Alma Mater Studiorum - Università di Bologna

SCHOOL OF ENGINEERING
DEPARTMENT of
ELECTRICAL, ELECTRONIC AND INFORMATION ENGINEERING
"Guglielmo Marconi"
DEI
Master Degree in Automation Engineering

Robotic Manipulation of DLOs for Wiring Harness Assembly: a Machine Learning Approach

Master Degree Thesis
in
Autonomous and Mobile Robotics

Supervisor:
Prof. Gianluca Palli

Co-Supervisor:
Kevin Galassi

Candidate:
Alex Pasquali

Academic Year
2021/2022
II Session

Abstract

In recent times, a significant research effort has been focused on how deformable linear objects (DLOs) can be manipulated for real world applications such as assembly of wiring harnesses for the automotive and aerospace sector. This represents an open topic because of the difficulties in modelling accurately the behaviour of these objects and simulate a task involving their manipulation, considering a variety of different scenarios. These problems have led to the development of data-driven techniques in which machine learning techniques are exploited to obtain reliable solutions. However, this approach makes the solution difficult to be extended, since the learning must be replicated almost from scratch as the scenario changes. It follows that some model-based methodology must be introduced to generalize the results and reduce the training effort accordingly. The objective of this thesis is to develop a solution for the DLOs manipulation to assemble a wiring harness for the automotive sector based on adaptation of a base trajectory set by means of reinforcement learning methods. The idea is to create a trajectory planning software capable of solving the proposed task, reducing where possible the learning time, which is done in real time, but at the same time presenting suitable performance and reliability. The solution has been implemented on a collaborative 7-DOFs Panda robot at the Laboratory of Automation and Robotics of the University of Bologna. Experimental results are reported showing how the robot is capable of optimizing the manipulation of the DLOs gaining experience along the task repetition, but showing at the same time a high success rate from the very beginning of the learning phase.

Contents

List of Figures

Chapter 1

Introduction

In recent years, automatic applications are widely used in a wide variety of scenarios, this due to the ever-increasing demand for productive and efficient capabilities of machines producing everyday goods such as food or medicine. Today's automatic machines consist of a multitude of drives and sensors connected by electrical wires that distribute power and signals. The handling of these wires is still done by hand, such as the organization of the wires into an electrical cabinet, where they are stored in a safe position, or the wiring of vehicles; in this work the wires are placed on a table to be organized. The wires are then grouped into different bundles, depending on the final destination of the wire in the vehicle.
Despite the large amount of automation, machine switchboards or wiring harnesses are still assembled by a human operator and no robots or automated processes are involved. It is important to study the feasibility of a possible robotic implementation in order to understand all the difficulties and limitations this may entail. It should be remembered that the operation of a robot must always be a conscious decision [1], in fact a robotic arm is a very complex mechanism and its control is a task that requires solid knowledge of various disciplines such as electronics, mechanics and computer science. For this reason, few companies have the resources to invest in these technologies and have a built in-house robotic arm. Acquiring the technology from an external company will inevitably raise the price of the machine for the end buyer and reduce the profit margin for the company, making the robot an unprofitable technology to use. The overall price increases further if the robot is designed and authorized to work with humans, due to the necessary sensors and certification. However, there are situations where the use of a robot is strictly necessary due to external causes or a possible dangerous or inaccessible environment. An example might be the handling or exposure of chemical and toxic material in the pharmaceutical industries, or the handling

of viruses for vaccine production. In applications requiring the handling of heavy loads, a robot may also be used, since the maximum weight that a human worker can move is strictly determined by government regulations.

Wiring a cable is not a heavy task nor dangerous to the health of the people involved, however the lack of automatic solutions for the manipulations of deformable objects make this application a possible investing opportunity, these technologies can open up a new frontier for the manipulation of deformable objects or the development of useful techniques and tools for other applications.
This thesis project was conducted at the Laboratory of Automation and Robotics (LAR) of the University of Bologna and the activities covered are related to the European REMODEL project. The aim of the project is to develop the technologies required to manipulate flexible objects in different industrial application, one of this include to complete a cabling operation 1.1.

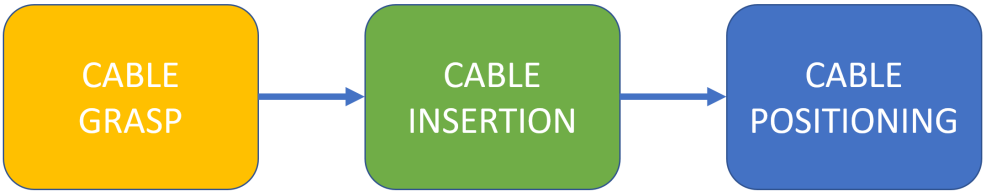


Figure 1.1: Cabling operation divided in sub-task

The cabling operation can be divided into three sequential steps:

1. The grasping of the cable from a magazine;
2. The insertion of the cable's connector into the connector block, so that the position of one end of the cable remains fixed;
3. The positioning of the cable, in the required clips, so that the cables are correctly organized.

The main topic of the thesis is the positioning of multiple cables along a given path; in this problem, it must also take into account the issue of wire tension by means of a force/torque sensor mounted on the end-effector. The robot used to tackle the problem is the 'Panda' robot, manufactured by the German company Franka-Emika. The robot belongs to the category of CoBots, an acronym for Collaborative Robots, i.e. machines designed to work in contact with humans in complete safety. Furthermore, the robot used is a 7 DOF robot and is able to move its end-effector in any position and orientation in its workspace with different configurations of its arm.

1 REMODEL Project

The REMODEL Project [2], acronyms for "Robotic tEchnologies for the Manipulation Of complex DEformable Linear objects" is a project financed by the European Union and started in 2020, involving several universities and commercial partners across the Europe.
The focus of the project is to develop the knowledge and the tools required for the manipulation of deformable object such as wires, electric cable and wiring harness by a robotic arm. This type of materials have a wide field of application, from the automotive manufacturing to the automatic machines sector, but they are also used in medical application. The objective is the realization of an automatic robotic platform capable to manage deformable linear objects, the identification through computer vision technologies of such objects in non-optimal conditions and their modelling through FEM software and numerical analysis. Part of the study will cover the design of specialized end-effectors capable to hold and manipulate the wires by means of special tactile sensor orientated toward the simulation of the human's fingertip interaction with the object; the designed EE will also accomplish other task required to complete a wiring operation such as securing a screw to hold the wires in position. The case studies are:

- Switch-gear wiring
- Wiring harnesses manufacturing
- Wiring harness assembly
- Hose packaging

Figure 1.2: REMODEL Project logo

The university of Bologna holds the role of project coordinator, apart that it will focus on the cable/wire detection, their grasping and their manipulation. The project involves important company from Europe, such as IEMA a company, held by IMA group, ELIMCO, a spanish company working on the aerospace and defence and ELVEZ, slovenian manufacturer of specialized products for the automotive industry and ENKI an Italian company working on the production of plastic medical hoses.

2 ELVEZ Case Study

Elvez is an advanced manufacturing company specialized in providing clients worldwide with plastic injection components, metallized parts and cable harness solutions for partners such as BMW, Volkswagen, Mercedes, Volvo, Scania, ZKW, John Deere, PSA, Renault, Mahle, McLaren, etc.
This company deals with the problem of wiring harness, this operation is still done manually by operators, so the goal of this thesis is to make an autonomous system that can accomplish this task.

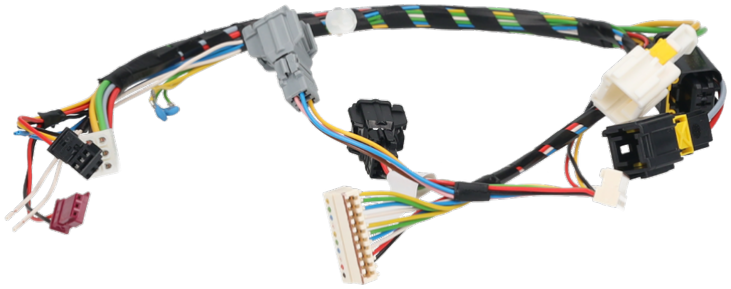


Figure 1.3: Wiring harness problem

The project focuses on the task of correctly positioning the cables in an organized manner in order to achieve the desired result. In particular, we have three groupings of cables distinguished by a different connector and characterized by a different diameter of the wires. To distinguish the sub-harnesses, they are identified by the numbers of wires connected at each connector:

- A six-pole connector (SPC);
- A ten-pole connector (TPC);
- An eleven-pole connector (EPC).

Chapter 2

Robot Set-up description

The project will investigate the possibility of introducing a robotic arm in operations where manipulation of deformable objects (DLOs) is required. The robot will have to deal with different types of cables and different paths, all in order to obtain a single ordered set of cables. The solution developed will be based on ROS, an open source operating system. In particular, a panda robot made by Franka Emika will be used. The robot will be equipped with a custom gripper capable of dragging, with a good compromise on the forces exchanged in contact, the cables.

1 ROS

ROS [3] is an open-source operating system for robotic applications. The idea of ROS is to develop an high level control that can communicate with the lower level of the robot. This high level controller is an abstraction, meaning that the program written for a robotic platform can be reused eventually with other robot even from different producers if their interface with ROS is provided. The key idea of ROS is the fact that it is a distributed framework of processes that run concurrently, each process, called "node", is meant to work independently of each-other and represents a different module. The nodes are capable to share information across-each other through a communication system based on the concept of "topic" and "messages".

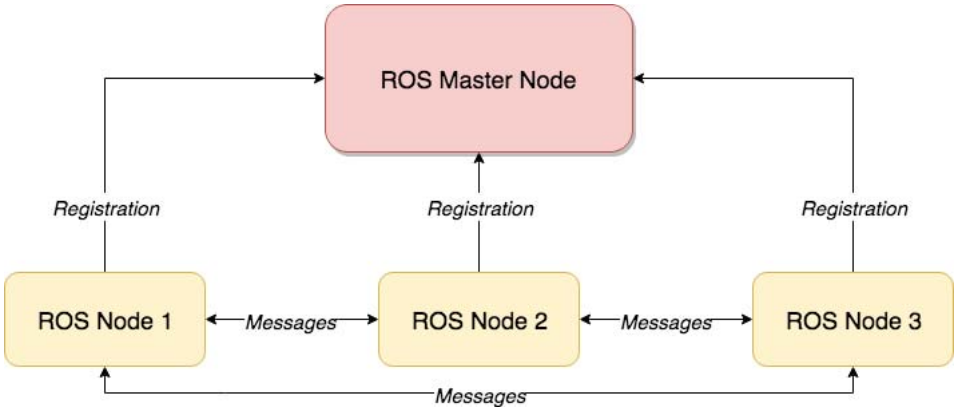


Figure 2.1: ROS communication scheme, nodes are registered to a central master node. During their execution, nodes communicate between each other with messages, services and actions

1.1 Nodes

In ROS each program executed is denoted as a Node, it can be seen as a single entity running concurrently in the system. Each node can communicate with the other nodes by using a server-client communication architecture, in which each node can work as client but also as server. To communicate, a node must be connected with a master node, which purpose is to enable the communication between all the node registered to it. Nodes can communicate between themselves through the direct invocation of another node services or actions or through publish/subscribe. A ROS−based robot control system is usually composed by many nodes. Each node should be designed to have a small and specific task. Nodes should perform their own tasks and exchange the results with other nodes. This net of elaboration of data will form a complex graph−like structure capable of solving demanding problems. Node−based architecture provides ROS with many benefits, where the biggest benefits are fault tolerance (as each node is an isolated part of the system) and reduced code complexity compared to monolithic systems, which are not decoupled.

1.2 Topics

ROS topics are the "mailboxes" used by nodes to communicate. This communication is based on a publish/subscribe mechanism. Each ROS topic has a unique name associated, so that ROS nodes can publish or subscribe to it. Any node is enabled to publish or subscribe to any ROS topic as long

it′ known the name of the topic. Furthermore, there is no limitation on the number of topics to which a node can be subscribed or publish to. As this communication is not direct, nodes are not aware who are they getting the data from or who are they sending the data to.

1.3 Messages

ROS messages are exchanged between ROS nodes using publish/subscribe mechanism. One ROS node would publish the ROS message to a certain ROS topic, while the other ROS node would subscribe to that ROS topic and obtain the ROS message sent. ROS Messages are described as .txt files inside a specific folder called msgs under in the ROS package folder structure. Each ROS message is described with a data structure which contains only primitive types like integers, floats or booleans, it also can include arrays of the primitive types listed earlier. A ROS message can also include other ROS message. In this way, it is possible to create complex and longer messages. Additionally, ROS message can contain the other ROS message or an array of ROS messages as a data type. ROS messages can also be exchanged in a direct communication between nodes. This mechanism is called "Services".

Figure 2.2: Visual concept of the message communication scheme in ROS

1.4 Services

Topics can be seen as named "mailboxes", where a node can publish or read messages. In this context there is no direct connections between the nodes or knowledge of the topics connected which are reading the published messages. ROS services instead are used when there is a wish for nodes to be able to communicate directly with each other. By using the ROS services, publish/subscribe mechanism is avoided and nodes can send requests and replies to each other directly using the defined srv. Like the messages, also the services have their target folder named "srv" in which a custom services can be defined, each srv is a .txt files in containing his two part, the request sent from the client node, and the response from the Server node. Since

the ROS services are a form of direct communication, they are increasing the performance of the system, however they are decreasing the system′s decoupling.

1.5 Action

ROS actions can be seen as an extension of the services capability, since in some case the services may be an operation that require a certain amount of time to be executed. Before the communication of the response from the server take place, it can occur an event that changes the system condition so that the execution of the service is no longer needed, in this case an Action is required. In a more general way, the action are needed for more complex group of operation. The program can call it and continue the execution of the program with still the possibility to cancel a request, get periodic feedback from the server or to have a queue of request. An action message is composed by three part: Goal, Feedback and Result. The Goal that is sent to the ActionServer by an ActionClient is the part of the message that contains information about the objective to execute, like a position or a series of parameters. With the Feedback, the ActionServer is able to tell to an ActionClient about the incremental progress of a goal. Finally, the result is sent to the ActionClient to conclude the action providing information about the correct execution of the task or some other feedback from the ActionServer.

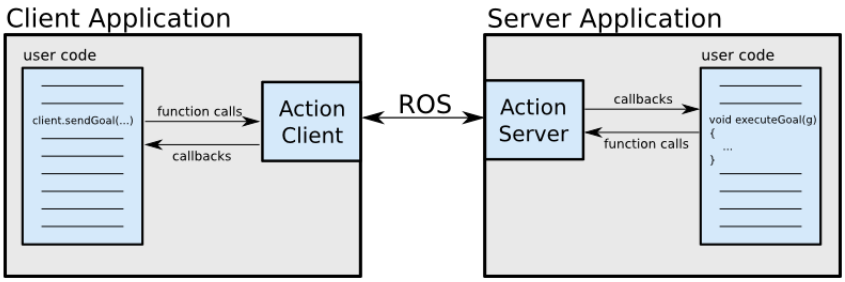


Figure 2.3: ROS Action working scheme, the action is triggered from a Client, the Server start the execution of the task and produce some feedback or a return value that the client can read.

2 Moveit

MoveIt! [4] is a plug−in package based on ROS that incorporates several functions for motion planning, manipulation, and kinematics. The Motion

planners are able to consider both the trajectory planning and the execution of the robot motion. In MoveIt! the motion planners are loaded using a plugin infrastructure, allowing the package to communicate at run-time with different planners from multiple libraries.MoveIt! offers user-friendly functionalities for most operations that a user may want to carry out, specifically setting joint or pose goals, creating motion plans, moving the robot, and adding objects into the environment. Indeed, the planning of a trajectory can be accomplished by just setting the current robot configuration and the specified target. The default motion planners in MoveIt! are configured to use OMPL [5], which is an open−source motion planning library that implements randomized planners. Depending on the planner used, MoveIt! can choose between joint space and Cartesian space for the representation of the problem. Natively, planning request switch orientation path constraints are sampled in Cartesian space. OMPL also provides several sample-based optimal planning algorithms. Some of them use a general framework to express the cost of robot configurations and paths, allowing to, e.g., maximize the minimum clearance along a path, minimize the mechanical work, or some arbitrary user−defined optimization criterion. However, the convergence to optimality is not guaranteed when optimizing over some index or metric which is not the path length.

3 Franka Emika: Panda

The robotic platform used in this project is the Panda [6]. This robot has multiple advantages. The robot has 7DOF, which is an important characteristic for a robotic arm, since it his able to positioning and orientate the end-effector in the whatever point inside the workspace, the extra degree of freedom let the robot able extent the position capability and eventually be able to perform motion avoidance algorithm. Another key feature of the design is the force sensors available, with that the robot is capable to detect if it enters in contact with another object or a person and then stop the execution of the operation with a low response time of $\sim$50ms to avoid possible dangerous situation. This characteristic is fundamental in order to design a human-machine interaction.

Figure 2.4: The 7DoF Panda robot from Franka Emika with the proprietary gripper installed

3.1 Franka ROS

Franka_ros is the integration of the libfranka package in ROS and ROS control, this enables the user to control the robot using the ROS structure, implementing ROS action for the gripper, or hardware abstraction of the robot for the control framework. This also enable to launch the robot arm and gripper controller as ROS node, controllable by using ROS action and reading the state from the node using the publisher/subscriber approach. The ROS implementation is also important because a working program based on ROS can be used also with different robot (of different manufacturer) with a proper ROS interface.

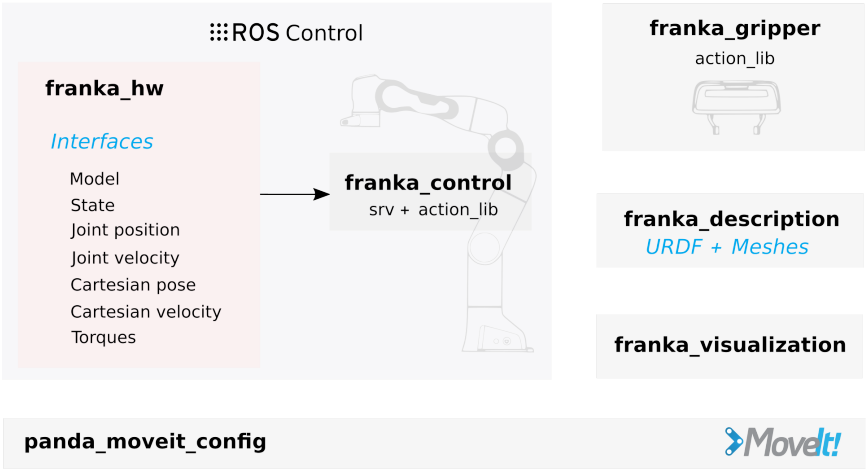


Figure 2.5: Franka control interface scheme

4 Custom Gripper Fingers

The gripper fingers, needed by this specific application, require three different fundamental properties:

- Keeping the body of the robot at a proper distance from the environment in which the positioning of the cables is carried out, to avoid collisions and reduce encumbrances;
- Having an elastic part that takes care of the contact with the cables, which can be of different diameters, allowing the approach of transporting the wires to remain flexible;
- Having rigid strikers that allow the cables to be inserted into the appropriate clips and at the same time not allow the wires to come loose from the gripper.

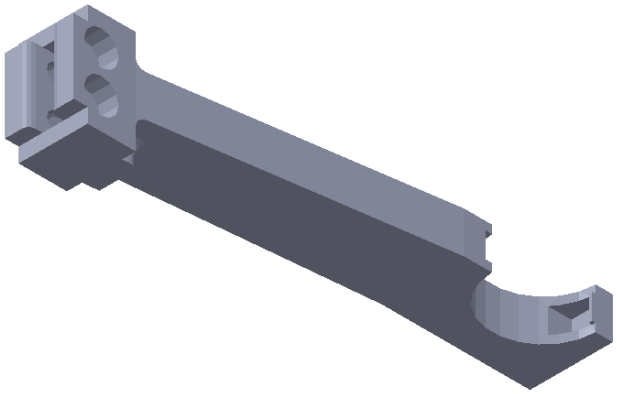


Figure 2.6: Custom gripper finger CAD view

Chapter 3

Environment Set-up description

This chapter focuses on the description of all the equipment and expedients that were used to create an optimal environment for the proper development of the application. This is a wired harness assembly application and thus we have the task of correctly routing different types of cable groupings, with common path sections, in order to obtain a single ordered grouping as final product.

1 Connector Blocks

In the Elvez case study (Sec. 2), we have seen how three different types of wiring will need to be handled: SPC, TPC and EPC. The application we desire to realize assumes as an initial condition that the groupings already have a terminal part locked in connector terminal blocks. The realization of these components was carried out by 3D printing after an in-depth study of the design in a CAD environment. The design of these elements was realized not only to hold the connectors in place, but also to enable the future development of a connector insertion application, which is an integral part of the overall project.

1.1 SPC Connector Block

In the specific case of the SPC, a connector block was created using a slot having the shape of the connector, limiting as much as possible the movement of the connector once it was inserted. At the top of the connector block there is a corridor for the passage of the wire, this allows the connector to be inserted from above, and the insertion of the connector into the slot is facilitated by the presence of appropriate chamfers and rounding.

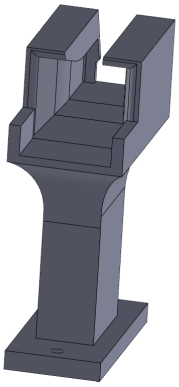


Figure 3.1: CAD SPC connector block

1.2 TPC Connector Block

The insertion principle of the TPC and thus the construction of its connector block are similar to those described for the previous design. What changes in this case is the shape of the connector shape, and having thinner cables, the insertion corridor is also thinner.

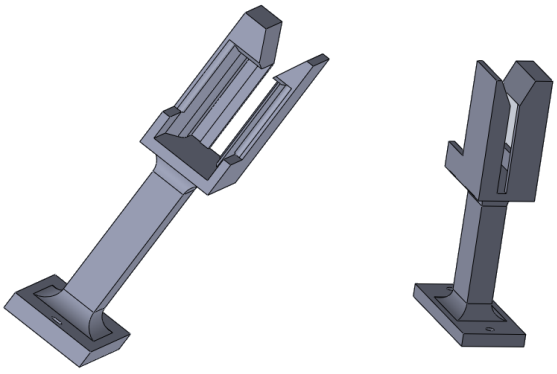


Figure 3.2: CAD view TPC connector block

1.3 EPC Connector Block

The last connector block has a completely identical design to the previous one except for the height which is a few millimetres higher, this is because the connector shape is the same but has an extra wire.

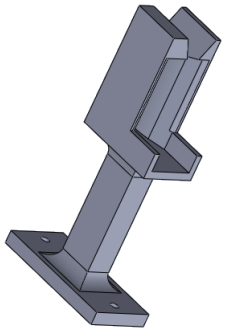


Figure 3.3: CAD view EPC connector block

2 Clips

In the work plane, we also needed components capable of grouping cables, so that we could apply tape closures between the various wires. To fulfil this purpose, clips were designed, again in a CAD environment and subsequently 3D printed, with passive retention and insertion. These clips must also work for different overall cable diameters, so their design was complex and required several attempts. It was decided to equip them with a flexible part with a wide chamfer to guide the insertion, this detail provided a very good chance of success even in the case of small misalignment in the cables' insertion.

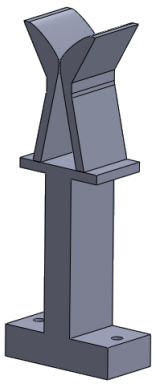


Figure 3.4: Clip CAD view

3 Cameras

For the application to work properly, it was necessary to consider a good vision system for two main reasons:

- To suppress misalignments of the various paths due to movements of the work table with respect to the robot;
- To understand what is happening on the work surface, i.e. how the cables are arranged and whether the insertions were successful or not.

Solving these two problems will allow the learning system, as explained in detail in the following chapters, to work in a facilitated manner and with an excellent knowledge of what is happening in the environment.

3.1 Camera for Alignment

To suppress misalignments, a RealSense 3D camera attached to the end-effector of the robotic arm was used. In this way, after appropriate calibration of the camera, the paths to be taken can be realigned on the basis of certain strategic points in the working plane, such as the arrangement of an Aruco marker in a corner of the plane.

Figure 3.5: RealSense 3D camera

3.2 Camera for Binarization

For the detection of the working surface, a simple 2D camera fixed at a sufficiently high position relative to the plane was used. The use of this camera allows, with the necessary precautions, the realization of a binary mask of the plane in which the cables will be clearly visible. It was necessary to create a small structure to keep the camera well positioned and at the correct height.

Figure 3.6: 2D camera

4 Background and Lights

As for the adjustments that are needed on the working plane to enable the 2D camera to be able to produce a mask, we certainly have the problem of contrast between the background and the cables. To solve this issue, it was decided to create a high-contrast background with multiples colours, ideally all the colors that the cables might have. The idea of this background is to create a high density chequerboard of squares where each cell takes on a random colour. In this way, the specific colour of a cable can only interfere in small areas of the background and thus the binarization is still efficient.

Figure 3.7: Working plane background

Another major obstacle to creation of a binary mask of an object is the problem of shadows. Shadows may be recognized as an object and thus binarized incorrectly in the mask. Anti-shadow lights are used to overcome this issue.

Figure 3.8: Anti-shadow lights

5 Force Sensor

To understand the interactions between the robot and the environment, a force sensor has been mounted on the gripper's end-effector. In particular, this sensor can be used to evaluate the forces exchanged between the gripper and the cables during the wired harness assembly process. The sensor used is a Nordbo Robotics and provides real-time knowledge of all the forces and torques it is subjected to.

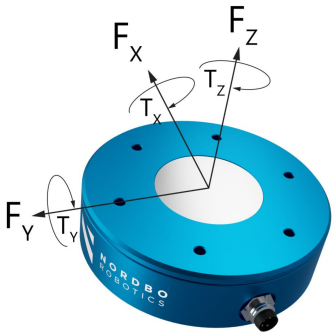


Figure 3.9: Nordbo force sensor

Chapter 4

Software Structure

The application is implemented on the ROS environment, initially it was appropriate to define a general software structure that could contain all the fundamental components. This was done in order to have a first complete idea of how the problem should be handled. From the implementation point of view, the objective is to transport each cable from their starting point (A connector or a clip) and insert them into the appropriate clips, for the purpose we need a software portion (ROS node) that takes care of checking the correct insertion. We also need a high-level node that takes care of the learning, i.e. managing the exploration and making decisions for improvement in the execution of the task. Finally, we need a low-level node to physically make the robot execute the paths and record the exchange of forces between the gripper and the cables in the trajectories made.

1 General Scheme

The diagram 4.1 presents the solution, in terms of software communication, required to perform the task.

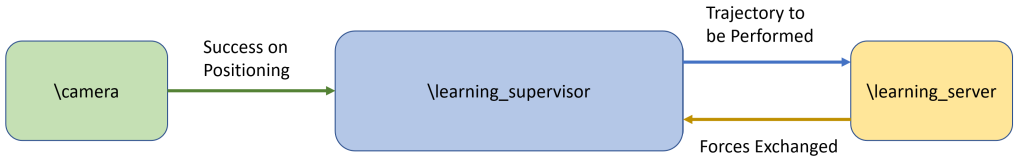


Figure 4.1: General nodes scheme

The camera node must take care of verifying the correct insertion of the cables, using video capture of what is happening in the environment. Each

time a cable is positioned, the node should send a message informing about the success or failure of the operation. For this reason this node has been constructed as a ROS service, every time another node needs this information, it will call the service and receive the feedback.
The learning server node, on the other hand, is the low-level software portion that has to deal with interacting to the robot interface and the force sensor, in order to work it needs a high-level command with the information regarding the path to be taken. This task is not as short-lived as that assigned to the camera node, which is why a ROS action server was chosen, which, unlike the ROS service, allows continuous interaction between the client and the server.
As far as the learning supervisor node is concerned, we can say that it is the heart of this software structure and that it has to take care of all the tasks involving learning, the next chapter will deal extensively with the realization of this node.

2 Nodes

This section aims to explain in more detail the realization of the necessary nodes, explaining how and what communication channels are constructed. It also shows in detail the computational operations performed by the nodes.

2.1 Binarization Camera

The camera node has the task of verifying the positions of the cables on the work surface. To do this, the idea was to use a simple 'find the differences' approach between two images. Starting with a static image of the work area and thus identifying the background, it is possible to analyse the differences between this image and a dynamic one in which the cables are positioned. What can be achieved with this approach is a binary mask that highlights the cables, or in any case everything that is different from the static image, and obscures the background.
Since this was the approach chosen, to make the most of it and increase the quality of the result, it is important to use a high-contrast background and use good lighting, as presented in the previous chapter.
The figure 4.2 shows the sequence of operations required to obtain the mask.

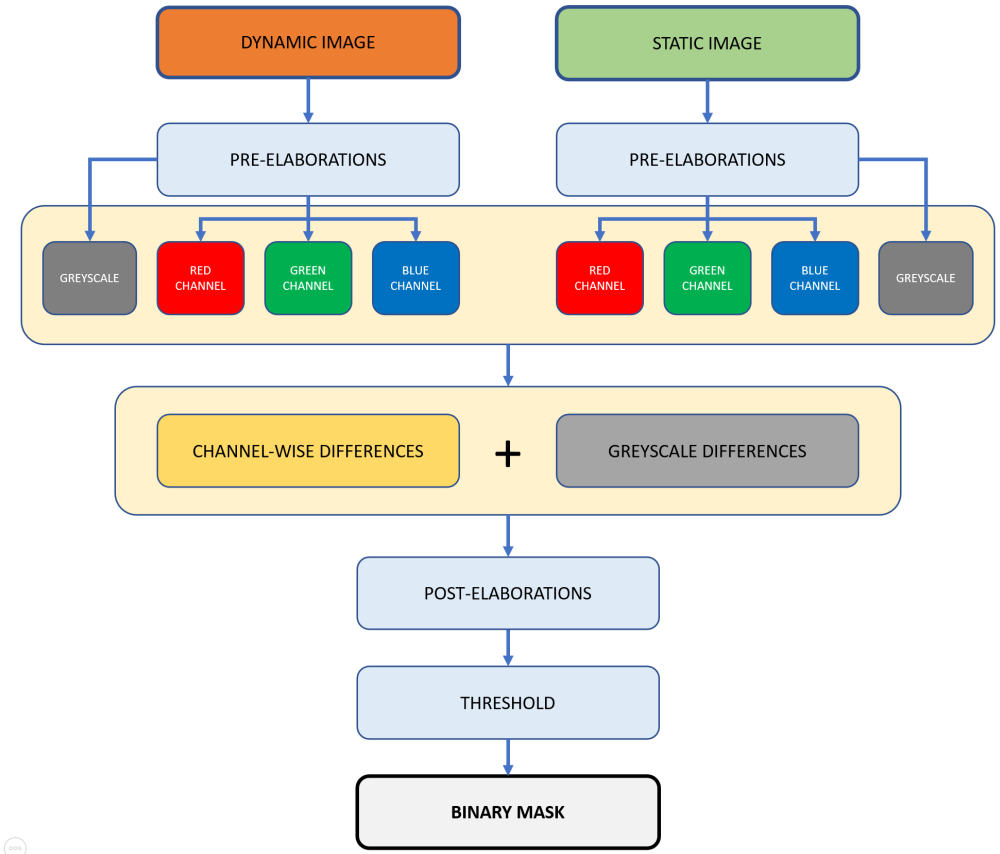


Figure 4.2: Camera node structure

It can be noted that there are pre-elaborations before making the comparison, designed to balance the illumination of the images through a normalization process on the greyscale images. This method allows to normalize and then rebalance the distribution of brightness in the images, favouring the identification of what should be considered as a background in the resulting mask. Defining $a = 0$ and $b = 255$ as the lower and upper extremes of the normalization process respectively, c as the lowest value in the image and d as the highest:

$$P_o = (P_i - c)\frac{b-a}{d-c} + a \tag{4.1}$$

P_o is the new value of pixel intensity in the normalized image and P_i is the original greyscale image value.
The comparison is the sum of the channel differences to which is added a contribution of three times the difference between the normalized greyscale images, all of which is then averaged to obtain an image. This was done to give more robustness to the solution, allowing the bulk of the work to be done

with greyscale and adjusting details using channel information.

$$R = \frac{|S_R - D_R| + |S_G - D_G| + |S_B - D_B| + 3|SG - DG|}{6} \tag{4.2}$$

Where:

R	Result Image
S_R	Red Channel Image of Static frame
S_G	Green Channel Image of Static frame
S_B	Blue Channel Image of Static frame
SG	Normalized Greyscale Image of Static frame
D_R	Red Channel Image of Dynamic frame
D_G	Green Channel Image of Dynamic frame
D_R	Blue Channel Image of Dynamic frame
DG	Normalized Greyscale Image of Dynamic frame

Table 4.1: Camera node variables

This new image undergoes post-processing before being thresholded. First, a Gaussian blur with a small kernel of dimensions (3,3) with a $\sigma = 1.5$ is used to reduce noise due to the *'salt and pepper'* without affecting the identification of the cables excessively.
Two-dimensional Gaussian function is given by:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.3}$$

Considering the kernel (3,3) and a $\sigma = 1.5$, the weights' matrix is:

$$W_{i,j} = G(i-2, j-2) = \begin{bmatrix} 0.045 & 0.057 & 0.045 \\ 0.57 & 0.07 & 0.057 \\ 0.045 & 0.057 & 0.045 \end{bmatrix} \tag{4.4}$$

$$i, j = 1, .., 3$$

The obtained Gaussian blurred image (R_F) of R is:

$$R_{Fw,h} = \sum_{i=1,j=1}^{3,3} W_{i,j} R_{w-2+i,h-2+j} \tag{4.5}$$

$$\forall (w, h) \in dim(R) : (w - 2 + i, h - 2 + j) \in dim(R)$$

As far as the threshold is concerned, a value ($T = 15$) was chosen by tuning, that is very tight on the foreground and thus favours cables identification. The binary mask (R_M) is:

$$R_{Mw,h} = \begin{cases} 255 & \text{if } R_{Fw,h} > T \\ 0 & \text{else} \end{cases} \tag{4.6}$$

$$\forall (w,h) \in dim(R_F)$$

With regard to the communication between server and client, the idea was to make it possible to obtain, from the client's point of view, the entire mask from this node. This allows those who desire to make use of the feedback message to be able to retrieve different information of their choice, such as the correct insertion of cables in specific clips.

Algorithm 1 Camera node

```
START Video Capture
Static Image = VideoCaptureFrame()
S_R, S_G, S_B = ChannelSplit(Static Image)
SG = Normalization(GreyscaleConversion(Static Image))
while (Node On) do
  if (Client Request) then
    if (New Static Image) then
      Image = VideoCaptureFrame()
      Static Image = Normalization(Image)
      S_R, S_G, S_B = ChannelSplit(Static Image)
      SG = Normalization(GreyscaleConversion(Static Image))
    end if
    Image = Video Capture Frame
    Dynamic Image = Normalization(Image)
    D_R, D_G, D_B = ChannelSplit(Dynamic Image)
    DG = Normalization(GreyscaleConversion(Dynamic Image))
    Total Differences = abs(S_R - D_R) + abs(S_G - D_G) + abs(S_B - D_B)
    + 3abs(SG - DG)
    R = int(Total Difference/6)
    R_F = GaussianBlur(R, (3,3))
    R_M = Threshold(R_F, 15)
    Client Result = R_M
  end if
end while
STOP Video Capture
```

The algorithm 1 shows the entire logical procedure performed by the node and shows the response that the server sends to the client.

Validation Results

Several tests were done to validate and improve the behaviour of the node, the following shows how the binary mask of the six-poles cable is constructed with two insertions in the clips.

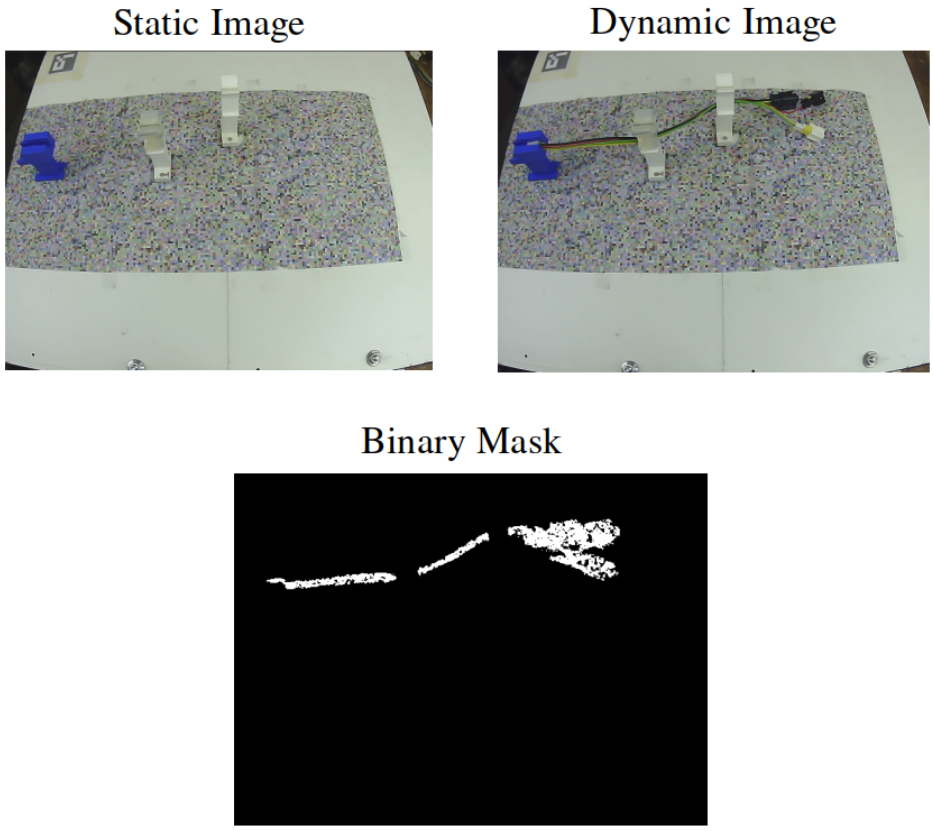


Figure 4.3: Cables binarization

It can be seen that, with the help of a high-contrast background and good shadow-free lighting, the software is able to generate a mask with good cable resolution.

2.2 Learning Server

The learning server node must perform the trajectories by directly commanding the robot. To drive the robot, we have seen how it is possible to directly exploit the ROS interface provided by Franka and MoveIt, in which we also have information that the motion has been successfully executed. During the execution of the path, we have set ourselves the task of gathering all the information regarding the forces exchanged between the gripper and the cables. To do this, we can exploit the force sensor and the related topic it publishes, where all the forces and torques to which the sensor is subjected are made explicit. Figure 4.4 shows the general structure of the node, where the client specifies the trajectory to be executed and obtains as feedback the success of the motion and the forces exchanged.

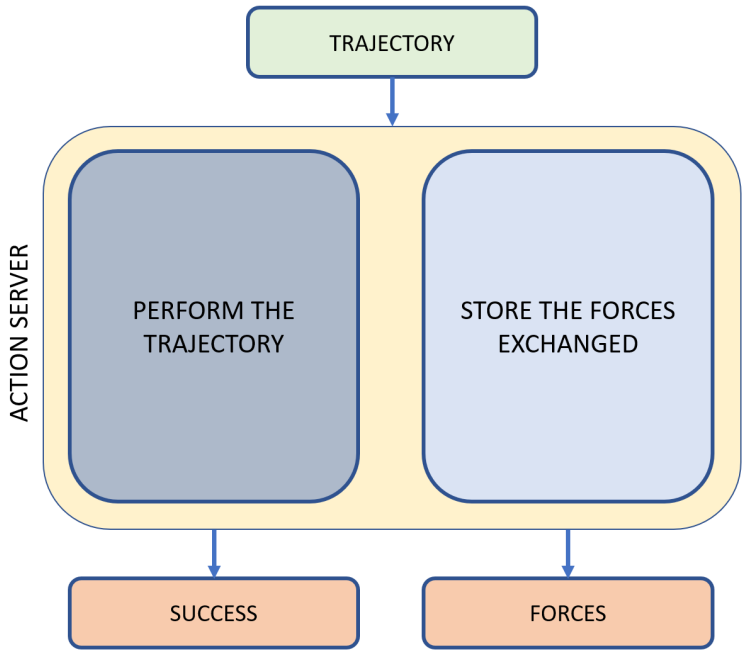


Figure 4.4: Learning server node structure

The forces and torques provided by the sensor, however, are raw information. From these values, the node must reconstruct the three-dimensional vector of forces to which the cable is subjected; in particular, we are interested in the module of this vector.
To obtain the force vector at the contact point, we need to perform a conversion of values taking into account the arrangement of reference systems.

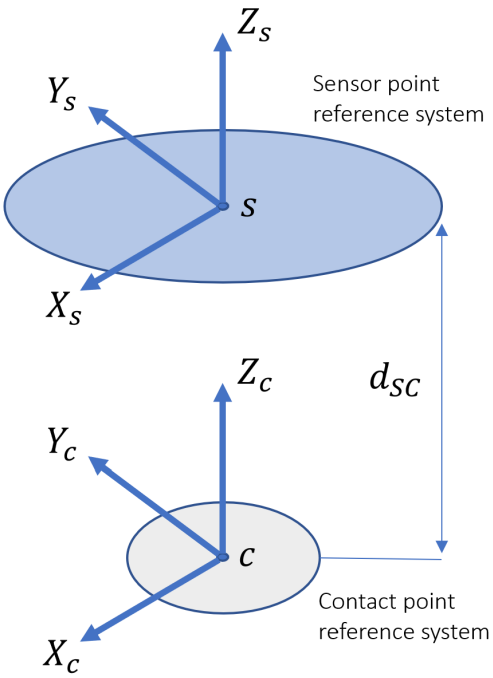


Figure 4.5: Sensor-Contact points: Forces and Torques reference systems

$$\begin{cases} F_X^C = \frac{\tau_Y^S}{d_{SC}} \\ F_Y^C = \frac{\tau_X^S}{d_{SC}} \\ F_Z^C = F_Z^S \end{cases} \tag{4.7}$$

F_X^S	Sensor x-axis force
F_Y^S	Sensor y-axis force
F_Z^S	Sensor z-axis force
τ_X^S	Sensor x-axis torque
τ_Y^S	Sensor y-axis torque
τ_Z^S	Sensor z-axis torque
F_X^C	Contact x-axis force
F_Y^C	Contact y-axis force
F_Z^C	Contact z-axis force

Table 4.2: Sensor-Contact points: Forces and Torques

Consequently, the scalar value of the contact force module is:

$$|F^C| = \sqrt{(F_X^C)^2 + (F_Y^C)^2 + (F_Z^C)^2} = \sqrt{(\frac{\tau_Y^S}{d_{SC}})^2 + (\frac{\tau_X^S}{d_{SC}})^2 + (F_Z^C)^2} \quad (4.8)$$

Where all the elements are considered as scalars. Applying this equation to each new sample, we can obtain a time vector of the force module.
The rationale of the node is to be able to provide the client with a good representation of the intensity of the contact forces exchanged, however the number of values we can collect during a path depends on the execution frequency of the node and the sampling time of the sensor. This number is far too high and always different even if we run the same path, as even the slightest slowdown would lead to more values.
What can be done is to standardize the number of samples, whose chosen window is one thousand elements, by reconstructing them from the initial sequence obtained. This number allows for a good information density and at the same time a good speed of computation, while managing to keep the number of computations to be performed low.
Considering N_s as the number of samples we have collected and N_r as the number of forces we desire to represent:

$$\begin{cases} F_{i+1} = \frac{1}{\lceil \frac{N_s}{N_r} \rceil} \sum_{k=i\lceil \frac{N_s}{N_r} \rceil + 1}^{(i+1)\lceil \frac{N_s}{N_r} \rceil} |F^C|_k & \forall i \in [0, N_r - 1) \\ F_{N_r} = \frac{1}{N_s - (N_r - 1)\lceil \frac{N_s}{N_r} \rceil} \sum_{k=(N_r-1)\lceil \frac{N_s}{N_r} \rceil + 1}^{N_s} |F^C|_k \end{cases} \quad (4.9)$$

Often the number of samples is not divisible by the number of representations, the choice was to portion homogeneously, by constructing an averaging function, all the elements of the new vector except the last one, which simply averages the remaining elements. This was done because the last part of the path is not explorable, as we shall see later, and therefore this small representation compromise will not affect the goodness of the learning.
With regard to the communication between action server and client, The server's response at the end of the path must be the entire time vector of force modules. This allows the supervisor to exploit this information to improve task execution performance.

Algorithm 2 Learning server node

```
1: while (Node On) do
2:   if (Req) then
3:     Server input = Path
4:     Starts to perform the Path
5:     while (Moving) do
6:       if (not Req) then
7:         Reset All
8:         Stop Moving
9:       else
10:        Obtain new samples from force sensor
11:        Collect a new sample of contact force module by computation
12:      end if
13:    end while
14:    if (Success) then
15:      F = Obtain the representation vector from force module samples
16:      Client Results = F , Success
17:    else
18:      Client Results = Empty vector , Success
19:    end if
20:  end if
21: end while
```

The algorithm 2 shows the entire logical procedure performed by the learning server node and displays the response that the action server sends to the client, also showing that, unlike a simple service, we have the possibility of interrupting the execution of the request at any time.
If the node is interrupted, it immediately stops performing the task and resets all memory elements it was collecting. This provides the possibility of restarting at any time. The client is provided with the two pieces of information of interest, the reconstructed time vector of forces and the actual success of the path. In the event of a negative outcome, the passed vector is simply null.

Validation Results

It is possible to show some tests that have been carried out to validate the correct functioning of the node.

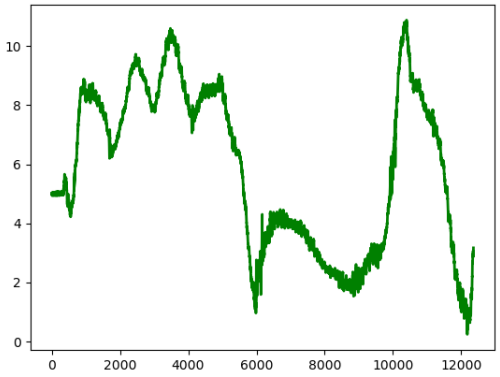


Figure 4.6: Original samples of forces [N]

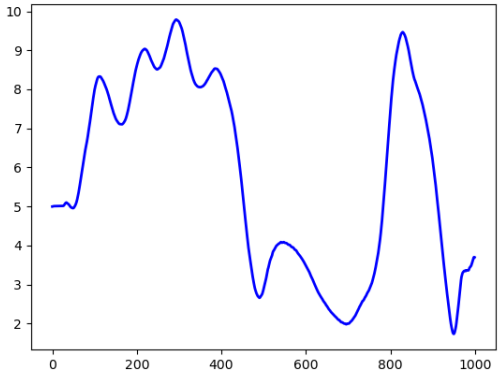


Figure 4.7: Representation samples of forces [N]

It can be seen that the representation vector (Figure 4.7) also remains consistent and informative with respect to the source sample vector.

2.3 Learning Supervisor

In this last section, the general structure of the learning supervisor node is analysed. This is an initial introduction to the subject, which will be treated exhaustively in the next chapter.
The learning supervisor node is the main part of the project and has the task of managing the entire learning stage in order to improve the execution of the tasks. The diagram presented in figure 4.8 shows the three main characteristics that a supervisor must have: *Initialization*, *Exploration*, *Learning Phase*.

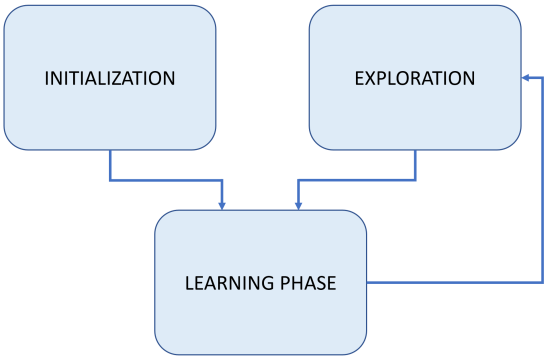


Figure 4.8: Learning supervisor node structure

The initialization in a machine learning approach is crucial, at this stage the designer has the opportunity to pass on to the software all his initial knowledge of how the task can be performed. This makes it much easier to find a solution and at the same time greatly reduces the time needed to achieve it.
The exploration is a necessary step to get closer and closer to the desired result. For this aspect, the concept of path characterization is important, as by characterizing a trajectory, we have the possibility of changing a few characteristics in order to find a new path to experiment with.
The last essential stage is the learning phase. In this portion of the node, a learning method must be chosen that is suitable for the problem to be solved. The decision process is not trivial, and often leaves compromises behind. Each learning phase, however, has some common traits; it must exploit initialization to have the first information from which to learn, and then exploit exploration to have new paths to evaluate. The way these methods learn is linked to the concept of a score function, which is a function that generates a score in order to evaluate a path, on the basis of the feedback available.
The correct creation of this function is what allows the system to obtain the expected result. In our specific case, the two nodes discussed above were created to provide the necessary information to the supervisor so that it could accommodate an effective score function.
We have at our disposal:

- The binary mask of the finished path cable arrangement provided by the camera node;
- The forces exchanged along this stretch and the correct execution of movements by the robot provided by the learning server node.

Chapter 5

The Supervisor: A Machine Learning Algorithm

The interest of researchers in recent years has focused on how to make a robot capable of learning just like humans do, trying to understand whether the human brain can be replicated through mathematical models and neural networks. A good starting point is to equip the robot with a pool of sensors that can be adapted to what humans have available to learn, for example with the use of force sensors to make contact with the surroundings and video cameras to obtain visual feedback of what is in the scene and possible changes. With this equipment, complex manipulation techniques can be implemented by collecting force data and analysing images [7]. An important note is certainly the fact that these algorithms can be used both online and offline [8]. While in the offline case the scene is reconstructed with the machine switched off once the image has been acquired, the online algorithm allows this to be done during the video acquisition and this would allow the robot to have the desired information in real time [9][10]. In this chapter, the strategy implemented in the supervisor node will be specifically analysed. The problem presents several critical issues and at the same time several initial assumptions. We have three different types of cable groupings, discriminated by a different type of connector. These cables are assumed to be already locked in the connector block and consequently our task is to insert the cables into the various clips of interest. If we subdivide this task, considering one clip at a time, then the task becomes uniform for all cables and thus standardizable in terms of approach. During this wiring harness assembly procedure, it is important that the cables are kept tensioned to reduce undesirable effects so as to avoid unexpected collisions, but we desire not to overstretch them in order not to risk damaging the final result.

1 Learning Algorithm

In this first section, we will deal with defining how the learning algorithm is realized from a software point of view. At the beginning is discussed the concept of path characterization, this aspect is important in order to be able to deal with the realization of the software components we discussed earlier: initialization, exploration and learning phase. Once the path structure has been defined, it will be possible to complete all the elements that the node needs.

1.1 Path Characterization

Since what we want to learn are paths, it is important to create a characterization in order to generate a software object that can represent them. We can see a path (Figure 5.1) as a set of ordered three-dimensional points connected by linear motion, to make the robot's behaviour totally deterministic. From a software point of view, these points can be set as mutable or non-mutable. The intermediate points between the start and the end are called way-points. The number of points characterizing a path is regarded as a free parameter available to the user.

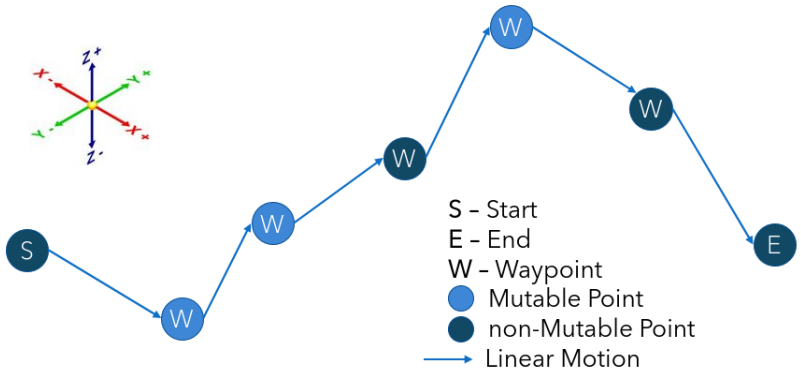


Figure 5.1: Path characterization

1.2 Path Initialization

The path initialization must be done by the user, at this stage one has the opportunity to decide on the initial path to be taken, which must be the starting point for learning. In the desired trajectories, there are known points, which it does not make sense to allocate to learning. In fact, the user is

responsible for defining for each task the starting point and end point for inserting the cables into the clips. There is also the possibility of inserting intermediate way-points, mutable or non-mutable, which can help learning and avoid collisions, based on a functional path from the start. The user also has the possibility of deciding the number of points that characterize a path according to his own experience; too many points would lengthen the learning time and may be too close together, while too few points may not be sufficient to perform the task. All non-fixed points are reconstructed by creating equidistant linear connections between those imposed. In this way we have obtained a complete first path, that if it is set up correctly is already able to perform the task but not optimally; the learning objective is not simply to solve the problem, but to solve it as good as possible with what is available.

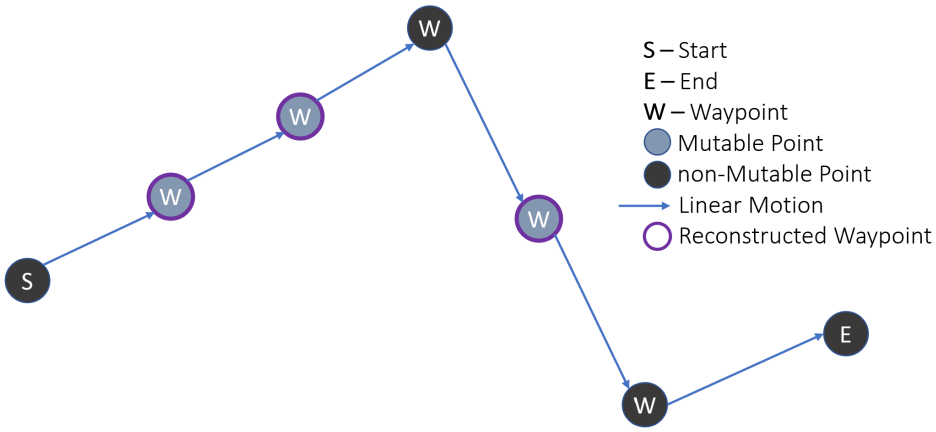


Figure 5.2: Path initialization: way-points reconstruction

1.3 Exploration and Learning Phase

Regarding strategy, the high-level software structure that has been created can be represented with the following diagram 5.3. The supervisor handles all the learning stages by choosing paths to take for the robot, the server takes care of the practical side of executing them, and in the environment we have the feedback of what is happening in reality, with the appropriate data to build a score function essential for the learning.

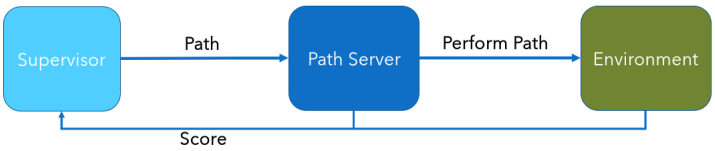


Figure 5.3: Learning structure

The idea is that the best path can be obtained from the path set by the user in the initialization. We have seen how there are mutable way-points and the purpose of learning is precisely getting the right position for all these points with respect to a specific score function.

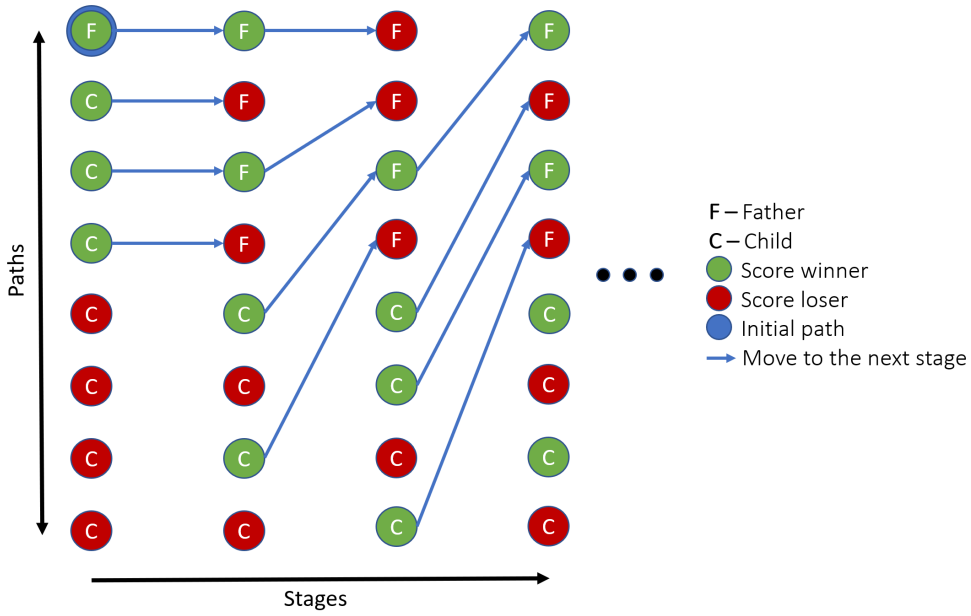


Figure 5.4: Paths learning and exploration

The learning strategy is derived from the so-called class of learning algorithm defined as genetic algorithm inspired by natural selection and designed to solve complex problem where the number of outcome are too many to be feasible to obtain an optimal solution. These algorithms are proven to solve problem such as the *travelling postman*. The algorithm is based on the concept of generating a set of possible solution named *father* (or parents) characterized by a set of information named *genes*, that in our specific case are represented by the points' trajectories. The changes of a way-point from a path leads to the creation of a *child* path. At each iteration of the algorithm, all trajectories are executed and evaluated with respected to a score-function

and the best one are selected as breeding parents for the creation of a new generation of child. In this way, only the best trajectories are allowed to be propagated, leading to the exploitation of the best paths founded. Only a better result can replace and old one. At the end of each trajectory, the cables are returned to the initial position via a user-set restore trajectory.

This approach may be redundant since the same trajectory can be executed several times, but it is necessary for the actual conditions under which learning must operate. The results on the same path may be different, as these scores depend on the friction between gripper and cables, the arrangement in which the cables are taken, small misalignments, etc.
The algorithm will terminate when the scores obtained begin to saturate, meaning that we have found a trajectory that maximizes the score and therefore represents a good solution for the task.
Since the learning procedure is time-consuming and could also be performed several times, the algorithm was equipped with memory. This allows learning to be resumed from where it was interrupted and then progress to be saved each time. The following scheme 5.5, that represent the algorithm structure, shows precisely this aspect, where we have the option of starting the learning from zero or taking advantage of a checkpoint saved in memory.

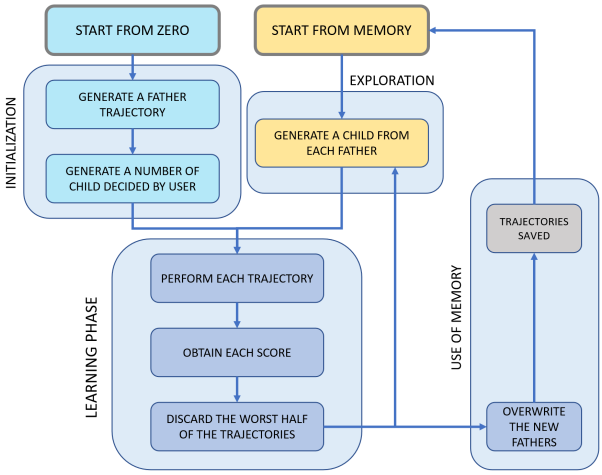


Figure 5.5: Supervisor software structure

It is also possible to show the algorithm executed by the supervisor node, highlighting its main aspects, such as the possibility of starting from the beginning or from paths saved in memory. During operation, all paths are passed to the learning server node, which performs them and returns the feedback for which it was designed. The camera node is also called when we

desire to verify the correct insertion of the cable, which returns the binary mask of the scene. The procedure is iterated, saving the best paths each time, until the improvement is considered sufficient; to do this, score saturation can be used as a stop criterion.
As far as scoring is concerned, the next section will go into detail on how to obtain these values, having to consider different information to construct them.

Algorithm 3 Supervisor node

```
1: if (Memory) then
2:    Ts[0:3] = Paths from memory
3:    Ts = GenerateChilds(Ts)
4: else
5:    Ts[0] = Initialization path
6:    Ts = GenerateChilds(Ts)
7: end if
8: while (not Improvement and Node On) do
9:    for (t in Ts) do
10:      LearningServer(t)
11:      WaitServer()
12:      Results = F, Success
13:      if (Success) then
14:         Move for picture
15:         Camera()
16:         WaitServer()
17:         Result = Picture
18:         if (CableInserted(Picture)) then
19:            Scores[t] = ForceScore(F)
20:         else
21:            Scores[t] = -1000000
22:         end if
23:         Perform the path to restore the cable conditions
24:      end if
25:   end for
26:   Ts[0:3] = Best four paths
27:   SavePaths(Ts)
28:   Ts = GenerateChilds(Ts)
29:   Improvement = ScoreSaturation(Scores)
30: end while
```

1.4 Path Score and Pseudo-Random Mutations

The score is a value that is assigned to an executed trajectory, in our case this value is conditioned by two parts:

- The correct insertion of the cable in the clips;
- A function on the forces exchanged imposed by the user.

In the first case, what we desire to obtain is a contribution that leads to the immediate discarding of those paths that are unable to fulfil the task. To do this, after the execution of the path, the supervisor requests the binary mask of the scene from the camera node. From this image, it is possible to take information on whether the insertion was successful. This can be done by obtaining a crop of the image in the area of the clip, its position is known, and assessing whether more than one object is present after appropriate processing. If the cable is correctly inserted then the clip will hide a portion of it, which is why two separate objects are created in the mask in that specific area.

Figure 5.6: Cable insertion mask

If the cable is not correctly inserted then the path will receive such a negative score that it will be discarded a priori, in the case of a positive outcome, however, the score will only be influenced by the function on the forces exchanged. This function is set by the user and it may be convenient to set it with always negative values, thus creating an upper limit value of zero, which would also be the ideal value as a score for a trajectory. An example for this function could be:

$$S(F) = -\sum_{i=1}^{N_r} f(F_i) = -\sum_{i=1}^{N_r} F_i^2 \tag{5.1}$$

That means trying to reduce the force module of the forces exchanged during the path as much as possible.
In an attempt to speed up the learning time, a ruse was used to ensure that path mutations were somewhat score-dependent. In this way, a child path tries to improve on its father's defects. When one desires to make a path mutation, one can think of randomly extracting one of its mutable way-points and changing its positional values slightly. The idea is precisely not to make this choice totally random, but to impose more probability of being extracted on those points that have had the greatest influence on making the score low. By unpacking the values that the force scoring function returns, it is possible to create weights that can be associated with the mutable way-points; by normalising these weights, it is possible to obtain probabilities of choice.
Considering N_p as the number of points in a path, the weights:

$$W_i = \sum_{k=i\lfloor \frac{N_r}{N_p} \rfloor - \lfloor \frac{N_r}{2N_p} \rfloor}^{i\lfloor \frac{N_r}{N_p} \rfloor + \lfloor \frac{N_r}{2N_p} \rfloor} f(F_k) \tag{5.2}$$

$$\forall i \in (1, N_p) : w_i \text{ is mutable}$$

Consequently, the probabilities:

$$P_i = \frac{W_i}{\sum W_k} \tag{5.3}$$

$$\forall k \in (1, N_p) : w_k \text{ is mutable}$$

The expressions for the weights and for the probabilities contain an approximation due to discretization, this approximation however, considering a high N_r and low N_p number, remains very valid for expressing the influence of way-points on the score.

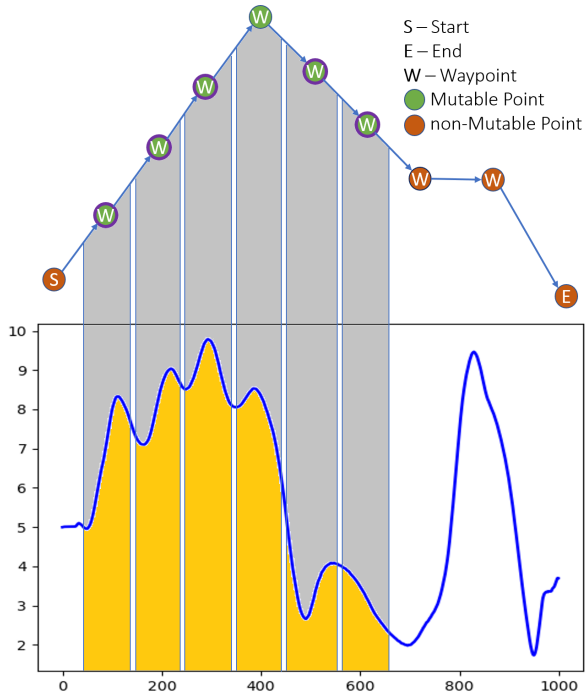


Figure 5.7: Separation of force contributions

The figure 5.7 shows how the force contributions are separated to obtain the probabilities, each mutable way-points generates a neighbourhood of force contributions, which can be used for the evaluations just explained.

1.5 Path Alignment

As the work table is not attached to the robot base, it is possible that during the execution or learning resumption of a given task, even days later, there may be unexpected path misalignments. For this reason, the robot is equipped with a RealSense camera, on the end-effector, and an Aruco marker is placed on the work surface. In this way it is possible, when starting the task for the first time, to obtain and save the position of the Aruco marker, and from time to time, using this information and the new position of the marker, to obtain the homogeneous transformations that allow the paths to be realigned with respect to the working base.

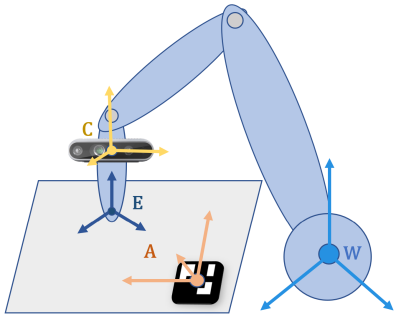


Figure 5.8: System references

Each path is defined as a succession of spatial points of the end-effector with respect to the world frame (P_W^E). So if we desire to recompute the path by realigning it to the working plane ($P_W^{E'}$) we must:

- Obtain the homogeneous transformations between Aruco and Camera (H_A^C) and between Camera and World (H_C^W);
- Find the pose of the End-effector with respect to the Aruco frame (P_A^E) and save it;
- Obtain the new homogeneous transformations between Aruco and Camera ($H_{C'}^{A'}$) and between Camera and World ($H_W^{C'}$);
- Calculate the pose of the End-effector with respect to the World frame ($P_W^{E'}$), exploiting P_A^E saved in memory.

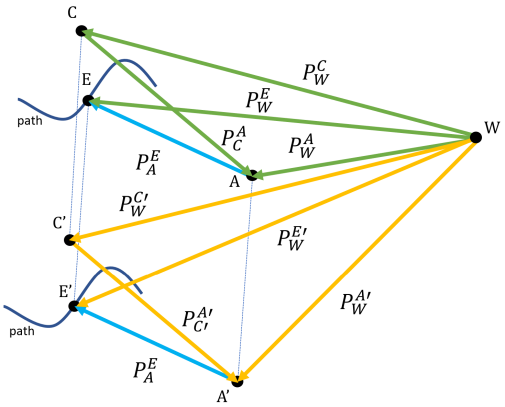


Figure 5.9: Pose vectors

In order to find P_W^A, it is necessary to calibrate the camera so as to obtain the homogeneous transformation from A to W (H_W^C). By means of a small ROS node, which uses the camera to find the homogeneous transformation from A to C H_C^A, it is possible to obtain H_A^W.

$$H_A^W = H_C^A H_C^W \tag{5.4}$$

It is now possible to find the static vector P_A^E by exploiting P_W^E, having H_A^W available.

$$P_A^E = H_A^W P_W^E \tag{5.5}$$

This procedure is done during initialisation and is necessary to have in memory the static references between Aruco and the zones of the working plane in which the End-effector is to operate. When we desire to perform the realignment, we need to compute $H_W^{A'}$ finding $H_{C'}^{A'}$ and $H_W^{C'}$, considering that the Aruco has moved. To obtain the coordinates of $P_W^{E'}$, which are the poses of the realigned path, we exploit the static vector we saved in memory P_A^E transformed with respect to the World frame.

$$P_W^{E'} = H_W^{A'} P_A^E = H_W^{C'} H_{C'}^{A'} P_A^E \tag{5.6}$$

2 Algorithm Validation

We can now move on to see the first experimental results of this approach. The cable used is the SPC cable and the parameters chosen are that the number of path points is 10 and the number of paths competing with each other is 8. In order to validate the learning algorithm and, consequently, test its properties, we set ourselves the first objective of being able to position the SPC cable in the clip with the connector already locked in the connector block.

2.1 Simple Score Function and Implementation

The wish is for the robot to learn to perform this task by trying to minimise the variance of the three-dimensional force vector, which is generated in the contact between the gripper and the cable. The idea is that in this way the robot is able to act uniformly on the force during the cable handling path.

$$S(F) = -\sum_{i=1}^{N_r} f(F_i) = -\sum_{i=1}^{N_r} (F_i - \mu_F)^2 \tag{5.7}$$

As far as implementation is concerned, the user must provide the specifications of the initial path, in particular he must define the starting point, the final point and, if desired, intermediate way-points which may be mutable or non-mutable. In this specific case it was decided to impose two non-mutable way-points near the end point and one mutable way-point in the middle of the path. The starting point was chosen to ensure correct cable grip in the proximity of the connector block. The intermediate way-point chosen is used to strategically indicate which direction of motion is favourable for performing the task, based on the user's experience.

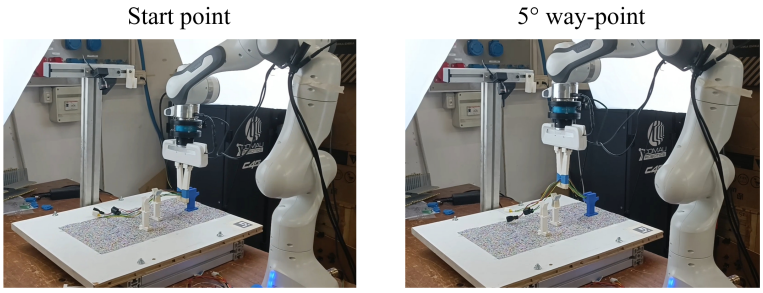


Figure 5.10: Start point and middle way-point

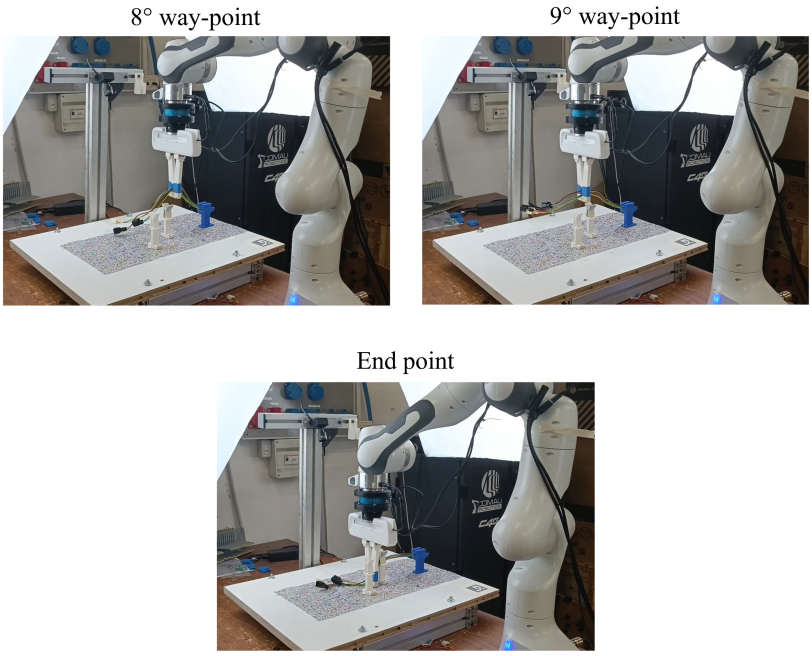


Figure 5.11: Final sequence of path points

The choice on the final sequence was made in order to have good results on cable insertion, since the last points of the path are very precisely imposed by the user.

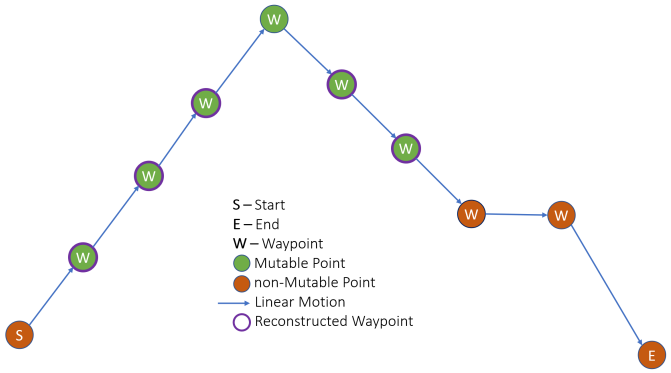


Figure 5.12: First path characterization

2.2 Results

Once the user has implemented the initial path, it is possible to proceed with the execution of learning. the three main nodes including the supervisor are turned on, which begins the exploration and learning phase. For this first experiment we wanted to verify that the scores saturated, with improvement, as the exploration progressed. To achieve the expected result, 240 winning paths (N_p) were explored.

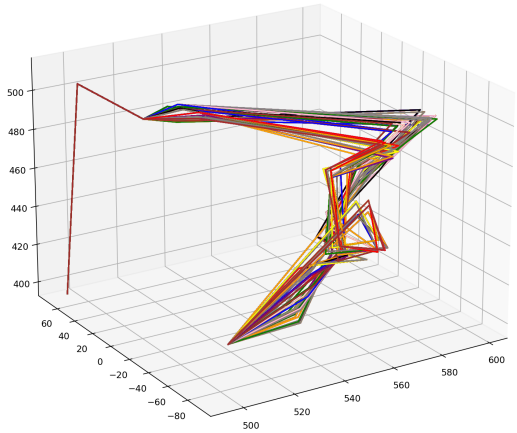


Figure 5.13: Paths performed [mm]

In order to be able to graph the scores by having a view of the improvement, a simple function was implemented that could express the improvement in absolute terms compared to the initial path. Considering the vector of scores S, it is possible to construct a vector of improvements I with the following function:

$$I_i = \frac{|\frac{1}{S_i}| - |\frac{1}{S_1}|}{|\frac{1}{S_1}|} \tag{5.8}$$

$$\forall i \in [1, N_p]$$

As can be seen from figure 5.14 the scores tend to saturate, the blue feature is the filtered scores to limit the visual effects due to data inconsistency. Green strokes show improvements, while red strokes show deteriorations.

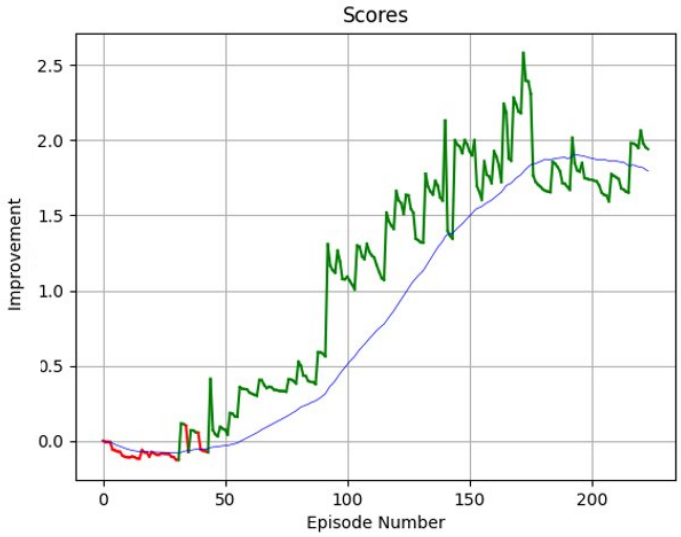


Figure 5.14: Scores Obtained

As explained in the previous paragraphs the same trajectory is executed several times if it is a winner, these trajectories, however, get very different scores from each other and this makes the graph look very fluctuating. For this reason, an improvement function referring to the best path of the previous learning phase will be implemented in the next chapter. This will allow the improvement of the paths to be better seen.

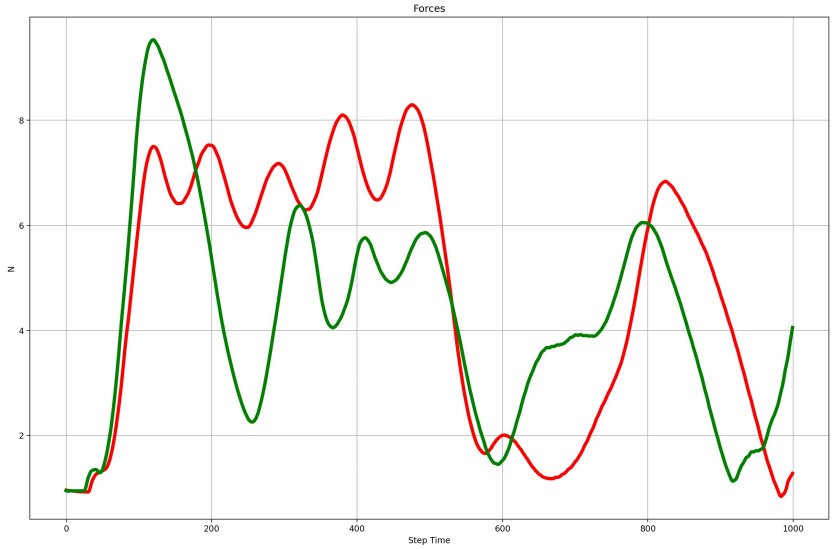


Figure 5.15: Forces exchanged

The figure 5.15 shows the results on the exchanged forces obtained from the first path (in red) and the learning solution (in green). The result is not what we had hoped to achieve; the robot did not make the treatment homogeneous in terms of forces on the cable, rather it generated a spike in the early part of the path. However, the result obtained minimizes the score function we imposed. The generated peak helps to raise the mean of the forces and thus decrease the variance in the final section of the path. The simple function on the score we used is not sufficient to obtain the desired behavior, however, it is sufficient to validate that the algorithm works properly.

Chapter 6

The Algorithm: Implementation and Results

In this last chapter, the practical implementation of the algorithm with its results is discussed. The previous chapters have shown all the software and hardware components required for the development of the application that will be used in the implementation phase. With regard to the results, the learning graphs will be shown, with particular detail on the improvement curve and the intensities of the vector of forces exchanged in the contact between gripper and cables. In the second section, we will use the first path to learn to introduce the results in detail, which considerations will then be replicated for subsequent paths.
The application aims to find a solution in terms of pseudo-optimal paths with short learning times. Applying learning directly in reality, without passing through a simulation environment that can speed up the learning process, it is unthinkable to find an optimal solution.

1 Tasks Subdivision and Score Function

The application requires the handling of three different groupings of cables, which must be organized in a specific harness. The environment was constructed in such a way that the application could be fulfilled, two clips and a suitable connector blocks arrangement were required. All three cable groupings must be inserted into the clips, but in a different order. Figure 6.1 shows the final arrangement of the cables and the worktop. The learning phase was organized into three different tasks, where each task represents the path a grouping of cables must take in order to be correctly arranged. In turn, these tasks are unpacked into sub-tasks, where the path is discovered in order to

perform the insertion in a single clip with the other part of the grouping fixed. Therefore, there will be six paths to learn, two for each cable type.

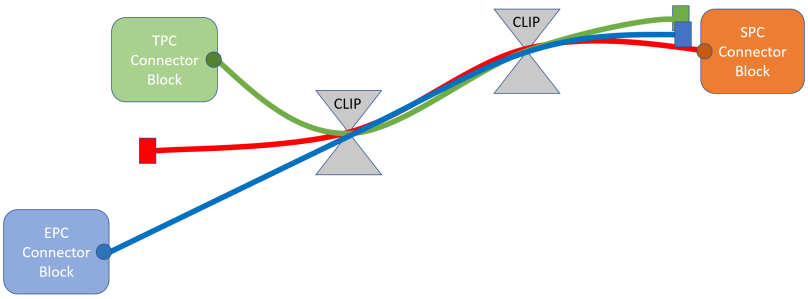


Figure 6.1: Representation of the tasks

Regarding the score, we saw how in the previous chapter the solution obtained with the cost function initially proposed was not suitable for the purpose, in fact the objective was to obtain a path that is able to keep the contact forces as homogeneous and low as reasonably practicable, reducing any force peaks where possible. For this reason, a score function which included these considerations was implemented, using the variance, mean and peak value of the contact forces.

$$S(F) = -\sum_{i=1}^{N_r} f(F_i) = -\sum_{i=1}^{N_r} F_{max}\mu_F(F_i - \mu_F)^2$$

2 Six-Poles Cable

The first path we desire to learn is for the six-pole connector cables. This path will be subdivided into two sub-paths to be learnt, when linked, will return the overall path. The connector block is located to the right of the two clips, in the proposed top view, and the goal will be to learn how to insert the group of cables into the first clip and then the insertion of the cables into the second clip.

2.1 SPC First Path

This first trait to learn is the same one we analysed in the previous chapter, when we set out to validate the supervisor algorithm.

Implementation

As far as implementation is concerned, the user will have to provide an initial path that can be adequately informative, so that the robot already has a good initial basis for learning.

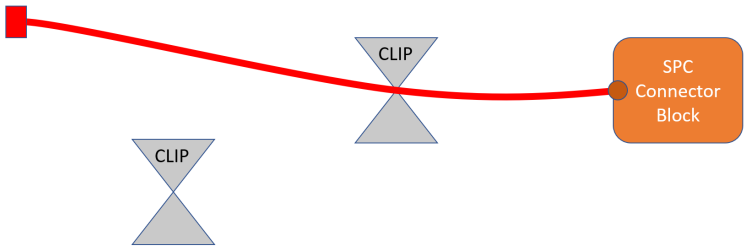


Figure 6.2: Representation of the SPC first path

It was decided, in the initialization, to impose two non-mutable way-points near the end point and one mutable way-point in the middle of the route. The starting way-point was chosen to ensure proper cable grip near the connector block. The middle way-point chosen is used to strategically indicate which direction of movement is conducive to task execution, based on user experience. This favours finding a path away from the clips initially to try to avoid collisions.

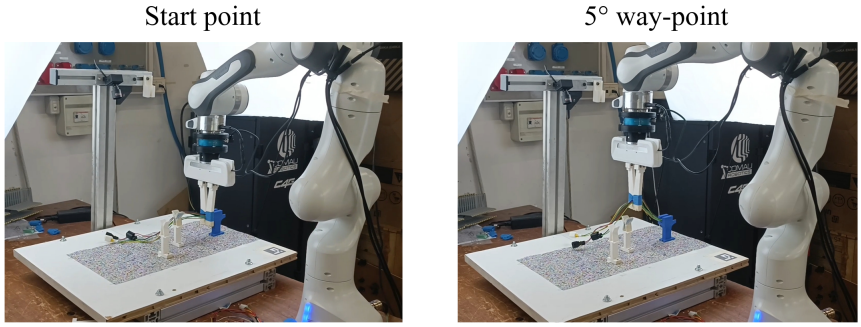


Figure 6.3: Start point and middle way-point of SPC first path

The choice on the final sequence was made in order to have good results on cable insertion, since the last points of the path are very precisely imposed by the user.

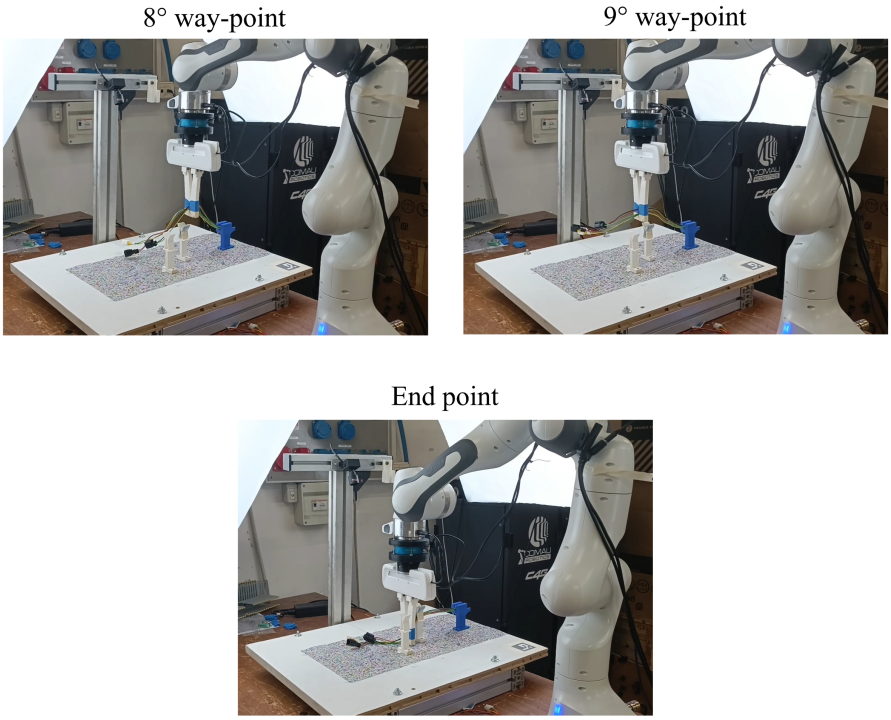


Figure 6.4: Final sequence of SPC first path points

Once the initial path has been set, the algorithm can proceed with the exploration and learning phase, following the logical structure that was shown in the previous chapters.

Results

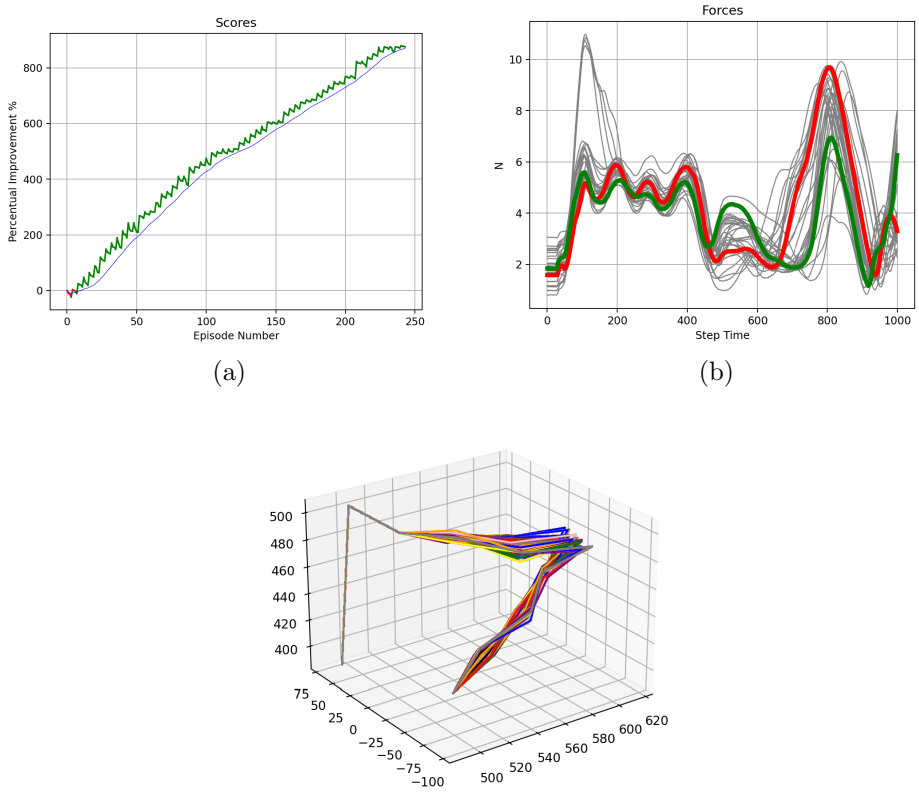
The robot took about 7 hours of continuous work to learn this first path by performing 244 winning paths, twice as many considering the losers. We have discussed how the algorithm forces improvement by trying to limit the effects of data inconsistency and for this reason it is appropriate to define a new improvement function, that allows the progress of the learning phase to be graphed in a more visible way.

$$I_i = 100\frac{|\frac{1}{S_i}| - |\frac{1}{S_{PS}}|}{|\frac{1}{S_{PS}}|} + I_{PS} \tag{6.1}$$

$$\forall i \in [1, N_p]$$

S_{PS} is the score of the winning path of the previous learning stage in the current stage, I_{PS} is the improvement value of the winning path of the

previous learning stage. With this new expression, this function is able to describe the improvement in relative terms and thus show how the solutions are better than those of the predecessors.



(a) (b)

(c)

Figure 6.5: Fig.6.5a Scores of the SPC first path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.5b Forces exchanged of the SPC first path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.5c Paths performed of the SPC first path in mm

Figure 6.5a shows that the slope of the improvement is clear from the beginning. These results are to be taken lightly, the application suffers greatly from data inconsistency, but the improvement has occurred. This time, compared to the previous chapter, the forces exchanged behaved as expected, reducing the peaks and the mean value as well as minimizing the variance.

2.2 SPC Second Path

In this second section, we desire to learn how to position the cable in the second clip considering that it has already been fixed in the first. This way, once this path has also been learnt, it will be possible to concatenate the two paths to obtain a single path for the cable.

Implementation

The first thing to do is to decide how to impose the initial path, based on human experience.

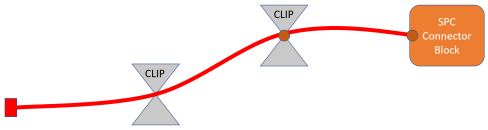


Figure 6.6: Representation of the SPC second path

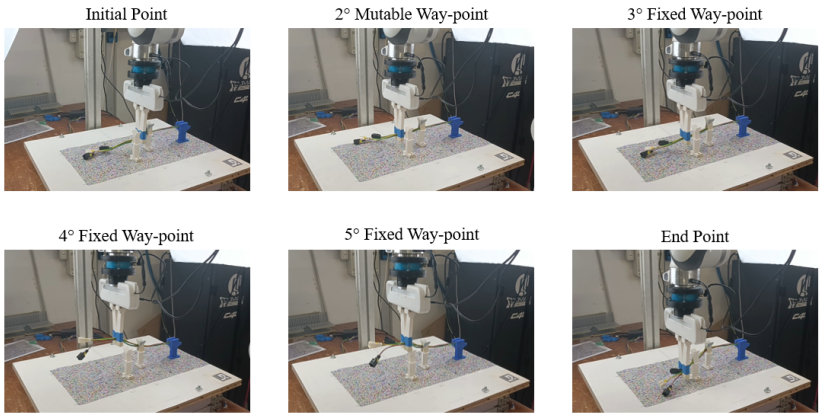


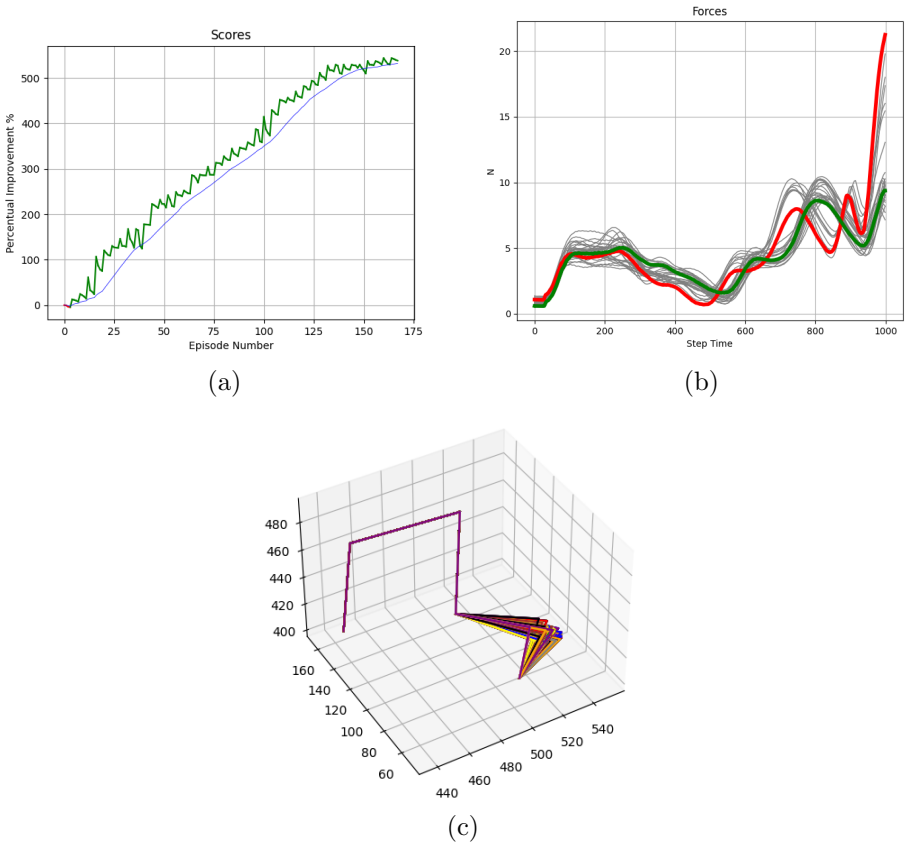
Figure 6.7: Points sequence of the SPC second path

This path is very short and must be carried out precisely so as not to pull the cable out of the previous clip and to make a correct insertion into the next clip. it was decided to characterise the path with six points: a start and an end point, a mutable way-point immediately after the start and four non-mutable way-points to follow. This choice was made in order to try to avoid collisions and free up the cable as much as possible in the initial section,

giving the possibility of learning to improve the treatment on the cable. The final leg, on the other hand, was precisely set to ensure proper insertion.

Results

mi io The robot took about 3 hours of continuous work to learn this second path by performing 168 winning paths, twice as many considering the losers.



(a) (b)

(c)

Figure 6.8: Fig.6.8a Scores of the SPC second path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.8b Forces exchanged of the SPC second path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.8c Paths performed of the SPC second path in mm

The path learnt shows excellent results in terms of the forces exchanged, significantly reducing the final peak and making the treatment more homogenous.

3 Ten-Poles Cable

The second path we wish to learn is that of the ten-poles connector cable. This path will be subdivided into two sub-paths to be learnt which, once linked, will return the overall path.

3.1 TPC First Path

The first path to learn is the one that leads the TPC cable from the connector block to the first clip.

Implementation

The first thing to do is to decide how to impose the initial path, based on human experience.

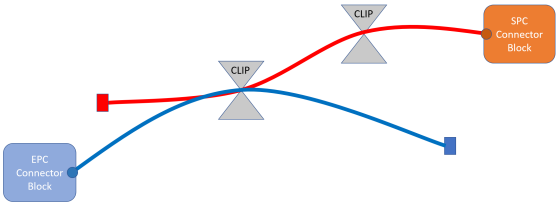


Figure 6.9: Representation of the TPC first path

This path is very similar to the one we learned in the previous section 2.1 when we analysed the first path of the SPC cable.

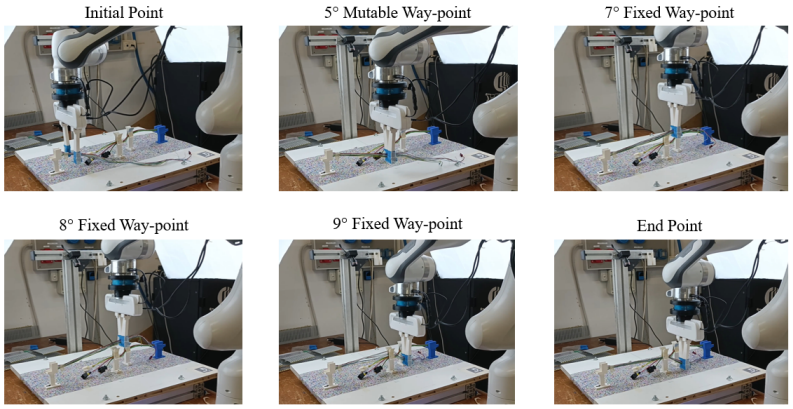
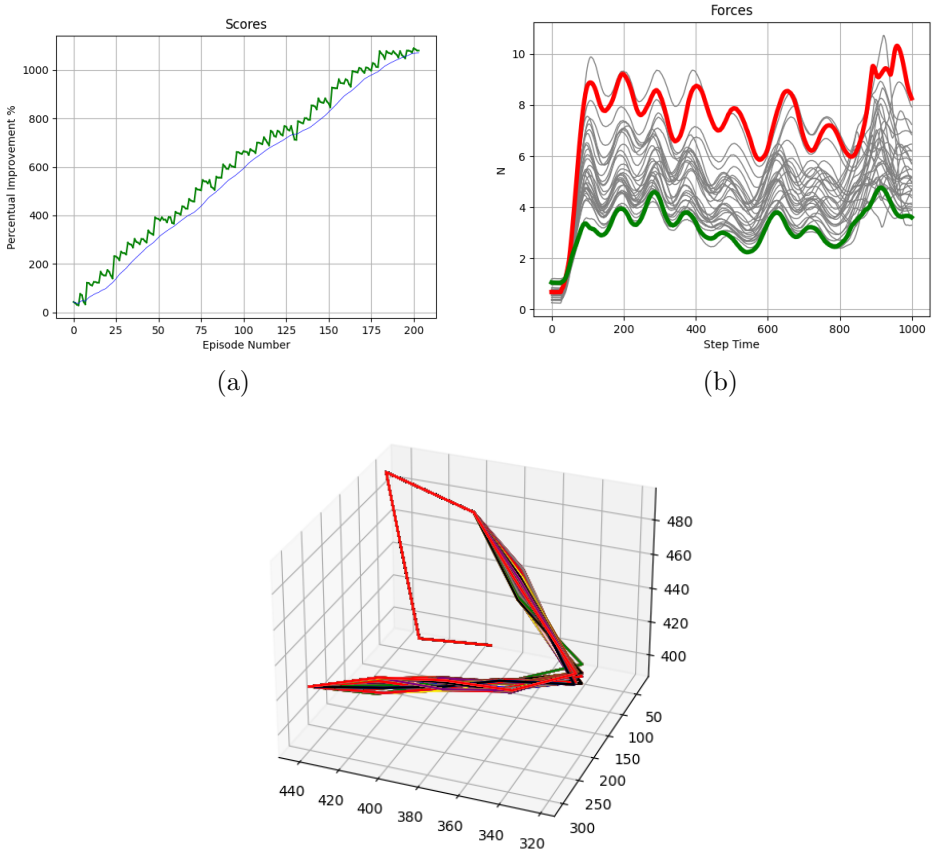


Figure 6.10: Points sequence of the TPC first path

The first part of the initial path will be set to ensure that the cable is free of collisions, with the possibility of changing the points in the learning phase. The final part, on the other hand, will be precisely set to ensure that the cable is correctly inserted into the clip.

Results

The robot took about 5 hours of continuous work to learn this first path by performing 208 winning paths, twice as many considering the losers.



(a) (b)

(c)

Figure 6.11: Fig.6.11a Scores of the TPC first path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.11b Forces exchanged of the TPC first path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.11c Paths performed of the TPC first path in mm

The learned path shows excellent results in terms of the forces exchanged, significantly reducing the variance and mean value, and avoiding spikes.

3.2 TPC Second Path

The second path to learn is the one that leads the TPC cable from the first clip to the second one.

Implementation

The first thing to do is to decide how to impose the initial path, based on human experience.

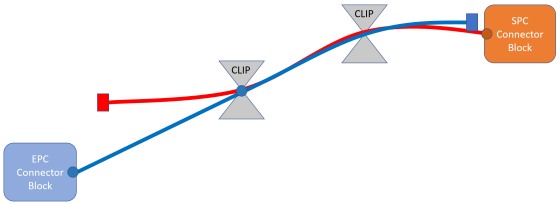


Figure 6.12: Representation of the TPC second path

This path is very similar to the one we learned in the previous section 2.2 when we analysed the second path of the SPC cable.

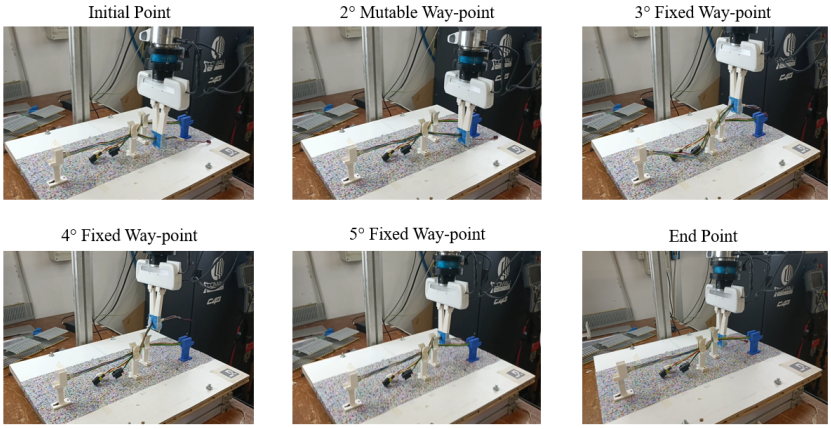


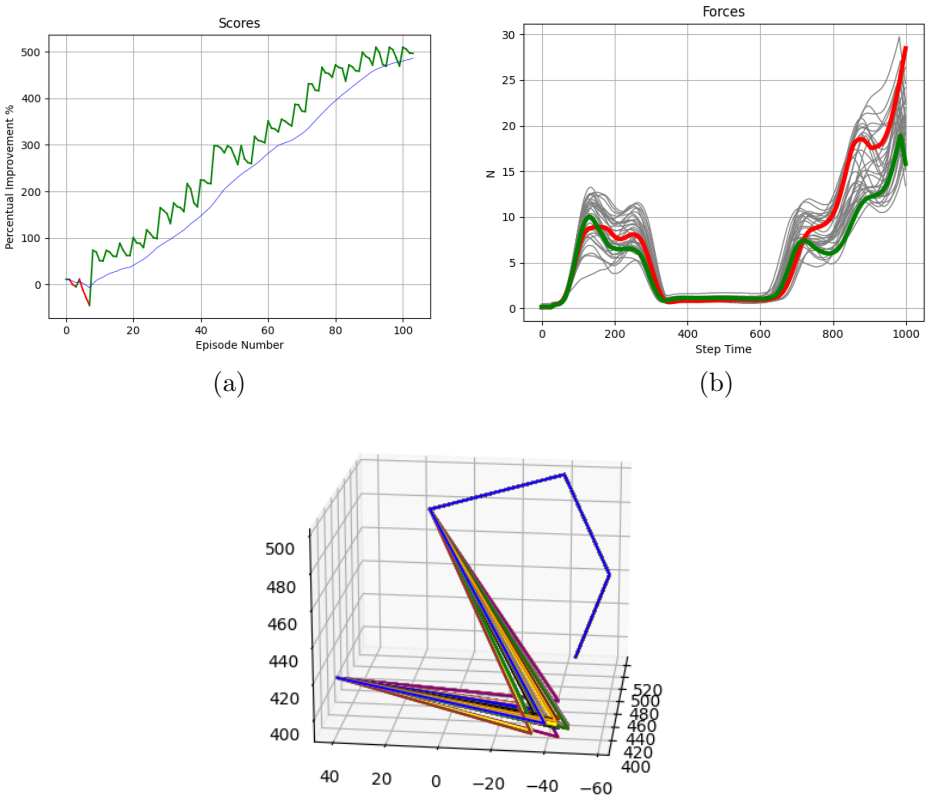
Figure 6.13: Points sequence of the TPC second path

It is very short path and must be carried out precisely so as not to pull the cable out of the previous clip and to make a correct insertion into the

next clip. it was decided to characterize the path with six points: a start and an end point, a mutable way-point immediately after the start and four non-mutable way-points to follow.

Results

The robot took about 3 hours of continuous work to learn this second path by performing 104 winning paths, twice as many considering the losers.



(a) (b)

(c)

Figure 6.14: Fig.6.14a Scores of the TPC second path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.14b Forces exchanged of the TPC second path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.14c Paths performed of the TPC second path in mm

The learned path shows good results in terms of the forces exchanged, reducing the variance and mean value, and avoiding spikes where possible.

4 Eleven-Poles Cable

The third path we wish to learn is that of the eleven-poles connector cable. This path will be subdivided into two sub-paths to be learnt which, once linked, will return the overall path.

4.1 EPC First Path

The first path to learn is the one that leads the EPC cable from the connector block to the first clip.

Implementation

The first thing to do is to decide how to impose the initial path, based on human experience.

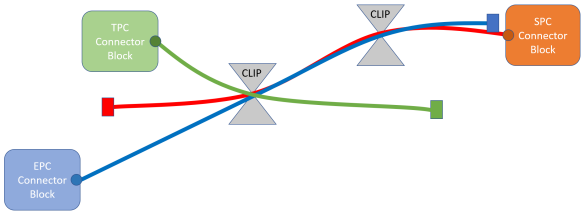


Figure 6.15: Representation of the EPC first path

This path is very similar to the one we learned in the previous section 3.1 when we analysed the first path of the TPC cable.

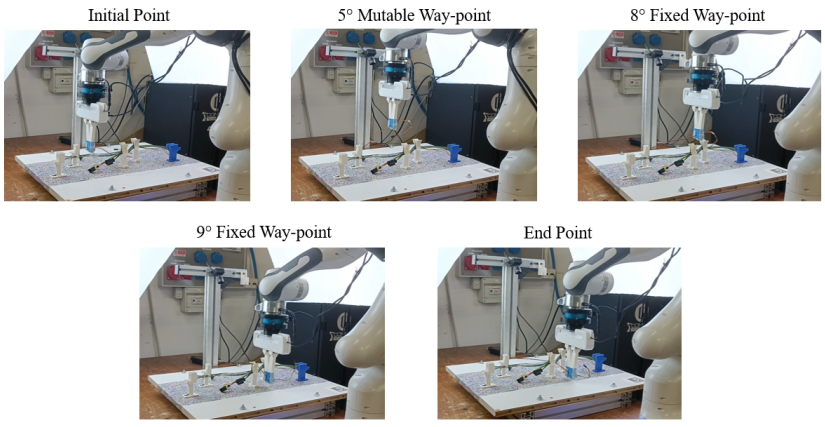
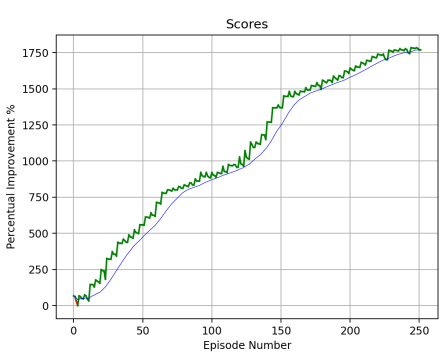


Figure 6.16: Points sequence of the EPC first path

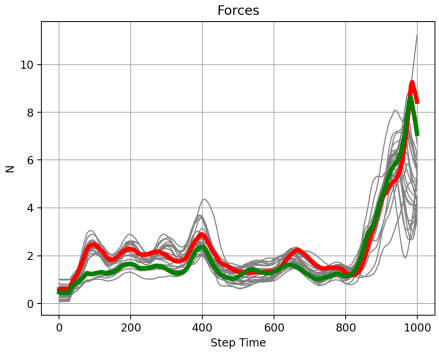
The first part of the initial path will be set to ensure that the cable is free of collisions, with the possibility of changing the points in the learning phase. The final part, on the other hand, will be precisely set to ensure that the cable is correctly inserted into the clip.

Results

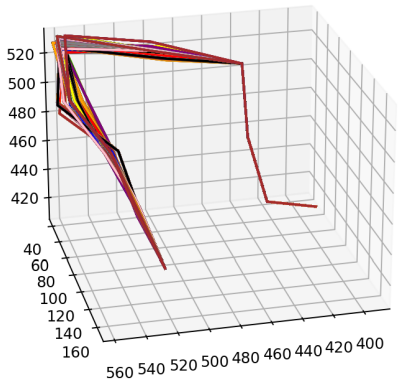
The robot took about 6 hours of continuous work to learn this first path by performing 252 winning paths, twice as many considering the losers.



(a)



(b)



(c)

Figure 6.17: Fig.6.17a Scores of the EPC first path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.17b Forces exchanged of the EPC first path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.17c Paths performed of the EPC first path in mm

The learned path shows good results in terms of the forces exchanged, reducing the variance and mean value, and avoiding spikes where possible.

4.2 EPC Second Path

The second path to learn is the one that leads the EPC cable from the first clip to the second one.

Implementation

The first thing to do is to decide how to impose the initial path, based on human experience.

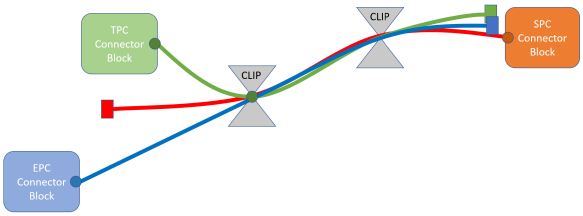


Figure 6.18: Representation of the EPC second path

This path is very similar to the one we learned in the previous section 3.2 when we analysed the second path of the TPC cable.

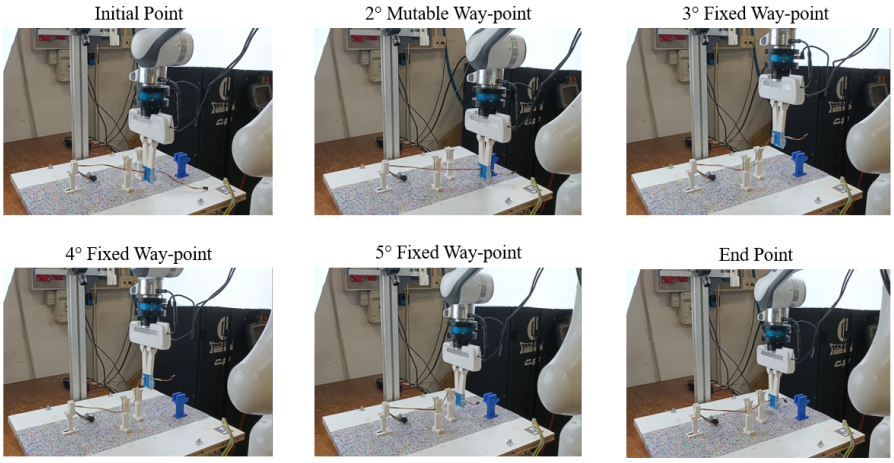
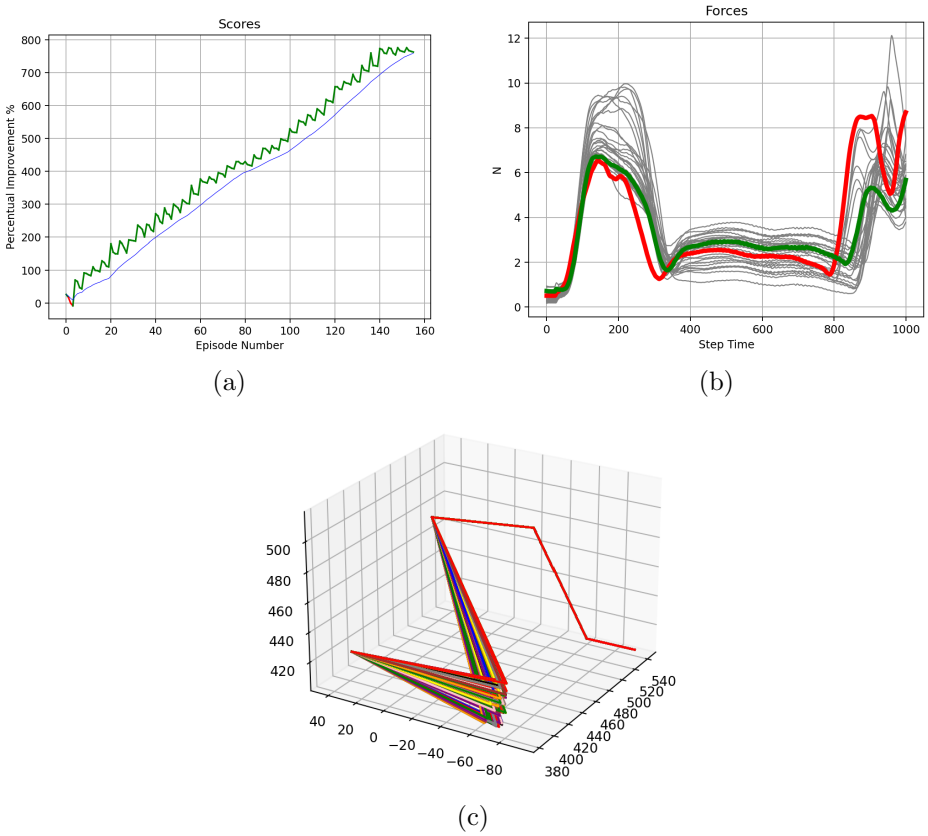


Figure 6.19: Points sequence of the EPC second path

It is very short path and must be carried out precisely so as not to pull the cable out of the previous clip and to make a correct insertion into the next clip. it was decided to characterise the path with the usual six points.

Results

The robot took about 4 hours of continuous work to learn this second path by performing 156 winning paths, twice as many considering the losers.



(a) (b)

(c)

Figure 6.20: Fig.6.20a Scores of the EPC second path, green strokes show improvements, red strokes show deteriorations and the blue curve is the filtered improvement. Fig.6.20b Forces exchanged of the EPC second path. (Red line is the initial path, Gray lines are the paths explored, Green line is the final path choose). Fig.6.20c Paths performed of the EPC second path in mm

The learned path shows good results in terms of the forces exchanged, reducing the variance and mean value, and avoiding spikes where possible.

5 Overall Results

This last section shows all the results regarding learning, in which the training times and values achieved on the forces, obtained from the chosen path compared to the initial one, are highlighted.

Path	Learning Time [h]	Force Peak [N]	Force Mean [N]	Force Variance	Score
SPC1 I	7	9.70	4.33	4.06	-171.02
SPC1 R		6.95	3.83	1.81	-48.24
SPC2 I	3	21.25	4.65	11.75	-1161.40
SPC2 R		9.38	4.42	4.3	-177.21
TPC1 I	5	10.32	7.31	3.57	-269.72
TPC1 R		4.77	3.20	0.58	-8.88
TPC2 I	3	28.47	6.76	45.94	-8853.21
TPC2 R		18.93	5.24	20.43	-2030.91
EPC1 I	6	9.27	2.29	2.35	-49.67
EPC1 R		8.61	1.96	2.68	-45.50
EPC2 I	4	8.69	3.57	4.98	-155.28
EPC2 R		6.72	3.40	2.49	-56.93

Table 6.1: Learning Results (I: Initial Path, R: Result path)

As can be seen from Table 6.1, the results are very satisfactory and testify to the effective operation of the algorithm, managing to improve paths with multiple mutable way-points but also those paths that have a single mutable point.

Chapter 7

Conclusions and Future Developments

During the study of this problem, an application was successfully realized to find an excellent path to solve the wiring harness assembly case study of Elvez. The project began by trying to realize the best possible environment to host the subsequent software implementation. 3D pieces were built to house the cables, cameras and force sensors were exploited to obtain the necessary information on what is happening in the environment during the execution of the task. Once all the hardware components were correctly arranged, it was possible to move on to implementing the software side. The software communication scheme and the implementation of a supervisor, capable of handling path learning techniques, made it possible to explore different possibilities and gradually improve the solutions. This project had several critical issues, such as the total lack of a model of the DLOs or the real possibility of inconsistency in the force sensor data. For these reasons, the implementation idea was to try to create an approach totally focused on avoiding these problems, which was successfully fulfilled. Regarding future developments, this thesis project was also a preliminary step for the research topic of the IntelliMan project in which the University of Bologna is coordinator. The aim of the IntelliMan project is to develop a robotic system for the manipulation of different elements with high performance and constant learning capability through a heterogeneous set of sensors, for advanced robotics, manufacturing and prosthetic services. This system will be able to adapt to the characteristics of its environment and decide how to perform a task autonomously, detecting flaws in its execution and requesting new knowledge through interaction. Ensuring safety and performance, IntelliMan's advances range from learning manipulation skills to abstract descriptions of a manipulation task, or discovering the functionality of an object.

Bibliography

[1] M.P. Groover. *Automation, Production Systems, and Computer-integrated Manufacturing*. Prentice Hall, 2008. ISBN: 9780132393218. URL: https://books.google.it/books?id=an-PTIoh%5C_NoC.

[2] REMODEL Project eu. *https://remodel-project.eu/*. 2020. URL: https://remodel-project.eu/.

[3] Stanford University. *https://www.ros.org/*. URL: https://www.ros.org/.

[4] Stanford University. *Moveit!,ROS base plug-in*. URL: https://moveit.ros.org. (accessed: 01.09.2020).

[5] I. A. Sucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library". In: *IEEE Robotics Automation Magazine* 19.4 (2012), pp. 72–82.

[6] Franka Emika GmBh. *Panda Robotic Arm*. URL: https://www.franka.de/technology. (accessed: 01.09.2020).

[7] Serkan Cabi et al. "Scaling data-driven robotics with reward sketching and batch reinforcement learning". In: (2019). DOI: 10.48550/ARXIV.1909.12200.

[8] L.Yen-Chen et al. "Learning to See before Learning to Act: Visual Pre-training for Manipulation". In: *IEEE* 10.1109/ICRA40945.2020.9197331 (2020).

[9] Vijay Kakani et al. "A critical review on computer vision and artificial intelligence in food industry". In: *Journal of Agriculture and Food Research* (2020), pp. 1–9.

[10] Di Stefano L. De Gregorio D. Palli G. "Let's take a Walk on Superpixels Graphs: Deformable Linear Objects Segmentation and Model Estimation". In: *Asian Conference on Computer Vision* arXiv:1810.04461v1 (10 Oct 2018), pp. 1–7.