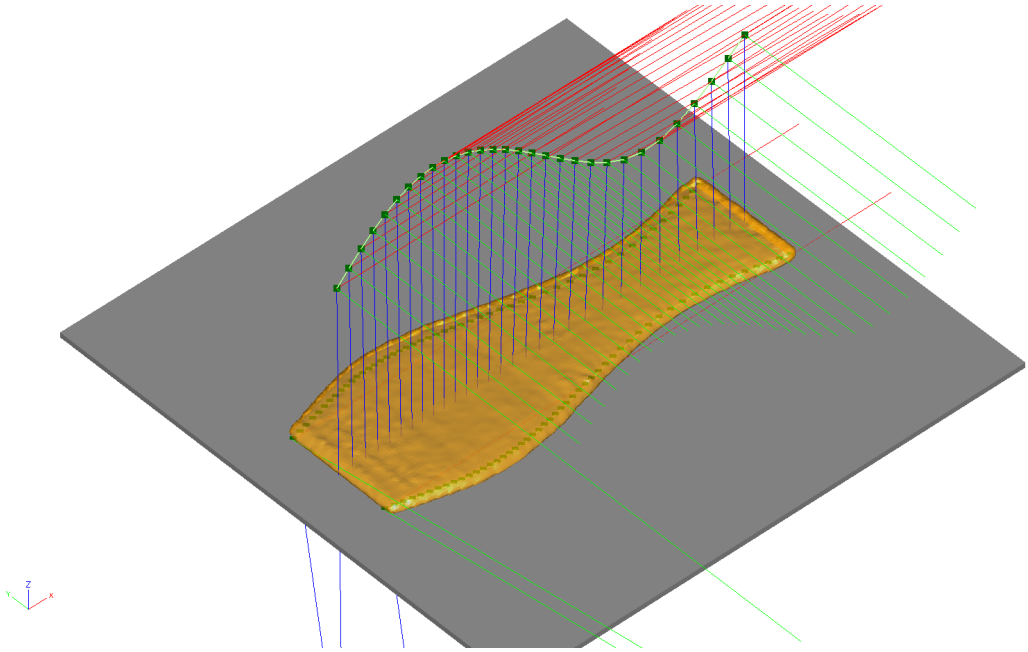
CHALMERS
UNIVERSITY OF TECHNOLOGY



Automatic Robot Trajectory Generation for Sealing Applications

Systems, control and mechatronics, MSc

Marcus Berg

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Automatic Robot Trajectory Planning for Sealing Applications

Marcus Berg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Automatic Robot Trajectory Generation for Sealing Applications
Marcus Berg

Supervisor: Robert Bohlin, Johan S. Carlson
Examiner: Knut Åkesson, Department of Electrical Engineering

Master's Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Automatic Robot Trajectory Generation for Sealing Applications
Marcus Berg
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Automatic robot trajectory generation can reduce production time and material use in automotive manufacturing. This thesis proposes a geometric approach to formulate a nonlinear constrained optimization problem. The objective is to compute an initial Tool Center Point (TCP) path that achieves a uniform height distribution along two curves that define the desired sealant edges. The initial TCP path yields sufficient trajectories for a wide range of curves. However, in scenarios involving sharp edges, some limitations related to the physical limitations of mechanical devices restrict the range of feasible sealants. The geometric approach does not consider all complex sealing behaviors. To capture the complexity of the sealing process while maintaining fast validation, a neural network was developed that predicts sealant cross-sections along a TCP path. An obstacle course that introduces unseen kinematic and spatial features was created to validate the network. The network performs well on simple geometries, but more development is required for the model to be deployable in practice. The geometric approach and the surrogate model build a framework for further optimization of the TCP path.

Keywords: Optimization, trajectory, sealant, CNN, surrogate, trajectory, pixel.

Acknowledgements

Firstly, I thank Fraunhofer Chalmers Centre for the opportunity to perform and support my thesis. I am especially grateful to Robert Bohlin and Johan S. Carlson for their insightful advice and for helping to shape the direction of this research. I also want to thank my examiner, Knut Åkesson, for their time and evaluation of this work. Lastly, I would like to express my gratitude to my family and friends for their support and encouragement.

Marcus Berg, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CNN	Convolutional Neural Network
FCC	Fraunhofer Chalmers Centre
IBOFLOW	Immersed Boundary Octree Flow Solver
IPS	Industrial Path Solutions
MLP	Multilayer Perceptron
MSE	Mean Square Error
SGD	Stochastic Gradient Descent
SLSQP	Sequential Least Squares Quadratic Programming
TCP	Tool Center Point

Nomenclature initial solution

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Discretizations index

Variables for calculation of optimal intial TCP path

$\mathbf{x_i}$	Left sealant edge position at index i
$\mathbf{y_i}$	Right sealant edge position at index i
$\mathbf{p_i}$	TCP position at at index i
$\mathbf{R_i}$	TCP rotation matrix at index i
a_i	Width of sealant at index i
b_i	Distance between $\mathbf{x_i}$ and $\mathbf{p_i}$ at index i
c_i	Distance between $\mathbf{y_i}$ and $\mathbf{p_i}$ at index i
d_i	Distance between $\mathbf{p_i}$ and substrate intersection point
β_i	Entry angle from $\mathbf{p_i}$ and s $\mathbf{x_i}$ at index i
γ_i	Entry angle from $\mathbf{p_i}$ and s $\mathbf{y_i}$ at index i
η_i	Angle between $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$
ζ_i	Angle between $\mathbf{y_i}$ and $\mathbf{y_{i+1}}$
u_i	Arc length along $\mathbf{x}$ at index i
w_i	Arc length along $\mathbf{y}$ at index i
$\dot{u}_i$	Velocity along $\mathbf{x}$ at index i
$\dot{w}_i$	Velocity along $\mathbf{y}$ at index i
ρ_i	Tilt at index i
κ_i	Drag at index i

t_i	Time at index i
g_i	Sealant height at point $\mathbf{x_i}$
h_i	Sealant height at point $\mathbf{y_i}$
$A_{x,i}$	Infinitesimally small area around point $\mathbf{x_i}$
$A_{y,i}$	Infinitesimally small area around point $\mathbf{y_i}$
$\mathbf{x_i'}$	Direction of $\mathbf{x}$ at index i
$\mathbf{y_i'}$	Direction of $\mathbf{y}$ at index i
$\mathbf{o_i}$	Center of drag circle at index i
ϱ_i	Radius of drag circle at index i
J	Cost function to be minimized
t_f	Time to perform TCP path
k_1	Cost variable penalizing spin
k_2	Cost variable penalizing drag
k_3	Cost variable penalizing distance of TCP path
$\dot{V}$	Volume flow of sealant

Variables for Surrogate model

$\mathbf{v_i}$	Local linear velocity for TCP at index i
$\boldsymbol{\omega_i}$	Local angular velocity for TCP at index i
$\mathbf{g_i}$	Local gravity vector for TCP at index i
$\mathbf{h_i}$	Distance vector at index i
$\boldsymbol{\gamma_i}$	Principal rotation vector for TCP for index i to index i+1
Φ_i	Principal angle for TCP for index i to index i+1
$\hat{e}_i$	Principal vector from index i to index i+1
N_m	Number of points considered when predicing cross section
$\mathbf{W}$	Weight matrix
$\mathbf{b}$	Biased vector
c_{in}	In channels
c_{out}	Out channels
n_u	Number of pixels per column of cross section
n_v	Number of pixels per row of cross section
n_h	Number of elements int height vector

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
1.1 Background . . . 1
1.2 Related work . . . 3
1.3 Research Question . . . 4
1.4 Research Objectives . . . 5
1.5 Limitations . . . 5

2 Theory **7**
2.1 Constrained continuous non-linear optimization problem . . . 7
2.2 Neural Networks . . . 7
2.2.1 Fully Connected Linear Layer . . . 8
2.2.2 Backpropagation . . . 8
2.2.3 Activation Function . . . 8
2.2.4 Batch Normalization . . . 9
2.2.5 Convolutional Neural Network (CNN) . . . 9
2.2.5.1 Residual block . . . 10
2.2.5.2 Squeeze and Excitation block . . . 10
2.3 IBOFlow Sealing Module . . . 10

3 Methodology **13**
3.1 Geometric Approach . . . 13
3.1.1 Problem description . . . 13
3.1.2 Derivation of Equations for Optimization . . . 15
3.1.2.1 Problem Setup . . . 15
3.1.2.2 Sealant Modeling . . . 16
3.1.2.3 Calculate TCP . . . 18
3.1.3 Optimization formulation . . . 19
3.1.3.1 Variable space . . . 20
3.1.3.2 Inequality constraints . . . 20

3.1.3.3 Equality constraints . . . 20
3.1.3.4 Bounds . . . 20
3.1.3.5 Initial guess . . . 21
3.1.3.6 Objective . . . 21
3.1.4 Evaluation metric . . . 21
3.2 Surrogate Model . . . 23
3.2.1 Problem description . . . 24
3.2.2 Data Collection . . . 24
3.2.3 Dataset . . . 25
3.2.4 Network architecture . . . 28
3.2.4.1 Linear Glass-Box Model . . . 28
3.2.4.2 MLP-CNN Network . . . 29
3.2.5 Training Process . . . 30
3.2.6 Evaluation Metric . . . 32

4 Results **33**
4.1 Geometric Approach . . . 33
4.1.1 Different widths . . . 34
4.1.2 Zic-Zac . . . 35
4.1.3 Large square . . . 36
4.1.4 Small square . . . 37
4.1.5 Plate circle . . . 38
4.1.6 Plate cube . . . 39
4.1.7 Ledge . . . 40
4.1.8 Stairs . . . 41
4.2 Surrogate Model . . . 42
4.2.1 Glass-box Model . . . 42
4.2.2 MLP-CNN Network . . . 44

5 Discussion **47**
5.1 Geometric Approach . . . 47
5.2 Surrogate model . . . 49
5.3 Research Question . . . 50
5.4 Ethical and Sustainability Aspects . . . 51

6 Conclusion **53**

List of Figures

1.1 A flowchart for generating a robot path to produce desired sealants. . 2
1.2 Simulation of the naive approach of using a constant offset, and velocity over the desired sealant center. 3

2.1 Convolutional filter sliding over arbitrary structured data with local dependencies. 9

3.1 Desired edge curves of the sealant from an above perspective. 14
3.2 Side view of the desired sealant curves, and robot nozzle with a constant volume flow over α. 14
3.3 Area created from bead over an infinitely small time dt and infinitely small angle $d\alpha$. 17
3.4 Circle introducing the drag $\boldsymbol{\kappa}$. 19
3.5 Comparison of the user defined sealant and the arc length parametrized sealant. 22
3.6 Test cases used to validate the geometric approach. 23
3.7 The objective for the surrogate model is to minimize the absolute cross section differnece between the surrogate model and the IPS simulation. 24
3.8 Distorted sealing center path. 25
3.9 Input and output parameters of the neural network. 26
3.10 Comparison of the initial and sampled TCP paths. 26
3.11 Description on obtaining height vector. 27
3.12 Illustration of obtaining a cross section. 28
3.13 Illustration of a dataset point. 28
3.14 Fully connected layer architecture 29
3.15 MLP CNN Hybrid Architecture. 29
3.16 The network architecture of the CNN block. 30
3.17 The network architecture of the Residual block. 30
3.18 The network architecture of the Squeeze-and-Excitation block. 31
3.19 Training pipeline used for the surrogate model. 31
3.20 Obstacle course created to validate the network. 32

4.1 Simulated sealant for a sealant changing the desired width over the path. 34
4.2 Parameters for a sealant changing the desired width over the path. . . 34
4.3 Simulated sealant for a zic zac path. 35
4.4 Parameter plots for a zic-zac path. 35

4.5 Simulated sealant for a square path. 36
4.6 Parameters for a square path. 36
4.7 Simulated sealant for a square, with short sides. 37
4.8 Parameters for a smaller square path. 37
4.9 Simulated sealant for a small seam created by a cylinder 38
4.10 Parameter plots for a small seam created by a cylinder. 38
4.11 Simulated sealant for a step created by a box on a plate. 39
4.12 Parameter plots for a step created by a box on a plate. 39
4.13 Comparison of sealant and TCP path depending on optimization of $\boldsymbol{\kappa}$. 40
4.14 Parameter plots for a ledge created by two plates. 40
4.15 Simulated sealant for steps created by a boxes on a plate. 41
4.16 Parameter plots for a stair geometry. 41
4.17 Training and validation loss for the glass box model for 500 epochs. . 42
4.18 Training evolution for the glass box model for 500 epochs. 43
4.19 MSE over validation paths. 44
4.20 Training and validation loss for the MLP-CNN hybrid model for 500 epochs. 44
4.21 Training evolution for the CNN model for 500 epochs. 45
4.22 MSE over validation paths for the validation set. 46

5.1 Limit in curve smoothing. 47
5.2 Effect of curve smoothening. 48
5.3 Curve smoothening effect on sealant width going over edges. 48
5.4 TCP location to prevent air bubbles. 49
5.5 Surrogate model predicts reasonable sealant for unexpected behavior in the simulation software. 50

List of Tables

3.1 Bounds and standard deviations for various physical parameters. . . . 25

4.1 Parameters used to validate the geometric approach. 33

5.1 Comparison of network performance. 49

1
Introduction

The thesis was conducted at the Fraunhofer-Chalmers Centre (FCC), a research institute dedicated to advancing various industrial applications, particularly focusing on modeling, simulation, and optimizing industrial processes. One of the main developments is a math-based software tool called Industrial Path Solutions (IPS) [1], which is widely used in the industry. IPS offers several modules, including cable simulation, path planner, robot optimization, virtual paint simulation, virtual sealing simulation, intelligently moving manikin, and IPS Immersed Boundary Octree Flow Solver (IBOFLOW). Currently, it is possible to plan collision-free robot paths within the software and simulate sealing beads for the planned robot path. However, the generated robot paths do not guarantee good sealing-bead results. The Master thesis focuses on connecting the IPS IBOFLOW sealing simulation module with the robot path planner to generate robot paths that produce desired sealing beads.

1.1 Background

The industry is striving to automate manufacturing processes. Some common objectives for automatic path planning are minimizing cycling time or ensuring surface coverage, which has been studied in previous work [2], [3], [4]. For application processes, the optimal robot paths differ depending on the situation and the type of applicator. An optimal path for spray paint differs from sealing processes. Research has been conducted on optimizing robot paths for spray paint applications, while there is limited research on optimal robot paths for sealing applications. Approximately 50 meters of sealant is applied to a car to prevent water leakage and reduce noise [2], making the application process an important aspect in increasing efficiency and precision. Optimizing the robot paths can decrease manufacturing time and material use by making the trajectories more executable and mathematically ensuring no extensive material use. It could also prevent mistakes in manually designing the robot paths.

Sealing spray processes include multi-phase and free surface flows and large moving geometries, which introduce challenging mathematics and modeling [2]. One of the leading solvers for sealing processes is the IPS IBOFlow solver, and complementing this, a robot path planner based on rapidly exploring random trees and probabilistic roadmap methods has been developed at FCC. Both the IBOFlow solver and robot path planner are implemented in IPS software. In [2], a framework was developed that produces collision-free robot paths and then uses the paths for sealing simu-

lations. The path planning framework creates a robot path by ensuring a minimal deviation from the original curve, with a minimum number of frames and proper frame orientation. All while avoiding collisions and kinematic issues. However, the task planner has a weak connection to generate desired sealants. The thesis addresses this issue by automatically designing initial curves that generate desired beads. The initial trajectory can be input to the IPS robot path planner to generate robot paths.

In the thesis, a simplified model is created based on a geometric approach projecting an equal volume flow along a fixed nozzle angle. The simplified model does not encapsulate all complex behaviors included in a sealing laydown process. On the other hand, the IBOFlow solver has proven to be accurate, but it takes time to perform the simulation. One meter of sealant is simulated in about an hour on a standard computer [2], which is not feasible for an optimization problem. To create a more accurate model while maintaining simulation speed, a surrogate model for the IBOFlow solver is developed. Figure 1.1 illustrates how the thesis fits in a larger scale project. The geometric approach objective is to calculate a trajectory that ensures equal sealing height distribution along the sealant edges. The surrogate model aims to mimic IPS IBOFlow as accurately as possible. To optimize the TCP path, the geometric approach could serve as an initial guess, and the surrogate model could be used to evaluate TCP paths. The blue stars correspond to the optimized TCP path.

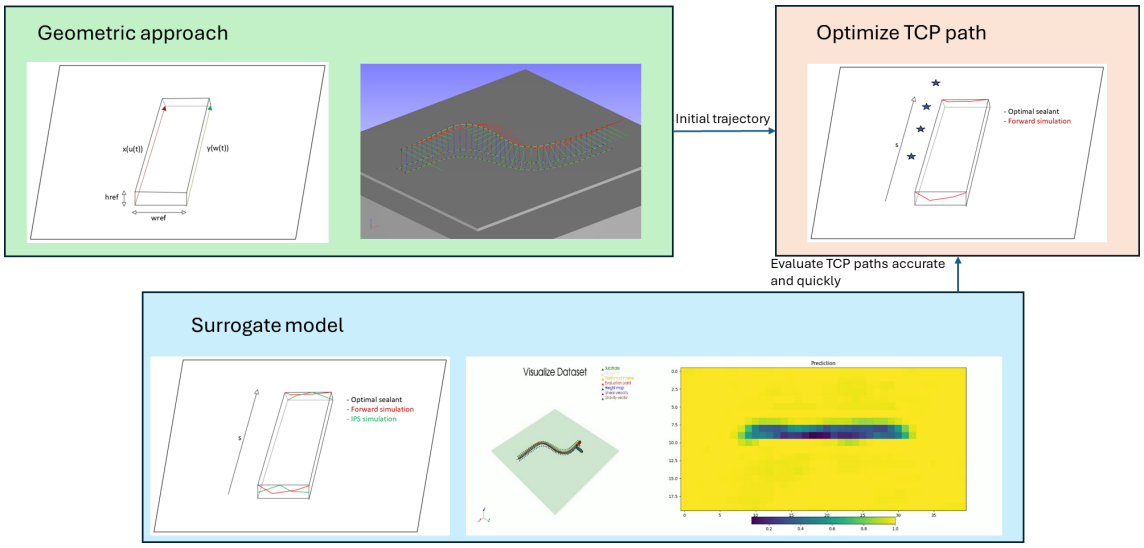
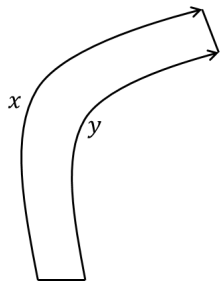
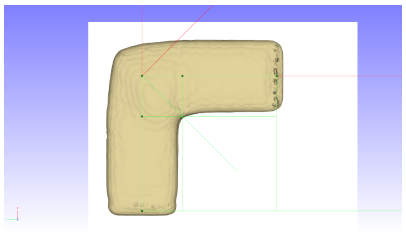


Figure 1.1: A flowchart for generating a robot path to produce desired sealants.

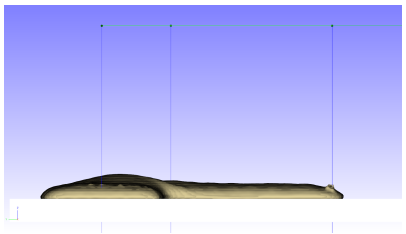
An intuitive initial solution is to move the robot tool path with a constant offset above the desired sealant center with a constant velocity. This strategy introduces unwanted behavior, especially around corners. Figure 1.2a demonstrates two curves where $\|\mathbf{x}\| > \|\mathbf{y}\|$, where $\mathbf{x}$ and $\mathbf{y}$ correspond to the desired sealant edges, and $\|\|$ denotes the length of a curve. Since $\mathbf{x}$ is longer than $\mathbf{y}$ means that more sealant needs to be applied along $\mathbf{x}$ to maintain an equal height distribution along the sealant edges. Figure 1.2b and Figure 1.2c show a sealant geometry for a corner using the naive approach of a constant offset and velocity.



(a) Curves defining the desired edges of a sealant.



(b) Top view of simulated sealant using the naive approach going over a right angle.



(c) Side view of simulated sealant using the naive approach going over a right angle.

Figure 1.2: Simulation of the naive approach of using a constant offset, and velocity over the desired sealant center.

1.2 Related work

In [3], the authors developed a method to generate a TCP path that minimizes spray paint on the target area to match a user-defined target thickness. The authors argue that it is computationally expensive to simulate the painting process, which is impractical in an optimization formulation. Therefore, they used a simplified model projecting a profile to a target geometry. Their research shows promising results on flat surfaces and a tractor fender, but several potential issues were observed, such as sensitivity to initial problem setup. Another observed limitation is that the projection-based modeling did not capture all physical features of the paint, such as the electrostatic effect. In [4], the authors developed a framework for automatic path generation for spray paint. The model requires a point cloud of the target geometry and spray painting parameters, including the spray gun model, paint flux, desired paint thickness, and allowed variation. The framework showed promising results for less complex geometries, but for more complex surfaces, there were problems containing a paint thickness within the allowed variation. In [5], the authors proposed conceptual approaches taking advantage of existing data sets to create tasks similar to the training sets to automatically generate robot paths.

For spray paint, it is simple to define what constitutes an optimal target geometry. Previous articles have discussed that the optimal laydown includes a target

surface geometry and a desired thickness [3], [4]. It is more challenging to define what constitutes an optimal sealing bead. In [6], the authors optimize the cross-section of automotive door sealing to improve door-closing comfort, while having sufficient sealing qualities. However, research is far from designing the optimal sealing geometry. Improving the sealing process usually depends on experts knowledge and experimental testing, given the complexity of the sealing process [2].

An Riemannian motion policy (RMP) is a second-order dynamical system that, for instance, can input position and velocity and calculate acceleration [7]. In [7], a method is proposed that combines several motion policies from one subspace to another. The term "motion generation" was used to describe RMP, which is suggested as an umbrella term for motion planning and reactive motion. By defining different motion policies, the authors managed to navigate through cluttered environments. In [8], RMPs were used for reactive path planning for robotic silicone sealing. The authors used a two-camera reactive path planning framework and edge detection to detect the sealing seam. Thereafter, they used a neural network to predict RMPs. The authors calculated the control commands by combining the RMPs with the forward kinematics of the robot system.

A force vision based end effector was developed in [9] to ensure the quality of the sealing bead during the laydown. The system used computer vision to approximate the width of the sealant and showed some interesting behaviors for undesired sealants. However, it was only tested on a flat surface with a straight sealing bead. In [10], an RGB-D sensor was employed to automate the sealant dispensing process. The RGB-D sensor is used for part recognition and pose estimation of an object, thereafter RobotStudio [11] was then used to automatically generate a robot path. A literature survey on coverage path planning was performed in [12]. The literature survey covered several approaches ranging from simple methods, such as classical cellular decomposition methods, to more complex ones, including multi-robot methods.

In [13], a convolutional neural network was used to predict optimized collision-free paths for a drone. The input of the network was the drone's Inertial Measurement Units and a depth image. The Inertial Measurement Unit and depth image were sent through two separate backbones, then fused, and put through a prediction head including a convolutional network. The training process included a privileged expert having full knowledge of the surroundings and calculated trajectories at each timestep using Metropolis-Hastings [14] to sample points and represent the trajectory using cubic B-spline with three control points. The different curves get a score based on the distance to an obstacle and deviation from the reference trajectory. The three trajectories with the lowest score were used to train the network.

1.3 Research Question

The thesis aim is to connect the IPS IBOFlow solver with the IPS robot path planning module. The thesis explores how AI can be used to build a surrogate model for sealing simulation. It also explores how a geometric approach can be used to

calculate an initial path.

The research questions investigated in the thesis are:

- How do the simulated sealants using a TCP path calculated by a geometric approach compare to the desired sealant?
- How do sealing simulation predictions made by the surrogate model compare to simulations performed in IPS IBOFlow?
- How do simple neural networks compare to deep neural networks for sealing simulations?

1.4 Research Objectives

The research objectives are divided into main objectives and secondary objectives:

Main objectives

- **Calculate the optimal TCP path using a geometric approach:** Develop a simplified model and use it to calculate an initial optimal path, ignoring more complex sealant behaviors.
- **Develop a surrogate model for sealing simulations:** Develop and train a neural network to predict sealing geometries based on different TCP paths and substrates.

Secondary objectives

- **Validate the TCP paths achieved by using the geometric approach:** Analyze the generated TCP path by performing IPS IBOFlows simulations and analyze the generated bead.
- **Data collection:** Create a diverse dataset by performing IBOFlow simulations with different TCP paths and substrate geometries that is used to train the surrogate model.
- **Post processing of sealing simulation:** The sealing simulation needs to be post-processed to access the data used in the surrogate model.
- **Validate surrogate model:** Compare the surrogate model predictions with the results from the IBOFlow simulations.
- **Analysis of network architecture:** Develop and train simplified models to validate if the complexity of the deep neural network increases performance compared to simple interpretable models.

1.5 Limitations

The thesis assumes that the IBOFlow solver is correct and does not address any limitations the solver has. The surrogate model does not account for varying fluid properties, such as viscosity, different robot nozzles, or variations in flow rate. A single set of properties is used throughout the work. The problem of generating optimal collision-free robot paths is considered a well-studied area and will be out of scope within this thesis.

2
Theory

The section begins with a brief overview of nonlinear optimization, followed by an explanation of neural networks and an introduction to the IBOFlow simulation framework.

2.1 Constrained continuous non-linear optimization problem

A constrained continuous nonlinear optimization problem is set up in Section 3.1. The thesis focuses on setting up the problem formulation, and the details behind how the solver works are beyond the scope of this thesis. However, a brief understanding is relevant. The general form of a nonlinear optimization problem is stated in Equation 2.1.

$$\min_{x \in X} \; J(x), \tag{2.1}$$

$$\text{such that } g(x) \geq 0, \tag{2.2}$$

$$h(x) = 0. \tag{2.3}$$

J is a cost function minimized in the variable space X that satisfies the inequality constraint g(x) and the equality constraints h(x).

Due to the complexity of the problem, a numerical solver is typically used. Many numerical solvers are based on Newton's method [15] but utilize more complex concepts such as Karush-Kuhn-Tucker conditions [16] to handle the constraints. The numerical solvers do not guarantee a global minimum, and different solutions can be reached depending on the initial guess. However, the solution ensures that the constraints are satisfied and that no small movements will improve the cost function.

2.2 Neural Networks

Neural networks perform well at learning complex patterns from a large amount of data. The neural network consists of neurons that are interconnected in layers. Linear transformations are performed on the input of the neurons using learned weights and biases and are usually passed through an activation function to introduce nonlinearities. To train the model, the neural network predicts an output that

is compared to the ground truth, and the internal weights are updated by minimizing the loss. Each internal weight contribution to the loss can be calculated through backpropagation. Thereafter, some optimizer, often utilizing gradient descent, is used to update the internal weights.

2.2.1 Fully Connected Linear Layer

A fully connected layer is a foundational component of neural networks. Every neuron in the previous layer is connected to each neuron in the next layer, and every connection has an associated weight, and every neuron has a bias. The output $\mathbf{y}$ from the linear connected layer can be expressed as described in Equation 2.4.

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{2.4}$$

$\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, and $\mathbf{x}$ is the input vector. Fully connected layers are often used at the end of the network to combine learned features [17]. The fully connected layer can be inefficient for high-dimensional data such as images since it requires a large number of parameters, which led to the development of more complex models such as CNNs [18].

2.2.2 Backpropagation

Backpropagation enables the network to learn from data and is used to compute how each weight and bias contributes to the loss, where the loss describes the error between the prediction and the ground truth [19]. A prediction is calculated through a forward pass. A backward pass is applied to calculate the gradient of the loss with respect to each parameter. The gradients are computed layer by layer, starting from the output and moving toward the input.

A common problem when training neural networks is vanishing and exploding gradients [20]. Vanishing gradients describe the process where gradients shrink during backpropagation, which makes the early layers of the network learn slowly. Gradients can also become exponentially large, leading to unstable weight updates, which are referred to as exploding gradients [21]. Research has been performed to avoid unstable gradients, such as batch normalization [22], residual connections [23], and Rectified Linear Unit activations [24].

2.2.3 Activation Function

Activation functions are incorporated in neural networks to introduce nonlinearity into the model. An activation function is applied to each neuron to decide whether the neuron is activated. The most common activation function used is Rectified Linear Unit (ReLU) due to its simplicity and ability to avoid vanishing gradient [24]. To avoid inactive neurons, Leaky ReLU [25] was introduced, which is described in Equation 2.5, where leaky refers to the small leak of values when the input is negative defined by α and x is the input to the activation function.

$$\text{Leaky ReLU:} \quad f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \tag{2.5}$$

Vanishing gradients are avoided by ensuring the gradients to be 1 [26] for positive inputs and α for negative as described in Equation 2.6.

$$\text{Derivative:} \quad f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ \alpha & \text{if } x \leq 0 \end{cases} \tag{2.6}$$

2.2.4 Batch Normalization

A deeper neural network often suffers from internal covariance shift where the distribution of activations changes during training due to parameters of previous layers being updated. Equation 2.7 that is introduced in [22] normalizes the output for each layer during the training process.

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad y^{(i)} = \gamma \hat{x}^{(i)} + \beta \tag{2.7}$$

Where μ_b is the batch mean and σ_b is the batch covariance, ϵ is a small constant to avoid division by zero, γ and β are learned parameter weights.

2.2.5 Convolutional Neural Network (CNN)

Convolutional networks are a specialized class of neural networks that process structured data with local dependencies. The convolutional layer applies filters with trainable parameters over an input by sliding a window with a stride, which enables the network to learn local features [18]. Figure 2.1 illustrates a convolutional filter sliding over an input with a stride of 1 and kernel size of (3,3).

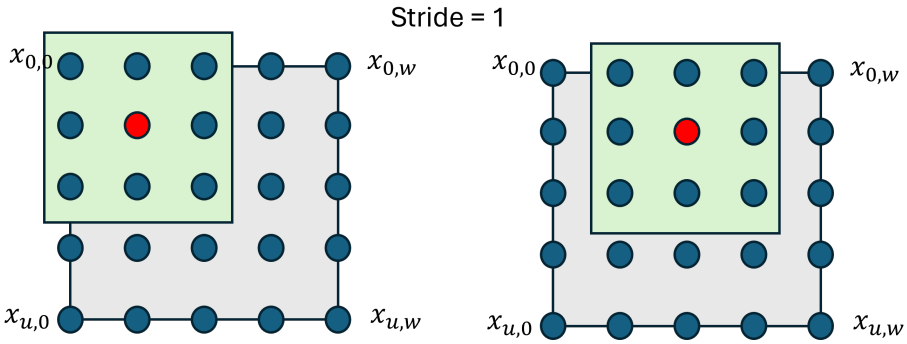


Figure 2.1: Convolutional filter sliding over arbitrary structured data with local dependencies.

Equation 2.8 is used to calculate a single output pixel where (N,M) is the kernel size.

$$y_{i,j} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{i+m,j+n} \cdot w_{m,n} \tag{2.8}$$

2.2.5.1 Residual block

Residual nets were introduced in [23] and addressed the vanishing gradient problem in deep convolution neural networks by skipping connections. The residual block is described in Equation 2.9.

$$\mathbf{y} = \mathbf{F}(\mathbf{x}) + \mathbf{x} \tag{2.9}$$

Where $\mathbf{F}(\mathbf{x})$ usually consists of convolutional layers, followed by batch normalization and ReLU layer. Skipping connections allow improved gradient flow during backpropagation.

2.2.5.2 Squeeze and Excitation block

Convolutional neural networks focus on extracting local features, and all channels are treated equally. However, the features might be of varying importance. Squeeze-and-Excitation [27] address this limitation by assigning different weights to each channel. Firstly, the channels are squeezed by applying global average pooling to each channel. Thereafter, the pooled features are passed through two fully connected linear layers to generate a set of channels-wise weights. Lastly, element-wise multiplication is performed between the feature maps and channel-wise weights to emphasize important features and suppress the irrelevant ones.

2.3 IBOFlow Sealing Module

The IPS IBOFlow solver is used both to validate the geometric approach presented in Section 3.1 and to generate data to the surrogate model presented in Section 3.2. Further theory about the IPS solver is described in [2], but a short summation is presented below.

IPS can be used to perform accurate sealing simulations for various robot paths, sealing materials, target geometries, and robot nozzles by utilizing finite volume discretizations [28] on a cartesian octree grid that can be dynamically refined. The volume of fluids [29] is used to track the fluid interface between air and sealant. To avoid having to create meshes for complex geometries, IPS uses an immersed boundary method discussed in [30]. The immersed boundary method was first introduced in [31] and is a method that modifies the fluid equations close to the solid without having to modify the mesh.

The Navier Stokes Equation [32] is used to model the motion of an incompressible fluid and is presented in Equations 2.10, 2.11. ρ is the fluid density, p is the pressure, μ is the viscosity, $\mathbf{f}$ is the external force, and $\mathbf{u}$ is the fluid velocity. The viscosity behaves differently under stress. As described in [33], a material can be modeled as a Bingham fluid, where the fluid behaves like a solid at low shear stresses and as a fluid at higher stresses.

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} \tag{2.10}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.11}$$

3
Methodology

In Section 3.1, an optimal TCP path calculated based on a geometric approach is introduced. The simplified model does not encapsulate all complex sealant behavior while running a full sealing simulation is time-consuming and is not feasible for optimization problems. Therefore, a surrogate model was developed to capture more complex sealant behaviors while maintaining quick validation, as described further in Section 3.2.

3.1 Geometric Approach

The following section covers how the initial TCP path is adjusted based on a simplified modeling of the sealant. First, the problem is described in Section 3.1.1, thereafter a derivation of the equations used in the optimization formulation is explained in Section 3.1.2, then the optimization formulation is introduced in Section 3.1.3. Lastly, the evaluation metric is introduced in Section 3.1.4.

3.1.1 Problem description

Given two curves $\mathbf{x}$ and $\mathbf{y}$ describing the desired sealant edges and a nozzle with constant volume flow over the nozzle angle α. The robot path has freedom in its offset to the surface, velocity profile, tilt, drag, and spin. The aim is to calculate a Tool Center Point (TCP) path that uses a geometric approach to achieve equal sealant height distribution along $\mathbf{x}$ and $\mathbf{y}$, ignoring what is happening between the sealant edges.

The sealant is illustrated from an above perspective in Figure 3.1. Where $\mathbf{x}$, and $\mathbf{y}$ are curves describing the desired sealant edges. $\mathbf{x_0}$ and $\mathbf{y_0}$ are the start points, while $\mathbf{x_N}$ and $\mathbf{y_N}$ are the end points. The system is discretized in N time steps from t_0 to t_N. The sealant edge locations where the spray should intersect the desired sealant edges at timestep t_i are denoted $\mathbf{x_i}$, $\mathbf{y_i}$, while $\mathbf{x_i'}$, $\mathbf{y_i'}$ are the sealant edge directions. η_i is the angle between $\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ and $\mathbf{x_i'}$, and ζ_i is the angle between $\mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i}$ and $\mathbf{y_i'}$. The distance between $\mathbf{x_i}$ and $\mathbf{y_i}$ is labeled a_i.

The TCP point $\mathbf{p_i}$ together with the points $\mathbf{x_i}$ and $\mathbf{y_i}$ at time t_i is illustrated from a side perspective in Figure 3.2. c_i denotes the distance between $\mathbf{p_i}$ and $\mathbf{x_i}$, while b_i denotes distance between $\mathbf{p_i}$ and $\mathbf{y_i}$. β_i is defined as the angle between $\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ and $\mathbf{x_i}\vec{\mathbf{p}}_\mathbf{i}$, and γ_i is defined as the angle between $\mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i}$ and $\mathbf{y_i}\vec{\mathbf{p}}_\mathbf{i}$. d_i is the distance between

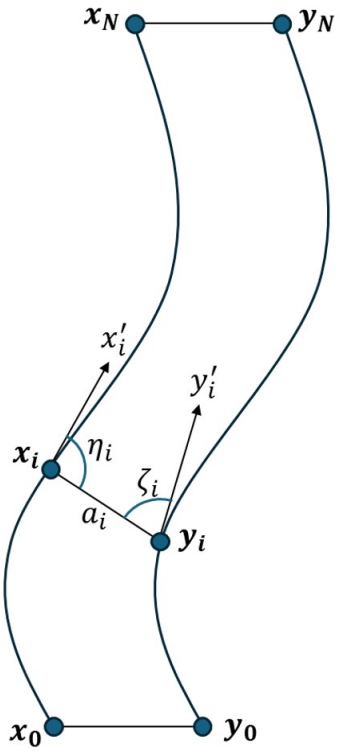


Figure 3.1: Desired edge curves of the sealant from an above perspective.

$\mathbf{p_i}$ and the intersection point between $\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ and the bisection of α, ρ_i is the tilt and is defined as the angle between $\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ and bisection of α.

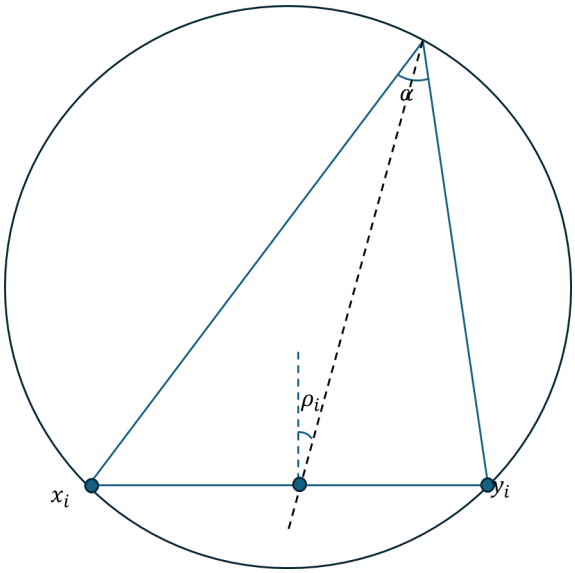


Figure 3.2: Side view of the desired sealant curves, and robot nozzle with a constant volume flow over α.

3.1.2 Derivation of Equations for Optimization

This section derives the equations that are used in the optimization formulation. The problem setup is explained in Section 3.1.2.1, the sealant modeling is described in Section 3.1.2.2, and the calculation of the TCP path is presented in Section 3.1.2.3.

3.1.2.1 Problem Setup

Firstly, $\mathbf{x}$ and $\mathbf{y}$ are arc length parameterized as described in Equations 3.1, 3.2.

$$\mathbf{x} = \mathbf{x}(s), \quad |\mathbf{x}'(s)| = 1, \quad \mathbf{x}(0) = \mathbf{x_0}, \quad \mathbf{x}(L_x) = \mathbf{x_N} \tag{3.1}$$
$$\mathbf{y} = \mathbf{y}(s), \quad |\mathbf{y}'(s)| = 1, \quad \mathbf{y}(0) = \mathbf{y_0}, \quad \mathbf{y}(L_y) = \mathbf{y_N} \tag{3.2}$$

Let $u(t)$ and $w(t)$ be the arc length at time t, where $u(t) \in [0, L_x]$ and $w(t) \in [0, L_y]$. Where $\mathbf{x}(u(t))$ and $\mathbf{y}(w(t))$ are described in Equations 3.3, 3.4.

$$\mathbf{x} = \mathbf{x}(u(t)), \quad u(0) = 0, \quad u(T) = L_x \tag{3.3}$$
$$\mathbf{y} = \mathbf{y}(w(t)), \quad w(0) = 0, \quad w(T) = L_y \tag{3.4}$$

The system is discretized into N time steps as described in Equations 3.5, 3.6, 3.7.

$$0 = t_0 < t_1 < \cdots < t_N = T \tag{3.5}$$
$$\mathbf{x_i} = \mathbf{x}(u(t_i)), \quad u_i = u(t_i), \quad u_0 = 0, \quad u_N = L_x \tag{3.6}$$
$$\mathbf{y_i} = \mathbf{y}(w(t_i)), \quad w_i = w(t_i), \quad w_0 = 0, \quad w_N = L_y \tag{3.7}$$

Given $\dot{u}_0 \cdots \dot{u}_N$ and $\dot{w}_0 \cdots \dot{w}_N$ the derivatives of $\mathbf{x}(u(t))$ and $\mathbf{y}(w(t))$ are calculated in Equations 3.8, 3.9.

$$\dot{\mathbf{x}}_\mathbf{i}(u(t_i)) = \mathbf{x}'_\mathbf{i} \cdot \dot{u}_i \tag{3.8}$$
$$\dot{\mathbf{y}}_\mathbf{i}(w(t_i)) = \mathbf{y}'_\mathbf{i} \cdot \dot{w}_i \tag{3.9}$$

Assuming continuous piecewise linear velocity u_i and w_i can be calculated as described in Equations 3.10, 3.11.

$$u_i = u_{i-1} + \frac{\dot{u}_{i-1} + \dot{u}_i}{2}(t_i - t_{i-1}), \quad u_0 = 0 \tag{3.10}$$
$$w_i = w_{i-1} + \frac{\dot{w}_{i-1} + \dot{w}_i}{2}(t_i - t_{i-1}), \quad w_0 = 0 \tag{3.11}$$

The arc lengths $\mathbf{u} = \begin{bmatrix} u_0 & \cdots & u_N \end{bmatrix}^T$ and $\mathbf{w} = \begin{bmatrix} w_0 & \cdots & w_N \end{bmatrix}^T$ can be calculated as described in Equations 3.12, 3.13, where $\dot{\mathbf{u}} = \begin{bmatrix} \dot{u}_0 & \cdots & \dot{u}_N \end{bmatrix}^T$ and $\dot{\mathbf{w}} =$

$\begin{bmatrix} \dot{w}_0 & \cdots & \dot{w}_N \end{bmatrix}^T$.

$$\mathbf{u} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \frac{t_1-t_0}{2} & \frac{t_1-t_0}{2} & 0 & \cdots & \vdots \\ \vdots & \frac{t_2-t_0}{2} & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \frac{t_1-t_0}{2} & \frac{t_2-t_0}{2} & \cdots & \frac{t_N-t_{N-2}}{2} & \frac{t_N-t_{N-1}}{2} \end{bmatrix} \dot{\mathbf{u}} \tag{3.12}$$

$$\mathbf{w} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \frac{t_1-t_0}{2} & \frac{t_1-t_0}{2} & 0 & \cdots & \vdots \\ \vdots & \frac{t_2-t_0}{2} & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \frac{t_1-t_0}{2} & \frac{t_2-t_0}{2} & \cdots & \frac{t_N-t_{N-2}}{2} & \frac{t_N-t_{N-1}}{2} \end{bmatrix} \dot{\mathbf{w}} \tag{3.13}$$

For any $\dot{u}_i$ and $\dot{w}_i$, we have u_i and w_i, thus for every $i = 1,\dots, N$ we can compute $\mathbf{x_i}$, $\mathbf{y_i}$, a_i, η_i, and ζ_i. Where a_i is calculated in Equation 3.14.

$$a_i = \|\mathbf{x_i} - \mathbf{y_i}\| \tag{3.14}$$

While η_i and ζ_i can be calculated from the definition of the cross product as described in Equations 3.15, 3.16.

$$sin(\eta_i) = \frac{\|\mathbf{x_i'} \times \mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}\|}{\|\mathbf{x_i'}\|\|\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}\|} \tag{3.15}$$

$$sin(\zeta_i) = \frac{\|\mathbf{y_i'} \times \mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i}\|}{\|\mathbf{y_i'}\|\|\mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i}\|} \tag{3.16}$$

$\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ and $\mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i}$ are calculated in Equations 3.17 3.18.

$$\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i} = \mathbf{y_i} - \mathbf{x_i} \tag{3.17}$$

$$\mathbf{y_i}\vec{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} - \mathbf{y_i} \tag{3.18}$$

$\therefore$ Given $\dot{\mathbf{u}}$, $\dot{\mathbf{w}}$, then $\mathbf{u}$, $\mathbf{w}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$, $\mathbf{a}$ can be calculated.

3.1.2.2 Sealant Modeling

Assuming equal volume flow across α, the volume for a small angle $d\alpha$ can be calculated as described in Equation 3.19.

$$V = \phi \cdot \frac{d\alpha}{\alpha} \cdot dt \tag{3.19}$$

For an infinitesimally small angle $d\alpha$ and infinitesimally short time dt, the corresponding area around $\mathbf{x_i}$ is illustrated in Figure 3.3.

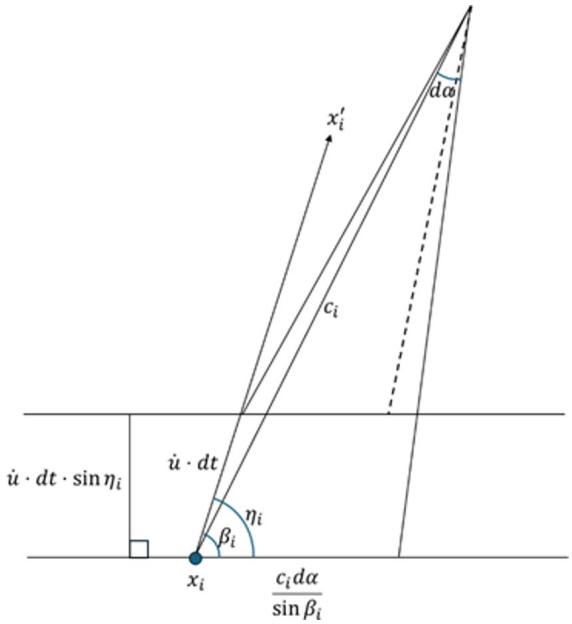


Figure 3.3: Area created from bead over an infinitely small time dt and infinitely small angle $d\alpha$.

The area corresponding to the points on both sides of the sealant is calculated in Equations 3.20, 3.21.

$$A_{xi} = \frac{\dot{u}_i \cdot dt \cdot sin(\eta_i) \cdot c_i \cdot d\alpha}{sin(\beta_i)} \tag{3.20}$$

$$A_{yi} = \frac{\dot{w}_i \cdot dt \cdot sin(\zeta_i) \cdot b_i \cdot d\alpha}{sin(\gamma_i)} \tag{3.21}$$

The heights at point $\mathbf{x_i}$ and $\mathbf{y_i}$ is calculated in Equations 3.22, 3.23.

$$g_i = \frac{V}{A_{xi}} = \frac{\phi \cdot sin(\beta_i)}{\alpha \cdot c_i \cdot \dot{u}_i \cdot sin(\eta_i)} \tag{3.22}$$

$$h_i = \frac{V}{A_{yi}} = \frac{\phi \cdot sin(\gamma_i)}{\alpha \cdot b_i \cdot \dot{w}_i \cdot sin(\zeta_i)} \tag{3.23}$$

Considering the triangle in Figure 3.2, the law of sines theorem results in Equation 3.24.

$$\frac{sin(\alpha)}{a_i} = \frac{sin(\beta_i)}{b_i} = \frac{sin(\gamma_i)}{c_i} \Rightarrow \frac{sin(\beta_i)}{sin(\gamma_i)} = \frac{b_i}{c_i} \tag{3.24}$$

Setting the heights g_i and h_i equal is shown in Equation 3.25.

$$g_i = h_i \Leftrightarrow \frac{sin(\beta_i)}{c_i \cdot \dot{u}_i \cdot sin(\eta_i)} = \frac{sin(\gamma_i)}{b_i \cdot \dot{w}_i \cdot sin(\zeta_i)} \tag{3.25}$$

Rearranging Equation 3.25, results in Equation 3.26.

$$\frac{b_i}{c_i} \cdot \frac{sin(\beta_i)}{sin(\gamma_i)} = \frac{\dot{u}_i \cdot sin(\eta_i)}{\dot{w}_i \cdot sin(\zeta_i)} \tag{3.26}$$

Using Equation 3.24, the left side of Equation 3.26 can be written as Equation 3.27.

$$\frac{b_i}{c_i} \cdot \frac{sin(\beta_i)}{sin(\gamma_i)} = \frac{b_i^2}{c_i^2} = \mu_i^2 \tag{3.27}$$

Combining Equation 3.26 and Equation 3.27 results in Equation 3.28. Since η_i and ζ_i are known from $\dot{u}_i$ and $\dot{w}_i$ as described in Equations 3.15, 3.16, μ_i is only dependent on $\dot{u}_i$ and $\dot{w}_i$.

$$\mu_i^2 = \frac{\dot{u}_i \cdot sin(\eta_i)}{\dot{w}_i \cdot sin(\zeta_i)} \tag{3.28}$$

From Equation 3.27, b_i can be expressed from c_i and μ_i as described in Equation 3.29.

$$b_i = \mu_i \cdot c_i \tag{3.29}$$

Using Equation 3.29 and the law of cosines in the triangle illustrated in Figure 3.2, it is possible to describe c_i from μ_i and a_i, as described in Equations 3.30, 3.31.

$$a_i^2 = b_i^2 + c_i^2 - 2 \cdot b_i \cdot c_i \cdot cos(\alpha) \tag{3.30}$$

$$c_i = \sqrt{\frac{a_i^2}{\mu_i^2 - 2 \cdot \mu_i \cdot cos(\alpha) + 1}} \tag{3.31}$$

By using b_i and c_i, and Equation 3.24. β_i and γ_i can be calculated as described in Equations 3.32, 3.33.

$$\beta_i = sin^{-1}\left(\frac{b_i \cdot sin(\alpha)}{a_i}\right) \tag{3.32}$$

$$\gamma_i = sin^{-1}\left(\frac{c_i \cdot sin(\alpha)}{a_i}\right) \tag{3.33}$$

From γ_i, ρ_i can be calculated as described in Equation 3.34

$$\rho_i = \frac{\pi}{2} - (\pi - \frac{\alpha}{2} - \gamma_i) = \frac{\alpha}{2} + \gamma_i - \frac{\pi}{2} \tag{3.34}$$

$\therefore$ Given $\dot{\mathbf{u}}$ and $\dot{\mathbf{w}}$, then $\mathbf{u}$, $\mathbf{w}$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{\rho}$, $\boldsymbol{\eta}$, $\boldsymbol{\zeta}$, $\mathbf{g}$, $\mathbf{h}$ can be calculated.

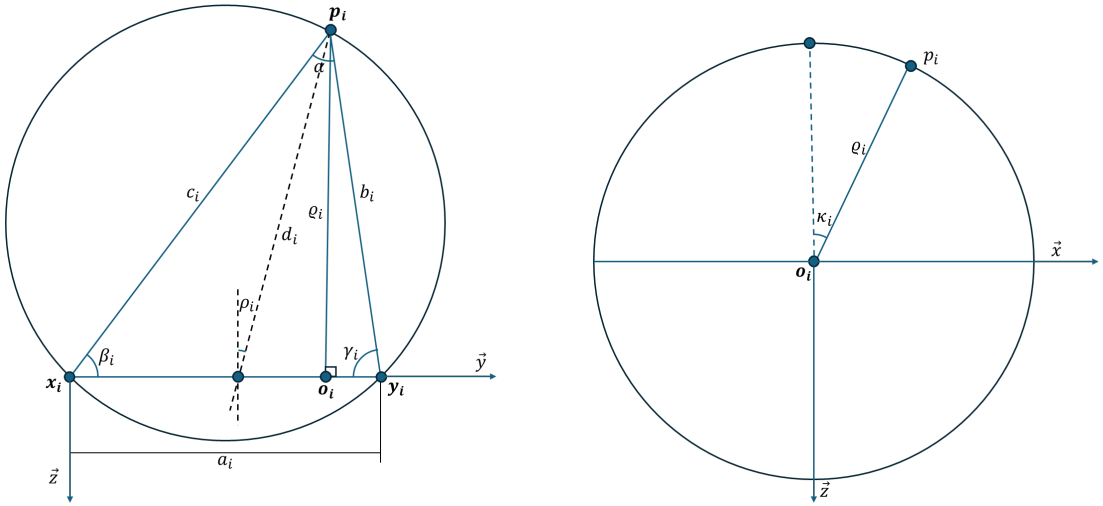
3.1.2.3 Calculate TCP

Considering Figure 3.4a, the point on the circle $\mathbf{p_i}$ can be calculated based on $\mathbf{x_i}$, β_i, ρ_i and c_i. Thereafter, consider the sealants xz-plane illustrated in Figure 3.4b where another circle centered in $\mathbf{o_i}$ with the radius ϱ_i. Where ϱ_i is the perpendicular distance between $\mathbf{p_i}$ and $\mathbf{x_i}\vec{\mathbf{y}}_\mathbf{i}$ in the yz-plane. The drag angle is denoted $\boldsymbol{\kappa_i}$.
The TCP point $\mathbf{p_i}$ is calculated in Equation 3.35.

$$\mathbf{p_i} = \mathbf{x_i} + cos(\frac{\pi}{2} - \kappa_i)\varrho_i\vec{\mathbf{x}}_\mathbf{i} + cos(\beta_i)c_i\vec{\mathbf{y}}_\mathbf{i} - sin(\frac{\pi}{2} - \kappa_i)\varrho_i\vec{\mathbf{z}}_\mathbf{i} \tag{3.35}$$

Where the sealant frame can be calculated by Equations 3.36, 3.38, 3.37.

$$\vec{\mathbf{y}}_\mathbf{i} = \frac{\mathbf{y_i} - \mathbf{x_i}}{\|\mathbf{y_i} - \mathbf{x_i}\|} \tag{3.36}$$



(a) Side view of TCP, in the sealants yz-plane.

(b) Side view of TCP, in the sealants xz-plane.

Figure 3.4: Circle introducing the drag $\boldsymbol{\kappa}$.

The vector $\vec{\mathbf{z}_\mathbf{i}}$ of the sealant is calculated in Equation 3.37.

$$\vec{\mathbf{z}_\mathbf{i}} = \frac{(\mathbf{x}'_\mathbf{i} + \mathbf{y}'_\mathbf{i}) \times \vec{\mathbf{y}_\mathbf{i}}}{\|(\mathbf{x}'_\mathbf{i} + \mathbf{y}'_\mathbf{i}) \times \vec{\mathbf{y}_\mathbf{i}}\|} \tag{3.37}$$

The direction of the bead is calculated in Equation 3.38.

$$\vec{\mathbf{x}_\mathbf{i}} = \frac{\vec{\mathbf{y}_\mathbf{i}} \times \vec{\mathbf{z}_\mathbf{i}}}{\|\vec{\mathbf{y}_\mathbf{i}} \times \vec{\mathbf{z}_\mathbf{i}}\|} \tag{3.38}$$

The rotation matrix of the sealant frame can be defined as described in Equation 3.39.

$$\mathbf{R_{si}} = \begin{pmatrix} \vec{\mathbf{x}_\mathbf{i}} \\ \vec{\mathbf{y}_\mathbf{i}} \\ \vec{\mathbf{z}_\mathbf{i}} \end{pmatrix}^T \tag{3.39}$$

The TCP rotation matrix is calculated in Equation 3.40

$$\mathbf{R_i} = \mathbf{R}(-\rho_i, \vec{\mathbf{x}_\mathbf{i}})\mathbf{R}(\kappa_i, \vec{\mathbf{y}_\mathbf{i}})\mathbf{R_{si}} \tag{3.40}$$

$\therefore$ Given $\dot{\mathbf{u}}$, $\dot{\mathbf{w}}$ and $\boldsymbol{\kappa}$, the TCP path $\mathbf{p}$ and its rotation matrices $\mathbf{R}$ are determined.

3.1.3 Optimization formulation

The problem can be formulated as a constrained continuous non-linear optimization problem. The general form of the optimization problem is stated in Equation 3.41.

$$\min_{x \in X} \; J(x), \tag{3.41}$$

$$\text{such that } g(x) \geq 0, \tag{3.42}$$

$$h(x) = 0. \tag{3.43}$$

Where J is a cost function minimized in the variable space X that satisfies the inequality constraints g(x) and the equality constraints h(x). `Scipy` [34] optimization package with the Sequential Least Squares Programming (SLSQP) is used as the solver.

3.1.3.1 Variable space

As described in Section 3.1.2, the TCP points $\mathbf{p}$ and orientations $\mathbf{R}$ depends on $\dot{\mathbf{u}}$, $\dot{\mathbf{w}}$, and $\boldsymbol{\kappa}$. Equations 3.44, 3.45 show the fully indexed notation where N is the number of waypoints and spans the free variable space.

$$\dot{\mathbf{u}} = \begin{bmatrix} \dot{u_0} & \cdots & \dot{u_N} \end{bmatrix} \tag{3.44}$$

$$\dot{\mathbf{w}} = \begin{bmatrix} \dot{w_0} & \cdots & \dot{w_N} \end{bmatrix} \tag{3.45}$$

$$\boldsymbol{\kappa} = \begin{bmatrix} \kappa_0 & \cdots & \kappa_N \end{bmatrix} \tag{3.46}$$

3.1.3.2 Inequality constraints

The inequality constraints described in Equations 3.47, 3.48, 3.49 specify the distance and tilt constraints.

$$b_{min} \leq \mathbf{b} \leq b_{max} \tag{3.47}$$

$$c_{min} \leq \mathbf{c} \leq c_{max} \tag{3.48}$$

$$\rho_{min} \leq \boldsymbol{\rho} \leq \rho_{max} \tag{3.49}$$

3.1.3.3 Equality constraints

The equality constraints are described in Equations 3.50, 3.51, 3.52, 3.53, 3.54. Where the height is supposed to be equal along the sealant, end at the endpoints, and have an initial and end drag of zero. The constraint on κ_0 and κ_N was introduced considering the objective of minimizing the TCP trajectory distance. Without the constraint, most trajectories start and end close to their bounds to minimize the distance, which is an undesired behavior.

$$g_i = g_{i-1} \tag{3.50}$$

$$u_N = L_x \tag{3.51}$$

$$w_N = L_y \tag{3.52}$$

$$\kappa_0 = 0 \tag{3.53}$$

$$\kappa_N = 0 \tag{3.54}$$

3.1.3.4 Bounds

Drag and velocities outside certain intervals result in undesired sealants. Therefore, bounds on the free variables were introduced. The bounds is described in Equations 3.55, 3.56, 3.57.

$$v_{min} \leq \dot{\mathbf{u}} \leq v_{max} \tag{3.55}$$
$$v_{min} \leq \dot{\mathbf{w}} \leq v_{max} \tag{3.56}$$
$$\kappa_{min} \leq \boldsymbol{\kappa} \leq \kappa_{max} \tag{3.57}$$

3.1.3.5 Initial guess

The initial guess for the optimization problem is specified as the average velocity along the curves and initial drag of zero, which is shown in Equations 3.58, 3.59, 3.60.

$$\dot{\mathbf{u}}_\mathbf{0} = \frac{L_x}{t_f} \tag{3.58}$$
$$\dot{\mathbf{w}}_\mathbf{0} = \frac{L_y}{t_f} \tag{3.59}$$
$$\boldsymbol{\kappa}_\mathbf{0} = \mathbf{0} \tag{3.60}$$

The final time t_f is a user input and serves as a scale of the sealant height. A shorter final time results in a smaller sealant height.

3.1.3.6 Objective

The cost function is introduced in Equation 3.61, where k_1, k_2, and k_3 are chosen weight constants.

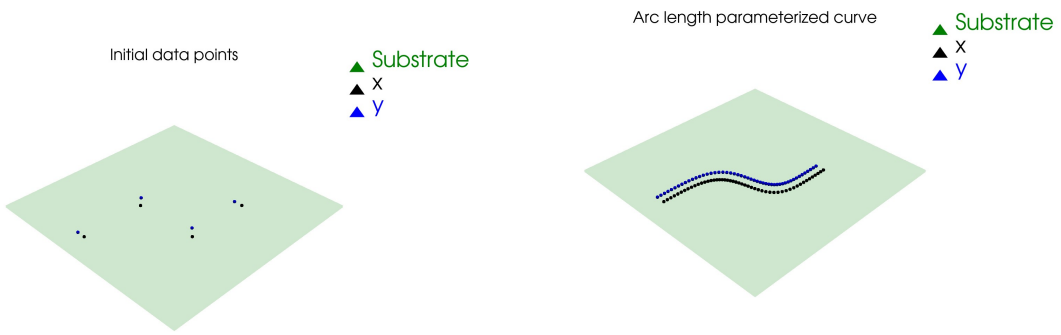
$$J = k_1(|sin(\eta_i) - 1| + |sin(\zeta_i) - 1|) + k_2|\kappa_i| + k_3\|\mathbf{p_i} - \mathbf{p_{i-1}}\| \tag{3.61}$$

The term $|\sin(\eta_i) - 1| + |\sin(\zeta_i) - 1|$ minimizes how much the TCP direction to diverge from x' and y', and the $\|\mathbf{p_i} - \mathbf{p_{i-1}}\|$ minimizes the robot nozzles distance and ensures smooth TCP movements. Lastly, the $|\kappa_i|$ term maintains the drag to oscillate around 0.

3.1.4 Evaluation metric

The geometric approach is validated by creating test cases and then visually examining the TCP path so the solution behaves as expected. An IBOFlow sealing simulation is then performed to validate the resulting sealant. Test cases were obtained by manually defining the desired $\mathbf{x}$ and $\mathbf{y}$ as illustrated in Figure 3.5a. Secondly, the curves were interpolated, and Bezier was used to smooth the curve. The accuracy of how the curve follows the original curve depends on the number of evaluation points. Lastly, the curve was arc length parametrized as illustrated in Figure 3.5b using the Euclidean distance between the points as arc length.

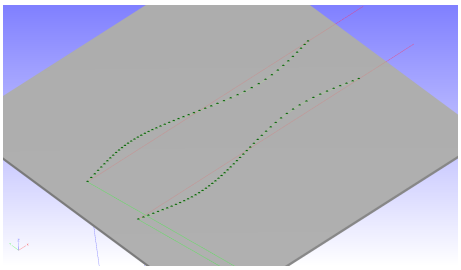
The test cases consist of the eight cases illustrated in Figure 3.6. The cases persist of a zic-zac pattern, large square, small square, half circle with a gap, seam, varying



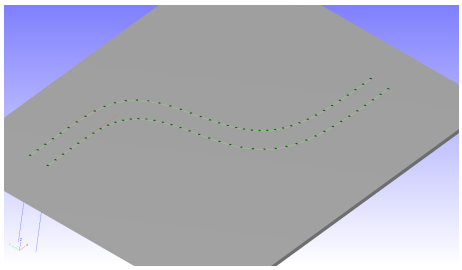
(a) Initial user defined sealant. **(b)** Arc length parametrized sealant

Figure 3.5: Comparison of the user defined sealant and the arc length parametrized sealant.

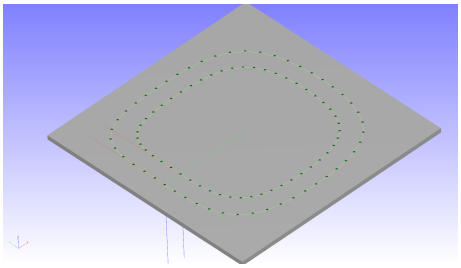
width, and staircase. The left line is the arc length parametrized curve $\mathbf{x}$, and the right corresponds to $\mathbf{y}$. For each case, the graphs corresponding to the line velocities $\dot{\mathbf{u}}, \dot{\mathbf{w}}$, the arc locations $\mathbf{u}, \mathbf{w}$, the constraint distances $\mathbf{b}, \mathbf{c}, \mathbf{d}$, the width $\mathbf{a}$, the sealant heights $\mathbf{g}, \mathbf{h}$, the tilt $\boldsymbol{\rho}$, the line spin angles $\boldsymbol{\eta}, \boldsymbol{\zeta}$, the entry angles $\boldsymbol{\beta}, \boldsymbol{\gamma}$, and drag $\boldsymbol{\kappa}$ is illustrated and is critically examined to see if they follow the expected behavior, which is discussed more in Section 4.1.



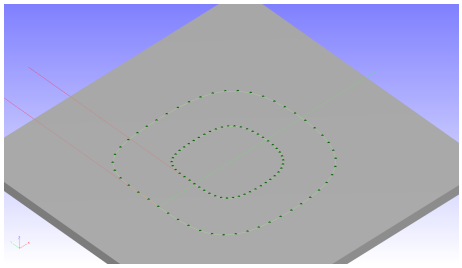
(a) The desired edges of a sealant with a varying width.



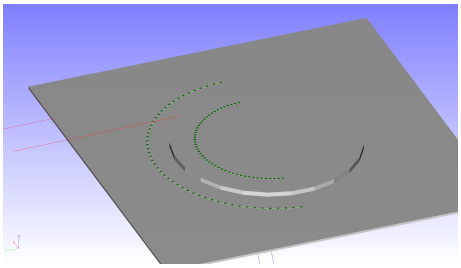
(b) The desired edges of a sealant in a zic-zac pattern.



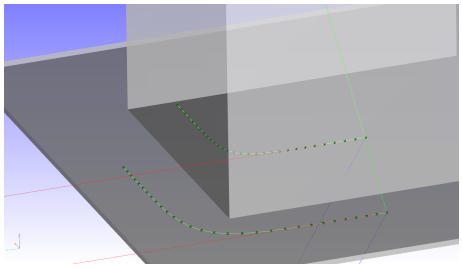
(c) The desired edges of a sealant as a square.



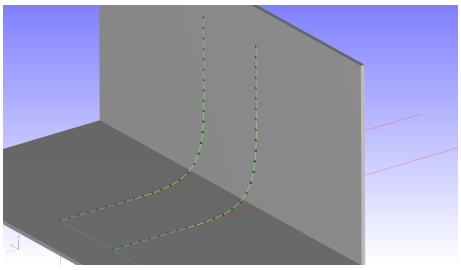
(d) The desired edges of a sealant as a smaller square.



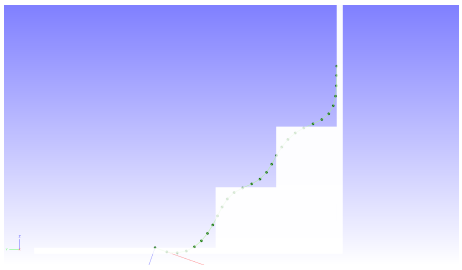
(e) The desired edges of a sealant as a half circle with a small seam.



(f) The desired edges of a sealant in corner.



(g) Desired edges of a sealant for going over a corner.



(h) Desired sealant for going over a stair geometry.

Figure 3.6: Test cases used to validate the geometric approach.

3.2 Surrogate Model

While modifying the initial TCP path described in Section 3.1 is sufficient for generating an initial trajectory based on simple geometric considerations, it does not

include some complex sealant behavior. A surrogate model was developed to encapsulate more complex behavior for the sealant while maintaining validation speed.

3.2.1 Problem description

Based on a given TCP path and a target geometry, an approximate sealing geometry is determined. It is possible to use a training set of known sealing geometries, target geometries, and TCP paths to predict future sealants, for instance, by using statistical curve fitting or neural networks. The goal is to minimize the accumulated cross section difference between the sealant predicted by the surrogate model and the IPS simulation illustrated in Figure 3.7. The green line corresponds to the IPS simulation, the red line is the surrogate models prediction, and the black cuboid corresponds to the desired sealant.

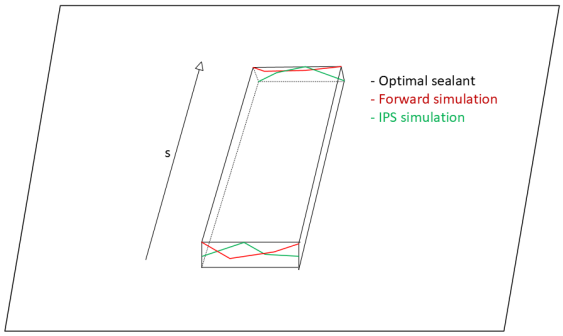
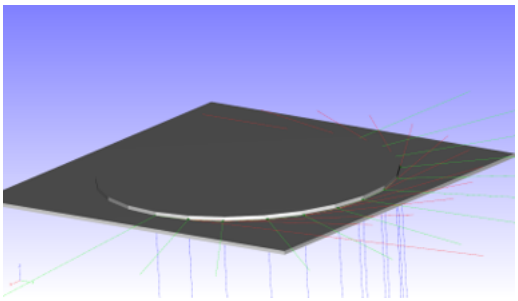


Figure 3.7: The objective for the surrogate model is to minimize the absolute cross section differnece between the surrogate model and the IPS simulation.

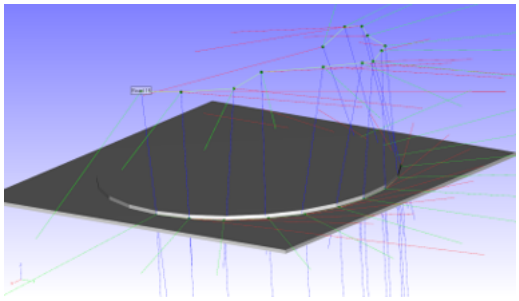
3.2.2 Data Collection

The training data was collected using the IPS IBOFlow simulation framework. The simulations used a mass flow of 15 g/s and a nozzle for flat beads. Initial sealing center points and substrate geometries were created by hand in IPS, where each point consists of a location and an orientation. An example of a sealing center path is illustrated in Figure 3.8a. The TCP is located along a sphere with its center in the sealing center point. The sphere radius is the offset, and the location and orientation of the TCP is described by the tilt, drag, and spin. To expand and ensure a diverse dataset, the TCP offset, velocity, drag, spin, and tilt were randomly initialized within the boundaries of the surrogate model, as presented in Table 3.1. The bounds in Table 3.1 were experimentally obtained to ensure reasonable sealant behaviors. The TCP path is distorted using a normal distribution. The next point is located along a sphere centered around the second sealing center point and was randomly varied from the initialized values drawn from a normal distribution and projected into the domain, where the standard deviations are presented in Table 3.1. For the training set, 25 sealant center paths were manually generated to encapsulate different spatial behaviors. Spanning geometries from a simple plate geometry to slightly more complex geometries such as the half circle or box geometry illustrated

Figure 3.6e, 3.6f. To include different kinematic behaviors, 100 randomized paths were generated for each sealing center path, which resulted in a data set of 2500 paths. A randomized TCP path is illustrated in Figure 3.8b. The network was trained on relatively simple data due to time constraints. More complex geometries could be incorporated into the dataset but would require a larger volume of data. Due to time constraints, the dataset was limited not to consider cases going over corners as illustrated in Figures 3.6g, 3.6h.



(a) A sealant center path located around a half circle.



(b) A randomly generated TCP path.

Figure 3.8: Distorted sealing center path.

Parameter	Min Bound	Max Bound	Standard Deviation
Offset	20 mm	40 mm	2.5 mm
Velocity	0.2 m/s	0.5 m/s	0.02 m/s
Tilt	$-30°$	$30°$	$7.5°$
Drag	$-15°$	$15°$	$5°$
Spin	$-7.5°$	$7.5°$	$3.75°$

Table 3.1: Bounds and standard deviations for various physical parameters.

3.2.3 Dataset

The input of the network is the local linear velocity $\mathbf{v}$, the local angular velocity $\boldsymbol{\omega}$, the local gravity vector $\mathbf{g}$, and a height vector $\mathbf{h}$ measuring the distance to the substrate. The sealant also depends on the past and the future of the TCP path. To address this, both past and future velocities, gravity, and height vectors are used to predict the sealant, the number of evalaution points are denoted N_m. The network will predict a cross-section of the sealant. Figure 3.9 summarizes the input and output of the neural network.

The TCP path consists of the global locations, rotation matrices, and time stamps. The original curve is interpolated to create samples with a 1-centimeter interval, assuming linear constant velocities between TCP points. Figure 3.10 illustrates an initial and a sampled TCP path. The green square is the substrate, and the coordinate frame centers illustrates the tool center points. The initial TCP path is shown

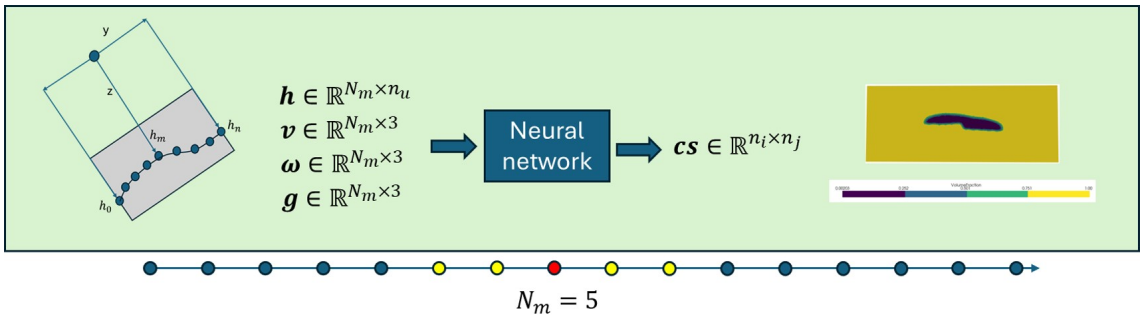
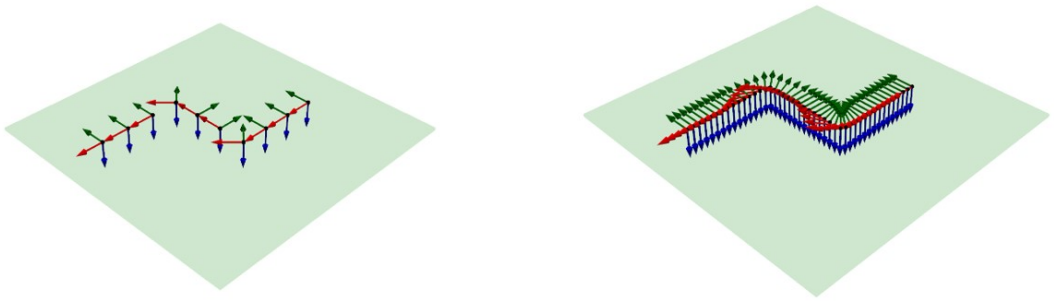


Figure 3.9: Input and output parameters of the neural network.

in Figure 3.10a, and the sampled path is portrayed in Figure 3.10b. The red arrows illustrate the x direction of the nozzle, the green arrow shows the y direction, and the blue arrow shows the z direction.

Let $\mathbf{p_i}$ denote the global position, $\mathbf{R_i}$ the global rotation matrix, and t_i the time at



(a) Initial TCP path. **(b)** Sampled TCP path.

Figure 3.10: Comparison of the initial and sampled TCP paths.

frame i. The input from the TCP file can be summarized as described in Equation 3.62.

$$\mathbf{p} = \begin{bmatrix} \mathbf{p_1} & \cdots & \mathbf{p_N} \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \mathbf{R_1} & \cdots & \mathbf{R_N} \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} t_1 & \cdots & t_n \end{bmatrix} \tag{3.62}$$

The global velocity can be calculated using Equation 3.63.

$$\mathbf{v_{g,i}} = \frac{\Delta \mathbf{p_i}}{\Delta t_i} \tag{3.63}$$

Where $\Delta \mathbf{p_i} = \mathbf{p_{i+1}} - \mathbf{p_i}$ and $\Delta t = t_{i+1} - t_i$. The local velocity vector is calculated in Equation 3.64 using that $\mathbf{R}^{-1} = \mathbf{R}^T$.

$$\mathbf{v_i} = \mathbf{R_i}^T \mathbf{v_{g,i}} \tag{3.64}$$

The local gravity vector is calculated using Equation 3.65, where $\mathbf{g_{global}} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T$.

$$\mathbf{g_i} = \mathbf{R_i}^T \mathbf{g_{global}} \tag{3.65}$$

The local angular velocity is calculated in Equation 3.66.

$$\boldsymbol{\omega_i} = \frac{\boldsymbol{\gamma_i}}{\Delta t_i} \tag{3.66}$$

Where $\boldsymbol{\gamma}$ corresponds to the principal rotation vector calculated in Equation 3.67.

$$\boldsymbol{\gamma_i} = \Phi_i \hat{\boldsymbol{e}}_i \tag{3.67}$$

Where $\hat{\boldsymbol{e}}$ is the principal vector describing the vector where only one rotation is required to go from frame i to i+1, which is the eigenvector corresponding to the eigenvalue 1 of the local rotation matrix between frame i and i+1. Φ_i is the principal angle describing the magnitude around the principal vector to go from frame i to frame i+1.

The height vector **h** is obtained using ray tracing and offsetting the base of the ray trace as illustrated in Figure 3.11. The gray rectangle portrays the cross-section, the blue dots are the raytrace intersection points with the substrate, and the red dot is the current TCP point. The TCP base is offset with an interval of 1 mm.

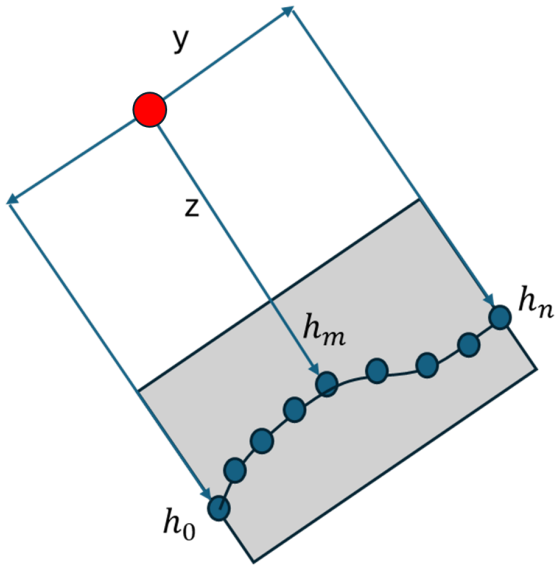


Figure 3.11: Description on obtaining height vector.

The cross-section is obtained by performing a ray trace from the TCP to the substrate illustrated in Figure 3.12 and then adding a window with the same frame orientation as the TCP. The window size is 4 cm in width and 2 cm in height, with one pixel per mm.

An example of the dataset point is illustrated in Figure 3.13. Where the purple arrow is the local linear velocity, the brown arrow is the local gravity vector. The blue points are the ray traces intersecting with the substrate. The angular velocity is not included in the illustration but is a part of the data point. Additionally, the same data is included in $\frac{N_m-1}{2}$ samples before and after the sampled TCP.

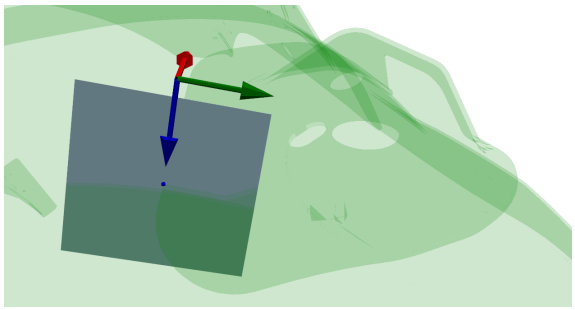


Figure 3.12: Illustration of obtaining a cross section.

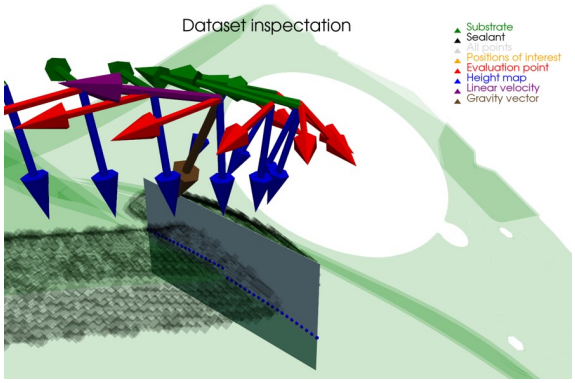


Figure 3.13: Illustration of a dataset point.

3.2.4 Network architecture

This section introduces the network architecture used in the project, Section 3.2.4.1 introduces the glass-box model, and Section 3.2.4.2 describes a neural network featuring Multi Layer Perceptrons (MLP) and convolutional neural network (CNN). Both network introduced do not encapsulate the sequential data and are using a $N_m = 0$.

3.2.4.1 Linear Glass-Box Model

A fully connected layer was chosen as a glass box model due to its interpretability. In contrast to black-box models, the linear layer offers direct inspection of the model parameters where each weight linearly contributes an input element, which enables transparent analysis of input features. The fully connected linear layer is illustrated in Figure 3.14.
The fully connected linear layer for $N_m = 1$ is described in Equation 3.68.

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{3.68}$$

Where $\mathbf{x} = \begin{bmatrix} h_1, \ldots, h_n, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, g_x, g_y, g_z \end{bmatrix}^T$ is the input vector, and $\mathbf{y} = \begin{bmatrix} p_1 & \cdots & p_{n_u \cdot n_w} \end{bmatrix}$ is the output vector. $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias vector which is described in Equation 3.69. Where n_h is the number of elements included in the height vector and $n_u \times n_v$ is the resolution of the output image.

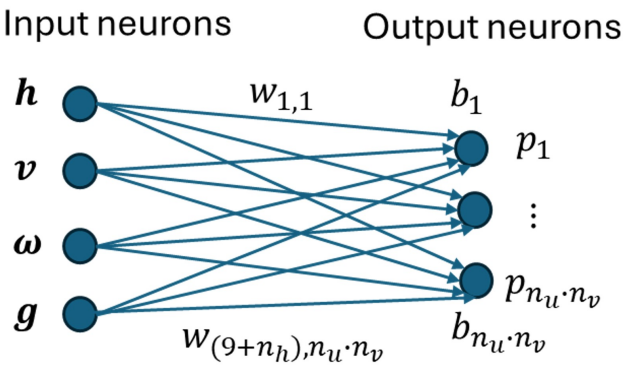


Figure 3.14: Fully connected layer architecture

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n_u \cdot n_v} \\ \cdots & \ddots & \vdots \\ w_{n_h+9,1} & \cdots & w_{n_h+9,n_u \cdot n_v} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_{n_u \cdot n_w} \end{bmatrix}^T \tag{3.69}$$

3.2.4.2 MLP-CNN Network

Multilayer perceptrons (MLP) were used to extract features from the input data and a convolutional network was chosen for pixel prediction because of its ability to capture local patterns and structure. The hybrid MLP CNN architecture is illustrated in Figure 3.15. Where UF stands for an unflattening operation, and CNN stands for convolutional neural network.

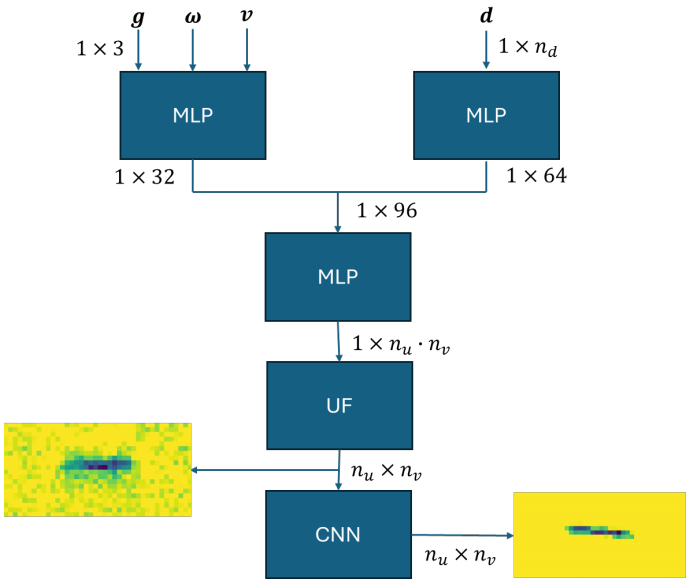


Figure 3.15: MLP CNN Hybrid Architecture.

Individual backbones are used for the kinematic and spatial input groups to extract specialized kinematic and spatial features. The gravity vector, linear, and

angular velocities are kinematically related and forms the kinematic input group, while the spatial input group consist of the height vector. Merging the kinematical states with the height information enables the network to reason the dynamic with the spatial states. An MLP combines the different feature inputs into a compact representation. Lastly, a CNN architecture was applied to consider nearby pixels.

The MLP blocks consist of two fully connected layers, where each layer is followed by a batch norm, and leaky ReLU layer. The batch norm layer reduces the internal covariance shift, making the network more reliable. The Leaky ReLU layer is used to introduce non-linearity, allowing the model to learn more complex nonlinear relations. The CNN incorporated residual blocks (RE block) [23] and squeeze excitement blocks (SE block) [27]. The CNN architecture is illustrated in Figure 3.16.

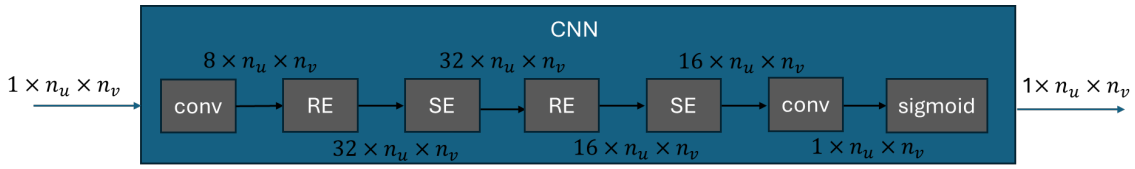


Figure 3.16: The network architecture of the CNN block.

The residual blocks skip connections to improve gradient flow and refinement learning. A residual block is shown in Figure 3.17. Where n_r and n_c denotes the input dimension to the block.

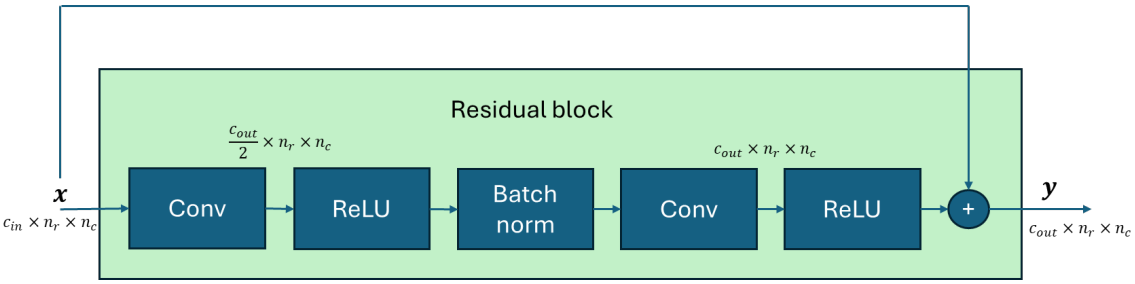


Figure 3.17: The network architecture of the Residual block.

Squeeze-and-Excitation (SE-blocks) were integrated into the network to apply channel-wise attention, enabling the network to emphasize the relevant features. A SE-block is visualized in Figure 3.18.

3.2.5 Training Process

To train the networks PyTorch [35] was chosen because of its user-friendliness and integrated GPU acceleration. Additionally, it is widely adopted in industry and academia and integrates with libraries such as TorchVision [36], a helpful library for image tasks.

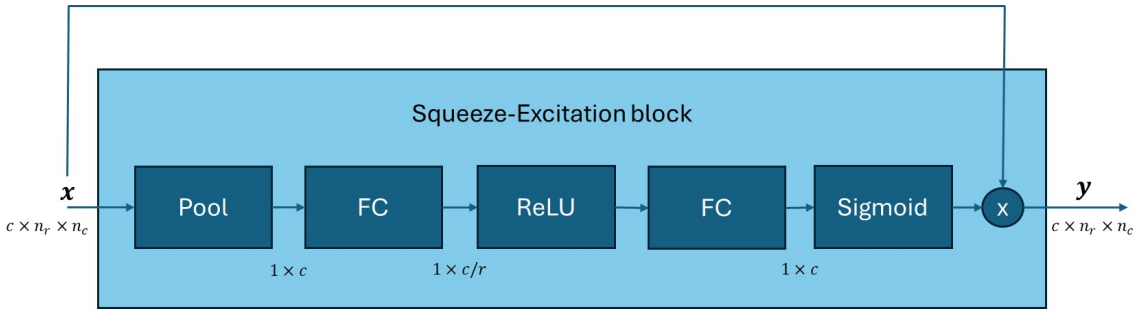


Figure 3.18: The network architecture of the Squeeze-and-Excitation block.

The training pipeline is summarized in Figure 3.19. Given a substrate geometry and a TCP path, a sealing simulation could be performed in IPS. From the TCP path, the substrate geometry and simulation, the input, and the ground truth, also called the label, can be obtained. The data is randomly split into mini-batches, which represent the number of data samples seen before updating the networks internal weights. The networks prediction is compared to the ground truth through a loss function. The loss function is chosen as the Mean Square Error (MSE) since it is intuitive and directly punishes the difference between the predicted pixel and the ground truth. Thereafter, backpropagation is performed to validate how much each internal weight contributed to the loss. The weights are then updated using an optimizer. Stochastic Gradient Descent (SGD) [37] was chosen as the optimizer because of its intuitiveness and generalization properties, making SGD a suitable choice for pixel prediction. While adaptive optimizers, such as Adams [38], outperform SGD in early training, the adaptive optimizers often result in worse generalization performance [39]. The training runs over epochs, which refers to the number of times the network will see the full dataset.

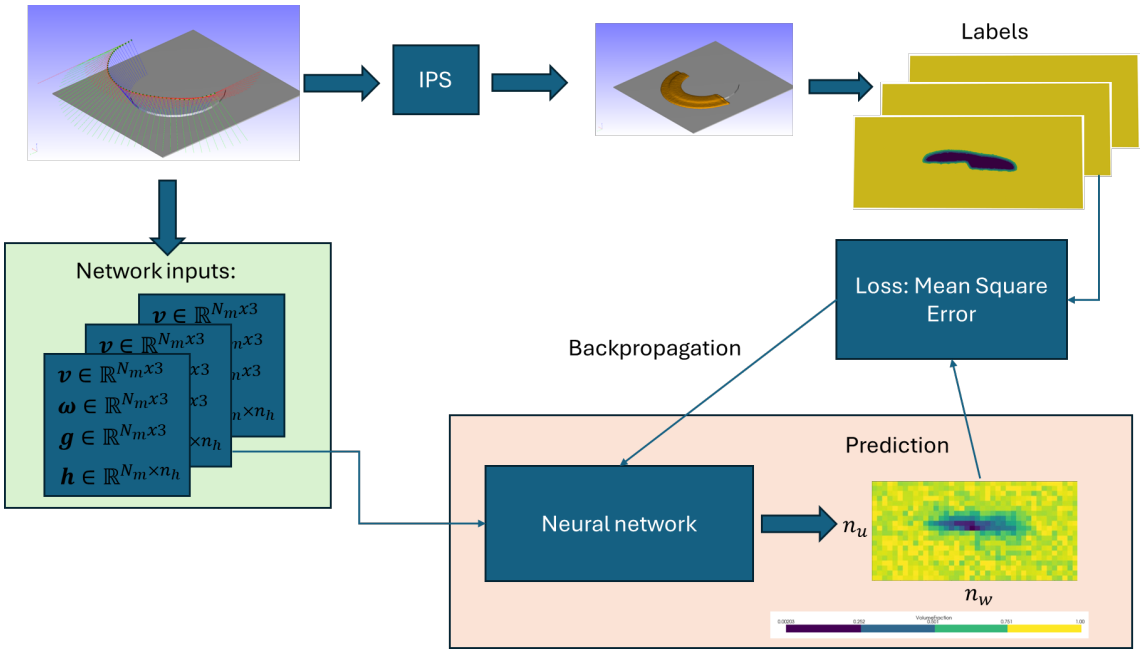


Figure 3.19: Training pipeline used for the surrogate model.

3.2.6 Evaluation Metric

The performance of the models will be evaluated through training error, validation error, and visual inspection. The training error refers to the model's MSE on the training set, and the validation error refers to the model's MSE on unseen data. Additionally, visual inspection is performed to identify potential defects not captured by MSE alone.

The model was validated on an obstacle course with geometries not included in the training process which is illustrated in Figure 3.20. Twelve paths were created where each path was randomly generated as described in Section 3.2.2.

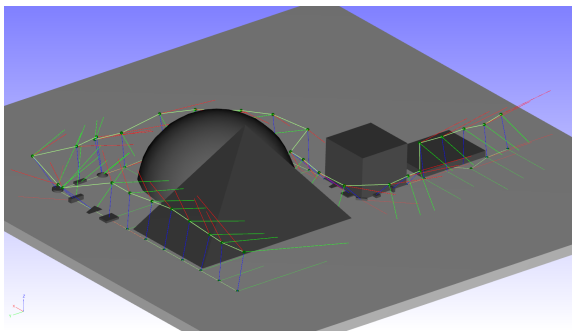


Figure 3.20: Obstacle course created to validate the network.

4
Results

This chapter presents the results of the geometric approach and the surrogate model. The calculated optimal TCP is visualized and simulated. Thereafter, the training and validation loss for the glass box model and CNN are presented, and the predicted cross-sections are visualized.

4.1 Geometric Approach

The following section demonstrates the results for each test case presented in Section 3.1.4. The constraints applied during the optimization are presented in Table 4.1. Scipy [34] Sequential Least-Squares Quadratic Programming (SLSQP) solver was chosen to solve the optimization problem.

As described in Section 3.1.4, eight test cases were introduced. For each case the graphs corresponding to the line velocities $\dot{\mathbf{u}}, \dot{\mathbf{w}}$, the arc locations $\mathbf{u}, \mathbf{w}$, the distances $\mathbf{b}, \mathbf{c}$, the width $\mathbf{a}$, the sealant heights $\mathbf{g}, \mathbf{h}$, the tilt $\boldsymbol{\rho}$, the line spin angles $\boldsymbol{\eta}, \boldsymbol{\zeta}$, the entry angles $\boldsymbol{\beta}, \boldsymbol{\gamma}$, and drag $\boldsymbol{\kappa}$ are illustrated. The described graphs are referred to as the parameter plots.

Parameter	**Constraint value**
a_{min}	10 mm
a_{max}	100 mm
ρ_{min}	$-\frac{\pi}{6}$
ρ_{max}	$\frac{\pi}{6}$
κ_{min}	$-\frac{\pi}{6}$
κ_{max}	$\frac{\pi}{6}$
$\dot{u}_{min}$	0.1 m/s
$\dot{u}_{max}$	1 m/s
h_{tol}	0.01 mm
k_1	1
k_2	10
k_3	100

Table 4.1: Parameters used to validate the geometric approach.

4.1.1 Different widths

The different width case exhibits expected simulation results, which is illustrated in Figure 4.1.

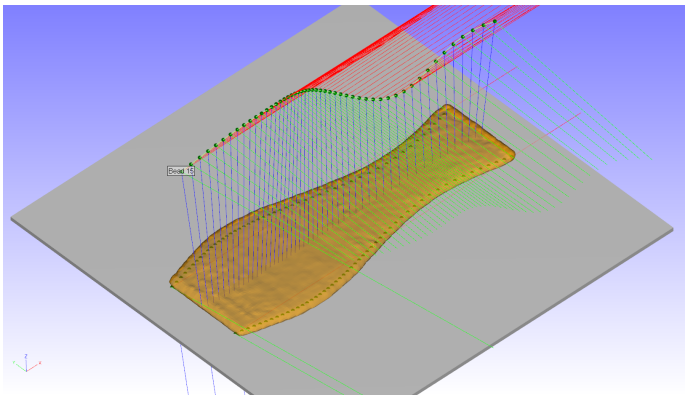


Figure 4.1: Simulated sealant for a sealant changing the desired width over the path.

The parameter graphs are shown in Figure 4.2. The line velocities increase while the offset decreases when the desired width is low, and the offset from the substrate increases while the velocity decreases when the sealant width is large. Constraints regarding the sealant height and geometric constraints are achieved. The geometric constraints refer to the distance $\mathbf{b}, \mathbf{c}$, and the angle constraints are the tilt $\boldsymbol{\rho}$ and the drag $\boldsymbol{\kappa}$.

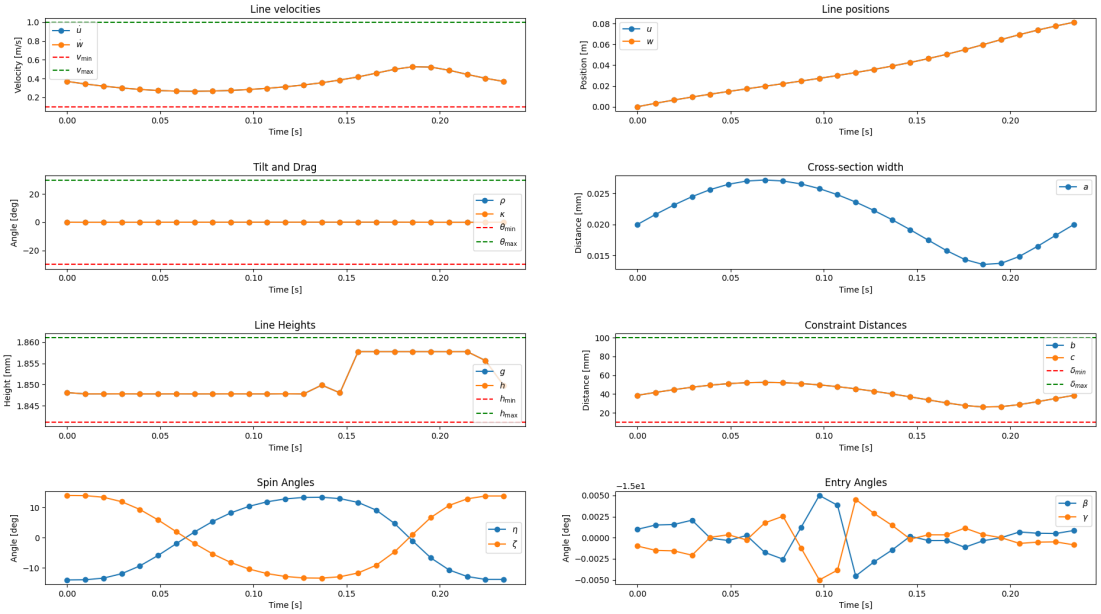


Figure 4.2: Parameters for a sealant changing the desired width over the path.

4.1.2 Zic-Zac

The zic-zac case reveals promising results as illustrated in Figure 4.3.

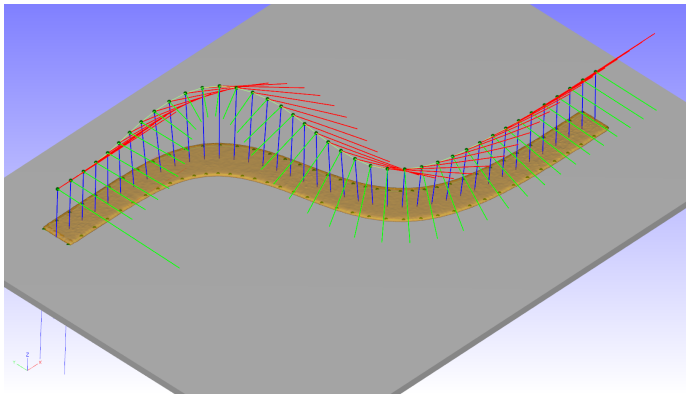


Figure 4.3: Simulated sealant for a zic zac path.

The parameter graphs are presented in Figure 4.4. Looking at the velocity profile along **x**, and **y**. The velocity of the outer curve increases when the curve turns. However, the sealing distribution is compensated by the tilt $\boldsymbol{\rho}$. All geometric bounds and sealant height constraints are fulfilled.

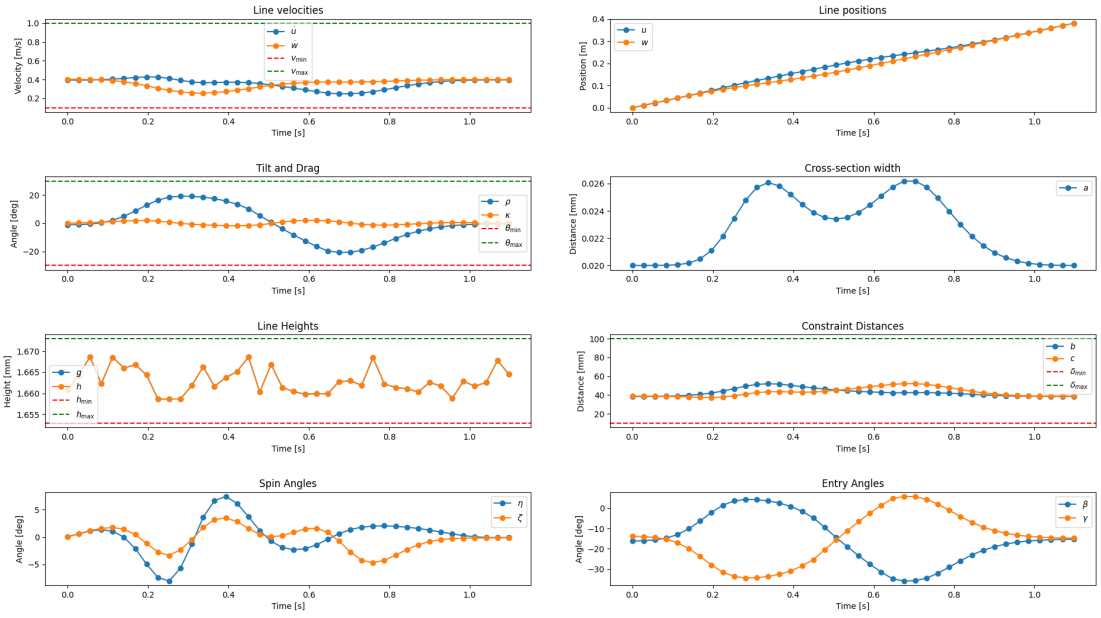


Figure 4.4: Parameter plots for a zic-zac path.

4.1.3 Large square

The large square shows results as expected which is presented in Figure 4.5.

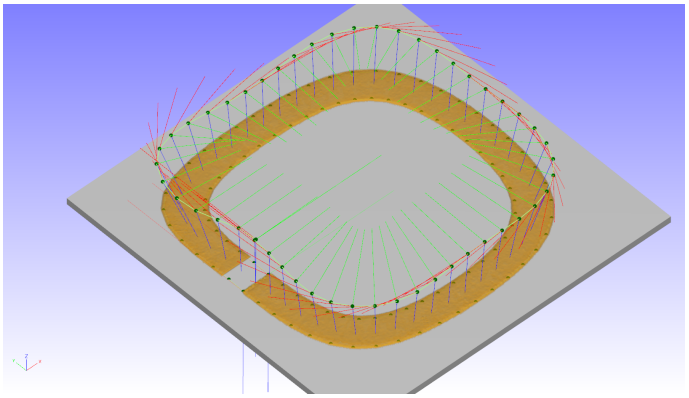


Figure 4.5: Simulated sealant for a square path.

The velocity along the outer curve is higher, especially around the corners. While the tilt slightly increases, which is displayed in Figure 4.6. All geometric parameters are within its constraints.

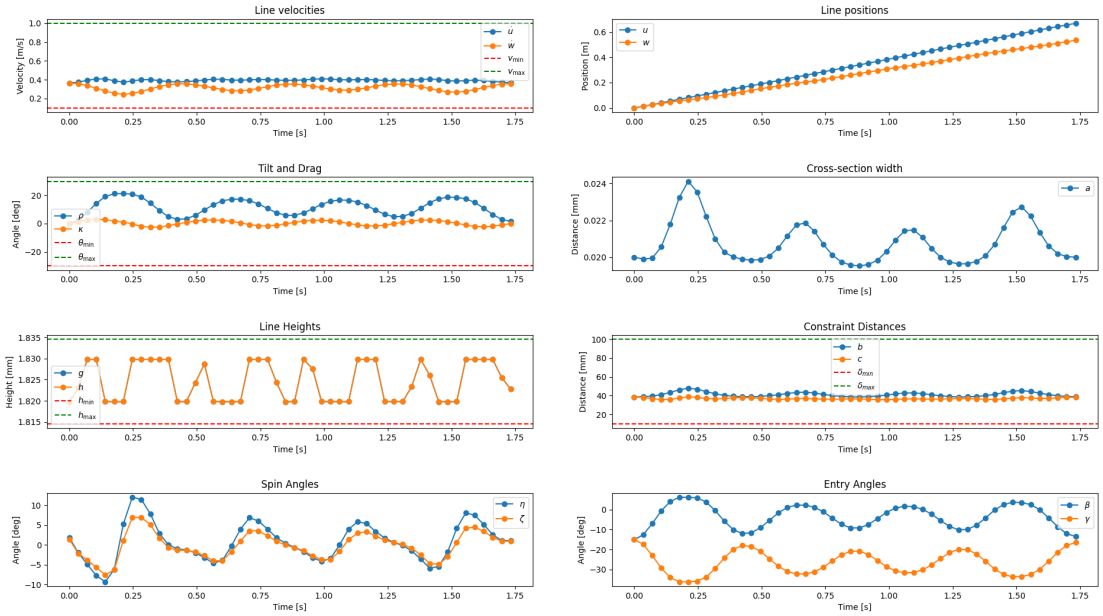


Figure 4.6: Parameters for a square path.

4.1.4 Small square

The path for the small square path is reaching an iteration limit, the best results while not validating any constraints is illustrated in Figure 4.7.

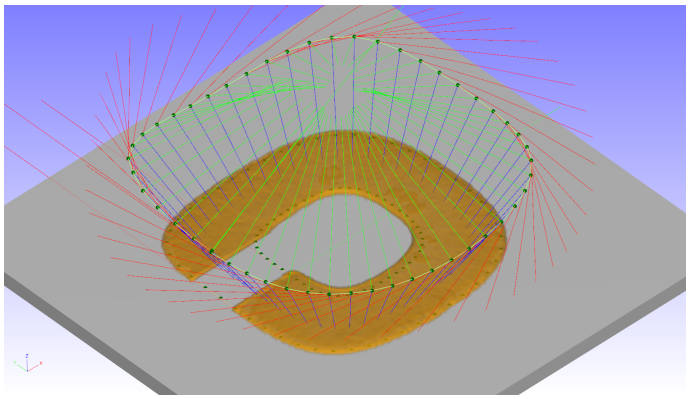


Figure 4.7: Simulated sealant for a square, with short sides.

The problem has no feasible solution, as the constraints are too conservative to find a solution with an equal height distribution along the sealant edges. Increasing the allowed tilt would make it possible for the solver to find a solution. The behavior is as expected. The tilt is at the limit of large parts of the trajectory to compensate for the excess amount of sealant applied on the outer sealant edge since $\|\mathbf{x}\| \gg \|\mathbf{y}\|$. More tilt is needed to achieve an equal height distribution along $\mathbf{x}$ and $\mathbf{y}$. The parameter graphs are visualized in Figure 4.8.

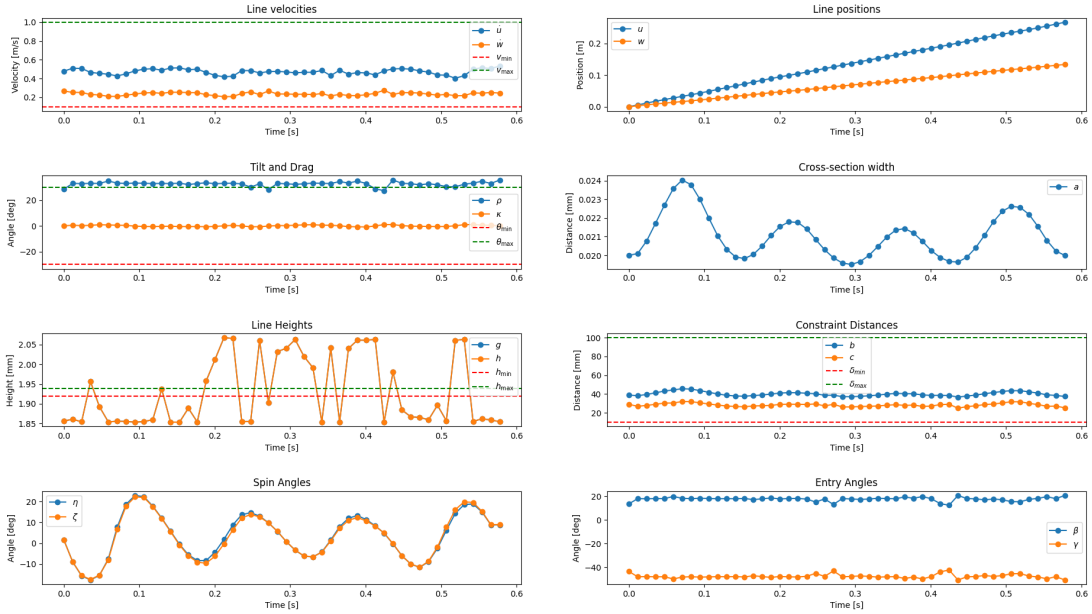


Figure 4.8: Parameters for a smaller square path.

4.1.5 Plate circle

The geometric approach ensures an equal height along $\mathbf{x}$ and $\mathbf{y}$, but what happens between the desired sealant edges is not considered. Therefore, a small seam between a plate and a cylinder was tested to see if the solution is feasible even with some geometric disturbances between $\mathbf{x}$ and $\mathbf{y}$. The simulated sealant is illustrated in Figure 4.9. This shows promising results for a small disturbance.

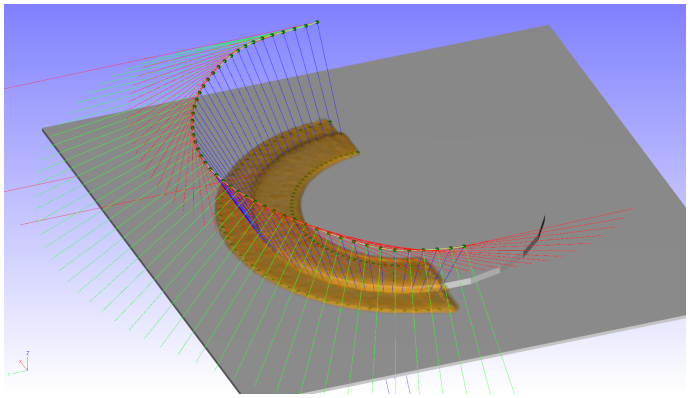


Figure 4.9: Simulated sealant for a small seam created by a cylinder

The parameter graphs are illustrated in Figure 4.10. It shows expected behavior by having a constant tilt and $\dot{\mathbf{u}} > \dot{\mathbf{w}}$ during the entire trajectory. All geometric constraints are within the bounds.

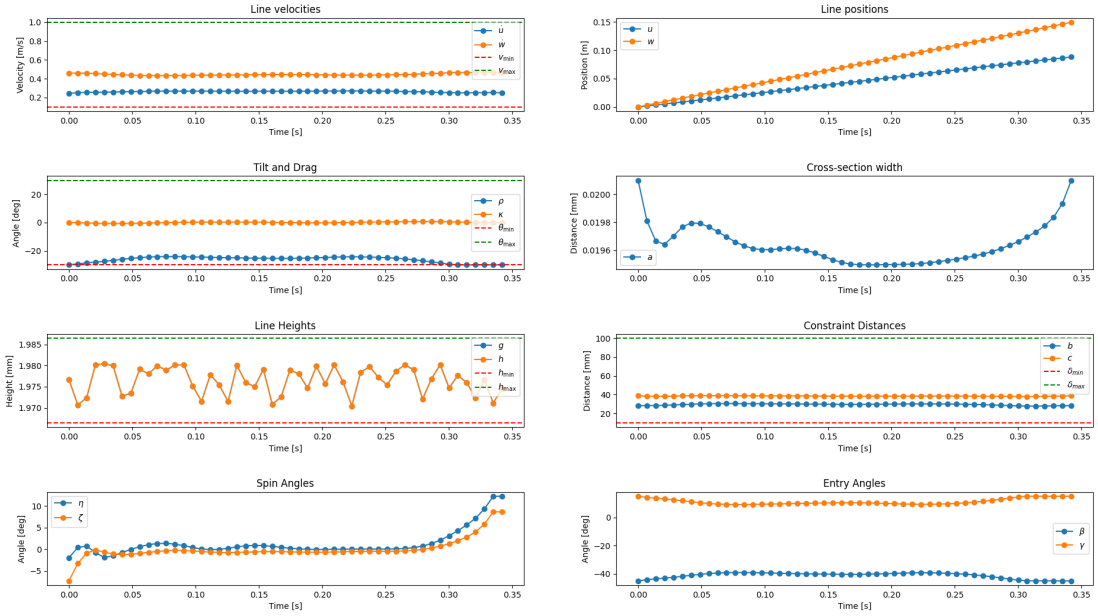


Figure 4.10: Parameter plots for a small seam created by a cylinder.

4.1.6 Plate cube

To analyze larger disturbances between $\mathbf{x}$ and $\mathbf{y}$, a seam between a plate and a box was created. The simulated sealant is portrayed in Figure 4.11, demonstrating good results even when the geometric disturbance is high. If the curvature is too high, the final time t_f can be adjusted, but the idea of an equal height distribution along $\mathbf{x}$ and $\mathbf{y}$ shows promising results.

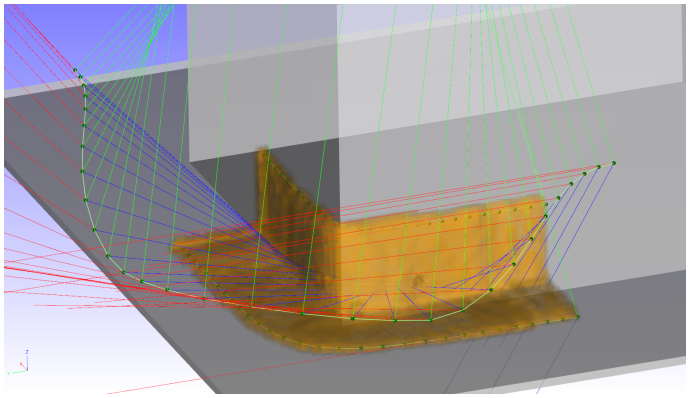


Figure 4.11: Simulated sealant for a step created by a box on a plate.

The parameter graphs shown in Figure 4.12 demonstrate good results. Where the tilt increase in the corner and $\dot{\mathbf{u}} > \dot{\mathbf{w}}$. All geometric constraints are within the bounds.

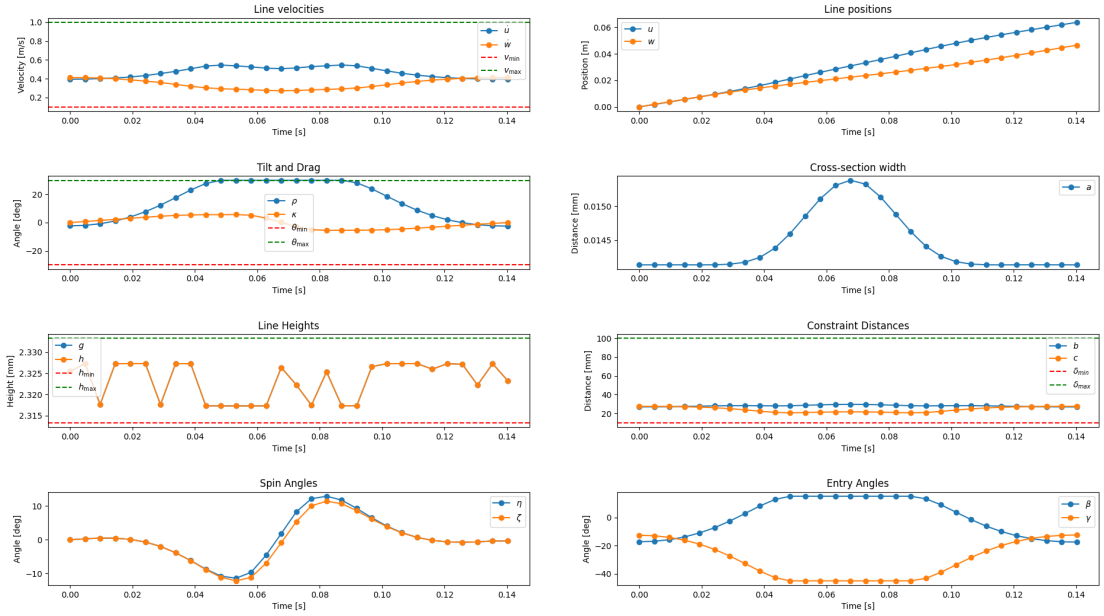
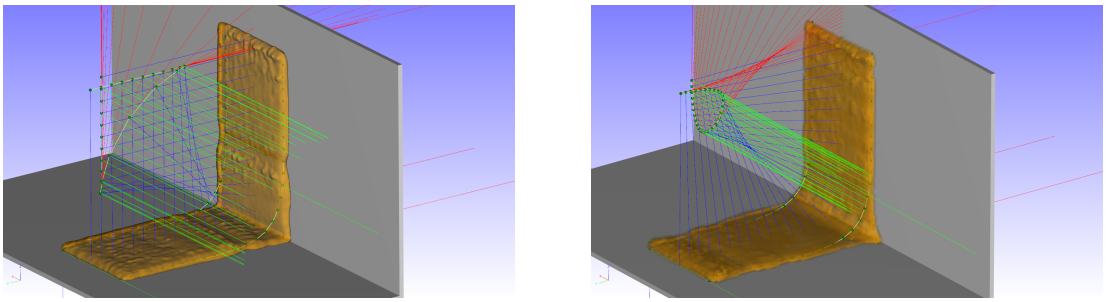


Figure 4.12: Parameter plots for a step created by a box on a plate.

4.1.7 Ledge

A ledge geometry was created to analyze the effect of $\boldsymbol{\kappa}$. If $\boldsymbol{\kappa}$ is locked to zero, the sealant geometry is showing good results, but the path is difficult for a robot to execute. Figure 4.13a illustrates an optimized path for $\boldsymbol{\kappa} = \mathbf{0}$. Figure 4.13b shows the simulated sealant, including optimization of $\boldsymbol{\kappa}$, maintaining acceptable simulation results while making the path executable. In the corner, there is a slight increase in the sealants width. This effect is described further in Section 4.1.8.



(a) Sealant geometry and TCP path when $\boldsymbol{\kappa} = 0$.

(b) Sealant geometry and TCP path when $\boldsymbol{\kappa}$ is optimized.

Figure 4.13: Comparison of sealant and TCP path depending on optimization of $\boldsymbol{\kappa}$.

The parameter graphs are portrayed in Figure 4.14, showing expected results with zero tilt, and no geometric constraints violated. While compensating with $\boldsymbol{\kappa}$ to keep the trajectory executable.

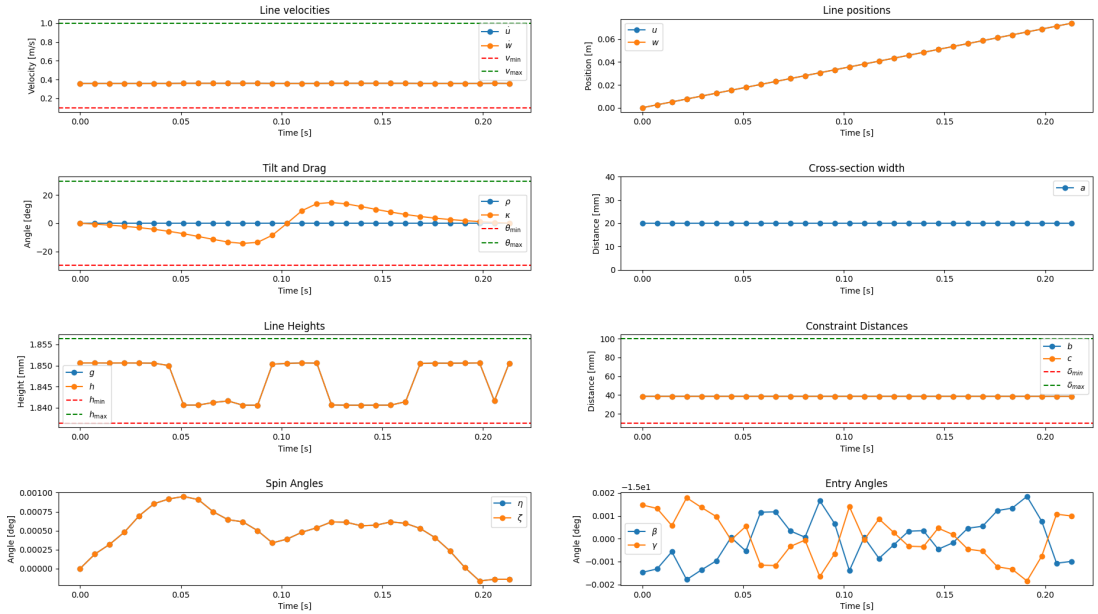
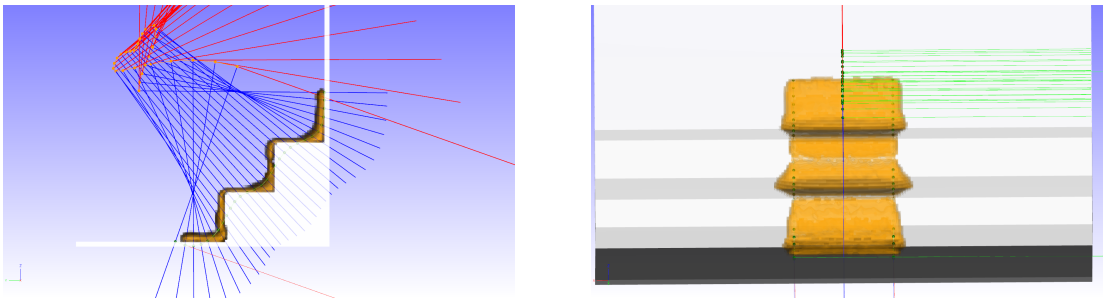


Figure 4.14: Parameter plots for a ledge created by two plates.

4.1.8 Stairs

To stress the freedom in drag, a stair geometry was created. Figure 4.15 shows the sealant geometry from a side perspective, and Figure 4.15b illustrates the sealant from a top perspective. From the side perspective, the effect of the curve smoothening described in Section 3.1.4 is seen. Where $\mathbf{x}$ and $\mathbf{y}$ can not follow the initial desired sealant. The simulation is illustrated in Figure 4.15b, which introduces a variation in width.



(a) Sealant geometry for a stair geometry.

(b) Parameter plots for a stair geometry.

Figure 4.15: Simulated sealant for steps created by a boxes on a plate.

The parameter graphs shown in Figure 4.16 demonstrate reasonable results, where $\boldsymbol{\kappa}$ is oscillating when going over the corners, arc length velocities along the two sealant edges are equal, and there is no tilt.

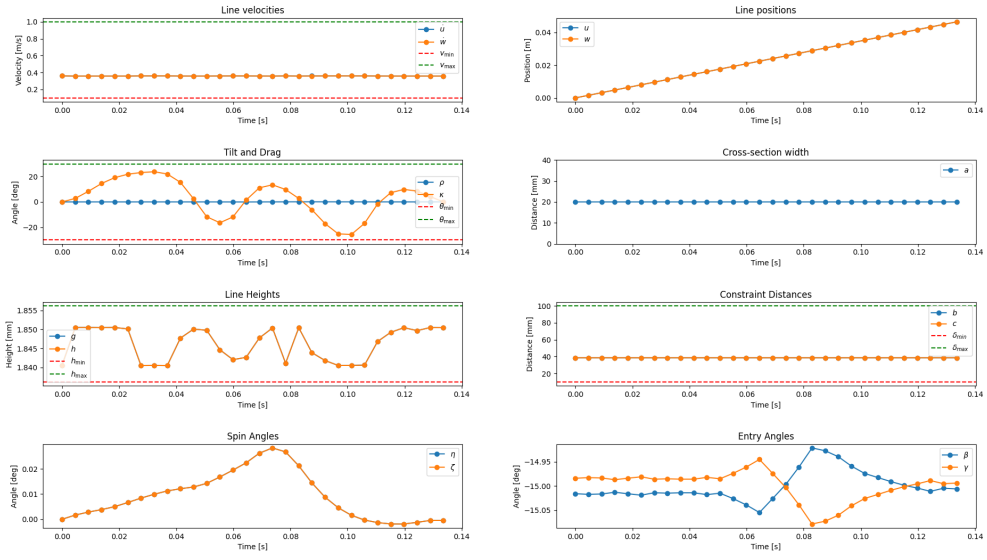


Figure 4.16: Parameter plots for a stair geometry.

4.2 Surrogate Model

In this section, the results of the forward simulation are presented. Demonstrating how well the glass-box model and the MLP-CNN architecture introduced in Section 3.2.4 perform on a validation set introducing unseen kinematics and geometries. The convolutional neural network performance is compared to a linear glass-box model to see if the depth in the MLP-CNN architecture extracts useful features.

4.2.1 Glass-box Model

Figure 4.17 illustrates the training and validation loss for the glass-box model. The orange line corresponds to the validation loss, and the blue line shows the training loss.

Figure 4.17: Training and validation loss for the glass box model for 500 epochs.

Figure 4.18 illustrates the evolution of the training in which a random cross section was chosen for each epoch. The two most left images correspond to the prediction and label of the validation set and the two images to the right correspond to the training set. The images suggest that a linear model can learn simple features regarding spatial and kinematic information.

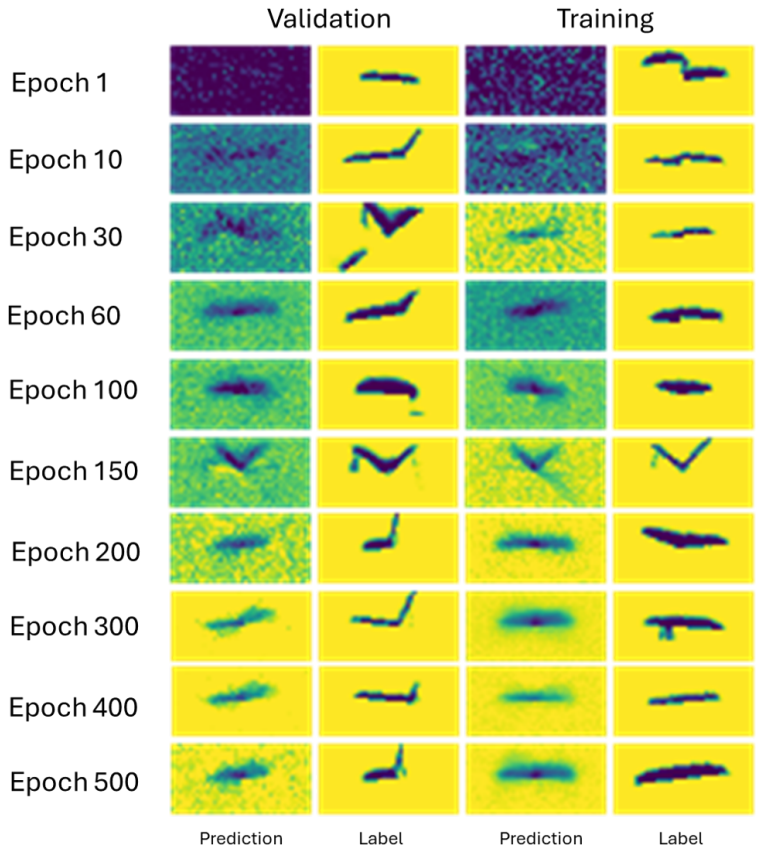


Figure 4.18: Training evolution for the glass box model for 500 epochs.

The pixel MSE for the 12 validation paths are illustrated in Figure 4.19. To validate if a cross-section is feasible, a pixel MSE of 0.02 was set as a threshold, meaning that the average pixel error is smaller than 15 %. This value was arbitrarily chosen and is merely used to compare the data presented in the thesis. The validation had an accuracy of 41.7 %. The red vertical lines correspond to each path. There is a trend where the network performs better at the beginning of the path. An explanation is that the start of the path has simpler geometries compared to the end of the path.

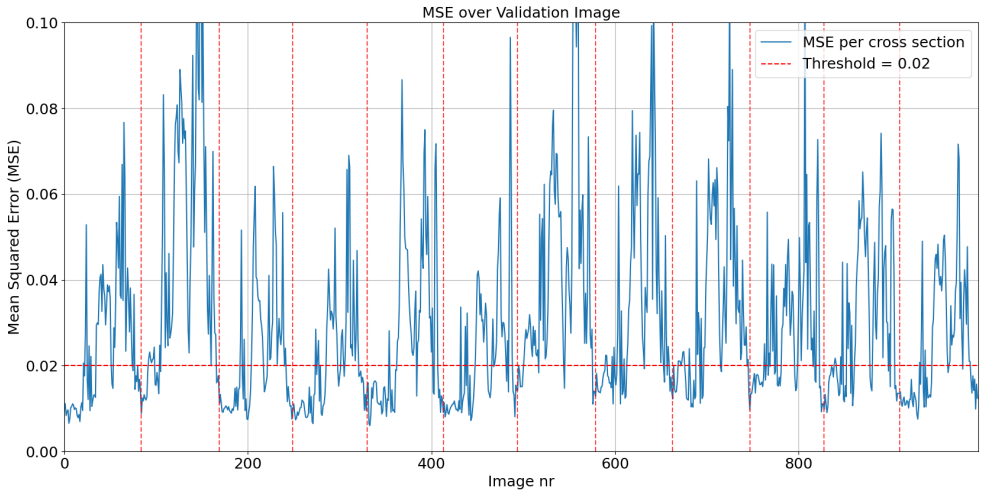


Figure 4.19: MSE over validation paths.

4.2.2 MLP-CNN Network

Figure 4.20 illustrates the training and validation loss for the MLP-CNN model. The blue line corresponds to the training loss, and the orange line shows the validation loss. The validation loss is not overfitting to the data, indicating either that the validation set is to close the training set or that the network learns meaningful features. Since the TCP path is randomly generated, each path is unique, and the model will perform well within the bounds presented in Table 3.1. A greater variation in the bounds could be experimented with.

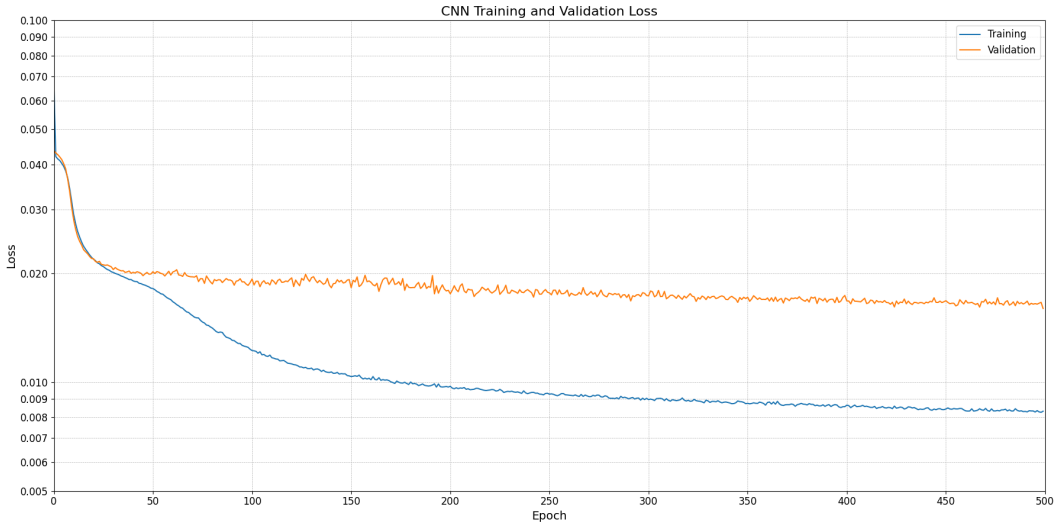


Figure 4.20: Training and validation loss for the MLP-CNN hybrid model for 500 epochs.

The training evolution is presented in Figure 4.21 to identify whether the network

learns useful features or has collapsed to predict a generalized output that only minimizes the training loss. The two most left images correspond to the prediction and label of the validation set and the two images to the right show the training set. The sample is chosen randomly for the epochs.

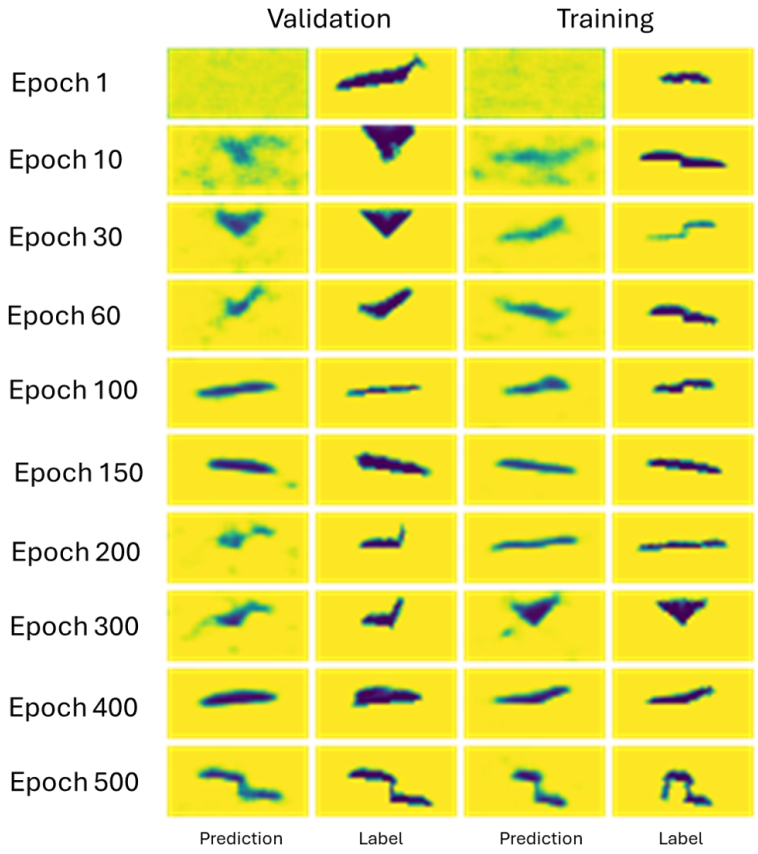


Figure 4.21: Training evolution for the CNN model for 500 epochs.

The pixel MSE for the 12 spatial validation paths are illustrated in Figure 4.22. The red horizontal line shows the threshold for the MSE, and the red vertical lines show the start and end of each validation path. The accuracy of a 0.02 MSE or better for the validation set was 70.38%. As illustrated in Figure 4.22, the network generally performs well at the beginning of the path but performs worse in the later stages, where the red dotted lines separate the twelve paths. An explanation is that the start of the path has geometries that are easier for the network to predict. For instance, at the beginning of the validation set, the path moves over a slope leaning from the right, which the network seems to handle well. While reaching more complex geometries, such as the sphere, the network performs worse. Indicating that spatial information has a high impact on network performance.

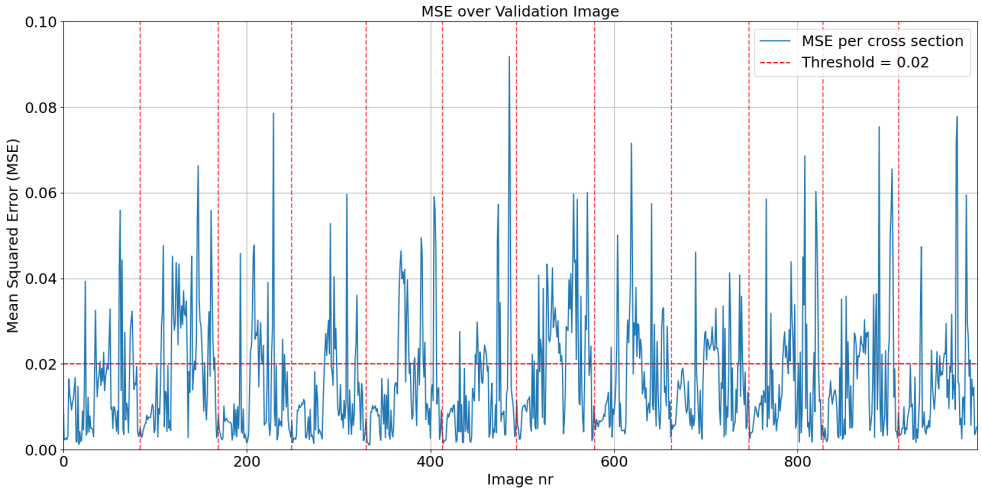


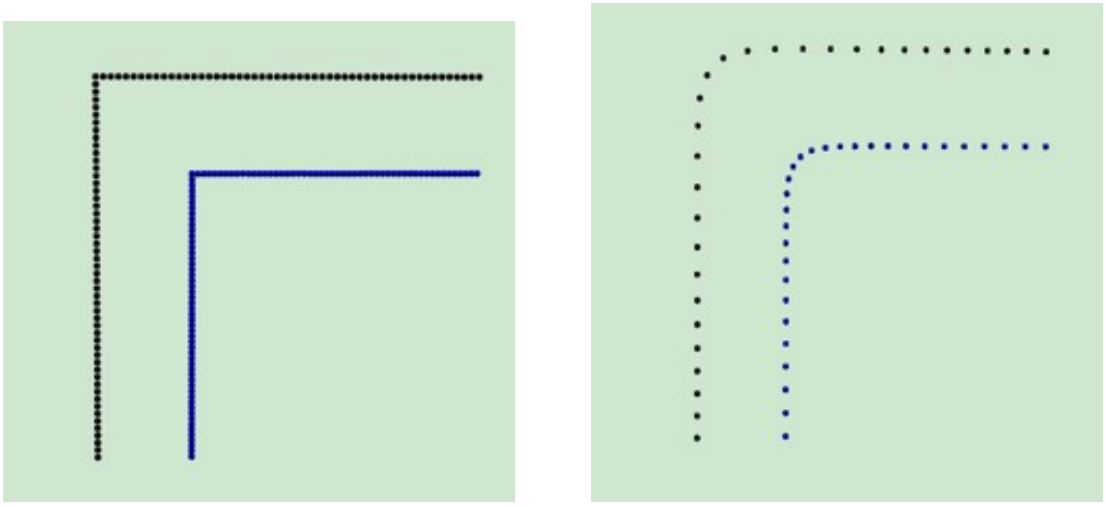
Figure 4.22: MSE over validation paths for the validation set.

5
Discussion

The geometric approach is discussed in Section 5.1, while a discussion regarding the surrogate model is presented in Section 5.2. The research questions introduced in Section 1.3 is discussed in Section 5.3.

5.1 Geometric Approach

While the geometric approach showed promising results for a large set of curves, it has some limitations related to the physical limitations of mechanical devices. Paths with sharp corners, for example, are impossible to follow with high velocity. For the same reason sharp changes in the desired sealant, as illustrated in Figure 5.1a, the solver can not find a feasible solution. Bezier curves are applied to smooth the curve. Figure 5.1b demonstrates the closest smoothing to the original curve while maintaining a feasible solution.



(a) Desired sealant over a sharp right angle.

(b) Closest Bezier curve that can yield feasible curves.

Figure 5.1: Limit in curve smoothing.

Furthermore, curve smoothing introduces additional limitations. For instance, going over an edge as described in Section 4.1.8, the width will not be constant. Due to the curve smoothening, some points of the desired sealant edges are located outside, and some will be located inside the substrate as illustrated in Figure 5.2.

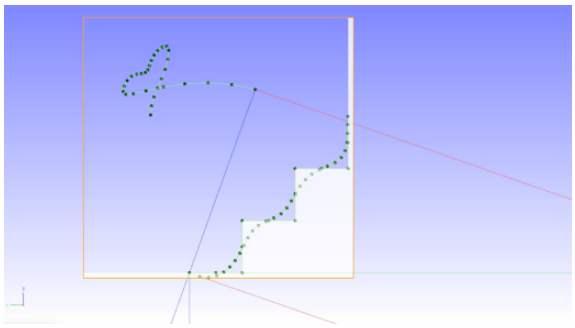


Figure 5.2: Effect of curve smoothening.

There will be a larger width when the desired sealant is located outside the substrate and a smaller width when it is located inside the substrate, which is illustrated in Figure 5.3.

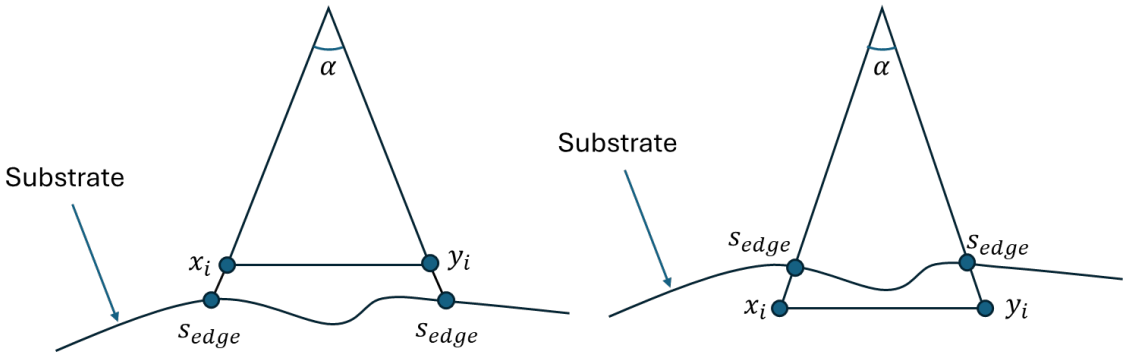


Figure 5.3: Curve smoothening effect on sealant width going over edges.

Reducing the nozzle volume flow would slow down the entire process, allowing more precise solutions in sharp corners like in Figure 5.3 and the stairs in Figure 4.15.

The constraints set due to executability in the TCP path limit some desired sealants. For example, as described in Section 4.1.4, when one of the sealant edges is much longer than the other, $\|\mathbf{x}\| \gg \|\mathbf{y}\|$. No feasible solution can be found without violating the tilt constraint. The same holds for the other constraints. The constraints set limitations on what sealants are possible. Because of the constraint set on $\mathbf{b}$ and $\mathbf{c}$, the sealant has limitations in its width. If the desired sealant is too wide, the distance to $\mathbf{b}$ and $\mathbf{c}$ will go out of bounds. If the desired sealant has a small height, the line velocity bounds set on $\dot{\mathbf{u}}$, and $\dot{\mathbf{w}}$ will be exceeded.

The simplified model ensures equal height distribution along the sealants edges, but what happens in between is not considered. The solver showed promising results for the seam created by the circle and box described in Sections 4.1.5, 4.1.6. However, cases exist where the solution is not optimal. For instance, given a seam along the geometry as illustrated in Figure 5.4. The TCP should be located so it can spray into the seam to prevent air bubbles.

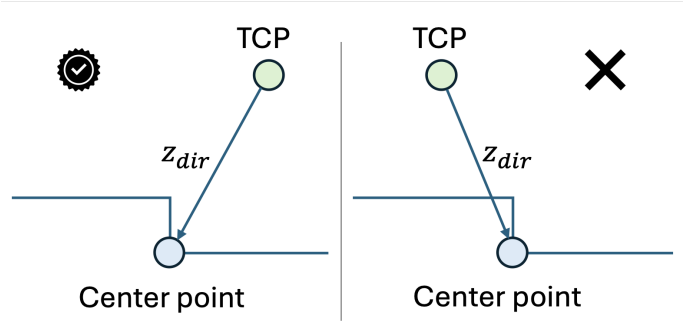


Figure 5.4: TCP location to prevent air bubbles.

5.2 Surrogate model

The surrogate model showed promising results for varying kinematics. However, for more complex geometries, an increased performance is required for the model to be deployable. The model was trained on 25 different geometries. Adding more geometries to the training data could help the network improve its performance.

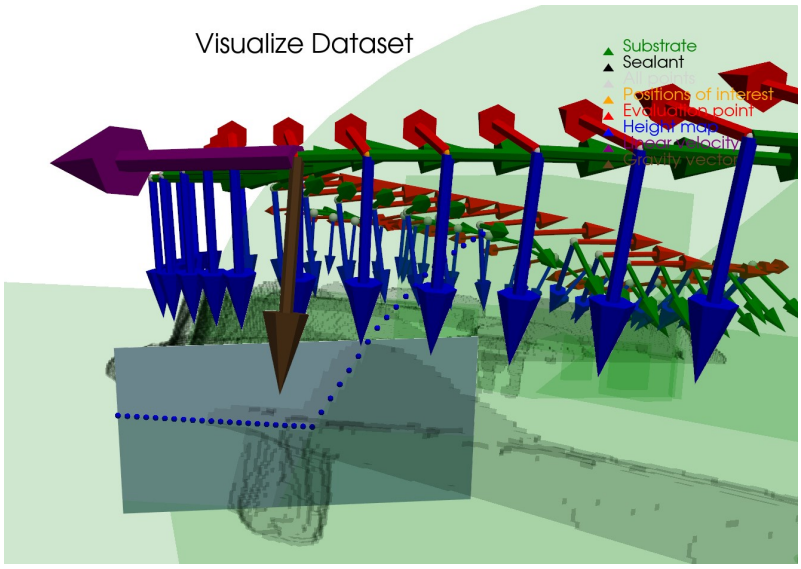
In this project, CNN was chosen to be included in the network architecture because of its strength in image tasks. For instance, AlexNet [40] is commonly used for image classification, You Only Look Once [41] for object detection, and U-Net [42] for image segmentation. The sealing results does not only depend on the current data points, but also on previous and later data. For increased performance this could be introduced in the input to the network. By introducing sequential data some recurrent neural network architecture could be considered. Some widely used recurrent neural networks include Gated Recurrent Units [43] and Long Short Term Memory [44].

While the glass-box model is able to learn some features to predict cross-section, the MLP-CNN outperforms the glass-box model in both accuracy, training, and validation loss. Table 5.1 presents the training loss, validation loss, and accuracy for the glass box model compared to the MLP-CNN.

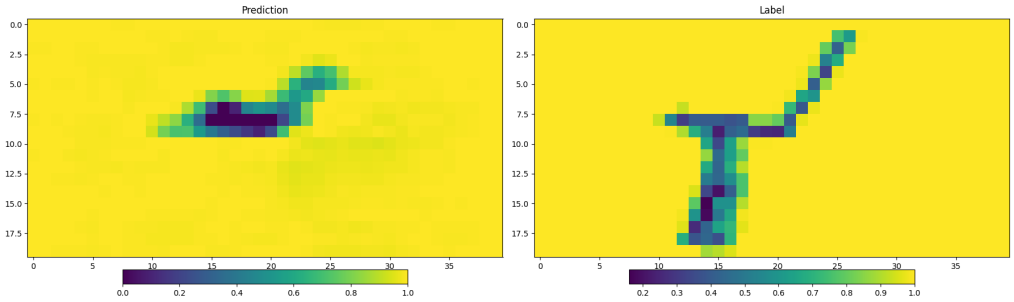
Table 5.1: Comparison of network performance.

Network	Training Loss	Validation Loss	Accuracy
Glass-box	0.019	0.027	41.7%
MLP-CNN	0.0083	0.016	70.4%

Given the network is trained on data that performs as expected, the surrogate model could result in more reasonable results when the simulation acts unexpectedly, which is illustrated in Figure 5.5. Where the sealant is going through the substrate, which is a limitation in the simulation software, the surrogate model still predicts a reasonable cross-section. The black geometry illustrates the simulated sealant and the green geometry is the substrate.



(a) Visualization of dataset point with error in simulation.



(b) Predicted cross section compared to faulty label.

Figure 5.5: Surrogate model predicts reasonable sealant for unexpected behavior in the simulation software.

5.3 Research Question

The research questions investigated in the thesis are:

- **How do the simulated sealants using a TCP path calculated by a geometric approach compare to the desired sealant?** As shown in Section 4.1, the sealant calculated by the geometric approach is sufficient for a large set of curves. However, the geometric approach does not consider discontinuities between the sealant edges, which introduce challenges for complex geometries. Additionally, the geometric approach has some limitations related to the physical limitations of mechanical devices, as illustrated in Figure 5.1.
- **How do sealing simulation predictions made by the surrogate model**

compare to simulations performed in IPS IBOFlow? As shown in Section 4.2.2, the sealing predictions made by the surrogate model showed promising results for further development. The model learns some useful features, but further development is needed for the surrogate model to be useful within an optimization formulation. The MSE over the validation path is shown in Figure 4.22, where a significant decrease in performance is seen when the validation geometry becomes more complex. The model can predict the shape of the sealants cross-section, but determining the cross-section with pixel-wise precision is challenging, as seen in predictions of the validation set in Figure 4.21.

- **How do simple neural networks compare to deep neural networks for sealing simulations?** Comparison of Figure 4.17 and Figure 4.20 shows that the CNN outperforms the glassbox model in prediction error, which is visually illustrated by comparing Figure 4.18 and Figure 4.21.l.

5.4 Ethical and Sustainability Aspects

The project focuses on developing a framework that increases the efficiency of robot trajectory generation. The project has no direct ethical or sustainable impacts, but some indirect aspects are considered.

The thesis does not presents any obvious moral issues. Developing a framework for robot trajectory generation with the intention of saving time and resources is considered a neutral technical advancement. The primary concern is that robots could replace human labor. However, the car industry does already use robots to apply sealants, and the goal of the thesis is to do so more effectively, which is seen as net positive.

The project does not have any clear environmental impact. However, the automotive industry is pushing for sustainability, and any advancement contributing to more efficient use of time could indirectly contribute to sustainability. However, it is valuable to acknowledge that saved time and resources are not guaranteed to be allocated toward ethical or sustainability initiatives.

6
Conclusion

Given two curves that define the edges of a desired sealing bead and certain process parameters, an initial trajectory was calculated using a direct geometric approach to maintain an equal sealant height distribution along the two sealant edges. Test cases that included curves with varying widths that turned and went over corners validated the geometric approach. Additionally, the solution was tested on geometries introducing cavities between the edges of the sealant. The initial trajectory was proven effective for a large set of curves. However, the solution has limitations, especially when navigating corners or when the desired sealant turns drastically.

To account for these limitations, a surrogate model was developed to capture more complex behaviors of the sealant while maintaining the speed of validation. The surrogate model showed improved cross-section prediction compared to the glass box model when validated against two validation sets. One introduces kinematic variation, and the other incorporates spatial variation. The surrogate model exhibited promising results on unseen kinematic data but struggled with complex geometries. Introducing more variations of geometries in the training set could result in better performance but requires a larger volume of data. An alternative network architecture could be considered, for instance, a recurrent network to encapsulate the time sequence of the dataset.

In conclusion, two essential blocks for automatic path planning for sealing applications have been developed. The geometric approach was proven efficient for a large set of curves, while the surrogate model showed promising results for further development.

Bibliography

[1] Fraunhofer-Chalmers Centre and Industrial Path Solutions AB. IPS - Industrial Path Solutions, 2025. Available at: `https://www.industrialpathsolutions.com` and `https://www.fcc.chalmers.se/software/ips/`.

[2] Andreas Mark, Robert Bohlin, Daniel Segerdahl, Fredrik Edelvik, and Johan S Carlson. Optimisation of robotised sealing stations in paint shops by process simulation and automatic path planning. *International Journal of Manufacturing Research 5*, 9(1):4–26, 2014.

[3] Daniel Gleeson, Stefan Jakobsson, Raad Salman, Fredrik Ekstedt, Niklas Sandgren, Fredrik Edelvik, Johan S. Carlson, and Bengt Lennartson. Generating optimized trajectories for robotic spray painting. *IEEE Transactions on Automation Science and Engineering*, 19(3):1380–1391, 2022.

[4] Saul Nieto Bastida and Chyi-Yeu Lin. Autonomous trajectory planning for spray painting on complex surfaces based on a point cloud model. *Sensors*, 23(24):9634, 2023.

[5] Julian R. Diaz Posada, Alexander Meissner, Gauthier Hentz, and Nikolai D'Agostino. Machine learning approaches for offline-programming optimization in robotic painting. In *ISR 2020; 52th International Symposium on Robotics*, pages 1–7, 2020.

[6] Wenfeng Zhu, Jie Wang, and Peijian Lin. Numerical analysis and optimal design for new automotive door sealing with variable cross-section. *Finite Elements in Analysis and Design*, 91:115–126, 2014.

[7] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[8] Peng Zhou, Pai Zheng, Jiaming Qi, Chengxi Li, Anqing Duan, Maggie Xu, Victor Wu, and David Navarro-Alarcon. Neural reactive path planning with riemannian motion policies for robotic silicone sealing. *Robotics and Computer-Integrated Manufacturing*, 81:102518, 2023.

[9] Franco Rocha Pereira, Caio Dimitrov Rodrigues, Hugo da Silva e Souza, José Oliveira Cruz Neto, Matheus Chiaramonte Rocha, Gustavo Franco Barbosa, Sidney Bruce Shiki, and Roberto Santos Inoue. Force and vision-based system for robotic sealing monitoring. *The International Journal of Advanced Manufacturing Technology*, 126(1):391–403, 2023.

[10] Perla Maiolino, Richard Woolley, David Branson, Panorios Benardos, Atanas Popov, and Svetan Ratchev. Flexible robot sealant dispensing cell using rgb-d sensor and off-line programming. *Robotics and Computer-Integrated Manufacturing*, 48:188–195, 2017.

[11] ABB Robotics. RobotStudio, 2024. `https://new.abb.com/products/robotics/robotstudio`.

[12] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[13] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[14] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[15] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[16] Harold W Kuhn and Albert W Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2013.

[17] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[21] George Philipp, Dawn Song, and Jaime G Carbonell. The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions. *arXiv preprint arXiv:1712.05577*, 2017.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[25] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA, 2013.

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[28] Henk Kaarle Versteeg. *An introduction to computational fluid dynamics the finite volume method, 2/E*. Pearson Education India, 2007.

[29] Cyril W Hirt and Billy D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[30] Andreas Mark, Robert Rundqvist, and Fredrik Edelvik. Comparison between different immersed boundary conditions for simulation of complex fluid flows. *Fluid dynamics & materials processing*, 7(3):241–258, 2011.

[31] Charles S Peskin. Numerical analysis of blood flow in the heart. *Journal of computational physics*, 25(3):220–252, 1977.

[32] Stephen B Pope. Turbulent flows. *Measurement Science and Technology*, 12(11):2020–2021, 2001.

[33] Robert Byron Bird, Robert Calvin Armstrong, and Ole Hassager. Dynamics of polymeric liquids. vol. 1: Fluid mechanics. 1987.

[34] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[36] TorchVision maintainers and contributors. Torchvision: Pytorch's computer vision library. `https://github.com/pytorch/vision`, 2016.

[37] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[43] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY