POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

Active Preference Learning for Velocity Optimization of Robotized Sealing Task

TESI MAGISTRALE IN MECHANICAL ENGINEERING – INGEGNERIA MECANICCA

AUTHOR: PALANIAPPAN VEERAPPAN

ADVISOR: GIUSEPPE BUCCA

ACADEMIC YEAR: 2021-2022

1. Introduction

In the wake of Industry 5.0 there is increasing requirement of automatizing tasks with taking into consideration sustainability, resilience and human centric service. Since robots are now being used in complex applications and it is not sufficient for them to just reproduce or replicate the task, it is also required for them to learn and adopt so that the task quality can be optimized in a different way (as a requirement of the user). Basically, this is required for the motion planning of the robot since providing a varying velocity reference based on the path feature could bring out expected task quality. Few of such tasks are the sealing and welding applications.

Industrial application to tasks where Robots are used, require optimization to make the task more replicable/comparable as a skilled operator. It is required to execute human centered solutions where the task skill of the operator is transformed to the robot. The Evaluation of the task quality by qualitative comparison between two experiments by the user is way too easier instead of using a reward function for the task performance of the Robot. Preference Optimization is used to satisfy this requirement. The task used to validate this procedure is a Sealing Task where the MAKITA sealing gun along with the sealing cartridge is used as the end effector for the Franka Emika panda robot. For each of the different kind of Path Feature, the optimal velocity is defined by using the Active Preference Learning Algorithm where the user judges the quality of deposition. In this part experimental trials are performed so that the user could give the preference on the deposition quality between the experiments. Then, for general path it is trained to implement this collected data of velocity of different path features. The final results showed the improved deposition quality as preferred by the user, where for certain complex paths it was necessary to implement an acceleration and deceleration constraint.

In this work, a path based velocity planner is employed which is divided into two sections- Preference Optimization Algorithm and the Curvature Analysis and Task Execution part. Taking advantage of the users' expertise on the deposition quality and using it in the Preference Optimization Algorithm, the velocity reference for different path feature is defined. The Preference Optimization Algorithm helps in reducing the number of iterations of physical demonstrations

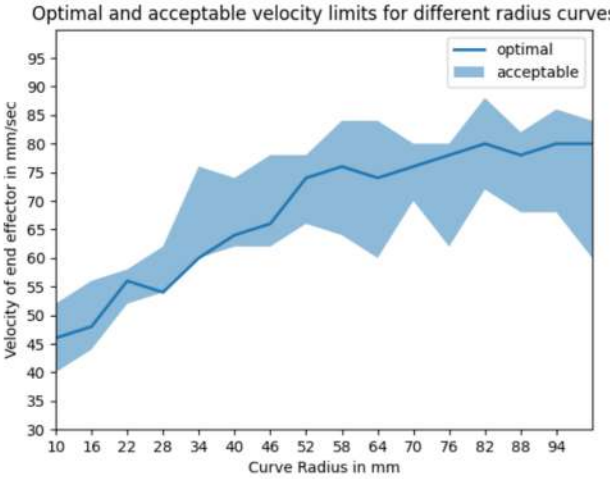


Figure 1: Relationship between the radius of curvature and the Optimum and Acceptable velocity values.

of each new trajectory. In [1], similar sealing task was validated for the velocity planner where the parameters are tuned by using pairwise preference optimization. Instead in the presented work we define the optimal parameter value beforehand for a wide range of path features.

2. The Preference Optimization Algorithm

A) The Algorithm

The Black box Optimization problems are generally accessed or approached based on simulations and evaluation of outputs through physical experiments. In most cases it is much easier if the human makes the assessment, considering into account the users need and their preference, hence making it a more user centered design process. User preferences-based learning techniques have been applied in a variety of domains. In [2], Bayesian optimization algorithm is used for various constrained or preference based optimization. It is proposed, a multi-objective Bayesian optimization algorithm that allows the user to express preference-order constraints on the objectives of the type "objective A is more important than objective B". The surrogate function is modeled using general radial basis functions (RBFs) in a recently developed approach [3] called Active Preference Learning based on Radial Basis Function. The surrogate is built at each iteration by satisfying the constraints, the preferences already expressed by the decision maker at sampled points. The method has been validated, and it is reported to be computationally lighter and closer to the global optimum than Bayesian active preference learning (PBO) in the same number of comparisons.

The algorithm [3] proposes a way to solve an optimization problem where decision makers cannot evaluate the objective function and can only express preferences such as "A is better than B" between the two candidate decision vectors. The algorithm described in this work, repeatedly proposes new comparisons to decision makers based on active learning of Surrogate substitutions of previously sampled decision vectors and pairwise preferences. The surrogate is adjusted using radial basis functions, under the constraint of satisfying, if possible, the preferences expressed by the decision maker over the existing samples. So, at each iteration the Radial Basis Function (RBF) weights were calculated based on the constraint of satisfying the available training set preferences given by the user. The substitution value (Surrogate) is used to propose a new sample of the decision vector to compare with the current best candidate based on two possible criteria: minimizing the combination of the surrogate values and the function of the inverse distance weighting (IDW) to bring a tradeoff between the exploitation (surrogate) and exploration of the decision variables, or maximize a function involving the probability that the new candidate is preferred. By this way the global optimal is reached iteratively based on the User's likelihood.

The initial step of the Active Preference Learning Algorithm is to get the preferences from the user.

Given two possible decision values $x_1, x_2 \in R^n$, consider the preference function $\pi : R^n \times R^n \rightarrow \{-1, 0, 1\}$ defined as:

$$\boldsymbol{\pi}(x_1, x_2) = \begin{cases} -1 & if\ x_1 \text{is better than } x_2 \\ 0 & if\ x_1 \text{is same as } x_2 \\ 1 & if\ x_2 \text{is better than } x_1 \end{cases} \quad (1)$$

where for all $x_1, x_2 \in R^n$ it holds $\pi(x_1, x_1) = 0$ and $\pi(x_1, x_2) = -\boldsymbol{\pi}(x_2, x_1)$. Assume that we have generated N ≥ 2 samples $\boldsymbol{X} = \{x_1, \ldots, x_N\}$ of the decision vector, with $x_i, x_j \in R^n$ such that $x_i \neq x_j, \forall i \neq j,\ i, j = 1, \ldots, N$. For each of the parameters the experiment is performed and is evaluated a preference vector $\boldsymbol{B} = [b_1 \ldots b_M]' \in \{-1,0,1\}^M$

$$b_h = \pi(x_{i(h)}, x_{j(h)}), \quad (2)$$

where M is the number of expressed preferences, $h \in \{1, \ldots, M\}, i(h), j(h) \in \{1, \ldots, N\}, i(h) \neq j(h)$. Note that the element b_h of vector $\boldsymbol{B}$ represents the preference expressed by the user between the experimental performance achieved with parameters $x_{i(h)}$ and $x_{j(h)}$.

The observed preferences are then used to learn a surrogate function $\hat{J}: R^{n_x} \rightarrow R$ of an (unknown) underlying performance index J. The surrogate $\hat{J}$ is parametrized as the following linear combination of Radial Basis Functions (RBFs):

$$\hat{J}(\mathrm{x}) = \sum_{k=1}^{N} \beta_k \varphi\big(\varepsilon d(x, x_i)\big), \quad (3)$$

Where $d: R^{n_x} \times R^{n_x} \rightarrow R$ is the squared Euclidian distance

$$d(x, x_i) = ||x - x_i||_2^2 \quad (4)$$

$\varepsilon > 0$, is a scalar parameter, $\varphi : R \rightarrow R$ is an RBF and $\boldsymbol{\beta} = [\beta_1 \ldots . \beta_N]$ are the unknown coefficients to be computed based on the available users preference. Examples of RBFs are $\varphi(\varepsilon d) = \frac{1}{1+(\varepsilon d)^2}$ (inverse quadratic), and $\varphi(\varepsilon d) = e^{-(\varepsilon d)^2}$ (Gaussian).

The surrogate J must meet the following requirements, according to the preference relation:

$$\begin{aligned} \hat{J}\big(x_{i(h)}\big) &\leq \hat{J}\big(x_{j(h)}\big) - \sigma + \varepsilon_h, \\ &\qquad if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = -1 \\ \hat{J}\big(x_{i(h)}\big) &\geq \hat{J}\big(x_{j(h)}\big) + \sigma - \varepsilon_h, if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = 1 \end{aligned}$$

$$\big|\hat{J}\big(x_{i(h)}\big) - \hat{J}\big(x_{j(h)}\big)\big| \leq \sigma + \varepsilon_h, \quad if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = 0 \quad (5)$$

for all h= 1,…,M, where $\sigma > 0$ is a given tolerance and ε_h are positive slack variables which are used to relax the preference constraints. Based on the above given preference constraints, the coefficient vector $\boldsymbol{\beta}$ describing the surrogate $\hat{J}$ are obtained by solving the Quadratic Programming (QP) problem

$$min_{\beta,\varepsilon} \sum_{h=1}^{M} c_h \varepsilon_h + \frac{\lambda}{2} \sum_{k=1}^{N} \beta_k^2 \quad (6)$$

Figure 2 : The Experimental Setup. The Close up image in the right shows the rack and pinion mechanism- The red part is connected to the EE, while the grey part contains the motorized mechanism

$$s.t. \quad \sum_{k=1}^{N}(\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k) \leq -\sigma + \varepsilon_h, \quad \forall h: b_h = -1$$

$$\sum_{k=1}^{N}(\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k) \geq \sigma - \varepsilon_h, \quad \forall h: b_h = 1$$

$$\left|\sum_{k=1}^{N}(\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k)\right| \leq \sigma + \varepsilon_h, \quad \forall h: b_h = 0$$

$$h=1,\ldots,M$$

where c_h are positive weights, for example, $c_h = 1, \forall\, h = 1, \ldots, M$. The scalar $\lambda > 0$ in the cost function is a regularization parameter which gaurentees uniqueness in the solution of the Quadratic Programming problem. If $\lambda = 0$ problem becomes a linear program (LP), whose solution may not be unique.

Once a surrogate $\hat{f}$ has been calculated, this function can be minimized in order to identify the best velocity parameter x. More specifically, the following steps can be followed: (i) generate a new sample by pure minimization of the estimated surrogate function $\hat{f}$, i.e.,

$$x_{N+1} = argmin\, \hat{f}(x) \quad s.t. \quad x \in \Theta; \qquad (7)$$

(ii) ask the user to express a preference $\boldsymbol{\pi}(x_{N+1}, x_N^*)$, where $x_N^* \epsilon R^{n_x}$ is the best value of the velocity parameter found so far, (iii) update the estimate of $\hat{f}$; and (iv) iterate over N. Such a technique, which only uses the current available data to find the best parameter vector, could easily overlook better performing velocity values. As a result, a function that encourages parameter space exploration should be considered. The exploration function is constructed by using the inverse distance weighting (IDW) function z(x),

$$a(x) = \frac{\hat{f}(x)}{\Delta\hat{f}} - \delta z(x) \qquad (8)$$

Where,

$$\Delta\hat{f} = \max_i\{\hat{f}(x_i)\} - \min_i\{\hat{f}(x_i)\}$$

The next parameter x_{N+1} of the velocity planner to test is computed as the solution of the (non-convex) optimization problem

$$x_{N+1} = arg\, \min_{x\in\Theta} a(x) \qquad (9)$$

B) The Experimental Setup

The preference optimization of the velocity parameter for the deposition task is done using the experimental setup Figure 1. The setup consists of the MAKITA sealing gun for the implementation of the silicon deposition attached to the Franka Emika Panda robot with a flange. The designed flange firmly connects the gun to the robot EE, while also providing a control of the silicone extrusion using a servomotor and a rack and pinion mechanism controlled by Arduino. The robot is controlled exploiting its torque control (control frequency: 1000 Hz).

Algorithm 1 : Velocity Optimization using APL

Data: Upper(u) and lower(l) limit bounds for velocity parameter (x), discretization steps for the velocity parameter, Initial sample point numbers (N_{init}), maximum number of iteration (N_{max}), preferences for the initial sample points.

1: **while** $N<N_{max}$ **do**:

2: Solve (6) with available user preference data and Compute $\hat{f}$ as in (3).

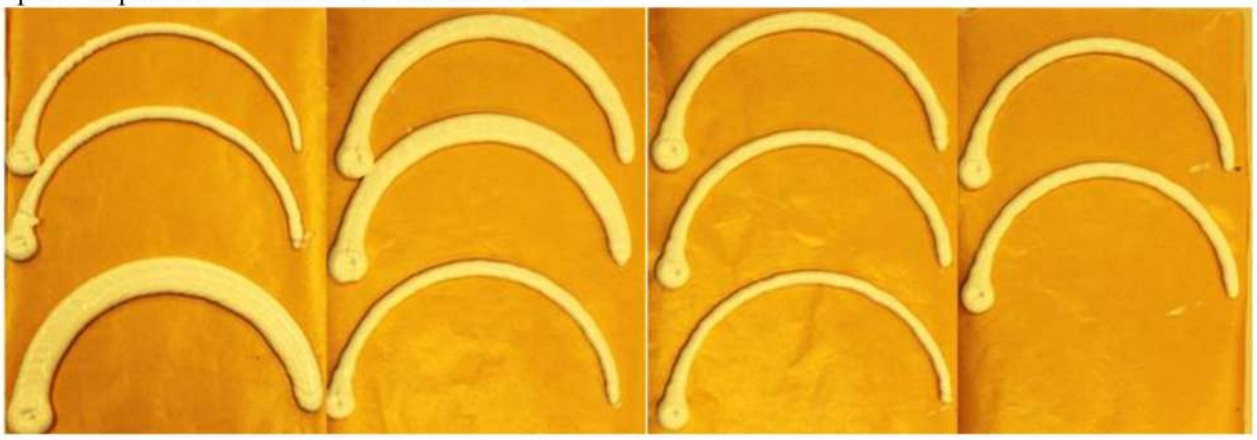


Figure 3: The Iterative Preference Optimization of the Velocity of the deposition task. Velocities(mm/s) in order : 98,92,26,44,32,60,64,68,84,72,74(best)

3: Compute the exploration function (8) and solve the optimization problem x_{N+1} as in (7).
4: Obtain preference from the user with comparison of deposition outcome of x_{N+1} and the previous best x.
5: Update N=N+1
6: **end while**

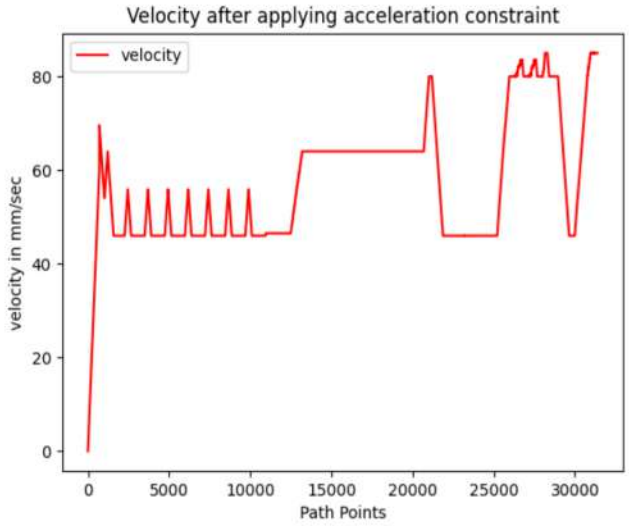
The parameter to be optimized is the velocity of deposition and the upper and lower bound limits for the velocity is set to be 5mm/s to 100mm/s which are discretely considered with a step of 5mm/s. The initial sample points for the optimization are considered to be 4 and the maximum iteration limit for the optimization was set to 20. This procedure of Active Preference Learning Optimization was implemented for radius of curves ranging from 10mm to 100mm with discrete steps of 6mm and also for the straight line. The preference optimization of the velocity parameter for the deposition task for the curve of 64mm is shown in Figure 3, optimum parameter being 74 mm/s. The data of relationship between the optimum velocities corresponding to its radius of curvature values is shown in Figure 2.

3. Curvature Analysis and Task Execution

The radius of curvature is estimated, given the generic path points. It is the reciprocal of the curvature. For any curve with equation y = f(x), the radius of curvature (R) can be given as follows:

$$Radius\ of\ Curvature\ R = \frac{(1+\frac{dy}{dx}^2)^{\frac{3}{2}}}{\left|\frac{d^2y}{dx^2}\right|} \qquad (10)$$

The Final Velocity Planner (Figure 5) works in the following way. For a given generic path (with the x, y points already discretized) it is necessary to find the varying radius of curvature at each discrete point of the path. For each of this radius of curvature values we find the optimum velocity value from the relation data acquired from the Active Preference learning Algorithm Figure 2.
To test the implication of the velocity profile it is required to experimentally validate the algorithm on various test paths/complete trajectories. The output (deposition quality) for the more complicated generic path, was not uniform and desirable as the user had already preferred. This was due to the fact that the robot did not have enough time to increase its velocity to a desirable one and also the transition was not smooth enough to have a proper uniform deposition



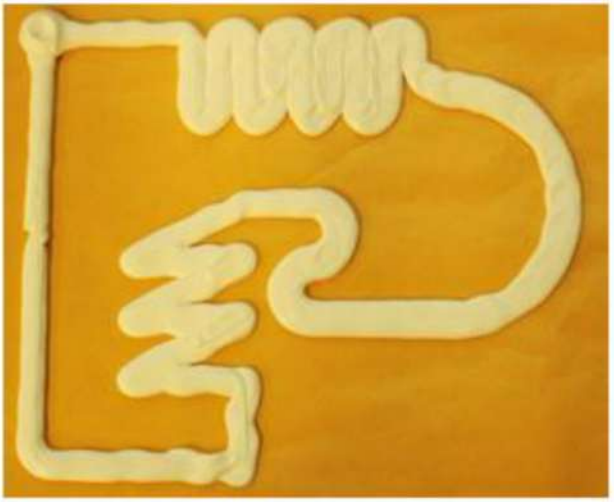


Figure 4: (a) Velocity value along the generic path after constraints implication. (b) Deposition for the generic path with constrained velocity values for a uniform deposition

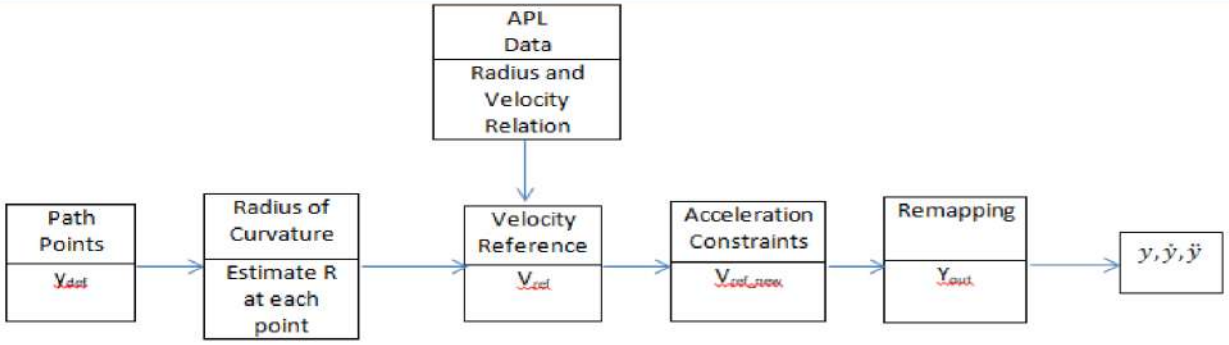


Figure 5: Block Diagram of the Path based Velocity planner for the robot

velocity. Hence the acceleration and deceleration limit was set to 150mm/s². By this way the velocity at each point of the path (v_i) is compared with the next one (v_{i+1}) and is being checked with the acceleration/deceleration limits, and is modified if the limit condition is not satisfied. The complex trajectories, its implementation and output is shown in Figure 4.

4. Conclusion

The presented work proposes a solution to create a varying velocity reference defined from the preferences of the user for different path features. This is required in applications where the task quality varies depending on various path features. The Active Preference Learning Algorithm successfully helps in optimizing the velocity for each curve and this helped in forming the data required for the relation between the velocity and curve radius. For a generic path the radius at each point of the path is calculated and the corresponding velocity reference is implied. The proposed framework along with the acceleration constraints is experimentally validated and the deposition quality is as expected by the user.
This work allows future possibilities of improving the deposition quality by introducing a Model Predictive Control which takes into consideration also the acceptability range velocity values and modify the reference velocity based on the acceleration limits.

Bibliography

1. *Pairwise preferences-based optimization of a path-based velocity planner in robotic sealing tasks.* **L, Roveda, et al., et al.** 4, s.l. : IEEE Robotics and Automation Letters, 2021, Vol. 6.

2. *Multi-objective bayesian optimisation with preferences over objectives.* **M, Abdolshah.** s.l. : Advances in Neural Information Processing Systems, 2019.

3. *Global optimization based on active preference learning with radial basis functions.* **A, Bemporad and D, Piga.** s.l. : Machine Learning, 2020, Vol. 110, pp. 417-448.

POLITECNICO
MILANO 1863

School of Industrial and Information Engineering

ACTIVE PREFERENCE LEARNING FOR VELOCITY OPTIMIZATION OF ROBOTIZED SEALING TASK

This Thesis is submitted in Fulfilment of the requirments for the degree in:

Laurea Magistrale in Mechanical Engineering

By

PALANIAPPAN VEERAPPAN-10649224

Thesis Advisor

Prof. Giuseppe Bucca

Supervisor

Ph.D. Loris Roveda

April 2022

ACKNOWLEDGEMENT

First and foremost, I am immensely grateful to my thesis advisor, Prof. Giuseppe Bucca and my research supervisor, Loris Roveda (Ph.D.) for guiding me with the thesis work with their guidance and continuous support. I would also like to thank Prof. Mario Covarrubius for allowing me to work in the Open lab. Thanks also to Marco Maccarini for helping me get familiar with the optimization algorithm. Also, I would like to extend my gratitude to the university, faculty members and professors for imparting knowledge during this course work. My gratitude extends to the equipment I used during the course work.

I would prefer to dedicate this thesis to my mother Alaguswarna and my late father Veerappan, from whom I sought inspiration and to whom I owe a great deal. Finally, I extend my deepest gratitude to almighty God, for the blessings I am blessed with.

Palaniappan Veerappan,
Lecco, April 2022

ABSTRACT

Industrial robots performing tasks as equivalent to the skilled operators in terms of the task quality requires a complex velocity planning along with a path reference. Evaluating the task quality by creating Robot performance functions is way too tedious than the user's comparative evaluation input between test task qualities. This thesis work is centered on the optimization of a sealing task, which requires an advanced velocity planning to achieve the quality standard of the final products. The final Algorithm allows the Robot to reproduce the specified Path by generating a trajectory planner which executes the task with an optimized velocity reference defined for specific path features with the help of user's preference on quality.

User's comparative evaluation demands are met by the Active Preference Learning algorithm, which optimizes the velocity for different path features. This algorithm takes in the task quality knowledge from the user preference between the test tasks performed with different velocities for a specific path feature (curves of various radius, sharp edges, and straight lines). Then, the algorithm provides an acceptable range of velocities along with an optimal velocity for each curvature. The problem optimized in the present work is a sealing task performed by a Franka Emika Panda robot. The acceptable and optimal velocity is then tracked, considering the acceleration and deceleration limits, to reproduce the task with target quality.

The presented work is aimed to create a control which executes the different path feature with an optimized velocity each, by also taking into account the acceleration limits and physical constraints of the setup which also provides a smoothing effect on the reference.

Keywords: Machine Learning, Autonomous Robotics, Active Preference Learning, Preference Based Optimization, Trajectory Planning

Contents

List of Figures

1. Introduction

Considering the Industry 4.0 scenario, manipulators are taking in charge onerous and repetitive tasks [1] to relieve humans from mental and physical heavy operations [2]. Manufacturers in many industries have long used robots to tackle complex assignments, but robots are evolving for even greater utility. They are becoming more autonomous, flexible, and cooperative. Eventually, they will interact with one another and work safely side by side with humans and learn from them. Although they have been around for some time in industrial production and assembly, robots have for decades had less attention than they receive nowadays, and they are expected to spread to other and less manufactural realms in the near future. The current debate on robotics takes up artificial intelligent approaches in robotics, war robots, and even smart algorithms. The industrial robots are often required to be highly productive, of high quality and adaptability as the result of competitive manufacturing. Since high productivity can be obtained by finding minimum time planning strategies for the manipulators, the execution time of a specific task attains high value for commercial robot manipulators.

In the field of robotics, there is a need for the tracking of a known path for robot manipulators in the Cartesian space with maximum velocity or minimum-time algorithms for their economic benefits. Minimum-time path planning problems for robotic manipulators have widely been studied in the past, especially for industrial applications. Many techniques have been proposed in the past for this problem; however, due to factors such as non-linearity, coupling dynamics, and torque limitation, the task of finding a minimum-time path for robot manipulators is quite complicated. Therefore, an efficient motion planning technique for robot manipulators to move along a pre-defined trajectory is required.

The thesis is based on the optimization of an automatized sealing task, which is executed by a collaborative robotic arm in a human-robot collaborative workspace. This can lead to higher economic benefits, such as cost cutting and higher production rate. The current work introduces a new approach to the Trajectory definition part of the Task Planning. A continuous varying reference execution velocity based on an input path is crucial in sealing applications since the path needs to be defined based

on time characterization also. Parameters such as end effector velocity and deposition flow must be precisely controlled in order to achieve a proper sealant distribution. A manual setting of the reference velocity might be beneficial to transfer the task knowledge from an expert operator to the manipulator, but this is not always feasible because of the effort required to demonstrate each new trajectory. The proposed analysis allows setting the deposition velocity by a control algorithm which is developed based on the Users' preference on the deposition quality of the test tasks. By the way, it is just enough to define the path to be executed and the algorithm provides the reference velocity corresponding to each part of the path, instead of requiring a manual definition by the operator. The algorithm will then be able to approach any generalized path other than the given reference path.

This algorithm containing the trajectory planning can be used as a generalized application for numerous setups such as welding and cutting applications too. Once the algorithm is tuned on the productive setup, it can approach any path, without the need to manually tune the controller every time the reference path is slightly changed.

This introduction aims to give a brief description of the presented thesis work, explaining the topics of each chapter and the workflow from the state of the art on trajectory planning strategies to the experimental validation of the proposed novel approach.

In the following chapter [ch. 2], some basic concepts are remarked. Starting from the background related to the manipulation definition and its constitutive equations, the already existing general trajectory planning algorithms and industrial applications similar to the reference task are then presented. Various studies, investigations and proposals related to the same are explained briefly. In ch. 3 the experimental setup and the working of the robot environment along with the end-effector for the sealing gun is explained. Ch. 4 gives the description of how the Active Preference Learning algorithm along with the Radial Basis Functions works and also the experiments of how the velocity is optimized for each radius of curve. It is also provided the data relation between the radius of curvature and the optimal velocity value along with the acceptable limits. Ch. 5 provides the insight on the final velocity planner algorithm by describing how a general path is analyzed for calculating the radius of curvature at each point on the path thereby defining the velocity at point of the path and its experimental execution.

2. State of the art

The increasing need of automatization for traditional manual processes requires the use of machines and robots that are precisely trained and programmed to execute the planned task. For standard series productions automatized industrial machines were preferred over robots, but in a framework of high variability of applications and tasks, robots are gaining space over the market. The International Federation of Robotics (standard ISO/TR 8373) defines a manipulator as follows [3]:

"A manipulating industrial robot is an automatically controlled, re-programmable, multipurpose manipulator, programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications."

A robotic system is in reality a complex system, functionally represented by multiple subsystems [Figure 2.1]. The essential component of a robot is the mechanical system endowed, in general, with a locomotion apparatus (wheels, crawlers, mechanical legs) and a manipulation apparatus (mechanical arms, end-effectors, artificial hands). The capability to exert an action, both locomotion and manipulation, is provided by an actuation system which animates the mechanical components of the robot. The capability for perception is entrusted to a sensory system which can acquire data on the internal status of the mechanical system (proprioceptive sensors, such as position transducers) as well as on the external status of the environment (exteroceptive sensors, such as force sensors and cameras). The capability for connecting action to perception in an intelligent fashion is provided by a control system which can command the execution of the action in respect to the goals set by a task planning technique, as well as of the constraints imposed by the robot and the environment.

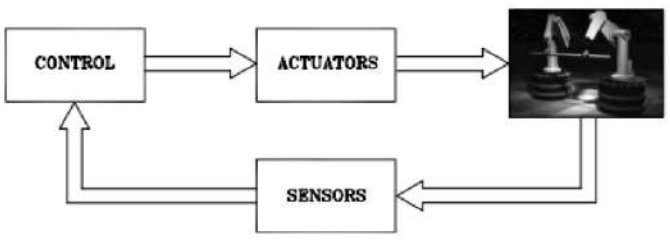


Figure 2.1: Components of a Robotic system

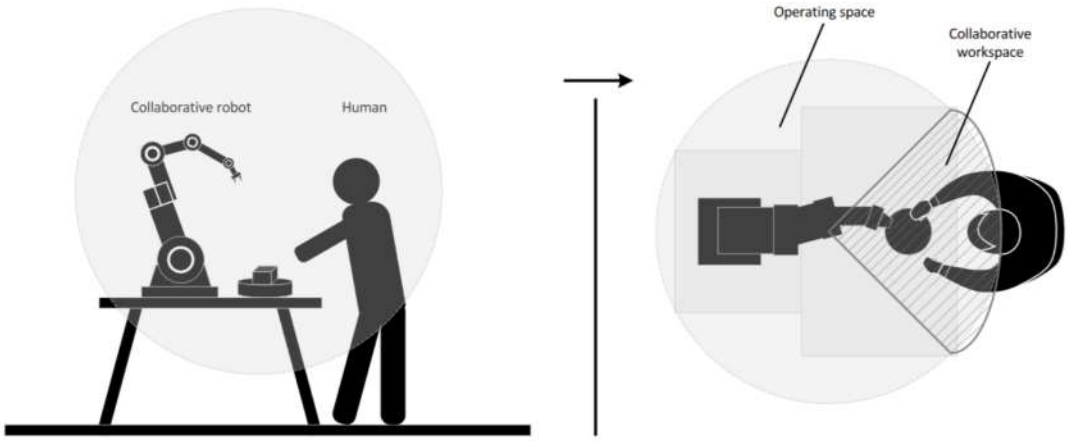


Figure 2.2: Human-Robot collaborative workspace. Source: [4]

In this framework, this study is focused on the fixed place robots, and more specifically on collaborative ones, which are conceived as "mechanical colleagues" of the human, where a relaxed human-robot interaction can be created, and the machine can be an assistant for reaching the final goal [4]. This thesis analyses the use of a collaborative robot for an industrial process, which is the sealing of an aerospace panel. The selection of this type of machine is made with the aim of accomplishing the task while being close to a human collaborator which can enter in the robot operating space and act on the environment [Figure 2.2].

Trajectory execution is a quite common task performed by robots, thanks to their ability to firmly reproduce the provided route. Manipulators must at least convey a movement forms the starting to goal position, and then the selected motion control algorithm will characterize the transition.

Several algorithms have been developed, and they are always referred to as trajectory planning. Before analyzing possible methods, it must be clarified the difference between path and trajectory [5]: “A path denotes the locus of points in the joint space, or in the operational space, the manipulator has to follow in the execution of the assigned motion; a path is then a pure geometric description of motion. On the other hand, a trajectory is a path on which a time law is specified, for instance in terms of velocities and/or acceleration.”

2.1 Robot: mechanical aspects

Robots are complex electro-mechanical structures composed of multiple sub-systems. The mechanical skeleton is typically made of a locomotion apparatus, for mobile robots, and of a manipulation one, typically mechanical arms and end-effector (EE). The motion of this structure is conveyed through actuators, like servomotors, drives and transmission that are controlled by a power electronic which imposes the motion control strategy. Robots are also equipped with sensors that are necessary to perform measures and acquire data both related to the executed movement, both to the external environment, like force sensors or cameras.

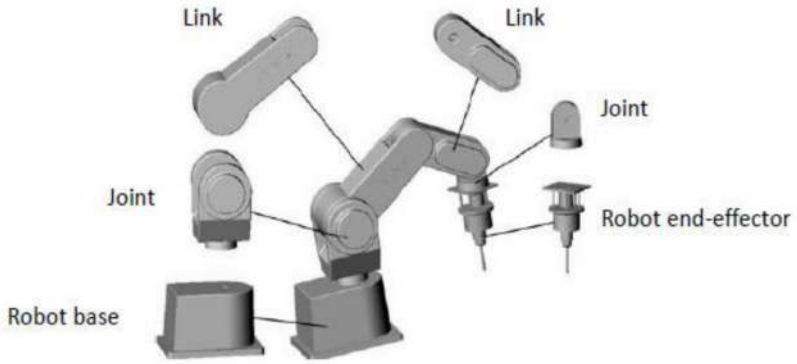


Figure 2.3: Robot structure

The manipulator structure is composed of rigid bodies (links) connected by hinges or articulations, the joints [Figure 2.3]. The arm is the sequence of links that ensures mobility, while the EE is the component that performs the task. All movements are conveyed through the joints, which can perform:

- Revolute motion, a relative rotation between the two links;
- Prismatic motion, a relative translation between the two links.

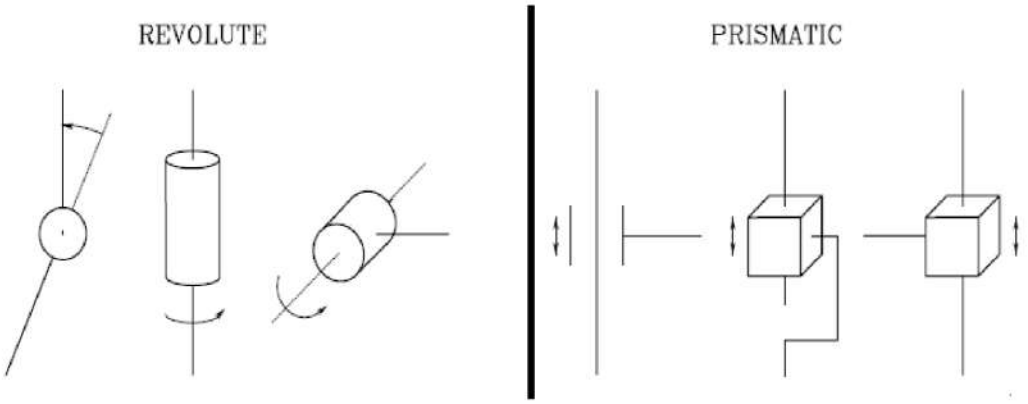


Figure 2.4: Conventional representation of joints. Source: [5]

Those two types of joints are graphically schematized as in Figure 2.4.

The minimum number of real values to specify the position of all the joints of a robot is called the robot's degrees of freedom (DOF). The position of each point on a robot with n degrees of freedom is fully specified by its n-dimensional configuration q $\in$ C, where C $\subseteq R^n$ is called the configuration space, or joint space. The structure of the robot influences the complexity of the system. For open-kinematic-chain robots the number of joints corresponds to the number of degrees of freedom, while for closed-kinematic-chain the number of DOF differs from the joints one, due to closed-loop physical constraints [5]. In the scope of this thesis, the former structure will be selected, because it is in accordance with the given setup [ch. 3]. These two types of configurations are graphically schematized as in Figure 2.5.

The selection of the manipulator must be related to the task that has to be computed. If a positioning and orientation of the EE is required in three-dimensional space (3D), then 6 DOF are required, since 3 of them characterize the position (x,y,z) while the other 3 characterize the orientation with respect to the reference frame (φ,ϑ,ψ).

Common applications in 2D space can be achieved either with simple robots as Cartesian or Selective Compliance Assembly Robot Arm (SCARA) ones, that perform movements on parallel planes, or with more complex manipulators, as anthropomorphic ones, that typically have more DOF, and sometimes are even redundant, as they dispose a number of DOF higher than the one needed to characterize a position in space (i.e. manipulator of 7 DOF for a 3D movement).

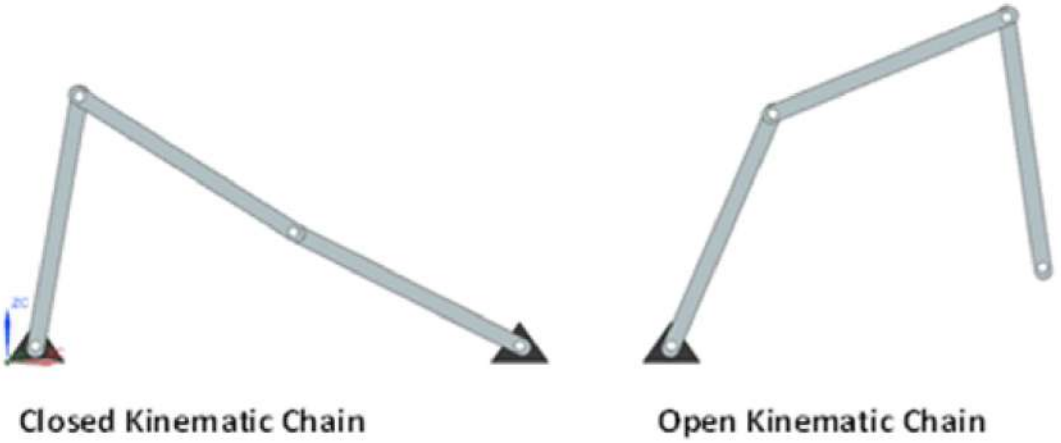
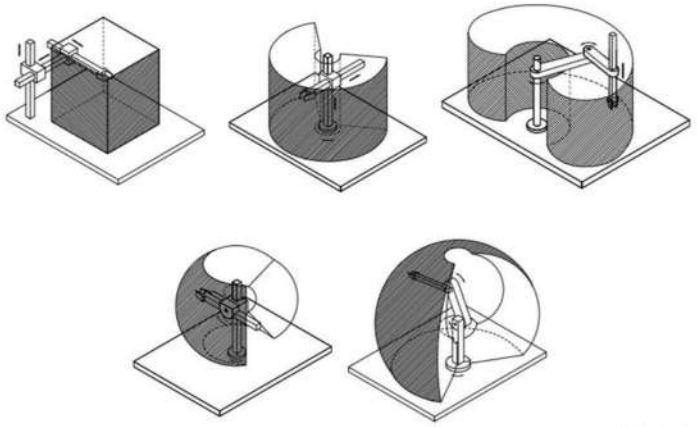


Figure 2.5 : Comparison of the two kinematic chain structures of manipulators

Thanks to these possible multiple rotations of arms, these latter robots can execute more complex movements and can cover larger task areas. In this framework the concept of workspace has to be introduced as the locus of points that can be reached by the manipulator Tool Center Point (TCP) for executing a task. A graphical representation of the different workspaces related to different robots is reported in Figure 2.6.

Very simple robots, as the Cartesian one, reach univocally each point, while others with more complex structure (SCARA, Anthropomorphic, etc.) can display multiple possible configurations. In case of anthropomorphic robots, this extended motion is directly related to the mechanical structure, which is typically a sequence of links and pure revolution joints, which occupy a reduced space while permitting extensive motion in 3D. A scheme of a 3 DOF anthropomorphic manipulator is reported in [Figure 2.7], where each joint is characterized by its reference frame (x_i; y_i; z_i) and the admitted movement which in this case is a pure rotation ϑ. The EE frame is: (x_3; y_3; z_3). Knowing the position of the TCP is fundamental to describe its motion in the space, and so to define a proper trajectory planning algorithm. To evaluate this position, there are two possible approaches: direct kinematics and inverse kinematics.



(a) Cartesian manipulator and its workspace, **(b)** Cylindrical manipulator and its workspace, **(c)** SCARA manipulator and its workspace, **(d)** Spherical manipulator and its workspace, **(e)** Anthropomorphic manipulator and its workspace.

Figure 2.6 : Comparison of different workspaces of manipulators. Source: [5]

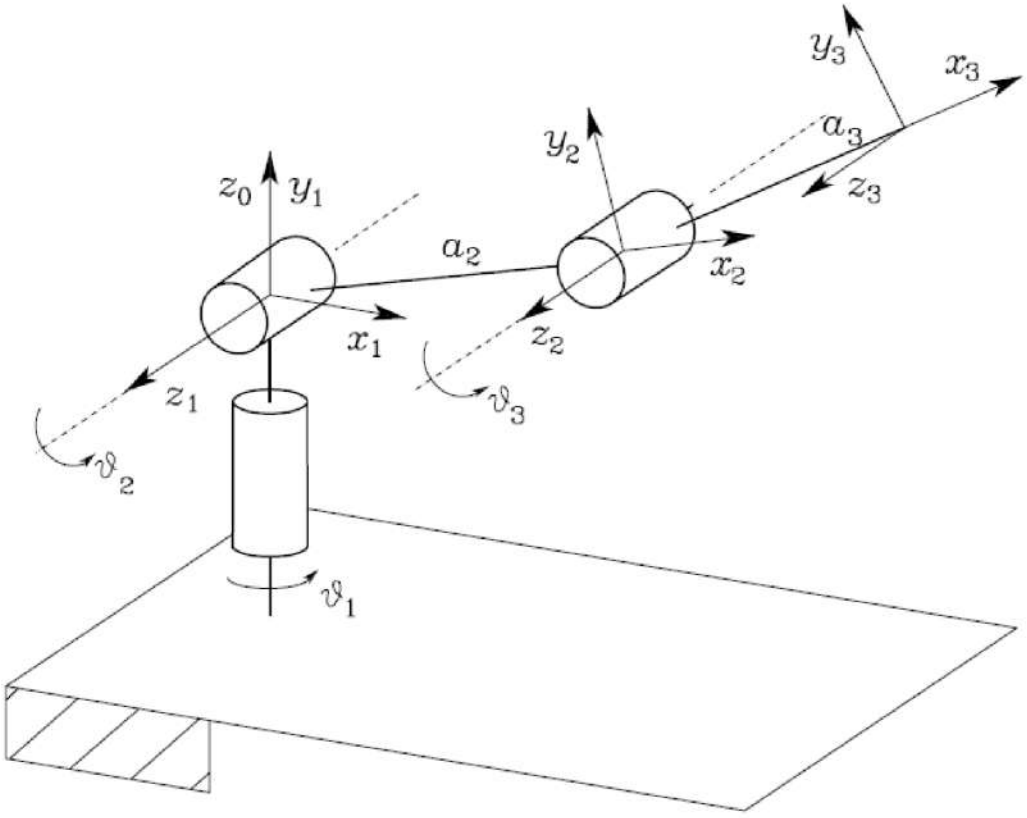


Figure 2.7 : Anthropomorphic arm representation. Source: [5]

2.1.1 Direct and Inverse Kinematics

Finding the pose of the EE is fundamental to describe the desired tasks that must be reproduced by the manipulator. In Direct Kinematic (DK), the EE position is computed as a function of joint variables.

The posture of the EE can be described through a vector $\mathbf{x} \in \Re^m$, which collects the positions, $\mathbf{p_e}$ and orientations, $\boldsymbol{\varphi_e}$ of the robot hand:

$$\mathbf{x_e} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{bmatrix} = \begin{bmatrix} p_e \\ \varphi_e \end{bmatrix} \qquad (2.1)$$

while joint positions, $\mathbf{q} \in \Re^n$, which are represented with rotations and translations according to the type of joint, are:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ . \\ . \\ . \\ q_n \end{bmatrix} \qquad (2.2)$$

where $m \leq n$ so that all points are reachable. Direct Kinematic (DK) equations describe the position of the hand gripper as function of the joint ones:

$$\mathbf{x} = k(\mathbf{q}) \qquad (2.3)$$

In Inverse Kinematic (IK) instead, the target is commonly to define the joint positions while knowing the EE ones. This permits to define the joints movements, and then compute the torques and force actions to drive motion. Mathematically, these variables $\mathbf{q}$ are evaluated through an inverse relationship with respect to DK:

$$\boldsymbol{q} = k^{-1}(\boldsymbol{x}) \qquad (2.4)$$

The definition of these equations is not trivial when the number of DOF increases, because the relationships are mostly non-linear and the evaluation of the EE orientations $\boldsymbol{\varphi_e}$ may require the use of rotation matrices.

2.1.2 Rotation Matrices, Euler Angles and Quaternions

A rotation matrix is a compact notation used to describe the orientation of a point with respect to a selected reference frame [Figure 2.8].

The base frame is (x, y, z), while the object frame is (x'; y'; z'). To describe the orientation of the body, the versors describing its position must be calculated referring to the base frame:

$$\begin{cases} \hat{x}' = x'_x\hat{x} + x'_y\hat{y} + x'_z\hat{z} \\ \hat{y}' = y'_x\hat{x} + y'_y\hat{y} + y'_z\hat{z} \\ \hat{z}' = z'_x\hat{x} + z'_y\hat{y} + z'_z\hat{z} \end{cases} \qquad (2.5)$$

where a'_a terms represent the direction cosines with respect to each axis.

By switching to a matrix compact formulation, it is possible to obtain the rotation matrix:

$$\boldsymbol{R} = [\hat{x}' \quad \hat{y}' \quad \hat{z}'] = \begin{bmatrix} x'_x & x'_y & x'_z \\ y'_x & y'_y & y'_z \\ z'_x & z'_y & z'_z \end{bmatrix} \qquad (2.6)$$

The terms of the matrix represent the direction cosines needed to define the orientation of the new reference frame with respect to the base one. When multiple rotations are done, it is better to evaluate a rotation matrix for each rotation angle and then compute the overall one.

The definition of the 3 rotational coordinates $\boldsymbol{\varphi_e}$ that describe the orientation of a body in space is done though Euler angles. Euler angles are a set of three angles:

$$\boldsymbol{\varphi_e} = [\varphi, \vartheta, \psi] \qquad (2.7)$$

which represent the rotation of a reference frame with respect to a base one. They are commonly used to compute the direction cosines of the rotation matrices.

The definition of the Euler angles is not trivial, because 12 different sets of angles can be used to describe an orientation. These sets depend on the sequence of rotations performed to reach the final position. To better explain this concept, refer to [Figure 2.9] and to the following calculus [6].

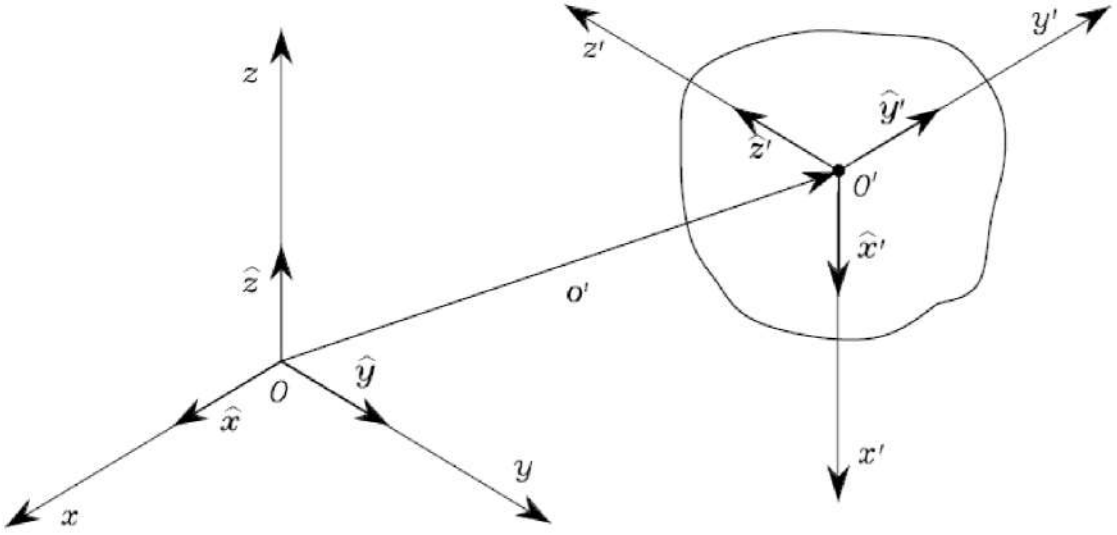


Figure 2.8 : Orientation and position of a rigid body in space. Source: [5]

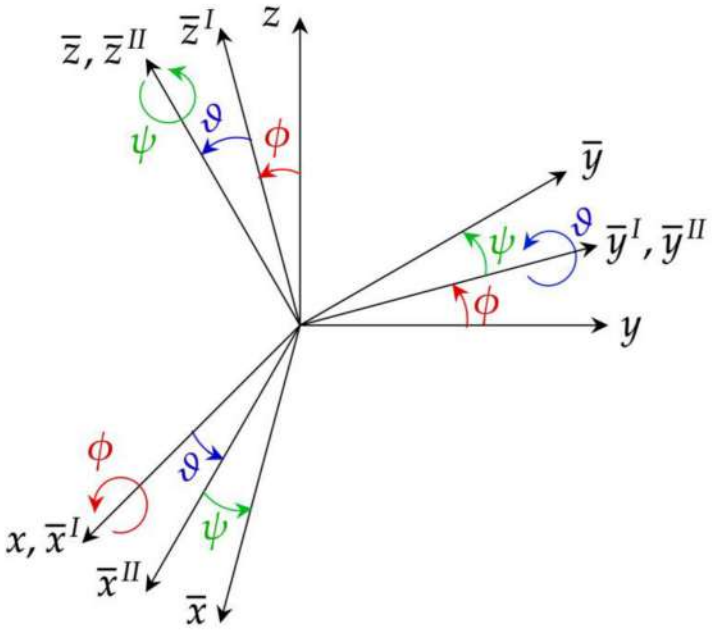


Figure 2.9 : Reference Frame rotation through Euler Angles. Source: [5]

Considering to perform a rotation from the reference frame **x** = (x; y; z) to the final reference one $\bar{\mathbf{x}}$ = ($\bar{x}$; $\bar{y}$; $\bar{z}$), it is possible to perform three separate rotations using

three rotational matrices. Considering the case showed in Figure 2.9, the procedure is:

- Perform a rotation - around x-axis, $x = X\bar{x}^{I}$:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{bmatrix} \begin{bmatrix} \bar{x}^{I} \\ \bar{y}^{I} \\ \bar{z}^{I} \end{bmatrix} \qquad (2.8)$$

- Perform a rotation ϑ around y-axis, $\bar{x}^{I} = Y\bar{x}^{II}$:

$$\begin{bmatrix} \bar{x}^{I} \\ \bar{y}^{I} \\ \bar{z}^{I} \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & 0 & \sin(\vartheta) \\ 0 & 1 & 0 \\ -\sin(\vartheta) & 0 & \cos(\vartheta) \end{bmatrix} \begin{bmatrix} x^{II} \\ y^{II} \\ z^{II} \end{bmatrix} \qquad (2.9)$$

- Perform a rotation ψ around z-axis, $\bar{x}^{II} = Z\bar{x}$:

$$\begin{bmatrix} \bar{x}^{II} \\ \bar{y}^{II} \\ \bar{z}^{II} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix} \qquad (2.10)$$

By combining the three equations, it is possible to obtain the final total matrix required to perform the framework rotation:

$$x = (X\ Y\ Z)\bar{x} = \boldsymbol{M}\bar{x} \qquad (2.11)$$

Where M(φ, ϑ, ψ) is the rotation matrix function of the three Euler angles.

The selection of the sequence of rotations defines the M matrix. Considering each possible combination, there will be 12 possible sets of Euler angles. In aeronautical applications, the typical sequence is ZYX which means to perform a first rotation around z-axis, then a rotation around y-axis and the last rotation around x-axis, the angles are typically called: roll, pitch and yaw.

Quaternion formulation can be selected alternatively. This representation uses four parameters to univocally select an orientation for a point in space. They are formally defined as the vector [w; x; y; z] [7]:

$$q = w + xi + yj + zk \qquad (2.12)$$

where w, x, y, z $\in \Re$, and the norm of the quaternion is 1:

$$w^2 + x^2 + y^2 + z^2 = 1 \qquad (2.13)$$

According to the definition, it is possible to call $q_w = w$ the scalar part of the quaternion, and $\mathbf{q_v} = (x, y, z)$ the vectorial one.

Robot controllers often require the use of quaternions to define orientations. Considering the Euler sequence ZYX, where the angles in sequence are ϑ for z-axis, ψ for y-axis and φ for x-axis, the quaternion is obtained as:

$$\begin{cases} w = \cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\ x = \sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\ y = \cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\psi}{2}\right) \\ z = \cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\vartheta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\vartheta}{2}\right)\cos\left(\frac{\psi}{2}\right) \end{cases} \quad (2.14)$$

Mind that for each sequence of Euler angles, the law that links them to quaternions is different. All the 12 equations are reported in [6].

2.2 Trajectory planning

Task execution for manipulators is commonly controlled through motion planning algorithms, called trajectory planning. This topic is widely studied in robotics because the selection of the proper planning strategy can substantially improve the execution of the task [8]. Planning consists in generating a time-sequence of points or positions that must be executed either by the EE or by the joints.

According to the provided path description, these algorithms can set a:

- Point-to-point motion or position control, where only starting and goal positions are provided, and the objective is simply to reach the end position regardless of the followed trajectory;
- Path motion if the robot EE motion is specified with a defined sequence of points parametrized in time.

In Figure 2.10 the two different approaches are reported with a graphical representation of the target motion. In the first case, the control strategy simply imposes the three points to be reached and then the robot controller chooses the proper path that must be performed; in the latter case the input is exactly the red 2D trajectory that must be followed.

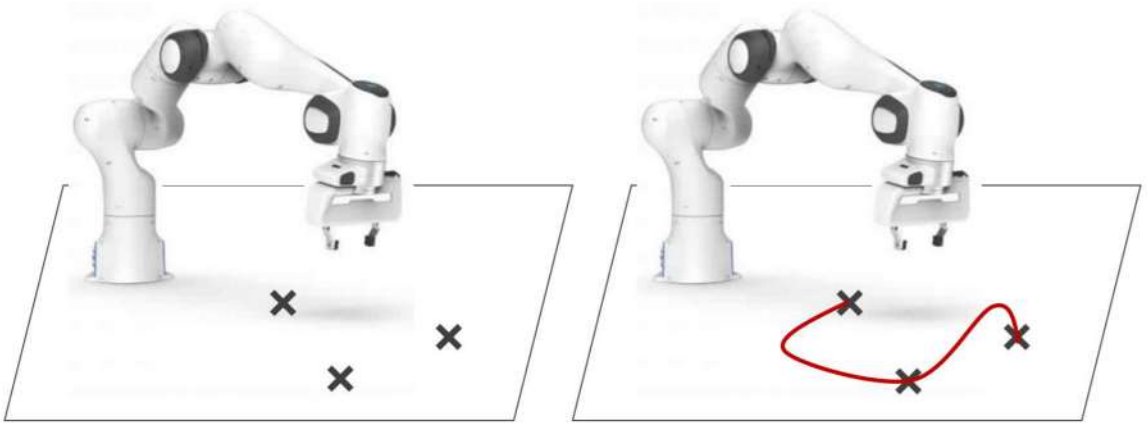


Figure 2.10 : Position Control on the left, where only three intermediate points are given VS Path Control on right, where all the path is provided (red).

The goal of trajectory planning control strategies is to generate a reference trajectory for the robot controller, which is a time-sequence of positions, velocities and accelerations that map the desired trajectory. Planning can be done either in joint space or in task one (Cartesian space), according to the movement that must be executed. For applications like welding, sealing or laser cutting, the joints of the robot can move almost freely in the workspace, with limitations only for singularity positions and physical spatial constraints, while proper planning for the TCP is imposed in Cartesian space.

Trajectory planning algorithms can be schematized considering the planning steps and input parameters of the activity. In case of task space tracking, the operational space trajectory must be converted into a joint motion signal through the use of IK such that the robot motion control system will generate the torques and forces needed to activate motors [Figure 2.11].

Instead, joint space planning results in a faster motion because directly creates the signal $q_{out}(t)$ that must be fed into motion controller but it is not an optimal solution for precise path tracking since it describes a joint motion.

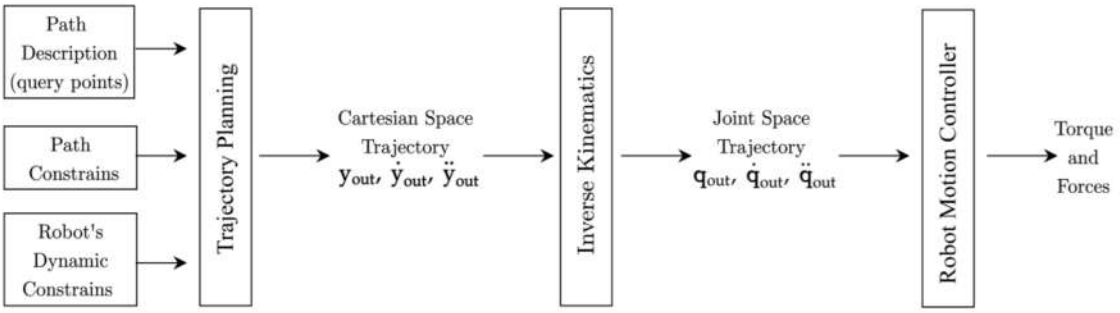


Figure 2.11 : Steps of trajectory (planning activity)

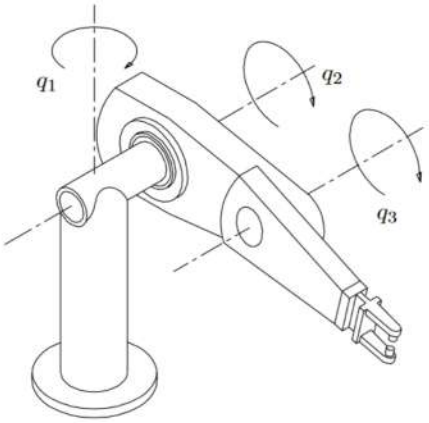


Figure 2.12 : Joints on a manipulator. Source [3]

2.2.1 Joint space control

Joint space planning maps a set of positions for the TCP by controlling joint ones. This solution is preferable when dealing with motion near possible manipulator singularities, or when a certain joint movement must be described. Example applications can be found in point-to-point motion or in motion through a sequence of points, where joints starting, intermediate and final positions are specified and the planning algorithm controls the interpolation and time execution in joint coordinates between one point and another.

Joint space motion is characterized by computation of joint positions of the robot, traditionally referred to with the variable q. A simple representation is proposed in Figure 2.12 where it is possible to observe a 3 DOF manipulator with pure rotational joints.

Considering a more general open-kinematic-chain manipulator with n-DOF, characterized by n joint coordinates: $\mathbf{q} \in \Re^n$.

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \tag{2.15}$$

where each coordinate is related to the movement generated by the physical link, revolute joints **q** will be represented by a rotation, while prismatic joints by translations.

Joint space trajectory planning algorithms interpolate the **q**(t) positions while respecting the imposed constraints, which can be physical limits of the manipulator (velocities, accelerations) or objective constraints imposed by the chosen interpolating function (i.e. Jerk minimization [9], [10] Time minimization etc.).

Point-to-Point Motion

The simplest planning algorithm is the point-to-point motion. The joints have to move from a starting configuration $\mathbf{q}(t_o) = q_o$ to a final one $\mathbf{q}(t_f) = q_f$ in a certain time interval t_f while respecting the imposed constraints.

Cubic polynomial interpolation algorithm is used to minimize the energy dissipated on each motor. The joint movement is obtained through a third-order polynomial function, such that the velocity profile is represented by a parabola:

$$\begin{cases} \boldsymbol{q}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ \dot{\boldsymbol{q}}(t) = a_1 + 2a_2 t + 3a_3 t^2 \end{cases} \tag{2.16}$$

to characterize the motion law, 4 constraints must be imposed: initial and final joint position and velocity. Higher order polynomials, such as fifth order polynomial are commonly used to impose also the acceleration limits.

Another common algorithm for interpolation is the trapezoidal velocity profile. This method is preferred when the motion must be executed at an imposed cruise velocity level, with a specific acceleration value during only starting and final stages [Figure 2.13].

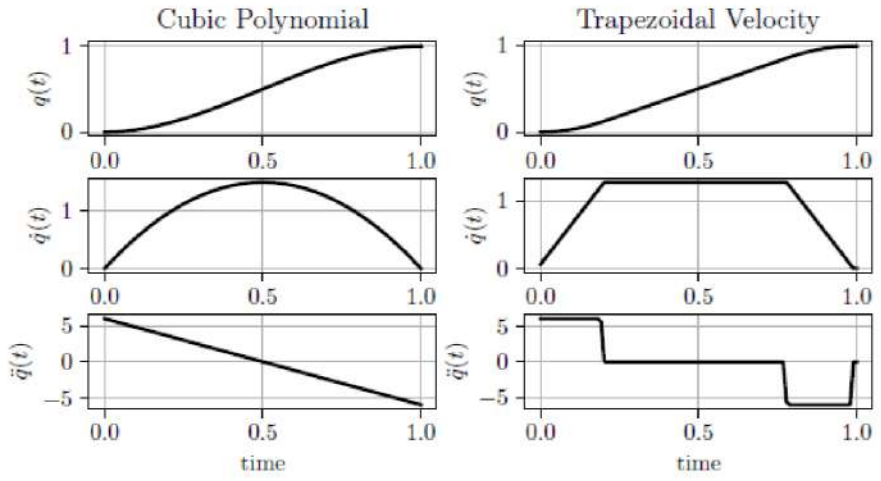


Figure 2.13 : Comparison between cubic polynomial interpolation and trapezoidal velocity profile with null velocity at starting and ending conditions, a_{max} = 6, t = 1 and q_f = 1

Trapezoidal velocity profile is defined as:

- Acceleration:

$$\ddot{\boldsymbol{q}}(t) = \begin{cases} \ddot{q}^{+}_{max}, & t \in (t_0, t_1) \\ 0, & t \in (t_1, t_2) \\ -\ddot{q}^{-}_{max}, & t \in (t_2, t_f) \end{cases} \tag{2.17}$$

- Velocity:

$$\dot{\boldsymbol{q}}(t) = \begin{cases} \ddot{q}^{+}_{max}(t - t_0) + \dot{q}_0, & t \in (t_0, t_1) \\ \dot{q}_{max}, & t \in (t_1, t_2) \\ -\ddot{q}^{-}_{max}(t - t_2) + \dot{q}_{max}, & t \in (t_2, t_f) \end{cases} \tag{2.18}$$

- Position:

$$\boldsymbol{q}(t) = \begin{cases} \frac{1}{2}\ddot{q}^{+}_{max}(t - t_0)^2 + \dot{q}_0(t - t_0) + q_0, & t \in (t_0, t_1) \\ \dot{q}_{max}\,(t - t_1) + q_1, & t \in (t_1, t_2) \\ -\frac{1}{2}\ddot{q}^{-}_{max}(t - t_2)^2 + \dot{q}_{\max}(t - t_2) + q_2, & t \in (t_2, t_f) \end{cases} \tag{2.19}$$

where at t_0 the motion is starting, at t_1 the acceleration step has ended, at t_2 starts deceleration and at t_f motion ends. The velocity of execution is identified by $\dot{q}_{max}$.

Motion through a Sequence of Points

More complex tasks require a path characterization with more than just two points. Intermediate points or sequences are necessary to describe with higher precision the motion in space, indeed this planning approach is used whenever there are obstacles to be avoided or when a particular trajectory must be performed. The intermediate points between the starting and goal position are called via-points. The higher the number of via-points, the higher the definition of the motion (i.e. curve characterization requires more points with respect to a straight line motion).
Provided a sequence of N points that must be reached by the manipulator at certain time instants, it is necessary to define an interpolating function Π, to provide a continuous motion between the via-points. Mathematically it must be used a $(N-1)$-order polynomial, but while N increases, several problems may arise:

- Initial and final velocities cannot be assigned;
- High order polynomials show unnatural oscillatory behaviors (Runge's phenomenon);
- The solution of the system for defining polynomials coefficients results computational expensive;
- The polynomial definition strictly depends on the via-points, if one should be changed, all the coefficients must be re-computed.

Moreover, whenever a dense sequence of points is provided, such as to describe a proper path definition, the via-points are very close one to another, resulting in a quasi-continuous trajectory. Especially in this situation, selecting a $(N-1)$-order polynomial is not advised. The best solution is to use $(N-1)$ polynomials of lower order to describe a motion between one point and the following one: Π_i, $i \in [1; N-1]$, [Figure 2.14].

Cubic polynomials are advised for maintaining velocity continuity between one point and the following one, but they require a velocity characterization at each via-point. Also, lower order polynomials can be used, but they generate a non-smooth transition between one interval movement and another, and they lose the continuity in velocity.

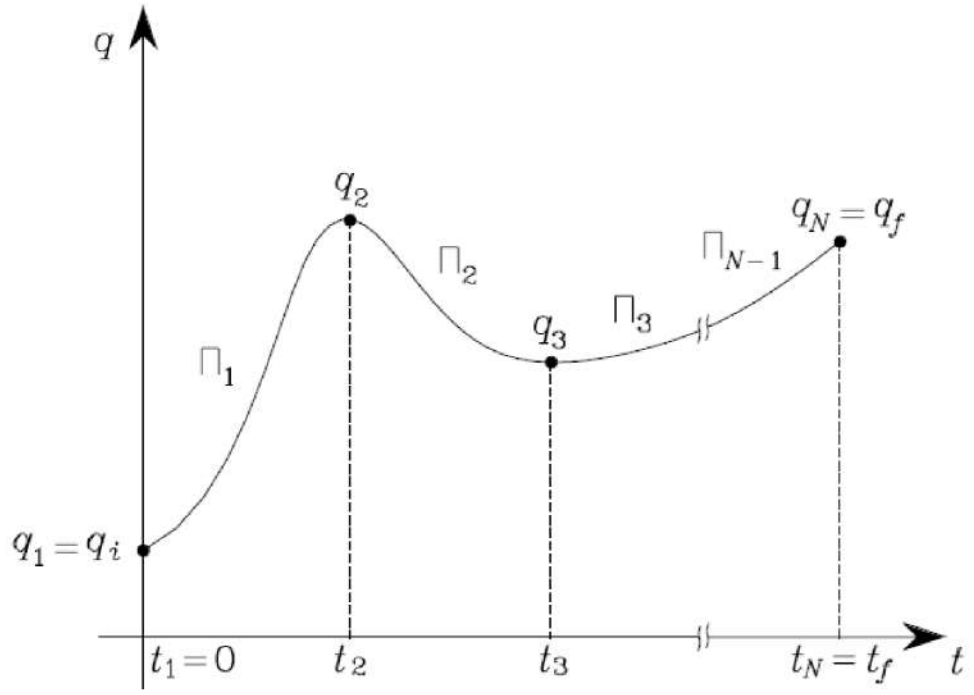


Figure 2.14 : Via-points interpolation with ($N - 1$) polynomials. Source: [5]

Whenever via-points are very dense, the choice of the interpolating polynomial results less critical from the displacement point of view, because the movement to be executed in between is not large enough to show a characteristic difference in results [Figure 2.15], but the velocity profile will still show a non-smooth behavior that will result in a non-optimal movement of the joints. Non continuous transitions can be the cause of vibrations during the execution of the task, especially if it requires a good precision. In [11] it is studied a trajectory generation algorithm that permits to achieve smooth transitions while controlling also joint positions.

2.2.2 Task space control

In task space planning the motion of the manipulator is mapped by controlling the motion of the EE. This control strategy is preferred whenever a precise movement of the TCP is requested, i.e. in precise geometry following applications, like welding and sealing.

Planning in operational space can be done with two main approaches:

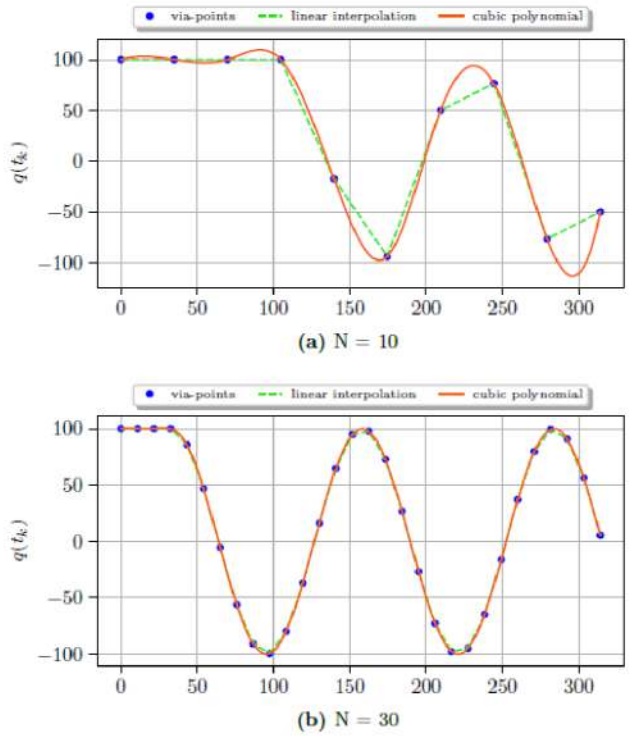


Figure 2.15 : Comparison between different interpolation methods when the number of via-points increases [N = 10 and N = 30]

- By performing interpolations between numerous via-points with the same method presented for the joint-space motion, typically using linear micro-interpolations with a point density equal to the robot frequency;
- Through generation of analytical motion primitives.

Interpolation has been exploited in [chap 2.2.1]. In the same way it is possible to describe several via-points in the Cartesian space, such that the movement of the EE is directly controlled.

Path-Velocity Parameterization

Path primitives are analytical functions used to precisely and analytically describe the motion of the EE. These functions characterize features of the path while providing a time law to determine the motion execution. According to the formal definition of path primitives, it is possible to refer to them through the equation:

$$\mathbf{p} = \mathbf{f}(s) \qquad (2.20)$$

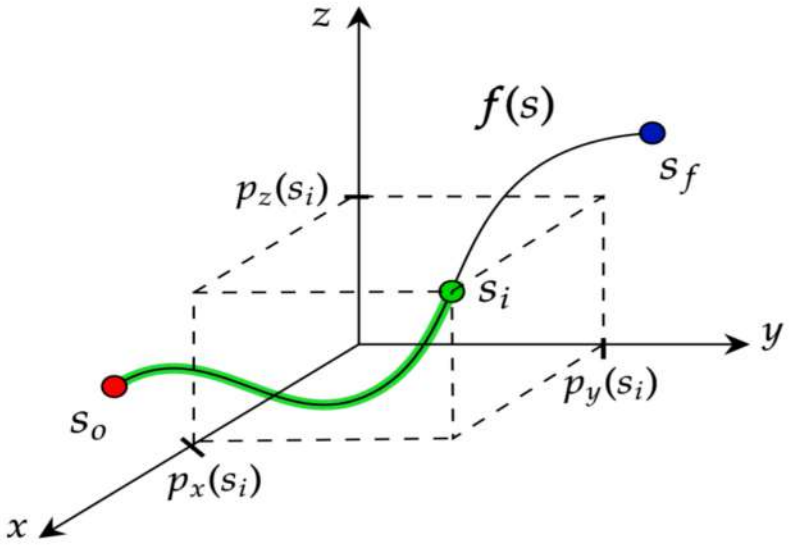


Figure 2.16 : Completion of the analytical function f(s), in green the percentage of path already executed, s_i is the actual position along the path.

where **p** is a three-dimensional vector that describes the space position, **f** is a continuous vector function parametrized over the variable s, the natural coordinate, which represents the percentage of completion over all the path. If s increases, the execution along the cartesian path described by the function **f** proceeds [Figure 2.16].

The characterization in time of the motion execution is provided by the mathematical law that links the natural coordinate with time: s = s(t). Typically, when $t = 0 \rightarrow s = 0$, while when $s = s_f$ = path length the overall time requested to perform the entire path is reached, $t = t_f$.

The execution velocity along the path depends on the time parametrization of the natural coordinate.

$$\dot{\mathbf{p}} = \frac{d\boldsymbol{p}}{dt} = \frac{d\mathbf{f}(s(t))}{dt} = \frac{d\mathbf{f}}{ds}\frac{ds}{dt} \qquad (2.21)$$

where $\frac{ds}{dt}$ represents the magnitude of the velocity vector in correspondence of the selected time instant, while $\frac{d\mathbf{f}}{ds}$ is the tangent vector to the path.

Traditionally a motion primitive provides a mathematical description of a precise motion, i.e. they are commonly used on humanoid robots to describe basic

movements, such as walking or catching; they are then joined together to obtain a complex outcome [12]. A motion primitive describes the execution of a proper movement with related constraints, both in space and velocity. In [13] motion primitives are exploited to analyze the possible outcomes while combining three basic primitives. A proper analytical parametrization of each path is requested, and the overall motion is obtained as a sequence of these sub-paths.

With the proper formulation, even one path primitive can be used to obtain a complex motion, without the need to subdivide the path in smaller intervals. Obviously, this approach can be selected only in certain situations, not for all the numerous cases of task planning. This specific analysis will be deepened in the next chapters of the thesis, since it recalls the selected method for planning the task execution.

2.3 Robotic Industrial Processes

Now that the basic concepts of trajectory planning have been explained, it is possible to analyse several methods already implemented in industrial applications that require a precise movement of the EE.

Sealing procedures include numerous types of applications, from silicone sealing to welding and gluing. Traditionally those activity are performed by a human operator, but automatization is requiring novel approaches with the use of robot manipulators. Welding procedure can be the main reference application for the sealing tasks, as it has already several automatized processes, but it is also possible to analyse other similar applications, like Fused Deposition Modeling (FDM) 3D printing or laser cutting.

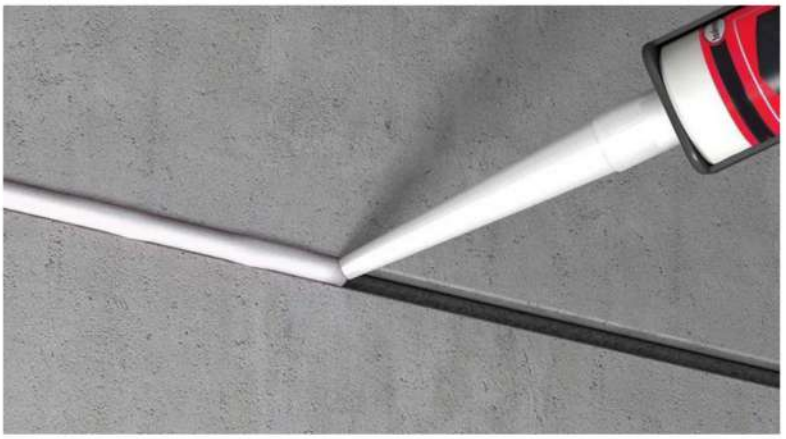


Figure 2. 17 : Human performing a silicone deposition

2.3.1 Sealing Applications

Sealing is a traditional manual operation that requires the deposition of a ribbon of sealant material that will permanently join separate pieces [Figure 2. 17]. This operation requires a precise path tracking to deposit the material in the correct zones, and a proper characterization of the motion to generate a correct deposition.

Skillful manual operators can achieve a good quality level by slowly performing the task; robots can achieve a higher quality deposition in a faster way.

The main problems of silicone deposition are related to:

- Velocity of deposition;
- Caulking gun orientation during deposition.

The first problem for human operators is partially solved through a trial and error movement; for robots it is the object of the planning algorithm, that must characterize and optimize the execution velocity.

The gun orientation changes for the human and robotic case, because the two cases have different features and control strategies. A human cannot have the precision in position, orientation and velocity of a manipulator, so it must perform a deposition in a way that both the path and the sealant material ribbon can be semi-optimal. To achieve that, a proper deposition of the sealant material can be obtained sliding the gun nozzle on the path to be executed while keeping the gun with a certain inclination, such that the nozzle cannot be obstructed and the material can properly flow.

Instead, robots do not need to slide on the path since an accurate positional controller permits to perform a sealing application moving above the deposition plane with a fixed offset. Since the offset prevents the nozzle to be clogged, the caulking gun can be maintained vertical during all the execution. This way, the deposition can be done similarly to a FDM 3D printing process, with a constant and precise vertical distance from the plane and a completely perpendicular orientation.

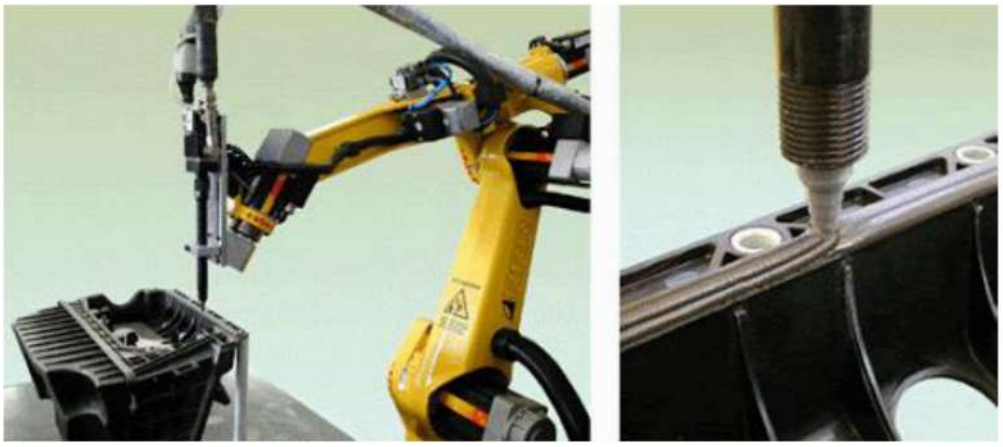


Figure 2.18 : Gluing application. Source: http://www.rrrobotica.it/incolla.htm

Figure 2.19 : Innovative sealant deposition. Source: https://www.durr.com/en/products/sealing-gluing-technology/sealing-gluing-robots

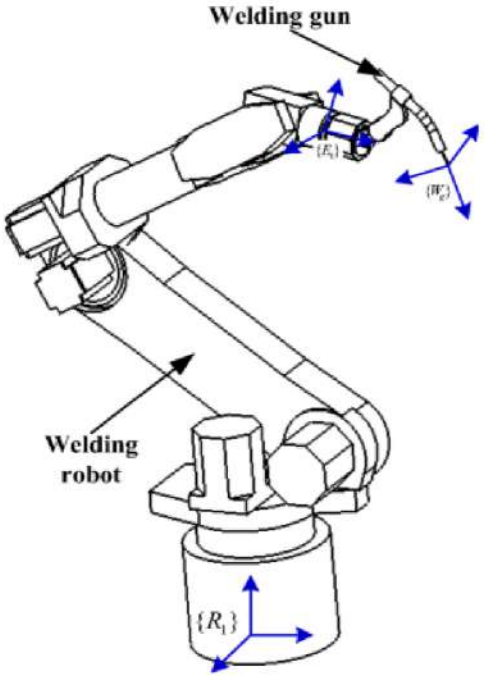


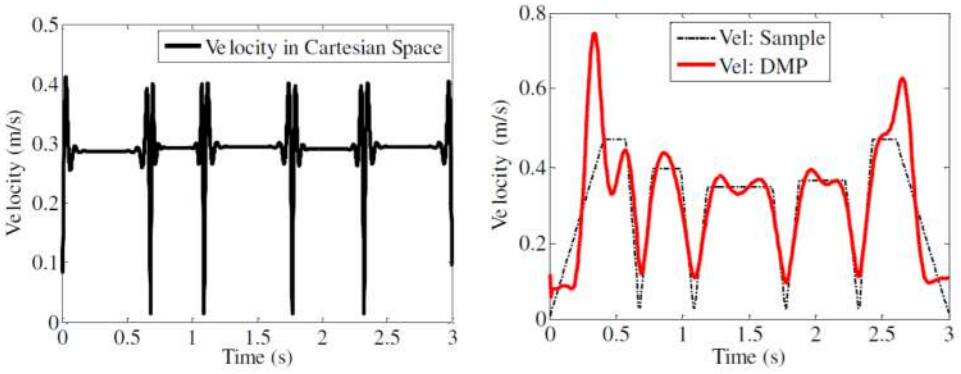
Figure 2.20 : Welding Robot scheme. Source: [14]

A gluing industrial application is reported in [Figure 2.18], while an alternative version which uses an improved sealing and gluing process is reported in [Figure 2.19].

2.3.2 Welding Applications

Welding is a traditional mechanical procedure used to join two or more metallic or thermoplastic parts. The pieces are heated up at the edges that must be joined, and then the connection is sealed with the cooldown of the material. The execution of this activity requires the use of arc welding torches that are fixed at the EE and that must be precisely moved along the joining edge. The cartesian movement must be described for the coordinate framework of the tip of the torch, the TCP and not for the robot flange reference one [Figure 2.20].

A known robotic algorithm for welding applications is the Descartes Algorithm [15]. This algorithm finds the proper movement for the manipulator by checking a timing constraint for the EE and joints positions. Through this algorithm it is possible to avoid motion redundancy by imposing a cost function on the joint positions, even if the motion is controlled in cartesian space. The planning strategy provides a sequence



(a) Spline Interpolation (b) DMP Reproduction

Figure 2.21 : Comparison between velocity outputs obtained for Spline interpolation and DMP motion with a high number of via-points. Source: [16]

of TCP poses without any velocity or acceleration information and then characterizes the motion in time by introducing a timing constraint.

This approach is proper for large displacements in the workspace that need a time optimization for the joint movement, but it does not assess the problem related to the point velocity execution. In the scope of this thesis, this method does not provide any advantage because the sealing task will be executed on a collaborative robot, so a collaborative environment must be considered, in correspondence of a reduced workspace area and on a predominant 2D trajectory such that the robot internal controller will automatically optimize joint movements.

In [17] it has been studied an optimal planning for welding applications, through an analysis on the task execution that minimizes the time and interpolates cartesian via-points with B-spline method. This approach imposed a welding torch velocity fixed along the path and did not include a proper feature characterization, as this thesis aims to do. Another general approach which uses B-spline interpolation is studied in [18]: this analysis is not specifically made on welding problems but can provide an interesting online planning strategy.

Considering a more general robotic background, the interpolation method has been proved to not always be the best choice for path planning. A comparison between a motion primitive algorithm (in this case DMP) and a traditional spline interpolation is reported in [16]. The result shows that while the cartesian tracking execution is similar, the velocity behaviour differs and results less smooth and acceptable for a robotic manipulator [Figure 2.21]. According to these outcomes, it is

in the interest of this thesis to focus on the more robotic friendly method, which would compute smooth transitions during execution.

2.4 Preference optimization

The solution for an optimization problem, basically, is found by formulating the explicit formulation from the corresponding problem and then by defining and minimizing the cost function. In many applications, this is not possible since the explicit formulation of the problem maybe of unquantifiable nature or expensive to find. Hence these problems and the approach for them are defined as Black Box Optimization. These optimization problems are generally accessed or approached based on simulations and evaluation of outputs through experiments physically. In most cases, it is much easier if the human makes the assessment, considering into account the users need and their preference, hence making it a more user centered design process.

User preferences-based learning techniques have been applied in a variety of domains. The contribution in [19] uses active learning to rank the execution performance of a grasping task by querying a human expert. A classical learning algorithm interacts with the reward learning component, so that the action learner's evolution leads the reward learner's queries. Also in health-related technology, human engagement in the design and evaluation process is critical [20]. In this End users are included throughout the development process to ensure that technologies support tasks, are simple to use, and provide value to users. In [21], a preferences-based method for learning a reward function in the context of reinforcement learning is proposed. In simulation, a deep neural network is trained to predict a reward based on human preferences over pairs of trajectories for learning Atari games and robotic tasks. In relation to this context, it is being exploited more on study of preference query between two or more decision vectors for finding the optimal solution for an optimization problem. In [22] and [23], Bayesian optimization and Particle swarm Optimization algorithms are used for various constrained or preference based optimization. In [22] it is proposed, a multi-objective Bayesian optimisation algorithm that allows the user to express preference-order constraints on the objectives of the type “objective A is more important than objective B”. The surrogate

function is modelled using general radial basis functions (RBFs) in a recently developed approach [24] called Active Preference Learning based on Radial Basis Function. The surrogate is built at each iteration by satisfying the constraints, the preferences already expressed by the decision maker at sampled points. The method has been validated, and it is reported to be computationally lighter and closer to the global optimum than Bayesian active preference learning (PBO) in the same number of comparisons. Compared to preferential Bayesian optimization, the proposed approach is computationally lighter, due to the fact that computing the surrogate simply requires solving a convex quadratic or linear programming problem. Whereas in PBO one has to first compute the Laplace approximation of the posterior distribution of the preference function and then a system of linear equations, with size equal to the number of observations, has to be solved. In terms of comparison with PSO, the technique appears to arrive at the global optimal solution more quickly.

The work [24] proposes a way to solve an optimization problem where decision makers cannot evaluate the objective function and can only express preferences such as "A is better than B" between the two candidate decision vectors. The algorithm, GLISp described in this work, repeatedly proposes new comparisons to decision makers based on active learning of Surrogate substitutions of previously sampled decision vectors and pairwise preferences. The surrogate is adjusted using radial basis functions, under the constraint of satisfying, if possible, the preferences expressed by the decision maker over the existing samples. So, at each iteration the Radial Basis Function (RBF) weights were calculated based on the constraint of satisfying the available training set preferences given by the user. The substitution value (surrogate) is used to propose a new sample of the decision vector to compare with the current best candidate based on two possible criteria: minimizing the combination of the surrogate values and the function of the inverse distance weighting (IDW) to bring a tradeoff between the exploitation (surrogate) and exploration of the decision variables or maximize a function involving the probability that the new candidate is preferred. By this way the global optimal is reached iteratively based on the User's likelihood.

In the current work, the part of the problem is to find the velocity at which there is perfect or optimum deposition quality of the silicon material, for each and every radius of curves.

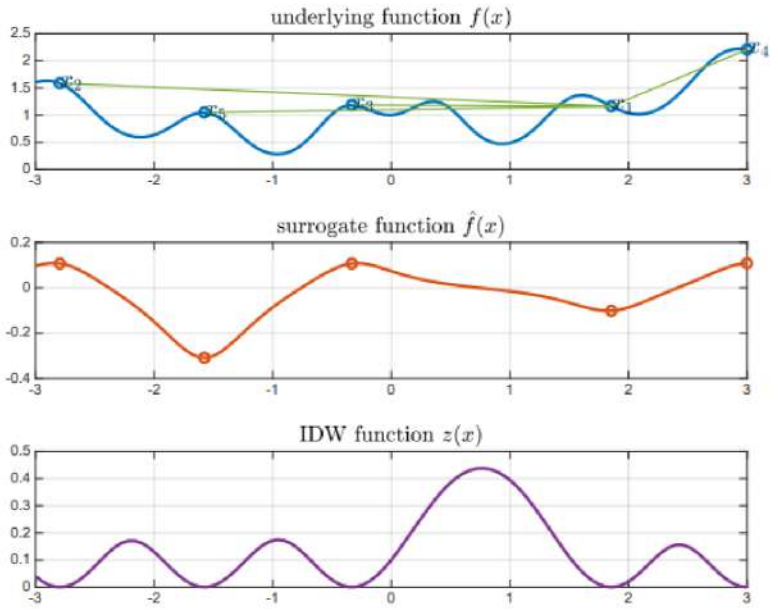


Figure 2. 22 : [24]Example of surrogate function $\hat{f}$ (middle plot) based on preferences resulting from function f. Pairs of samples generating comparisons are connected by a green line (Top plot). IDW exploration function, z(Bottom plot).

The following thesis structure is executed starting by finding the optimum and acceptable velocity values for different curves ranging from 10mm radius to 100mm radius, (4. Active Preference Learning). With this available data it is then executed a generic path by analyzing radius of curvature values at each discrete point of the path and then implying the velocity data for the corresponding radius of curvature, (5. Curvature analysis and task execution).

3 Experimental setup

This chapter explains and analyses the testbed used in all the physical experiments. The task specifications required the development of a trajectory planning algorithm for a collaborative robot, so the research focuses more on the algorithm and on its implementation, while only minor modifications to the provided experimental setup are made.

The testbed can be sub-divided into four main blocks [Figure 3.1] that interact with each other through communication signals managed by the workstation. All the algorithms that generate the trajectory and control the robot run into a Robot Operating System (ROS) environment. The robot used for the tests is a Franka Emika Panda, a collaborative robot with 7 DOF, which mounts at the EE a commercial caulking gun, a Makita DCG180, which is connected at the manipulator through a custom flange created ad-hoc [Figure 3.2].

All the physical arrangements and some useful building instructions for the development of the 3D printed flange and the motorized system that controls the sealant deposition, the Arduino code are available in [Appendix (i) : Trigger Control].

3.1 Robot Setup: Franka Emika Panda Robot

The robot provided for physical experiments is a Franka Emika Panda [Figure 3.3]. This collaborative redundant manipulator is often selected in numerous human-robot applications thanks to some peculiar features:

- Versatility: it can work in environments that require precision, force application and sensitive handling;
- Compact design and redundancy: its small dimension is not a limit since it can reach large workspaces [Figure 3.4];
- Immediate control: it is provided with an easy block programming environment for simple task execution, but it also has a complete library of control commands useful to program it for more complex activities.

Figure 3.1 : Scheme of the physical setup: from left to right there is an Ubuntu workstation which commands both the Arduino Nano and the Franka Controller that manage the robotic manipulator motion and sealant deposition flow

Figure 3.2 : Physical setup for the experiments: Franka robotic arm, Makita caulking gun, 3D printed custom flange

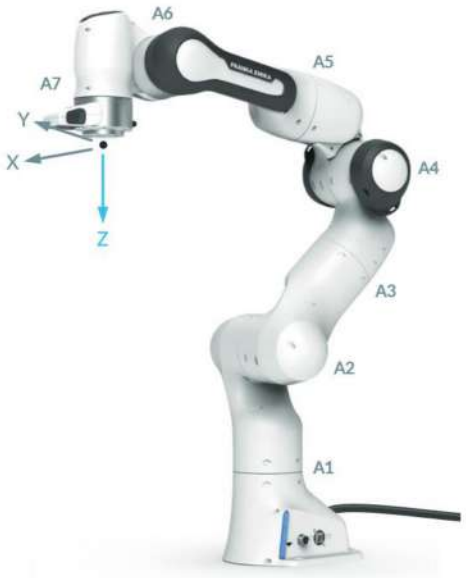


Figure 3.3 : Franka Emika Panda

Several mechanical characteristics are interesting for various applications, from tracking motion to pick and place tasks. Franka Emika Panda manipulator is provided of torque sensors on all the joints, which are useful both for impedance control activities, both for precise motion control, moreover, it has a high frequency controller (1 kHz) and high repeatability in position.

Thanks to the additional DOF, the same EE position can be reached with multiple configurations. Moreover, it is also possible to keep fixed in space the EE, while moving the arm, which is important in a human-robot environment because it permits to change configuration and adapt to the background (this characteristic is possible since the dimension of the operational space is smaller with respect to the one of joint space). The extra DOF allows also to avoid singularity configurations and to better distribute the motion above all the axes.

Generally speaking, working in a collaborative environment means that not only the robot must be collaborative, but also the task to be executed. Indeed, selecting the robot before knowing the task can be problematic; consider the welding application, even if a collaborative robot is chosen to perform the activity, the safety regulations of the mechanical procedure impose the use of fences around the robot workspace, and so it is impossible to work with human cooperation.

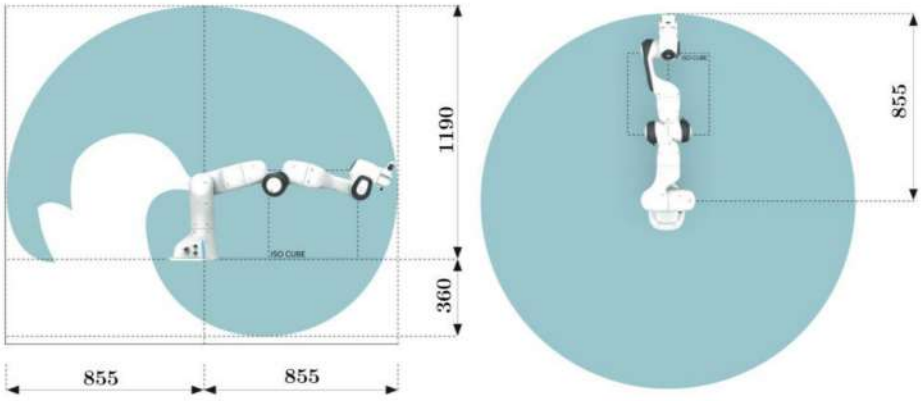


Figure 3.4 : Franka Emika Panda Workspace [mm] (side view and top view)

The sealing task does not present relevant risks for human operators, which means that a collaborative robot can be selected to accomplish and improve the reference task.

The collaborative nature of the robot refers to the manipulator capability to be inserted in an industrial environment where human operators can enter the robot workspace during the working activity. Typically, this robotic characteristic is achieved only when a safety degree can be assured, which means implementing collision-detection sensors, with low velocities and low detection times, which can stop the execution when an external force is sensed on the joints.

In the case of Franka robot, the safety condition is provided by very fast detection time (smaller than 2 ms) and high-resolution torque sensors, which are able to achieve [25]:

- Resolution < 0.05 N
- Accuracy < 0.8 N
- Repeatability < 0.05 N

The total reaction time is less than 50 ms according to datasheet information [Figure 3.5]. Moreover, rubber bumpers are placed on the robot arm to allow a reduced impact force if an unexpected collision happens.

DATA SHEET
ROBOT ARM & CONTROL

FRANKA EMIKA

Release Version: April 2020

Arm

Degrees of freedom	7
Payload	3 kg
Joint position limits	A1, A3, A5, A7: -166°/166° A2: -101°/101° A4: -176°/-4° A6: -1°/215°
Weight	~ 17.8 kg
Moving mass	~ 12.8 kg
Interfaces	• ethernet (TCP/IP) for visual intuitive programming with Desk • input for external enabling device • input for external activation device or safeguard • Control connector • Connector for end effector
Research interface	1kHz Franka Control Interface (FCI)

Interaction

Guiding force	~ 2 N
Collision detection time	<2 ms
Nominal collision reaction time [3,4]	<50 ms
Worst case collision reaction time [3]	<100 ms
Adjustable translational stiffness	0 – 3000 N/m
Adjustable rotational stiffness	0 – 300 Nm/rad
Monitored signals	joint position, velocity, torque cartesian position, velocity, force

Motion

Joint velocity limits	A1, A2, A3, A4: 150°/s A5, A6, A7: 180°/s
Cartesian velocity limits	up to 2 m/s end effector speed
Pose repeatabillity	<+/- 0.1 mm (ISO 9283)
Path deviation [3]	<+/- 1.25 mm

Force

Force resolution		<0.05 N
Relative force accuracy		0.8 N
Force repeatability		0.15 N
Force noise (RMS)		0.035 N
Torque resolution		0.02 Nm
Relative torque accuracy		0.15 Nm
Torque repeatability		0.05 Nm
Torque noise (RMS)		0.005 Nm
Minimum controllable force (Fz)		0.05 N
Force controller bandwidth (-3 dB)		10 Hz
Force range [N]	Nominal case	Local best case
Fx	-125 – 95	-150 – 115
Fy	-100 – 100	-275 – 275
Fz	-50 – 150	-115 – 155
Torque range [Nm]	Nominal case	Local best case
Mx	-10 – 10	-70 – 70
My	-10 – 10	-16 – 12
Mz	-10 – 10	-12 – 12

Figure 3.5 : Franka Emika Panda datasheet relevant information. Source: http://www.franka.de/technology

3.1.1 Franka Control Interface

The Franka Controller Interface (FCI) [Figure 3.6] provides the low-level control of the robot, both for the hand and for the arm. The connection is bidirectional, so it is possible to either provide commands to the robot, or to read measured data, such as joint positions and torque values with 1 kHz acquisition frequency [26].

Through the FCI it is possible to compensate gravity and friction, to command the robot cartesian pose (or velocity) or command joint position (or velocity). All those different interfaces are available through *libfranka*, which is the robot open source library, written in C++. A list of the robot controllers is shown in [Figure 3.7].

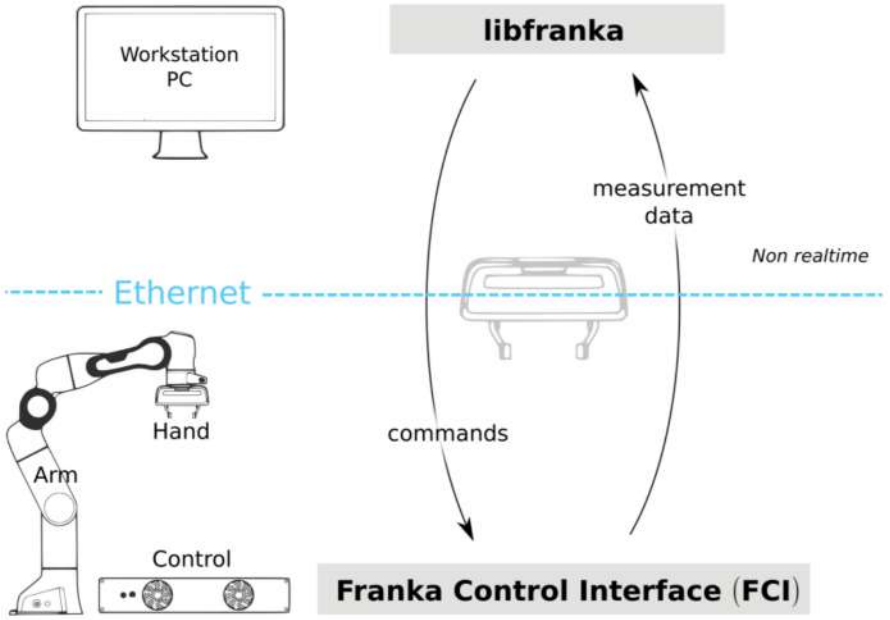


Figure 3.6 : Schematic of non real-time Franka Control Interface. Source: [26]

The library provides also measures of the joints' actual positions, of the applied torques and force sensing for collision detection from the external environment. Those data can be useful for closed-loop applications, where a feedback signal with the knowledge of robot states is necessary.

Controlling the robot and providing a proper execution signal, can be done in two ways:

- Using the web interface, useful to control simple basic movements, but not recommended for providing a complete motion execution;
- Using the ROS environment, which is an open-source operating system commonly used for robotic control activities.

This thesis uses the more complete environment, ROS, which permits to implement a complex motion and to analyse the output signals of the robot itself. Franka robots come with a build-in ROS package, called franka_ros, which provides the commands needed to interface the Python or C++ code with robot. The franka_ros library will be briefly described in [section: 3.2].

The use of ROS framework gives the possibility to plan robot movements and simulate them in a software simulator environment (Rviz).

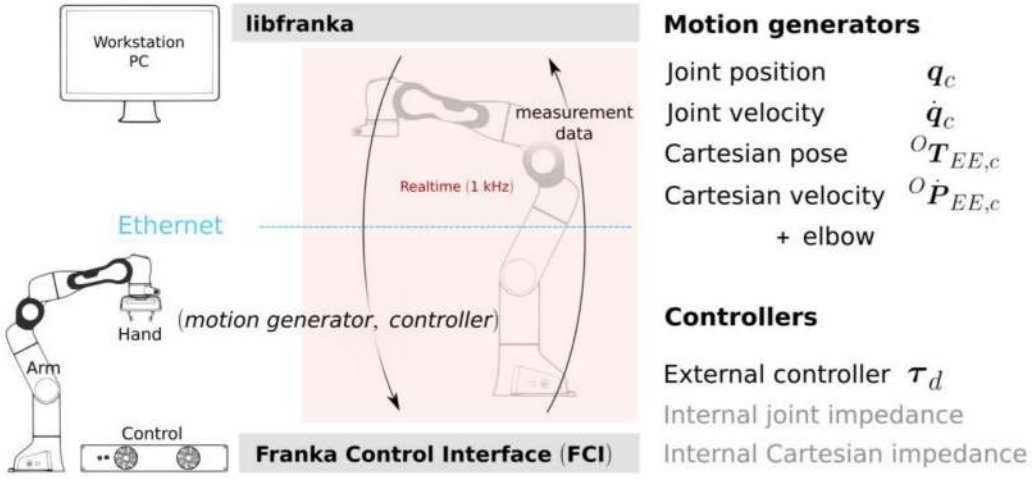


Figure 3.7 : List of the available real-time controllers for the robot. Source: [26]

3.1.2 Franka Emika Panda - Dynamic model

The dynamical structure of the Franka Emika Panda robot is in accordance with its description of collaborative manipulator. This means that some flexibility is admitted in the structure and in the specific case it is concentrated at the joints. This structure is in accordance with Flexible-Joint Robot (FJR), and so the joints must be modelled as elastic.

To derive the dynamics equations of the robot, several standard assumptions must be made, according to [27]:

- Rotors are homogeneous bodies with the center of mass on the rotation axis;
- In a N joints manipulator, each motor is mounted on the $i-1$ link and conveys a motion on the i-link, where i ∈ [1,N];
- Rotors have an angular velocity generated only by spinning action.

Joints can be modelled with two inertial contributions, rotor and link [Figure 3.8]. The rotation generated by the i-motor, ϑ_i, is converted in a joint rotation qi through reduction ratios and the connection between the two masses is considered elastic, such that a mass-spring-damper system can be modelled on every joint.

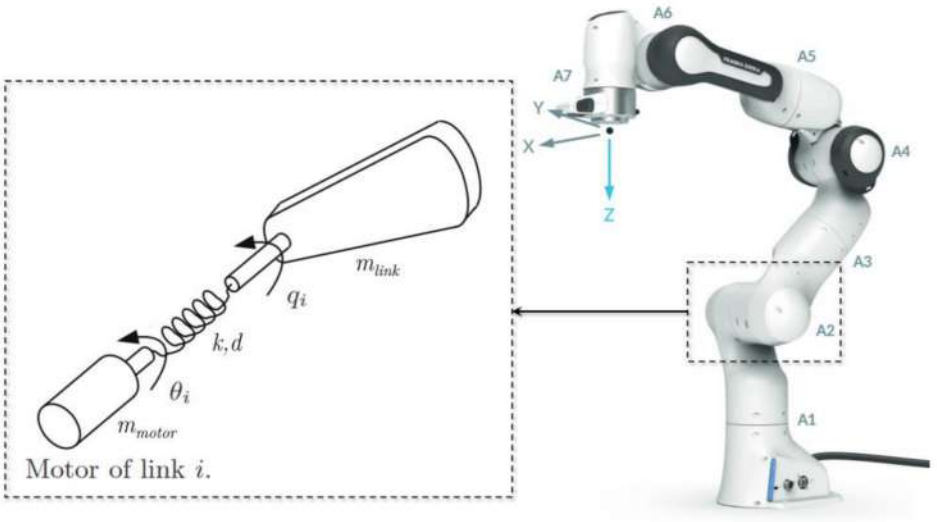


Figure 3.8 : Flexible joint kinematic model

The i-motor mass is defined as m_{motor}, while the link mass is m_{link}. The spring stiffness is modeled with the parameter k and the damping term related to friction forces is d.

The complete dynamic equation studies the motion of the robot dynamical system considering external torques and forces, damping actions and stiffness terms.

For a N-DOF system, the joint coordinates are $\mathbf{q} \in \Re^{N\times 1}$, and the dynamical model is a set of N equations [28; 29; 30]:

$$\begin{cases} \boldsymbol{B}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \boldsymbol{g}(\mathbf{q}) + \boldsymbol{K}(\mathbf{q}-\boldsymbol{\vartheta}) = \mathbf{0} \\ \boldsymbol{J}\ddot{\boldsymbol{\vartheta}} + \boldsymbol{K}(\boldsymbol{\vartheta}-\mathbf{q}) = \boldsymbol{\tau} \end{cases} \qquad (3.1)$$

where:

- $\boldsymbol{B}(q) \in \Re^{N\times N}$ is the manipulator inertial matrix;
- $\boldsymbol{C}(q,\dot{q})\dot{q} \in \Re^{N\times N}$ is the Coriolis vector and centrifugal forces term;
- $\boldsymbol{g}(q) \in \Re^{N\times N}$ is the vector of gravity terms.
- $\boldsymbol{K} \in \Re^{N\times N}$ is the joint stiffness matrix.

In the second equation is reported the rotor side:

- $\boldsymbol{J} \in \Re^{N\times N}$ is the rotor inertia;
- $\boldsymbol{\tau} \in \Re^{N\times N}$ is the vector of motor torques.

For a deeper analysis on the dynamics of a FJR, it is advised to look at [31]. This thesis does not focus on the impedance control or force analysis due to the nature of the task to be executed.

3.2 ROS Environment

The numerical code that controls the motions of the robot communicates with the manipulator itself and with the caulking gun using the ROS framework. This environment permits to command signals both to robot motion control system and to the Arduino board.

The communication between scripts written in traditional programming languages (Python, C++ etc.) and the robot is permitted by the franka_ros package, a build-in library that permits to receive numerical data and to send them to the robot. Instead, the Arduino is connected with the workstation through a Serial Port, that sends messages at fixed time rate to control the extrusion (this section is in open-loop, assuming that the servo motor does not saturates its torque).

3.2.1 ROS Nodes

The communication between the parts involved in the motion of the robot is permitted using nodes, which are sub-groups of the robot application that communicate with each other to retrieve data. Basically, nodes are executable programs that run inside the application. According to the ROS definition:

"A node is a process that performs computations. Nodes are combined together into a graph and communicate each other's using streaming topics, remote procedure call services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes."

In order to transmit a message from a node to another, ROS platform uses buses called topics. Whenever a node is sending a signal, the procedure is referred to as node publishing a topic, instead when a node is receiving a message coming from a topic, then the node is subscribing to a topic. A graphical representation of a single topic communication is reported in [Figure 3.9], where it is also schematized the registration to the ROS master environment. In complex systems, it is common to have multiple publishers and multiple subscribers which exchange different message types over many topics.

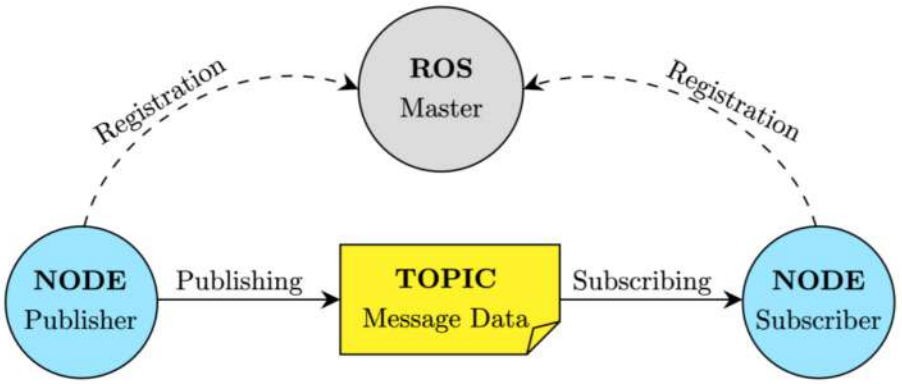


Figure 3.9 : ROS standard communication

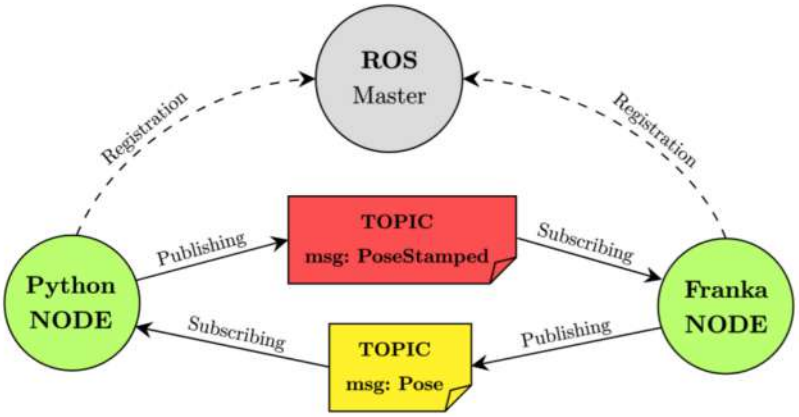


Figure 3.10 : ROS structure: the Python node commands the robot positions (PoseStamped) to the Franka node and it receives back the actual robot position (Pose)

The ROS implementation performed in this thesis for the off-line motion planning required the generation of one publishing node and one subscribing node needed to send position messages from the Python code to the robot manipulator. The ROS node-topic framework related to the control structure analysed in this work is reported in [Figure 3.10].

3.3 Flange for Caulking Gun

The provided testbed requires the use of a commercial caulking gun, the Makita DCG180, for the implementation of the silicone deposition. A physical limitation of the testbed was related to the mechanical connection between the robot EE and the caulking gun handle. The standard EE with the grasping hand [Figure 3.11] could not

provide a rigid connection between the two parts, so it has been studied and realized through FDM 3D printing a custom flange which permitted a stable and rigid application of the calking gun to the EE [Figure 3.12].

Figure 3.11 : Franka Emika Panda standard gripper

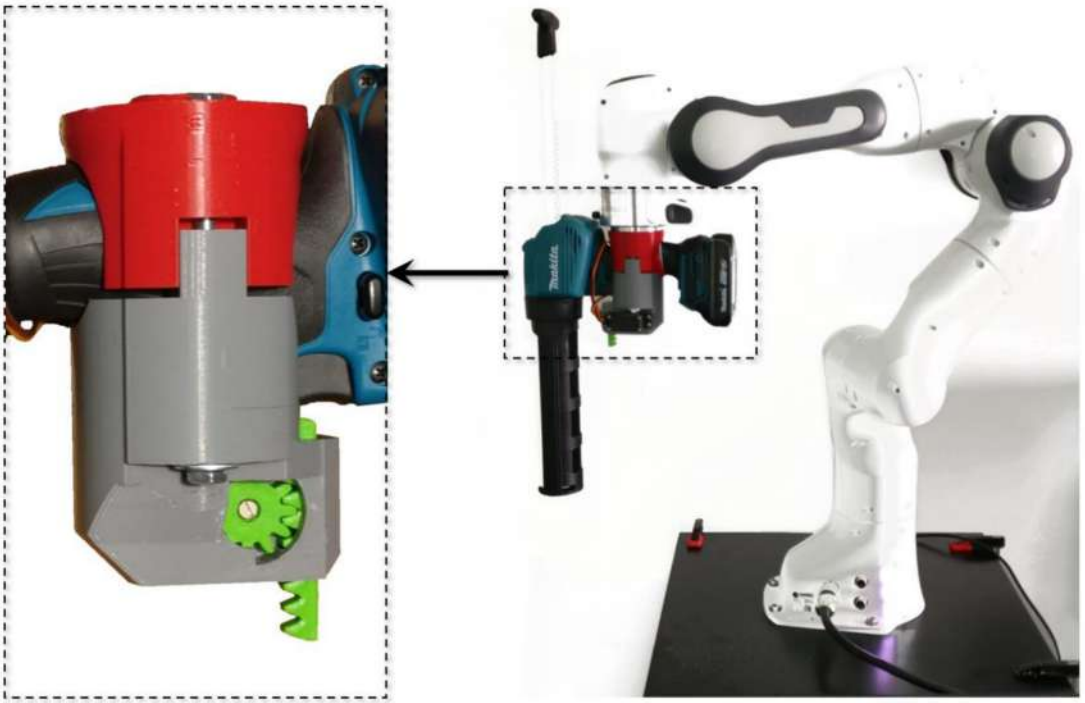


Figure 3.12 : Close-up of the custom flange on the rack-pinion mechanism side. The red part is connected to the EE, while the grey part contains the motorized mechanism

The designed flange firmly connects the gun to the robot EE, while also providing a control of the silicone extrusion using a servomotor. During the development of the flange, the following problem was considered:

- *Boundary matching*: the design had to match the profile of the robot flange (DIN ISO 9409-1-A50) and the shape of the gun handle;
- *Motor action on the trigger*: the motorized rotation-to-translation mechanism generated by the servomotor;
- *Stiff connection*, to prevent unwanted vibrations on the gun tip (which would result in poor quality of the deposition);
- *Compactness*, which intrinsically permitted to avoid collisions with the robot arm.

The flange has been fully produced through FDM 3D printing (refer Appendix (i) for 3D Diagrams and the setup) (Figure 3.12), while some commercial M6 screws have been inserted to connect the piece to the robot EE with a clamped configuration.

During the design stage, it has been defined the optimal system to perform the silicone extrusion: according to the datasheet of the caulking gun, the flow rate should have been proportional to the push on the trigger, while the experiments proved that the gun works with a mainly on-off system. In the current task the trigger is always on for the whole path to be performed.

To activate the extrusion, a rack-pinion mechanism motorized by a servomotor has been selected. The pinion is driven by an Arduino-controlled servomotor and the rack moves back and forth pressing the trigger [Figure 3.13].

For a complete implementation of the push mechanism, it has been discussed the proper control strategy, with manual or automatic activation. In the former case it has been used a potentiometer which worked as a knob, while in the latter one the command signal is directly fed into the Arduino by means of a serial communication with the computer.

The automatic activation through a command signal is preferrable to the manual one, because it is possible to numerically generate a proper control signal that precisely presses the trigger, while in the manual case, the operator controls the execution, causing reduction in the automatization of the task, and in reproducibility of the results. However, in both cases, it has been verified that the caulking gun never

saturates its pushing force (5 kN from datasheet) [32] since the overload warning light has never switched on.

Figure 3.13 : Trigger activation, rack-pinion mechanism

3.4 Positioning and Orientation

The sealing procedure must be examined before the effective planning strategy can be examined to define its orientation and positioning when attached with the robot. When a skilled operator conducts a manual sealing operation, the caulking gun is often turned at an angle α in the direction of travel with regard to the deposition plane's orthogonal axis [Figure 3. 14]. This manual arrangement allows for proper sealant application while avoiding nozzle clogging. Working vertically, like for a typical material extrusion of an FDM 3D printer, the collaborative robot will have no difficulty in maintaining a precise vertical offset from the deposition plane. Therefore, correct results can be produced by working vertically (i.e. α = 0). Since the gun's setup will not interfere with manipulator arm movements, keeping the End Effector orientation fixed provides for a larger bi-dimensional working space. Furthermore, because the Euler angles remain constant during the trip, establishing the orientation allows for better planning of the robot configuration.

The work at hand necessitates a Tool Centre Point coordinate definition in 3D space. However the problem can be simplified because the trajectory of interest is just bi-dimensional. Although the general End Effector pose in 3D is defined by six parameters for each point along the path, the End Effector pose has a fixed orientation (φ_ε) and vertical offset (z) for the entire path when the caulking gun is

perpendicular to the surface of the path. The suggested approach analyzes the execution velocity as a significant parameter that must be changed according to the path attributes (or path features) in order to improve the sealant deposition process in terms of time and quality.

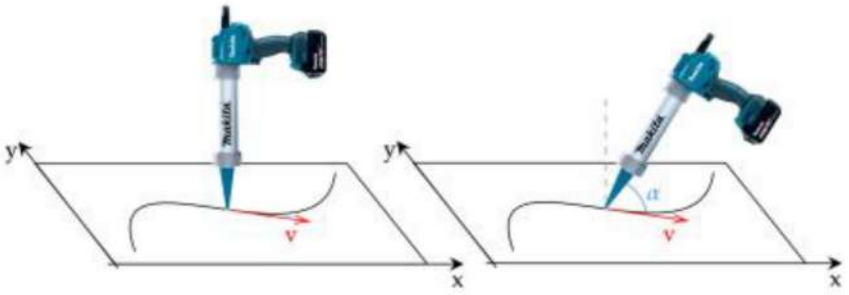


Figure 3. 14 : Different techniques to defining the gun pose: on the left, a perpendicular deposition is used, whereas on the right, the gun is inclined forward at an angle.

4 Active Preference Learning

Initially it was decided to modify the velocity profile of the deposition task and also the material flow of the gun by controlling the extrusion gun trigger. In order to do this, the silicone extrusion rate must be modulated with respect to the EE velocity in order to avoid material stockpiles when the EE moves at low speed and conversely material lacks when the EE is at full speed. An example is reported in Figure 4. 1. The trigger position was controlled from the Arduino commands to the motor and by this the material deposition rate was maximum when the trigger was continuously pushed, while it was minimum if the trigger was pressed intermittently with a very short push. But the downside of this was that every time the end-effector setup for holding the caulking gun was removed and refitted, it was difficult to reproduce the exact position of the trigger as before. Hence the intermittent position was varying every time and thus it was decided to proceed with maximum material deposition rate with the trigger being continuously pushed. The only parameter that is required to be optimized is the execution velocity of the deposition task relating to the different range of curves.

The optimization of the velocity is done for each kind of curve of radius ranging from 10mm to 100mm. So, for a single curve at a time, the algorithm suggests a new velocity point at each iteration (by satisfying the user preference constraints from previously sampled points), which is then executed by the robot to perform the deposition task. The user then judges the quality of deposition with respect to the best one so far. The users' input of preference to the algorithm allows searching for a new velocity point. By this way optimum velocity value is obtained for each and every curve.

The process of the Active Preference Learning Optimization Algorithm is structured below as The Problem Statement (4.1), The Surrogate Function (4.2) and The Acquisition Function (4.3). Then The Experiments (4.4) for the example curve 64mm and also the straight line are depicted.

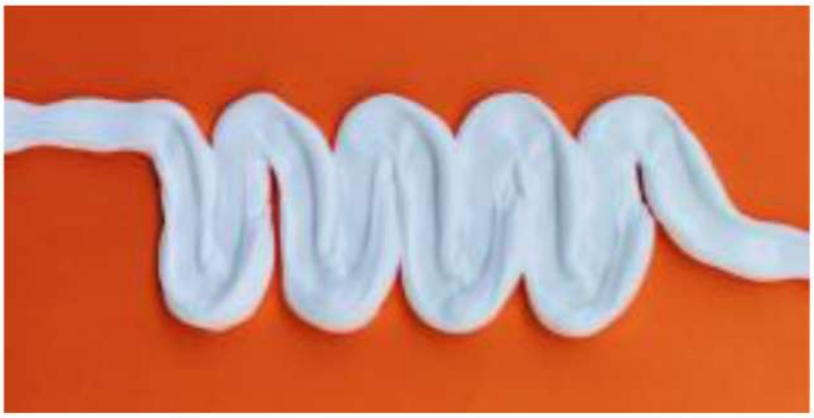


Figure 4. 1 : Excess deposition in hairpin curves.

4.1 The problem statement

The initial step of the Active Preference Learning Algorithm is to get the preferences from the user. Given two possible decision vectors $\mathbf{x_1}, \mathbf{x_2} \in R^n$, consider the preference function $\pi : R^n \times R^n \rightarrow \{-1, 0, 1\}$ defined as

$$\pi(\mathbf{x_1}, \mathbf{x_2}) = \begin{cases} -1 & if\ \mathbf{x_1} \text{is better than } \mathbf{x_2} \\ 0 & if\ \mathbf{x_1} \text{is same as } \mathbf{x_2} \\ 1 & if\ \mathbf{x_2} \text{is better than } \mathbf{x_1} \end{cases} \qquad (4.1)$$

where for all $\mathbf{x_1}, \mathbf{x_2} \in R^n$ it holds $\pi(\mathbf{x_1}, \mathbf{x_1}) = 0$ and $\pi(\mathbf{x_1}, \mathbf{x_2}) = -\pi(\mathbf{x_2}, \mathbf{x_1})$. Where the objective of the Active Preference Learning Algorithm is to solve the following constrained global optimization problem:

$$\text{find } \mathbf{x}^* \text{ such that } \pi(\mathbf{x}^*, \mathbf{x}) \leq 0,\ \forall x \in \chi,\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \qquad (4.2)$$

Vectors $\mathbf{l}, \mathbf{u} \in R^n$ in (4.2) define lower and upper bounds on the decision vector, and $\chi \subseteq R^n$ imposes further constraints on $\mathbf{x}$, such as:

$$\chi = \{\mathbf{x} \in R^n :\ g(\mathbf{x}) \leq 0\ \} \qquad (4.3)$$

Where g: $R^n \rightarrow R^q$, and $\chi = R^n$ *when* $q = 0$ (no inequality constraint is enforced).

The problem of minimizing an objective function f: $R^n \rightarrow R$ under constraints,

$$x^* = argmin_x \ f(\mathbf{x}) \qquad (4.4)$$
$$s.t. \ \ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$$
$$\mathbf{x} \in \chi$$

can be rewritten by defining,

$$\pi(\mathbf{x_1}, \mathbf{x_2}) = \begin{cases} -1 & if \ \mathrm{f}(\mathbf{x_1}) < \mathrm{f}(\mathbf{x_2}) \\ 0 & if \ \mathrm{f}(\mathbf{x_1}) = \mathrm{f}(\mathbf{x_2}) \\ 1 & if \ \mathrm{f}(\mathbf{x_2}) > \mathrm{f}(\mathbf{x_1}) \end{cases} \qquad (4.5)$$

By this, it is assumed that we do not have a way to evaluate the objective function f. The only assumption made is that for each given pair of decision vectors $\mathbf{x_1}, \mathbf{x_2} \in \chi$, $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, only the value $\pi(\mathbf{x_1}, \mathbf{x_2})$ is observed. The rationale of the problem formulation is that often one encounters practical decision problems in which a function f is impossible to quantify, but anyway it is possible to express a preference by a human operator, for any given presented pair $(\mathbf{x_1}, \mathbf{x_2})$.

The successive part, it is implicitly assumed that a function f exists but is unknown, and we seek to synthesize a surrogate function $\hat{f} : R^n \to R$ of f such that its associated preference function $\hat{\pi} : R^n \times R^n \to \{-1,0,1\}$ described in (4.5) coincides with on the finite set of sampled pairs of decision vectors.

4.2 The Surrogate Function

Assume that we have generated N ≥ 2 samples $\mathbf{X} = \{\mathbf{x_1}, \ldots\ldots, \mathbf{x_N}\}$ of the decision vector, with $\mathbf{x_i}, \mathbf{x_j} \in R^n$ such that $\mathbf{x_i} \neq \mathbf{x_j}, \forall i \neq j, \ i, j = 1, \ldots\ldots, N$. For each of the parameters, the experiment is performed and is evaluated a preference vector $\mathbf{B} = [b_1 \ldots\ldots b_M]' \in \{-1,0,1\}^M$

$$b_h = \pi(x_{i(h)}, x_{j(h)}) \ , \qquad (4.6)$$

where M is the number of expressed preferences, $h \in \{1, \dots, M\}, i(h), j(h) \in \{1, \dots, N\}, i(h) \neq j(h)$. Note that the element b_h of vector B represents the preference expressed by the user between the experimental performance achieved with parameters $\mathbf{x_{i(h)}}$ and $\mathbf{x_{j(h)}}$.

The observed preferences are then used to learn a surrogate function $\hat{J}: R^{n_x} \rightarrow R$ of an (unknown) underlying performance index J. The surrogate $\hat{J}$ is parametrized as the following linear combination of Radial Basis Functions (RBFs):

$$\hat{J}(\mathrm{x}) = \sum_{k=1}^{N} \beta_k \varphi\big(\varepsilon d(x, x_i)\big), \qquad (4.7)$$

where $d: R^{n_x} \times R^{n_x} \rightarrow R$ is the squared Euclidian distance

$$d(x, x_i) = ||x - x_i||_2^2 \qquad (4.8)$$

$\varepsilon > 0$ is a scalar parameter, $\varphi : R \rightarrow R$ is an RBF and $\beta = [\beta_1 \dots . \beta_N]$ are the unknown coefficients to be computed based on the available users preference. Examples of RBFs are $\varphi(\varepsilon d) = \frac{1}{1+(\varepsilon d)^2}$ (inverse quadratic), and $\varphi(\varepsilon d) = e^{-(\varepsilon d)^2}$ (Gaussian).

The surrogate J must meet the following requirements, according to the preference relation (4.5):

$$\begin{aligned} &\hat{J}\big(x_{i(h)}\big) \leq \hat{J}\big(x_{j(h)}\big) - \sigma + \varepsilon_h \quad if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = -1 \\ &\hat{J}\big(x_{i(h)}\big) \geq \hat{J}\big(x_{j(h)}\big) + \sigma - \varepsilon_h \quad if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = 1 \\ &\big|\hat{J}\big(x_{i(h)}\big) - \hat{J}\big(x_{j(h)}\big)\big| \leq \sigma + \varepsilon_h \quad if\ \pi\big(x_{i(h)}, x_{j(h)}\big) = 0 \end{aligned} \qquad (4.9)$$

for all h= 1,....,M, where $\sigma > 0$ is a given tolerance and ε_h are positive slack variables which are used to relax the preference constraints. Constraint feasibility might be due to an inappropriate selection of RBF due to the poor flexibility in parametric description of surrogate and also due to inconsistent assessments done by user. Based on the above given preference constraints, the coefficient vector β describing the surrogate $\hat{J}$ are obtained by solving the Quadratic Programming (QP) problem

$$min_{\beta,\varepsilon} \sum_{h=1}^{M} c_h \varepsilon_h + \frac{\lambda}{2} \sum_{k=1}^{N} \beta_k^2 \qquad (4.10)$$

$$s.t. \quad \sum_{k=1}^{N} (\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k) \leq -\sigma + \varepsilon_h, \quad \forall h: b_h = -1$$

$$\sum_{k=1}^{N} (\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k) \geq \sigma - \varepsilon_h, \quad \forall h: b_h = 1$$

$$\left| \sum_{k=1}^{N} (\phi(\gamma d(x_{i(h)}, x_k) - \phi(\gamma d(x_{j(h)}, x_k))\beta_k) \right| \leq \sigma + \varepsilon_h, \quad \forall h: b_h = 0$$

h=1,....,M

where c_h are positive weights, for example, $c_h = 1, \forall\, h = 1, \ldots, M$. The scalar $\lambda > 0$ in the cost function (4.10) is a regularization parameter which gaurentees uniqueness in the solution of the Quadratic Programming problem. If λ = 0 problem (4.10) becomes a linear program (LP), whose solution may not be unique.

4.3 The Acquisition Function

Once a surrogate $\hat{f}$ has been calculated, this function can be minimized in order to identify the best velocity parameter x. More specifically, the following steps can be followed: (i) generate a new sample by pure minimization of the estimated surrogate function $\hat{f}$, i.e.,

$$x_{N+1} = argmin\, \hat{f}(x) \quad s.t. \quad x \in \Theta;$$

(ii) ask the user to express a preference $\pi(x_{N+1}, x_N^*)$, where $x_N^* \epsilon R^{n_x}$ is the best value of the velocity parameter found so far, (iii) update the estimate of $\hat{f}$ through (4.10); and (iv) iterate over N. Such a technique, which only uses the current available data to find the best parameter vector, could easily overlook better performing velocity values. As a result, a function that encourages parameter space exploration should be considered.

In the GLISp algorithm, an acquisition function is employed to balance exploitation vs. exploration when generating the new sample x_{N+1}. The exploration

function is constructed by using the inverse distance weighting (IDW) function $z : R^{n_x} \rightarrow R$ defined by

$$z(x) = \begin{cases} 0 & if\ X\epsilon\{x_1, \ldots., x_N\} \\ \tan^{-1}\frac{1}{\sum_{i=1}^{N} w_i(x)} & otherwise \end{cases} \qquad (4.11)$$

Where $w_i(x) = \frac{1}{d^2(x,x_i)}$. Clearly $z(x) = 0$ for all parameters of x, and z(x) > 0 in $R^n \setminus X$. The arc tangent function in (4.11) avoids that z(x) gets excessively large far away from all sampled points. Given an exploration parameter δ ≥ 0, the acquisition function a: $R_n \rightarrow R$ is defined as

$$a(x) = \frac{\hat{J}(x)}{\Delta\hat{J}} - \delta z(x) \qquad (4.12)$$

where

$$\Delta\hat{J} = \max_i\{\hat{J}(x_i)\} - \min_i\{\hat{J}(x_i)\} \qquad (4.13)$$

is the range of the surrogate function on the samples in $\{x_1, \ldots., x_N\}$ and is used in (4.12) as a normalization factor to simplify the choice of the exploration parameter δ. As discussed below, given a set $\{ x_1, \ldots., x_N \}$ of samples and a vector B of preferences defined by (4.6), the next parameter x_{N+1} of the velocity planner to test is computed as the solution of the (non-convex) optimization problem

$$x_{N+1} = arg\ \min_{x\in\Theta} a(x) \qquad (4.14)$$

4.4 Experiments

As described initially in the section, the parameter to be optimized is the velocity of deposition and the upper and lower bound limits for the velocity is set to be 5mm/s to 100mm/s which are discretely considered with a step of 5mm/s. The initial sample points for the optimization are considered to be 4 and the maximum iteration limit for the optimization was set to 20. This procedure of Active Preference Learning Optimization was implemented for radius of curves ranging from 10mm to 100mm with discrete steps of 6mm and also for the straight line.

To start the deposition task on a curve it is required to re-discretize the created set of points on the curve. Initially the re-discretization is done in order to make the path for the deposition task in relation with the characteristic dimension of the sealing cartridge nozzle. Depending on the nozzle diameter the curve can be sharp and also long. Analysis of point density is done to obtain the optimum number of points per diameter (Figure 4. 2) and is set to be 4 points per nozzle diameter (which is in this experiment 8mm). Given a set of N points in X,Y coordinates for a curve,

$$\mathbf{y_{des,i}} = [x_i, y_i] \qquad i \in [0, N] \tag{4. 15}$$

they are re-sampled to be equally spaced, accordingly with the nozzle diameter re-parametrization (4 points per diameter).

$$point\ density = \frac{4}{Diameter\ of\ nozzle} \tag{4. 16}$$

$$spatial\ step, ds = \frac{1}{point\ density} \tag{4. 17}$$

$$N_{new} = point\ density \times length\ of\ path \tag{4. 18}$$

with N_{new}, the new set of equally spaced y_{eq} points are found by interpolating the previous y_{des} points with the N_{new} number of points. After this, the second resampling is done to apply the reference velocity implication for the robot to reproduce. The time step for the robot controller to receive the successive point is 1ms. The points y_{eq} are again refitted with this time step constraint so that the new set of points $y_{eq\text{-}final}$ is available for the robot to follow. Along with this there is an initial ramp up to reach the given velocity and a final descent to 0 velocity, a trapezoidal velocity profile to facilitate smoother initial movement for the robot and also the deposition task.

The preference optimization of the velocity parameter for the deposition task for the curve of 64mm is shown in Figure 4. 3. This process is repeated for all curves as mentioned initially in this section and the data of relationship between the optimum velocity corresponding to its radius of curvature values is shown in Figure 4. 4. At the same time the velocity of acceptable deposition quality is also recorded. Then these values are interpolated so that the velocity value can be obtained as the function of Radius of Curvature at any point.

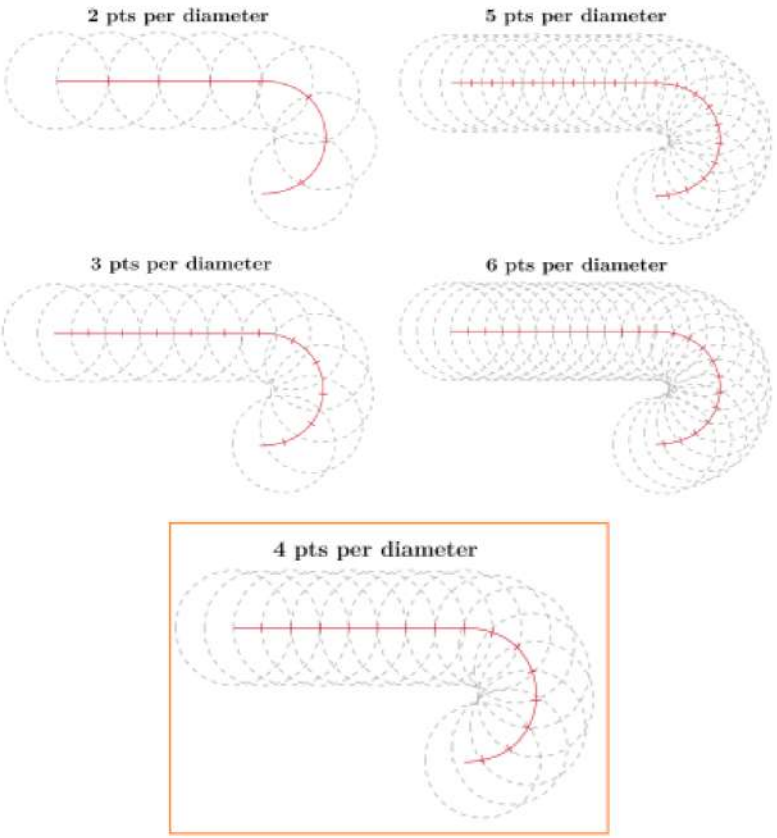


Figure 4. 2 : Comparable nozzle diameter with same sharp curve

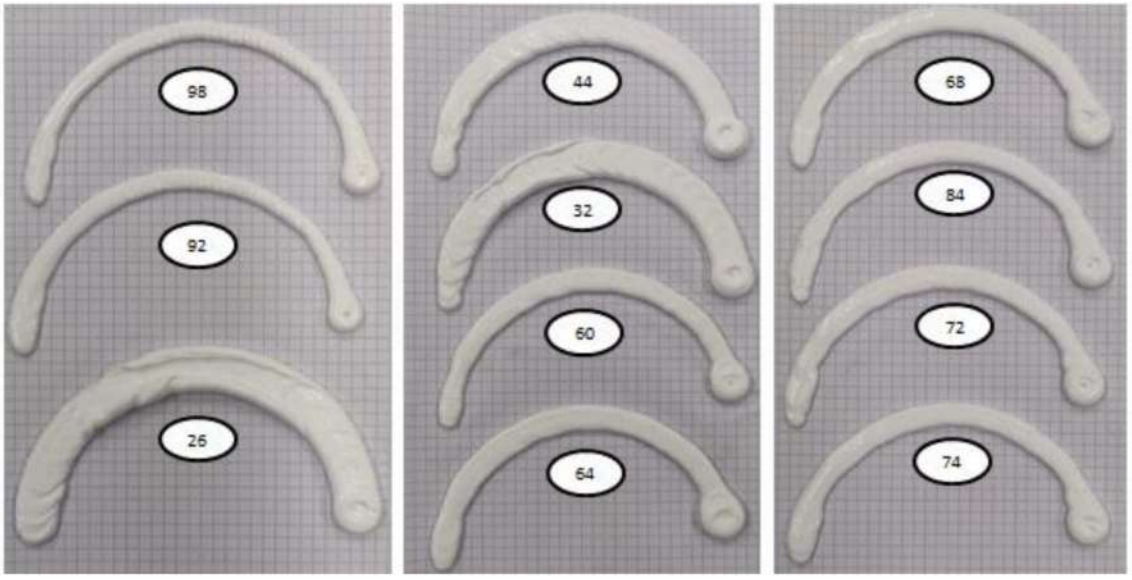


Figure 4. 3 : The Iterative Preference Optimization of the Velocity of the deposition task. (Numbers on the image indicate the velocity value in mm/s in the order of suggestion from the algorithm)

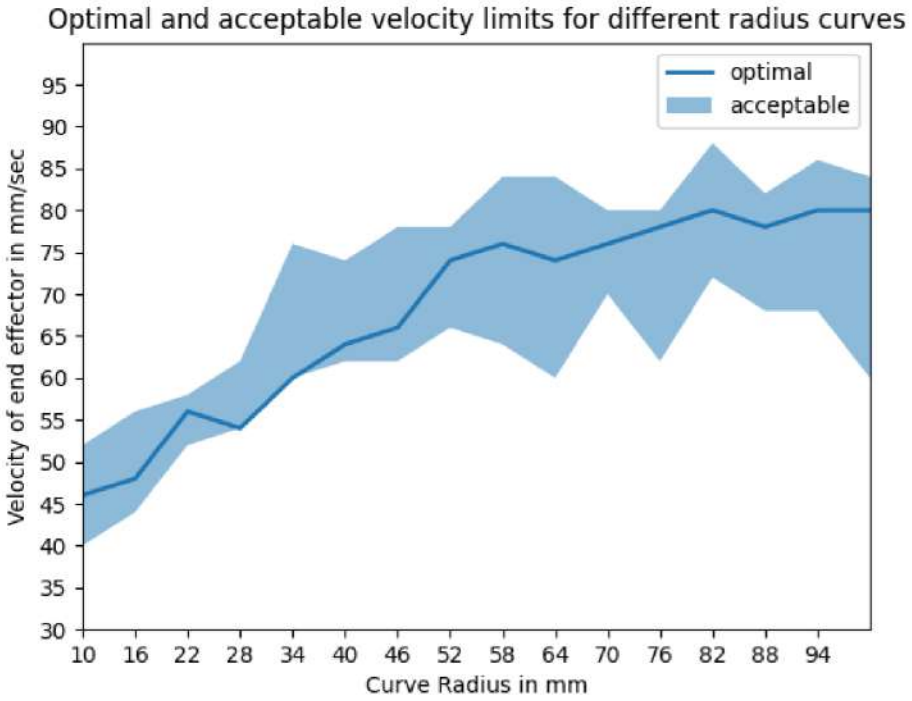


Figure 4. 4 : Relationship between the radius of curvature and the Optimum, Acceptable, Velocity values

5 Curvature analysis and task execution

This chapter is divided into 3 sections: mainly the curvature analysis where the radius of curvature is estimated given the generic path points (5.1), second is the experimental part where the velocity is implied on each point of the path corresponding to the radius of curvature for different test paths (5.2), third is the constraints implication on the velocity profile for a smoother movement of the robot.

5.1 Radius of curvature for a generic curve: curvature analysis

Any approximate circle's radius at any particular given point is called the radius of curvature of the curve. As we move along the curve, the radius of curvature changes. The radius of curvature formula is denoted as 'R'. The amount by which a curve derivates itself from being flat to a curve and from a curve back to a line is called the curvature. It is a scalar quantity. The radius of curvature is the reciprocal of the curvature. For any curve with equation $\mathbf{y} = f(\mathbf{x})$, with x as its parameter the radius of curvature can be given as follows. Suppose that the tangent line is drawn to the curve at a point M(x, y). The tangent forms an angle α with the horizontal axis (Figure 5. 1). At the displacement Δs along the arc of the curve, the point M moves to the point M_1. The position of the tangent line also changes: the angle of inclination of the tangent to the positive x-axis at the point M_1 will be $\alpha + \Delta\alpha$. Thus, as the point moves by the distance Δs the tangent rotates by the angle $\Delta\alpha$ (The angle is supposed to be increasing when rotating counterclockwise). The absolute value of the ratio $\frac{\Delta\alpha}{\Delta s}$ is called the mean curvature of the arc MM_1. In the limit as $\Delta s \to 0$ we obtain the curvature of the curve at the point M:

$$K = \lim_{s\to 0} \left|\frac{\Delta\alpha}{\Delta s}\right| \qquad (5.1)$$

From this definition it follows that the curvature at a point of a curve characterizes the speed of rotation of the tangent of the curve at this point. For a plane curve given by the equation **y**=f(**x**), the curvature at a point M(x,y) is expressed in terms of the first and second derivatives of the function f(**x**) by the formula:

$$K = \frac{|y''(x)|}{[1+y'(x)^2]^{\frac{3}{2}}} \qquad (5.2)$$

If a curve is defined in parametric form by the equations x = x(t) and y = y(t) then its curvature at any point M is given by

$$K = \frac{|x'y'' - y'x''|}{[x'^2+y'^2]^{\frac{3}{2}}} \qquad (5.3)$$

The radius of curvature of a curve at a point M is called the inverse of the curvature K of the curve at this point:

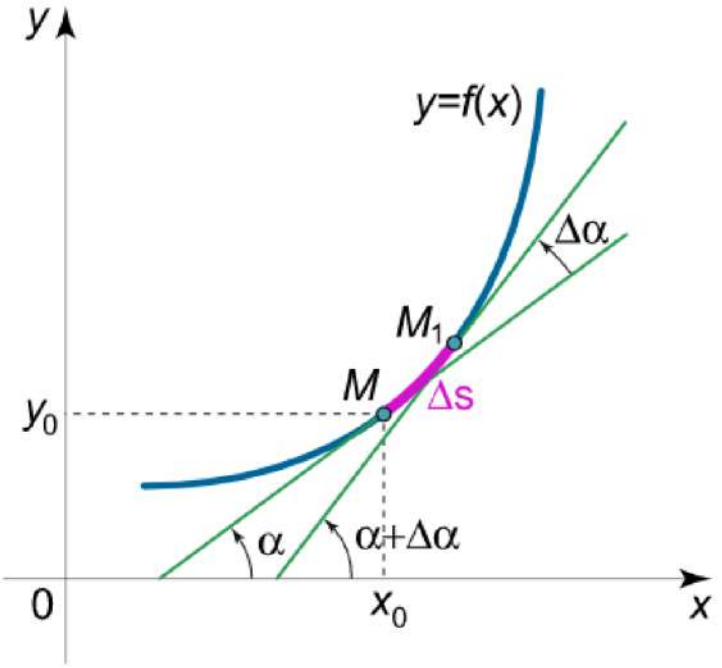


Figure 5. 1 : Radius of Curvature Estimation, display of how the angle of inclination of the tangent changes

$$R = \frac{1}{K} \qquad (5.4)$$

Hence for plane curves given by the explicit equation y=f(x) the radius of curvature at a point M is given by the following expression:

$$Radius\ of\ Curvature\ R = \frac{(1+\frac{dy^2}{dx})^{\frac{3}{2}}}{\left|\frac{d^2y}{dx^2}\right|} \qquad (5.5)$$

The Final Velocity Planner works in the following way. For a given generic path (with the x,y points already discretized) it is necessary to find the varying radius of curvature at each discrete point of the path. For each of this radius of curvature values we find the optimum velocity value from the relation data acquired from the Active Preference learning Algorithm (Figure 4. 4). So, at this stage we know the velocity value with which the robot should carry out the deposition task at each point of the path. To be noticed here is that the velocity values corresponding to the Radius of Curvature are directly implied and there are no constraints on them initially. In the proceeding parts the constraints are also discussed.

The path resampling with reference to the nozzle diameter is also carried out before calculation of radius of curvature to make sure the points are equally spaced. Finally, the path points are resampled again according to the velocity value at each point with reference to the time step of the robot (1ms).

5.2 Test paths

To test the implication of the velocity profile, it is required to experimentally validate the algorithm on various test paths/complete trajectories. In each of the test paths, it is depicted the radius of curvature profile, the velocity profile and also the result of the deposition task. The following paths had increasing difficulties in the velocity profile:

1. Spiral Path: A combination of curves with increasing radius of 5mm every 90 degrees to form a spiral path (Figure 5. 2). The velocity values are shown in Figure 5. 3.

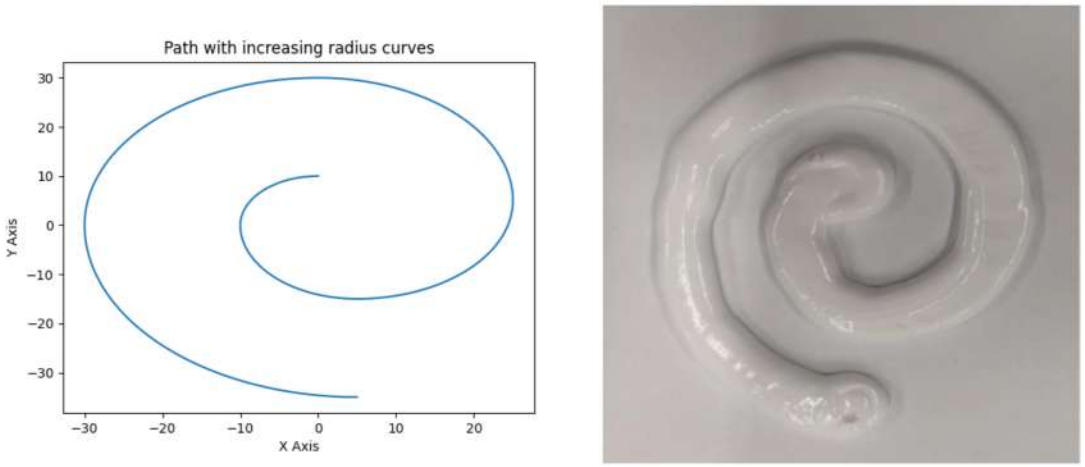


Figure 5. 2 : Spiral path with increasing radius every 90 degree (a) Path generation (b)deposition of spiral path

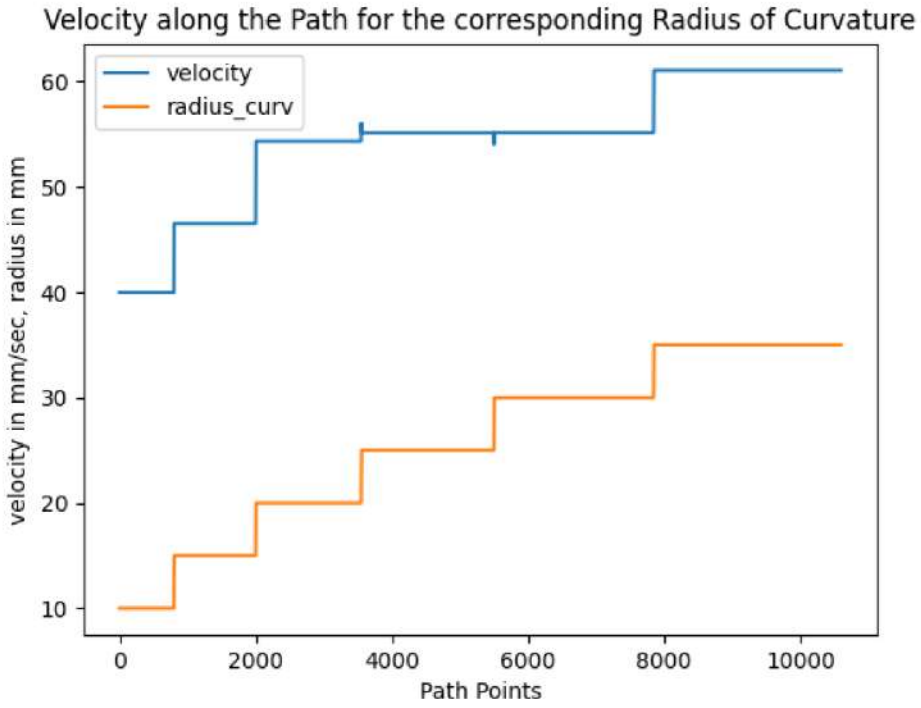


Figure 5. 3 : Velocity and Radius of Curvature along the Spiral Path

2. Similar radius Curves: A combination of straight line and sharp curve of radius 5mm (Figure 5. 4 : Same Radius curves: (a)Path Generation (b) Deposition). The velocity values are shown in Figure 5. 5 which has the initial ramp up for the smooth acceleration.

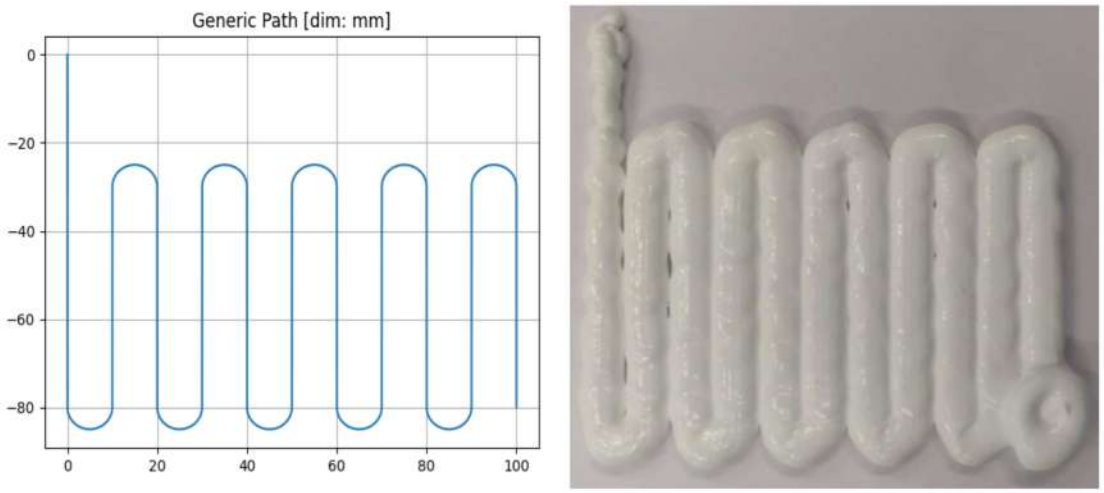


Figure 5. 4 : Same Radius curves: (a)Path Generation (b) Deposition

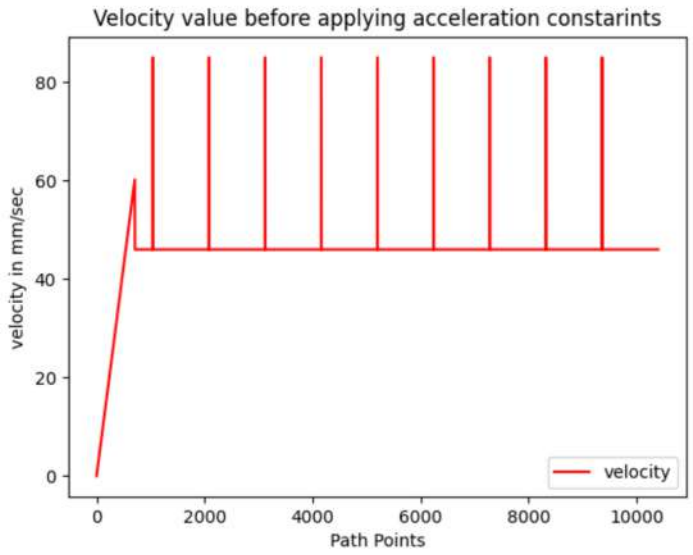
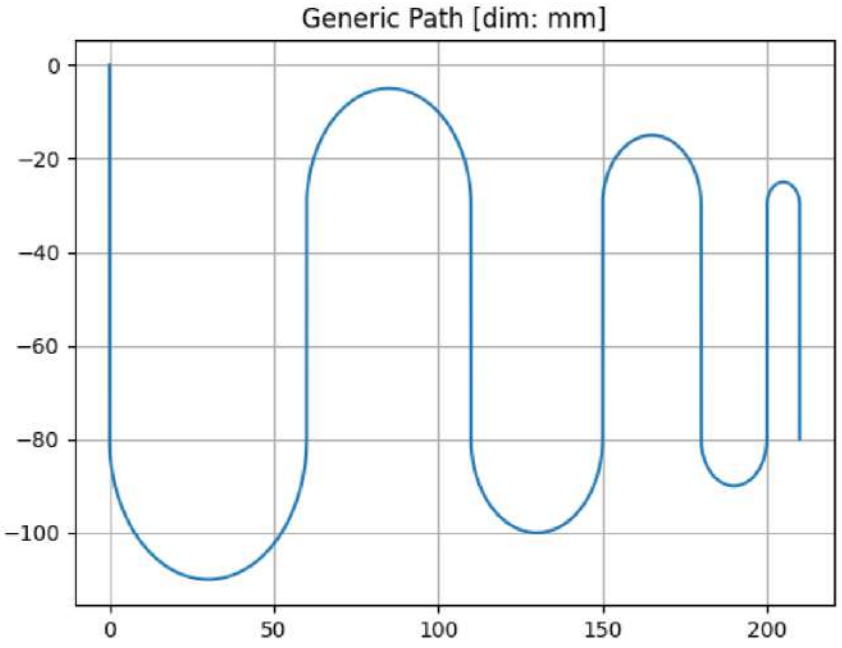


Figure 5. 5 : Velocity along the similar radius path

3. Different Radius Curves: A combination of straight lines and curves which are decreasing progressively at each end of the straight line (Figure 5. 6). The velocity values are shown in Figure 5. 7.



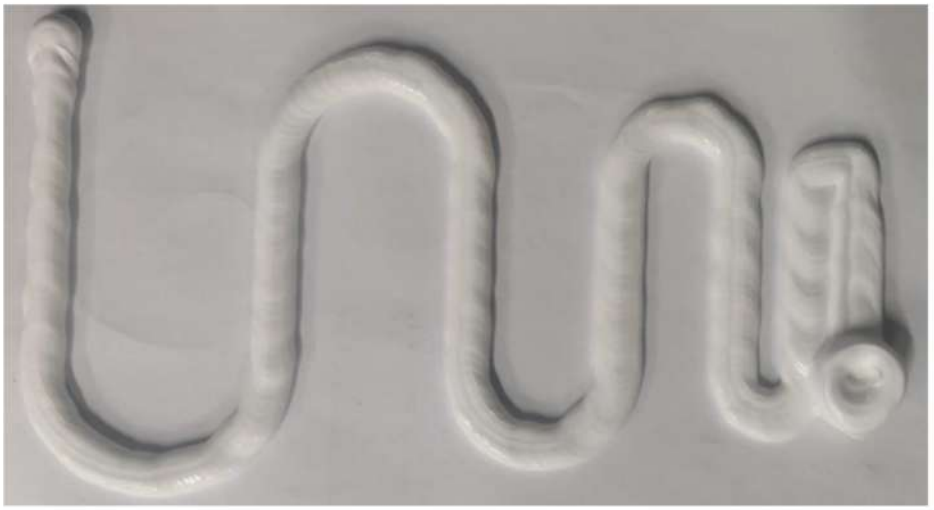


Figure 5. 6 : Different Radius curves : (a)Path Generation (b) Deposition

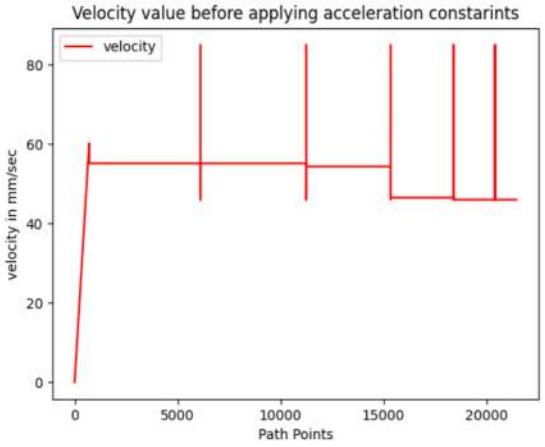


Figure 5. 7 : Velocity along the different radius path

4. A generic path: this has the combination of various sharp curves of radius 10mm, straight line, large curves of 50mm and saw tooth shaped structure having sharp turns at angles.

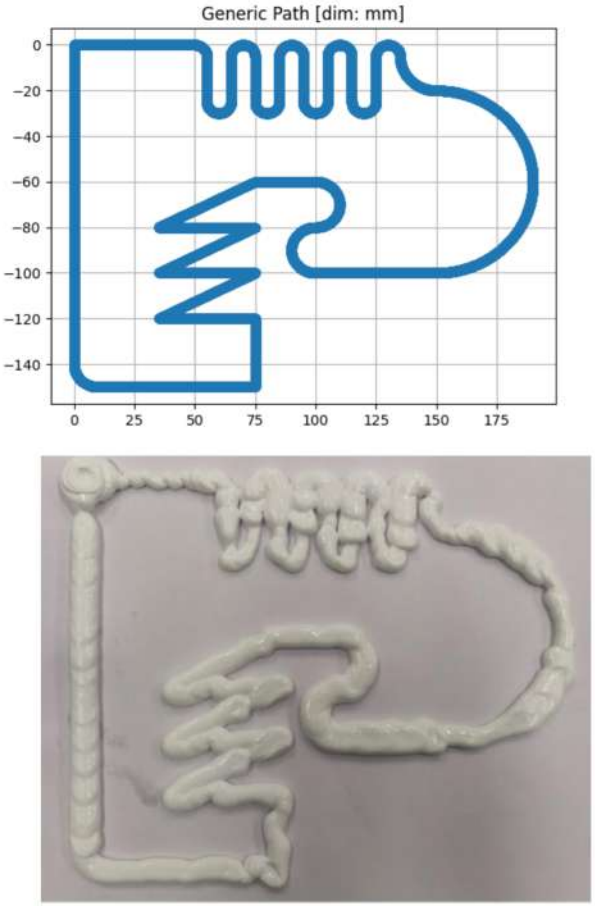


Figure 5. 8 : Generic Path: (a)Path Generation (b) Deposition

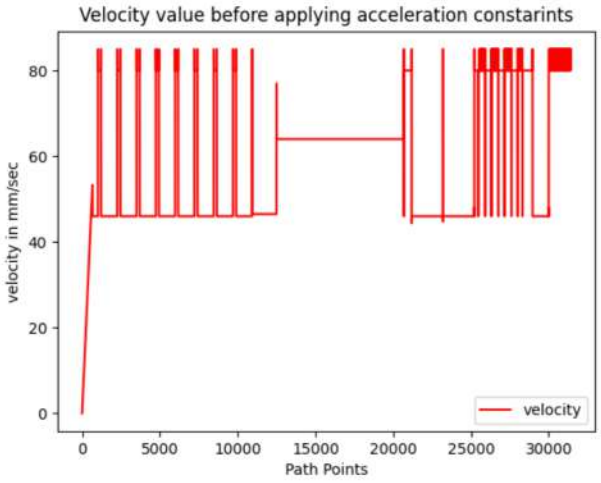


Figure 5. 9 : Velocity along the Generic path

The output for the generic path, that is the deposition quality (Figure 5. 8 (b)), was not uniform and desirable as the user had already preferred. This was due to the fact that the robot did not have enough time to increase its velocity to a desirable one and also the transition wasn't smooth enough to have a proper uniform deposition of velocity. This problem is being addressed in the following section where there are certain constraints implied to modify the velocity profile so that the hardware is able to get along with transition from sharp curves to straight lines in a more smoother way possible.

5.3 Acceleration and deceleration constraints

The Robot itself has its accelerations and deceleration constraints. The physical aspect of the current problem is that when the velocity values from Figure 5. 9 is implied for the robot, the motors at each joint of the robot tries to achieve the equivalent torque. But, when the velocity values are high in a short period of time, the motor at the joints reach their saturation level of torque trying to perform its maximum, thus creating an improper transition from the higher to lower velocities and vice-versa. This problem requires to smoothen the velocity profile so that we have a smoother transition to the maximum velocity value and also we have a maximum set limit for the acceleration value to control the torque value of the motors at joints (to prevent from reaching the maximum value). In Industrial robots the joint motors

are designed in such a way that they are able to handle higher torques and also lead to a smoother transition. But the current setup has a collaborative robot performing the task so the higher torque levels are not achievable. For this setup it was required to lower the acceleration values and also provide a smoother reference.

When there are higher acceleration limits implied to the robot (for eg., 180mm/s²) the robot starts to vibrate and most times the brakes are automatically applied on the joints for safety issues in case of further higher velocities. In our case we require a uniform deposition without vibration; hence the acceleration and deceleration limit was set to 150mm/s². By this way the velocity at each point of the path (v_i) is compared with the next one (v_{i+1}) and is being checked with the acceleration/deceleration limits. If the condition is satisfied the values remain unmodified, otherwise the velocity value is changed so as to satisfy this condition (Code in Appendix (ii)). The results of the test with 150 mm/s² are shown below (Figure 5. 10 and Figure 5. 11). The implication of acceleration constraints in the current setup led to better outputs in the quality of task even though the original velocity values were modified (Figure 5. 9 and Figure 5. 10). If the requirement is to track the original velocity reference without the constraints it may be possible with the help of a different experimental setup where the robot can handle higher velocities and its transitions.

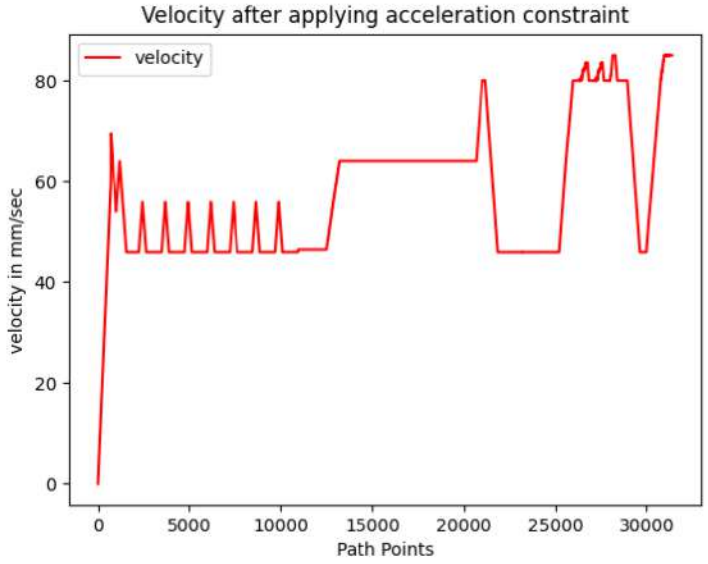


Figure 5. 10 : Velocity value along the generic path (Figure 5. 8(a)) after constraints implication

Figure 5. 11 : Deposition for the generic path with constrained velocity values for a uniform deposition

6 Conclusion

This thesis work deals with the trajectory planning of the robot where the velocity is planned depending on the path. The requirement for such a trajectory planner is needed for applications where the quality of task varies depending on the geometrical path. To enhance the task execution by reducing the task time and also not comprising on the task quality, by automatically characterizing the velocity at each point of the path is the main aim of this work. Since the caulking gun along with Arduino has poor repeatability and was no longer dependable for proper control over the material flow, it was decided to set the trigger position of the gun to maximum and then alter the velocity along the deposition path to obtain desired quality.

The initial experiments of the Active Preference Learning Optimization Algorithm are required to define the optimum and acceptable velocity ranges for the task based on the users likelihood for a range of curves. This requires the robot along with the deposition gun setup to perform the task on various curves, so that the user could choose his preference on various velocities and finally attain the optimal one. This data collected might vary from one user to another since the likelihood of a person is unquantifiable in a general way.

Once this data being collected, it is required a generic path for the deposition task and an algorithm which can find the radius of curvature at each point of the path. With the array of radius of curvature it can be implied the velocity at each point from the relation obtained from the data. With this procedure the deposition can be done basically in any kind of path.

This proposed approach is validated using the experimental testing procedure on the robot, where the varying velocities for the corresponding points of the path were implied and it allowed good deposition qualities on short curves and faster travelling speed on straight lines. For different test paths the results of sealant deposition, based on the varying velocity reference, was uniform and the output quality was as expected by the user. Also this approach did not require any manual tuning of parameters.

It was also required to implement an acceleration constraint to smoothen the velocity profile to have a more proper deposition in case of complicated paths. To improve the deposition quality on junction points (as from the straight line to sharp curve and then again straight line), it was also tried lower limits of acceleration, but limits lower than 95mm/s^2 showed poor deposition quality and the ones above that were the same quality of 150mm/s^2. After this smoother acceleration, the quality of deposition was enhanced with less jerks and vibrations. The downside with the setup was that since the robot is collaborative and not completely an industrial robot, there is a consequence of reduced precision. If there is a presence of external force, results deviate. The other downside was that the silicon cartridge responded with producing different material flow depending on the amount of material inside the cartridge. The fresh cartridge produced more flow of material while the almost finished cartridge produced less flow of material for the same amount of gun trigger position. For solving this issue, extrusion systems can be used where the flow of material can be controlled to be homogenous.

This work allows future possibilities of improving the deposition quality by introducing a Model Predictive Control which takes into consideration also the acceptability range velocity values and modify the reference velocity based on the acceleration limits. There is also the possibility of including the ROS node inside the Active Preference Learning algorithm to provide the robot iterative values directly, so that the whole process can be automated and time saving. Eventually, this procedure (as a general algorithm) can be adapted to any industrial process where a trajectory planning is required.

Appendix (i)

End effector mount

In this appendix there is a material list and some construction tips that may be useful for anyone who wants to build up the same setup used in this thesis, which is explained in [3.3]. The components needed to build and assemble the end effector are provided, together with the electrical circuit and also the code which runs on Arduino.

The 3D printed parts are the Base Flange (that is directly connected to the robot final joint); Top Flange (Figure A. 2 (b)) (which closes on the gun handle and by means of long screws, it clamps the gun. Both the motor and the mechanism have been inserted inside this component); Gear Mechanism (Figure A. 2 (c)) (it is composed by a rack and a pinion. The pinion is mounted on the motor standard flange (which needs to be cut to adapt it to the pinion)). The electrical board and the commercial Components are the Arduino Nano (electronic controller of the system. It receives as input the potentiometer and controls in output the servomotor); High speed motor (MG995, servomotor with a stall torque of 0.8 Nm. It can be directly fed by an Arduino board); Potentiometer (any kind can be used, preferrable a linear one instead of logarithmic. The full scale of the potentiometer must be set in order to match the Arduino A/D converter); The Caulking gun (which makes the deposition of the sealant material).

Trigger Control

The gun trigger can be controlled by automatically feeding the Arduino serial port directly with the Python ROS node. An integer number is written in the Arduino serial port (with a fixed publication rate): the input range [0 ÷ 99] is correlated with the two limit positions of the trigger. The arduino code is shown below:

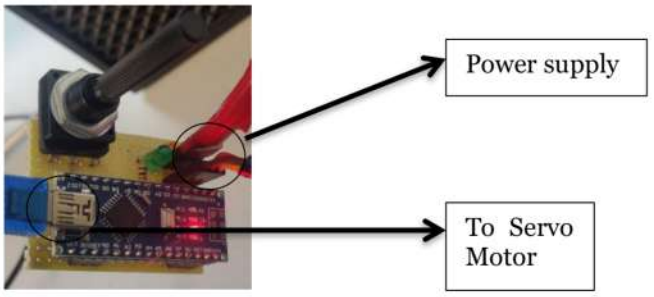
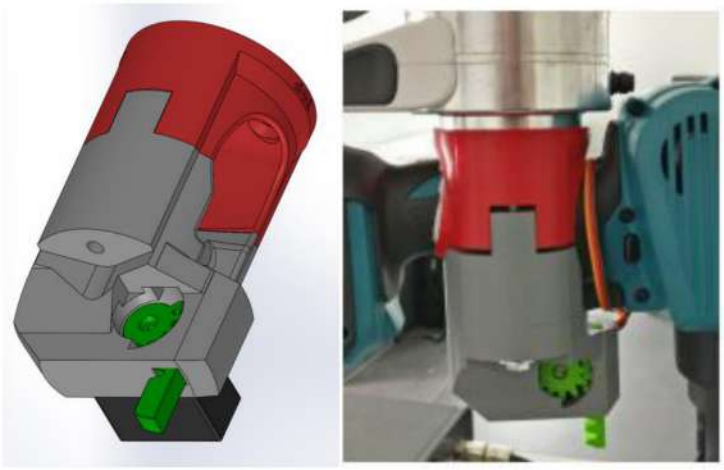


Figure A. 1 : Arduino Board and connections



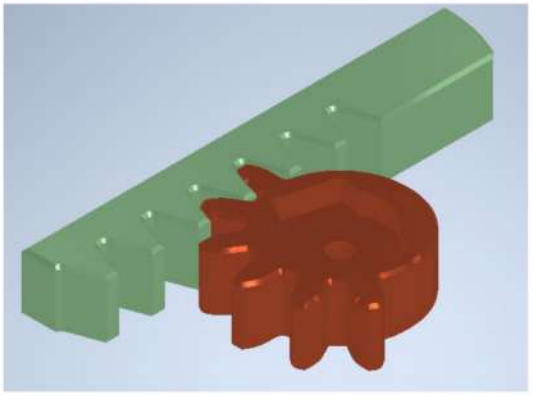


Figure A. 2 : (a) (Left) Design of The end effector mount with the caulking gun. (b) (Right)The real part of the end-effector. (c) Flange to hold the caulking gun. (d) the rack and pinion mechanism for trigger activation.

```
#include <Servo.h>
 const int pinServo = 4;
const int pinPot = A1;
const int pinLed = 3;   // just for debug
int pot = 512;    // potentiometer reading
String text = ""; // string value from the serial port
int alpha = 0;    // gun trigger pression [0-99]
int muS = 600;   // microseconds as input for the motor (1300)
bool MODE = 1;    // CHOOSE: 0 = potentiometer
                //        1 = serial input
void setup(){
  Serial.begin(115200);
  motor.attach(pinServo);
  // JUST TO SEE EVERYTHING IS WORKING
  digitalWrite(pinLed,HIGH);
  delay(100);
  digitalWrite(pinLed,LOW);
  delay(100);
  digitalWrite(pinLed,HIGH);
  delay(100);
  digitalWrite(pinLed,LOW);
  motor.writeMicroseconds(muS);
  }
void loop(){
  if(MODE==0) { // POTENTIOMETER
    pot = analogRead(pinPot);
```

```
  muS = map(pot,20,1000,1300,1930); // 1400-1930
  motor.writeMicroseconds(muS);
  delay(10);
  } else { // SERIAL READING
   while (Serial.available() > 0) {
   char inChar = Serial.read();
   text += inChar;

   // if you get a newline
   if (inChar == '\n') {
    alpha = text.toInt();

    // MOTOR COMMAND
    muS = map(alpha,0,99,150,1400); // 1400-1930 (1300 instead of 200)
    motor.writeMicroseconds(muS);

    // clear the string for new input:
    text = "";
    }
   }
  }
}
```

Appendix (ii)

Algorithmic development: The full length of the code is not reported, but the most significant parts which helps with the important working of the algorithm is shown here.

Code: Curvature Analysis:

As explained in the section (5.1), this part of the code is inserted where the X and Y coordinate points are available with the points refitted according to the diameter of the nozzle.

```
x_t = np.gradient(X)
y_t = np.gradient(Y)

xx_t = np.gradient(x_t)
yy_t = np.gradient(y_t)

curv = (np.abs(xx_t  y_t - x_t  yy_t) / (x_t  x_t + y_t  y_t)**1.5)
radius_curv=np.zeros(len(curv))
for i in range(len(curv)-1):
   if curv[i]==0:
      radius_curv[i]=100
   else:
      radius_curv[i]=1/curv[i]
```

Code: Acceleration Limits

As explained in Section (5.3) this code is inserted after the estimation of the reference velocity at each point of the path, so that there is a smoother transition of the deposition task.

```
for i in range (len(radius_curv)):
    t_inst=4/(velocity[i-1]+velocity[i])
    inst_acc=(velocity[i]-velocity[i-1])/t_inst
    if abs(inst_acc)>(acc_limit):
        if inst_acc > 0: #acceleartion
            while (inst_acc>acc_limit) :  #and (velocity[i]>=velocity_lower[i-1])
                velocity[i] = velocity[i] - (acc_limit*t_inst)
                inst_acc = (velocity[i] - velocity[i-1]) / t_inst
        else: #deceleration
            while(abs(inst_acc)>(acc_limit)):
                velocity[i] = velocity[i] + (acc_limit*t_inst)
                inst_acc = (velocity[i] - velocity[i-1]) / t_inst
```

Code: ROS Node

The Python ROS node which publishes the commanded position to the robot controller and also sends trigger position values to the Arduino board is reported here.

```
import rospy
import serial
import time
from geometry_msgs.msg import PoseStamped
from std_msgs.msg import Header

ard = serial.Serial('/dev/ttyUSB0',115200,timeout=0)
time.sleep(2.5)
ard.write('90\n')
time.sleep(1.2)
pwm = 1

try:
   pub = rospy.Publisher('/DMP_pose', PoseStamped, queue_size=1)
   rospy.init_node('DMP_planner', anonymous=True)
   rate = rospy.Rate(1/dt) # 1kHz

   p = PoseStamped()

   task_ended = False
   while not rospy.is_shutdown():
      if not task_ended:
         for i in range(len(x)):
```

```
            p.pose.position.x = round(x[i],5);
            p.pose.position.y = round(y[i],5);
            p.pose.position.z = round(z[i],5);
            p.pose.orientation.x = round(qx[i],6);
            p.pose.orientation.y = round(qy[i],6);
            p.pose.orientation.z = round(qz[i],6);
            p.pose.orientation.w = round(qw[i],6);
            p.header.stamp.secs=rospy.get_time()



            if i < (len(x)-3300):
               ard.write(str(90))
            else:
               ard.write('0\n')

            pub.publish(p)
            rate.sleep()

         task_ended=True
         ard.write('0\n')
         ard.close()

      pub.publish(p)
      rate.sleep()

except rospy.ROSInterruptException:
   pass
```

BIBLIOGRAPHY

1. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries.* **Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus.** 2015.

2. *Robots, Industry 4.0 and Humans, or Why Assembly Work Is More than Routine Work.* **Pfeiffer, Sabine.** s.l. : Societies , 2016.

3. *Control of robot manipulators in joint space.* **R. Kelly, V. S. Davila, and J. A. L. Perez.** s.l. : Springer Science & Business Media, 2006.

4. *Automation and robotics in the context of Industry 4.0: the shift to collaborative robots.* **Meshcheryakov, R and Galin, R.** IOP Conference Series: Materials Science and Engineering, s.l. : IOP Publishing, 2019, Vol. 537. 3.

5. *Robotics: modelling, planning and control.* **Siciliano, B.** s.l. : Springer Science & Business Media, 2010.

6. *Euler angles, quaternions, and transformation matrices for Space Shuttle analysis.* **Henderson, D. M. .** 1977.

7. *A tutorial on Euler Angles and quaternions.* **M, Ben-Ari.** Israel : Weizmann Institute of Science, 2014.

8. *Trajectory planning for automatic machines and robots.* **L, Biagiotti and C, Melchiorri.** s.l. : Springer Science & Business Media, 2008.

9. *FIR filter-based online jerk-constrained trajectory generation.* **P, Besset and R, Béarée.** 66, s.l. : Control Engineering Practice, 2017, pp. 169–180.

10. *Jerk-bounded manipulator trajectory planning: design for real-time applications.* **S, Macfarlane and E.A, Croft.** 19.1, s.l. : IEEE Transactions on robotics and automation, 2003, pp. 42–52.

11. *Smooth trajectory generation for industrial robots performing high precision assembly processes.* **A, Valente, S, Baraldo and E, Carpanzano.** 66.1, s.l. : CIRP Annals, 2017, pp. 17–20.

12. *Planning the sequencing of movement primitives.* **M, Kallmann, R, Bargmann and M, Mataric.** s.l. : proceedings of the international conference on simulation of adaptive behavior (SAB), 2004, pp. 193–200.

13. *Path-Parameterization Approach Using Trajectory Primitives for Three-Dimensional Motion Planning.* **A. J., Pachikara, J. J., Kehoe and R., Lind.** s.l. : Journal of Aerospace Engineering, 2013, Vol. 26.3, pp. 571–585.

14. *Simulation and trajectory generation of dual-robot collaborative welding for intersecting pipes.* **J., Xiong and al., et.** s.l. : The International Journal of Advanced Manufacturing Technology, 2020, pp. 1-11.

15. *Cartesian path planning for arc welding robots: evaluation of the descartes algorithm.* **J., De Maeyer, B., Moyaers and E., Demeester.** s.l. : 22nd IEEE International Conference on Emerging Technologies. IEEE, 2017, pp. 1-8.

16. *A novel trajectory generation method for robot control.* **K., Ning.** s.l. : Journal of Intelligent & Robotic Systems, 2012, Vol. 68.2, pp. 165–184.

17. *An Optimized Trajectory Planning forWelding Robot.* **Z., Chen.** 012009, s.l. : Materials Science and Engineering, 2018, Vol. 324.1.

18. *Online trajectory planning and filtering for robotic applications via b-spline smoothing filters.* **L., Biagiotti and C., Melchiorri.** s.l. : 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE., pp. 5668–5673.

19. *Active reward Learning.* **C., Daniel, et al., et al.**

20. *User-centered design and interactive health technologies for patients.* **A. D. V., Dabbs, et al., et al.** s.l. : CIN: Computers,Informatics, Nursing, 2009, Vol. 175.

21. *Deep reinforcement learning from human preferences.* **P., Christiano, et al., et al.** s.l. : arXiv preprint, 2017. 1706.03741.

22. *Multi-objective bayesian optimisation with preferences over objectives.* **M., Abdolshah, et al., et al.** s.l. : Advances in Neural Information Processing Systems (NeurIPS), 2019.

23. *A particle swarm pattern search method for bound constrained global optimization.* **A. I. F., Vaz and L. N., Vicente.** s.l. : Journal of Global Optimization volume, 2007, Vol. 39.

24. *Global optimization based on active preference learning with radial basis functions.* **A, Bemporad and D, Piga.** s.l. : Machine Learning, 2020, Vol. 110, pp. 417-448.

25. Franka Emika official website. [Online] https://www.franka.de/technology.

26. *Franka Control Interface documentation.* [Online] https://frankaemika.github.io/docs/overview.html.

27. *Dynamics Parametrization and Calibration of Flexible-Joint Collaborative Industrial Robot Manipulators.* **E., Madsen.** s.l. : Mathematical Problems in Engineering 2020, 2020.

28. *Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization.* **C., Gaz.** 4.4, s.l. : IEEE Robotics and Automation Letters, 2019, pp. 4147–4154.

29. *Robot dynamics constraint for inverse kinematics.* **E. M., Hoffman.** s.l. : Springer, 2018, Vol. Advances in Robot Kinematics 2016, pp. 275–283.

30. **M., Magni and M., Cantoni.** Human-robot collaboration in assembly task learning enhanced by uncertainties adaptation via Bayesian optimization. [Online] 2019. https://www.politesi.polimi.it/handle/10589/150736.

31. *Cartesian impedance control of redundant and flexible-joint robots.* **C., Ott.** s.l. : Springer, 2008.

32. *https://makita.com.sg/product/dcg180-cordless-caulking-gun/.* [Online]