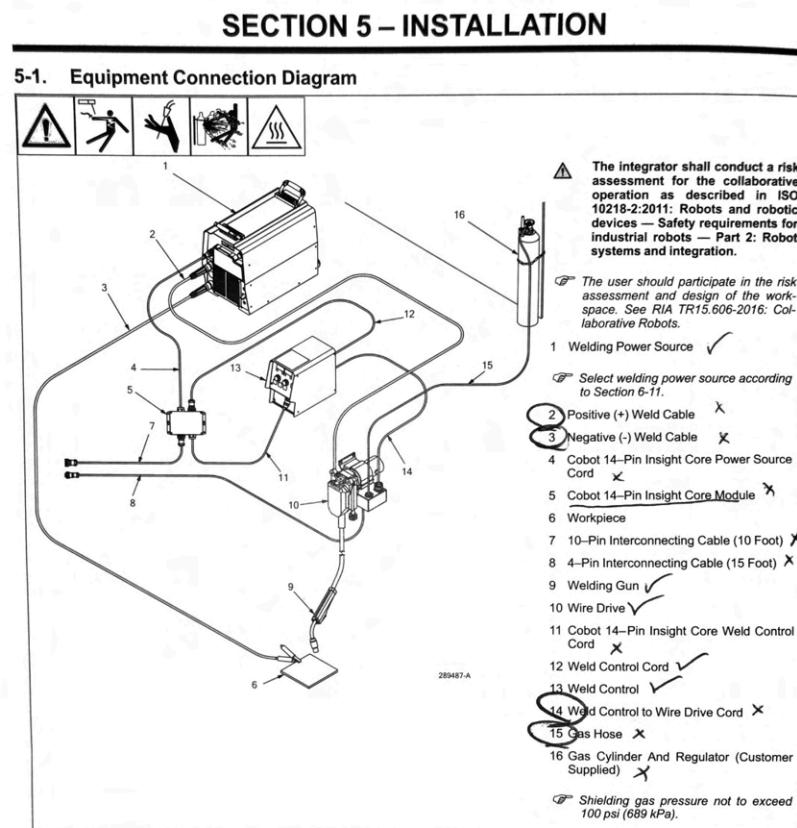


Figure B.1: Provided Miller Connection Diagram

3-7. Recommended Cobot Interface Connection Information

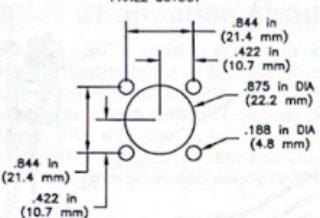
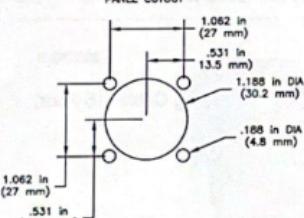
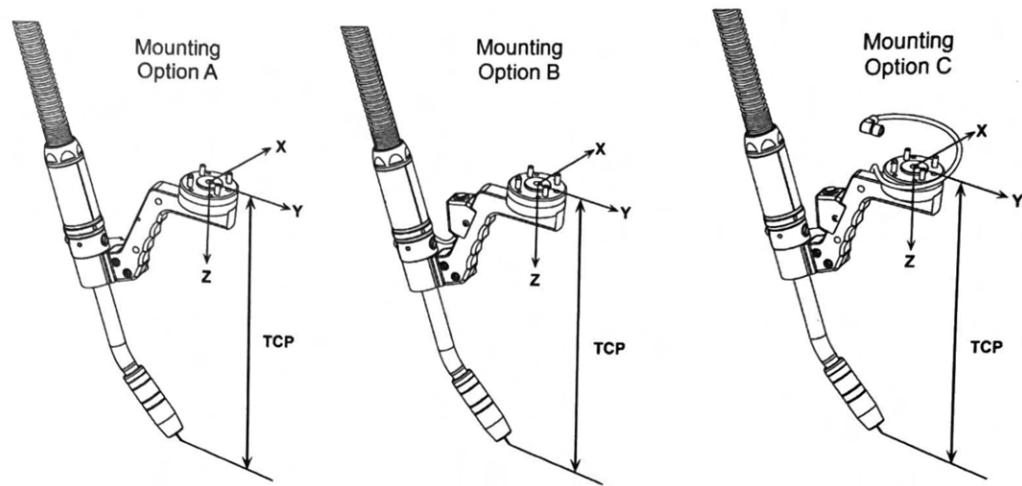
 PANEL CUTOUT  <p>844 in (21.4 mm) .422 in (10.7 mm) 875 in DIA (22.2 mm) .188 in DIA (4.8 mm)</p>	<table border="1"> <thead> <tr> <th>SOCKET</th><th>WIRE NUMBER</th><th>DESCRIPTION</th></tr> </thead> <tbody> <tr> <td>1</td><td>1A</td><td>JOG SOURCE</td></tr> <tr> <td>2</td><td>2A</td><td>JOG COMMAND</td></tr> <tr> <td>3</td><td>3A</td><td>PURGE COMMAND</td></tr> <tr> <td>4</td><td>4A</td><td>PURGE SOURCE</td></tr> </tbody> </table> <p>4-Pin Receptacle</p> <p>Contains wire feeder signals for control of Jog and Purge functions.</p> <p>1 Jog Relay Open = Jog Off Closed = Jog On</p> <p>2 Purge Relay Open = Purge Off Closed = Purge On</p>	SOCKET	WIRE NUMBER	DESCRIPTION	1	1A	JOG SOURCE	2	2A	JOG COMMAND	3	3A	PURGE COMMAND	4	4A	PURGE SOURCE																		
SOCKET	WIRE NUMBER	DESCRIPTION																																
1	1A	JOG SOURCE																																
2	2A	JOG COMMAND																																
3	3A	PURGE COMMAND																																
4	4A	PURGE SOURCE																																
 PANEL CUTOUT  <p>1.062 in (27 mm) .531 in (13.5 mm) 1.188 in DIA (30.2 mm) .188 in DIA (4.8 mm)</p>	<table border="1"> <thead> <tr> <th>SOCKET</th><th>WIRE NUMBER</th><th>DESCRIPTION</th></tr> </thead> <tbody> <tr> <td>A</td><td>20A</td><td>COMMON</td></tr> <tr> <td>E</td><td>19A</td><td>WELD OUTPUT COMMAND</td></tr> <tr> <td>C</td><td>18A</td><td>WELD AMP FEEDBACK</td></tr> <tr> <td>B</td><td>17A</td><td>WELD VOLT FEEDBACK</td></tr> <tr> <td>J</td><td>16A</td><td>MODE SELECT</td></tr> <tr> <td>I</td><td>15A</td><td>MODE SELECT COMMON</td></tr> <tr> <td>D</td><td>11A</td><td>WELD ENABLE SOURCE</td></tr> <tr> <td>G</td><td>12A</td><td>WELD OUTPUT ENABLE</td></tr> <tr> <td>H</td><td>13A</td><td>WFS COMMON</td></tr> <tr> <td>F</td><td>14A</td><td>WFS COMMAND</td></tr> </tbody> </table> <p>10-Pin Receptacle</p> <p>Contains welding power source signals for weld output command, weld voltage and amperage feedbacks, mode select, and weld output enable; contains wire feeder signal for wire feed speed.</p> <p>1 Weld Output Command DAC</p> <p>→ MIG Mode – 0 to 10 VDC = 10.0 to 38.0 V or 44.0 V depending on welding power source. See owner's manual.</p> <p>2 Weld Current Feedback ADC 0 to 10 VDC = 0 to 1000 Weld Amps</p> <p>3 Weld Voltage Feedback ADC 0 to 10 VDC = 0 to 100 Weld Volts</p> <p>4 Weld Mode Relay Open = MIG; Closed = Pulsed MIG</p> <p>5 Weld Output Relay Open = Weld Off; Closed = Weld On</p> <p>6 Wire Feed Speed DAC 0 to 5 VDC = 50 to 780 IPM</p> <p>☞ Analog and digital converters (ADC, DAC) must be isolated from earth (ground).</p> <p>☞ Wire Feed Speed DAC must be isolated from the other ADCs and DAC.</p> <p>☞ ADCs and DACs must be calibrated (gain and offset).</p> <p>☞ Recommended: use signal conditioner for analog inputs/outputs (I/O).</p>	SOCKET	WIRE NUMBER	DESCRIPTION	A	20A	COMMON	E	19A	WELD OUTPUT COMMAND	C	18A	WELD AMP FEEDBACK	B	17A	WELD VOLT FEEDBACK	J	16A	MODE SELECT	I	15A	MODE SELECT COMMON	D	11A	WELD ENABLE SOURCE	G	12A	WELD OUTPUT ENABLE	H	13A	WFS COMMON	F	14A	WFS COMMAND
SOCKET	WIRE NUMBER	DESCRIPTION																																
A	20A	COMMON																																
E	19A	WELD OUTPUT COMMAND																																
C	18A	WELD AMP FEEDBACK																																
B	17A	WELD VOLT FEEDBACK																																
J	16A	MODE SELECT																																
I	15A	MODE SELECT COMMON																																
D	11A	WELD ENABLE SOURCE																																
G	12A	WELD OUTPUT ENABLE																																
H	13A	WFS COMMON																																
F	14A	WFS COMMAND																																

Figure B.2: Provided Miller Connection Information

5-2 Center of Mass Coordinates



Standard Configurations	Tool Center Point (TCP)	Center of Mass			
		X	Y	Z	Weight
22 Degree Neck (405-22QC)					
Mounting option A: BAS2201A (Standard)	350 mm	1.10 mm	-149.92 mm	27.33 mm	2.86 kg
Mounting option B: BAS2201B (Free drive for controller I/O connection)	350 mm	1.21 mm	-149.70 mm	26.96 mm	2.91 kg
Mounting option C: BAS2201C (Free drive for UR wrist I/O connection)	350 mm	1.09 mm	-149.06 mm	27.48 mm	2.90 kg
45 Degree Neck (405-45QC)		X	Y	Z	Weight
Mounting option A: BAS4501A (Standard)	350 mm	1.11 mm	-75.67 mm	26.65 mm	2.83 kg
Mounting option B: BAS4501B (Free drive for controller I/O connection)	350 mm	1.25 mm	-75.56 mm	26.58 mm	2.89 kg
Mounting option C: BAS4501C (Free drive for UR wrist I/O connection)	350 mm	1.12 mm	-75.61 mm	26.85 mm	2.86 kg

Figure B.3: Provided TCP for BA1

9-3. Troubleshooting



Disconnect power before troubleshooting.

Trouble	Remedy
Pressing gun trigger does not energize feeder. Shielding gas does not flow and wire feeder does not feed.	Secure plug from gun control cable into Gun Control receptacle. Have nearest Factory Authorized Service Agent check optional water flow switch, if applicable.
Wire feeds, shielding gas flows, but welding wire is not energized.	Check to see if ground clamp or weld cable is connected.
Wire feeds erratically.	Verify proper wire size is selected (see Section 8-2). Check drive roll pressure in wire feeder and gun. Clean or replace drive rolls as necessary. Check and replace liner if necessary.
Arc varies and welding wire is kinked when feeding out gun.	Verify proper wire size is selected (see Section 8-2).
No weld output; gun/feeder does not work.	Check gun control cable connection.
Erratic weld output.	Tighten and clean all connections. Replace contact tip. Verify proper wire size is selected (see Section 8-2). Check drive roll pressure in wire feeder and gun. Check and replace liner if necessary. When welding aluminum, it is important to use minimal drive roll pressure and minimal brake tension to achieve consistent wire feeding.
Wire does not feed; burnback in contact tip.	Verify drive roll size is correct. Check drive roll pressure in wire feeder and gun. Check and replace liner if necessary.
Gun overheating (water-cooled models).	Be sure coolant flow rate is at least 1 qt/min. Corrosion buildup in gun decreasing coolant flow rate. Backflush coolant system, clean coolant system filter, and clean fittings.
Motor does not run.	Check drive roll pressure in wire feeder and gun. Check and replace liner if necessary. Have Factory Authorized Service Agent check feeder.

Figure B.4: Wire Feeder Common Problems

7-5. Pulsed MIG - Wire and Gas Selection Table

WIRE TYPES**		GAS TYPES
Steel	.035 (0.9) STL .045 (1.2) STL	ARGN CO2 (ARGON / CARBON DIOXIDE) 80 ARGN CO2 (ARGON / CARBON DIOXIDE) ARGN OXY (ARGON / OXYGEN)
Steel 100S	.035 (0.9) STL .045 (1.2) STL	100S C5 (95 ARGON / 5 CARBON DIOXIDE)
Metal Core	.045 (1.2) MCOR .052 (1.4) MCOR	ARGN CO2 (ARGON / CARBON DIOXIDE)
Stainless Steel	.035 (0.9) SSTL .045 (1.2) SSTL	TRI MIX (TRI-GAS MIXTURE) ARGN OXY (ARGON / OXYGEN) ARGN CO2 (ARGON / CARBON DIOXIDE)
Aluminum	.035 (0.9) AL4X (4000 Series) .040 (1.0) AL4X (4000 Series) 3/64 (1.2) AL4X (4000 Series) 1/16 (1.6) AL4X (4000 Series)	ARGN (ARGON)
	.035 (0.9) AL49 (4943) .040 (1.0) AL49 (4943) 3/64 (1.2) AL49 (4943) 1/16 (1.6) AL49 (4943)	
	.035 (0.9) AL5X (5000 Series) .040 (1.0) AL5X (5000 Series) 3/64 (1.2) AL5X (5000 Series) 1/16 (1.6) AL5X (5000 Series)	
Nickel	.035 (0.9) NI .045 (1.2) NI	ARGN HE (ARGON / HELIUM) ARGN (ARGON)
Copper Nickel	.035 (0.9) CUNI .045 (1.2) CUNI	HE ARGN (HELUM / ARGON)
Silicon Bronze	.035 (0.9) SIBR .045 (1.2) SIBR	ARGN (ARGON)
Titanium	.035 (0.9) TI-5 .045 (1.2) TI-5	ARGN HE25 (75 ARGON / 25 HELIUM)
	.035 (0.9) TI-5 .045 (1.2) TI-5	ARGN HE50 (50 ARGON / 50 HELIUM)

**Wire diameter in inches (mm). Refer to section 6-2 to change displayed units.

Figure B.5: Recommended Shielding Gas

SECTION 7 — TROUBLESHOOTING

7-1 Troubleshooting Table

PROBLEM	POSSIBLE CAUSE	CORRECTIVE ACTION
1. Free drive button does not work.	1. Faulty push button. 2. Faulty wiring.	1. a. Verify function using multimeter or pendant. b. Replace free drive assembly.
2. Electrode does not feed.	1. Feeder relay. 2. Broken control lead. 3. Poor adaptor connection. 4. Improper / worn drive roll. 5. Drive roll tension misadjusted. 6. Burn back to contact tip. 7. Wrong size liner. 8. Buildup inside of liner.	1. Consult feeder manufacturer. 2. a. Test and connect spare control lead. b. Install new cable. 3. Test and replace leads and/or contact pins. 4. Replace drive roll. 5. Adjust tension at feeder. 6. See 'Contact tip burn back' . 7. Replace with correct size. 8. Replace liner or clean out with compressed air, check condition of electrode.
3. Contact tip burn back.	1. Improper voltage and/or wire feed speed. 2. Erratic wire feeding. 3. Improper tip stickout. 4. Improper electrode stickout. 5. Faulty ground.	1. Adjust parameters. 2. See 'Erratic wire feeding' . 3. Adjust nozzle / tip relationship. 4. Adjust wire stickout. 5. Replace cables and/or connections.
4. Tip disengages from the gas diffuser.	1. Worn retaining head / diffuser. 2. Improper tip installation. 3. Extreme heat or duty cycle.	1. Replace tip and/or gas diffuser / retaining head / diffuser. 2. Install as per section 4-1 Changing Consumables on page 11 Replace with heavy duty consumables. See appropriate Spec Sheet for details.
5. Short contact tip life.	1. Contact tip size. 2. Electrode eroding contact tip. 3. Exceeding duty cycle.	1. Replace with proper size. 2. Inspect and/or change drive rolls. 3. Replace with properly rated Tregaskiss MIG gun.
6. Erratic arc.	1. Worn contact tip. 2. Buildup inside of liner. 3. Wrong tip size. 4. Not enough bend in neck.	1. Replace contact tip. 2. Replace liner, check condition of electrode. 3. Replace with correct tip size. 4. Replace with 45° neck.
7. Erratic wire feeding.	1. Buildup inside of liner. 2. Wrong size liner. 3. Improper drive roll size. 4. Worn drive roll. 5. Improper guide tube relationship. 6. Improper wire guide diameter. 7. Gaps at liner junctions. 8. Feeder malfunction. 9. Worn contact tip.	1. Replace liner, check condition of electrode. 2. Replace with new liner of proper size. 3. Replace with proper size drive roll. 4. a. Replace with new drive roll. b. Repair worn drive roll. 5. a. Adjust / replace guide as close to drive rolls as possible. b. Eliminate all gaps in electrode path. 6. Replace with proper guide diameter. 7. a. Replace with new liner trimmed as per Section 4-4 Changing the Liner on page 15. b. Replace guide tube / liner trimming as close to mating component as possible. 8. Consult feeder manufacturer. 9. Inspect and replace.*
8. Extreme spatter.	1. Improper machine parameters. 2. Improper tip installation. 3. Improper shielding gas coverage. 4. Contaminated wire or workpiece.	1. Adjust parameters. 2. Adjust nozzle / tip relationship. 3. a. Verify shielding gas coverage. b. Verify gas mixture. 4. Clean wire and workpiece.
9. Porosity in weld.	1. Insulator worn. 2. Gas diffuser damaged 3. Extreme heat or duty cycle. 4. Solenoid faulty. 5. No gas. 6. Flow improperly set. 7. Gas ports plugged. 8. Ruptured gas hose. 9. Control circuit loss. 10. Worn, cut or missing o-rings. 11. Loose fittings.	1. Replace nozzle / insulator. 2. Replace gas diffuser or o-rings. 3. Replace with heavy duty consumables. 4. Replace solenoid. 5. a. Install full tanks. b. Check supply. c. Check for hose leaks. 6. Adjust flow. 7. a. Clean or replace gas diffuser. b. Clean nozzle. 8. Repair or replace cable or line. 9. See 'Electrode does not feed' . 10. Replace o-rings. 11. Tighten gun and cable connections to specified torque. See Section 4 — Replacement starting on page 11.
10. Gun running hot.	1. Exceeding duty cycle. 2. Loose or poor power connection.	1. a. Replace with properly rated Tregaskiss MIG gun. b. Decrease parameters to within gun rating. 2. a. Clean, tighten or replace cable grounding connection. b. Tighten gun and cable connections to specified torque. See Section 4 — Replacement on page 11.
11. Liner is discolored.	1. Short circuit to electrode. 2. Broken copper stranding in power cable.	1. Isolate electrode reel from feeder and drive block. Consult feeder manufacturer's manual. 2. Replace unicable.
12. Sporadic feeding of aluminum electrode.	1. Tip galling. 2. Synthetic liner melting. 3. Wire deformed by feeder rolls.	1. Inspect and replace the contact tip.* 2. a. Replace liner. b. Replace with composite liner. c. Replace the neck and jump liner. 3. Adjust drive rolls as per feeder manufacturer's manual.

Figure B.6: Common Welder Issues and Solutions

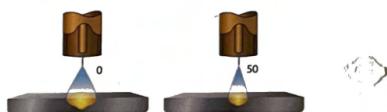


INVISION

Arc Control

Pulse MIG

- Adjusts **Sharp Arc** (arc focus) has a range from 0-50
- **Press setup once**
- *Factory default is **set at 25** (see picture)
- (Example select: 25)



MIG process

- Adjust inductance to add puddle fluidity (see picture)
- *Factory default is **set at 25** for steel or set 70 for Stainless Steel (Range 0-100)



Figure B.7: Miller Arc Control

B.2 Component List

Table B.1: Component List

Part	Description
WIRE MS 70S6 035 33# SP	33 lb reel of 0.035 inch diameter Mig Wire Mild Steel
WIRE REEL STD	Wire reel stand from Lincoln Electric
BERTT-A035CH	Tapered Contact tip machined into the measurement (10 tips)
TRE59D06 INSULATING DISC KUKA	Connecting polymer 'chuck' from weld gun to UR10e
MIL907431 MIG WELDER INVISION 352 MPA	Invision welder
CONTROL BOX S74 MPA PLUS COBOT MODEL	Weld Feeder Controller
TREBAS2201C MOUNTING ARM ASSEMBLY	Mounting arm from 'chuck' to weld gun
TREBA12AAANDOCM MIG GUN BA1 COBOT AIR COOLED	gun connected to the wire supply and the mountaing arm
MIL151026 DRIVE ROLL KIT 035 V GRV 4 RO ROLL	Wire feeder rollers for 0.035 inch wire
MIL300740 DRIVE ASSY 74MPA	wire feeder and gas supplier
ADAPTER TO WIRE FEEDER PIPWPT6	Adapter to wire feeder
MIL254864010 CABLE MOTOR GAS MPA PLUS	Cable to wire drive
MIL300405 CONNECTOR KIT	Weld cables not provided with other orders

B.3 Additional Equipment Images

In figure B.11 the lattice grid on the table was the original part to be tested on the second iteration of the project, this was abandoned for the prototype-parts shown in the results.



Figure B.8: Miller Invision 352 MPa



Figure B.9: Miller S-74 MPa, with crater and start functions



Figure B.10: Miller Insight Module

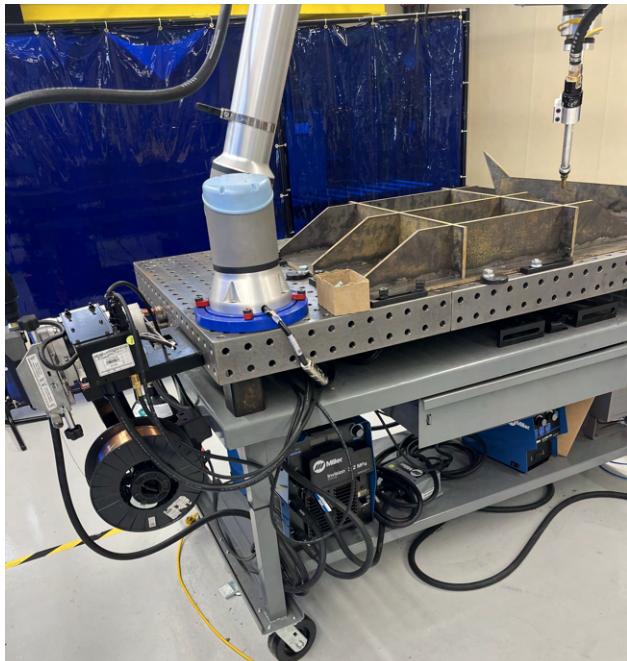


Figure B.11: Assembled System, with wire feeder on the left, and the S-74, Insight Module and Invision under the table, on the table is the real-size grid model



Figure B.12: Under-side of Weld Table with: Invision 352, Insight Module, S-74, and UR10e controller

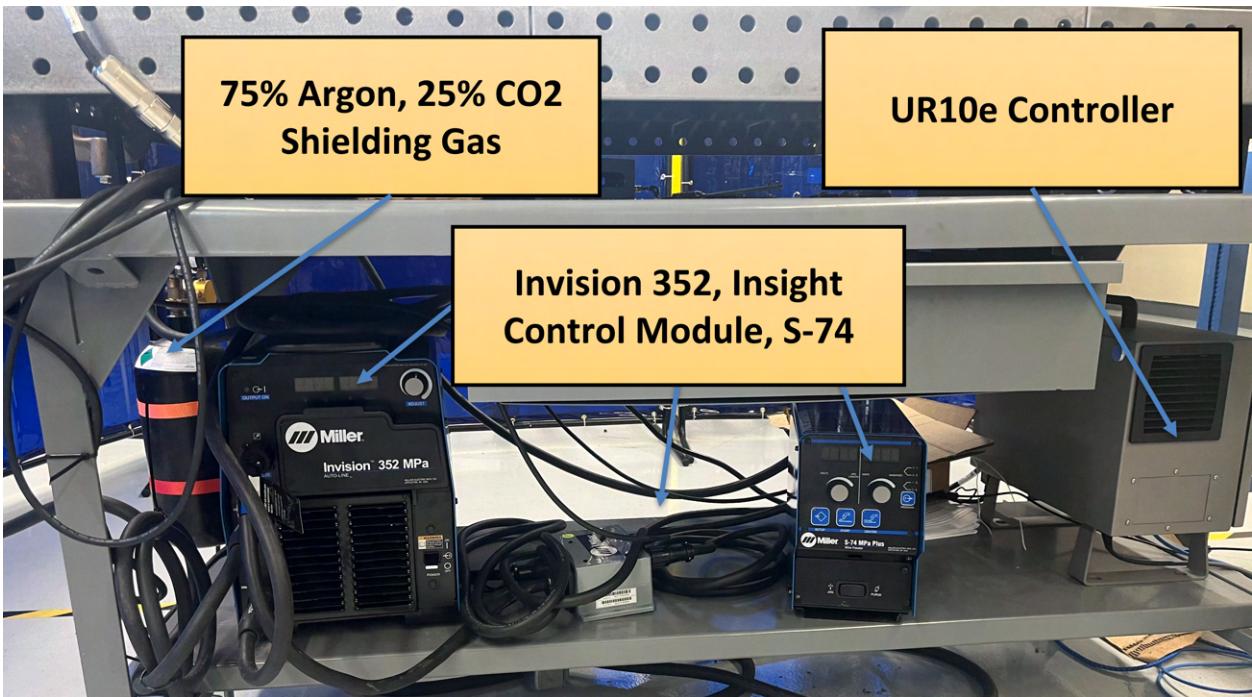


Figure B.13: Under-side of Weld Table: labeled