An approach for the bimanual manipulation of a deformable linear object using a dual-arm industrial robot: cable routing use case

Pablo Malvido Fresnillo
FAST-Lab, Faculty of Engineering and Natural Sciences
Tampere University
Tampere, Finland
pablo.malvidofresnillo@tuni.fi

Saigopal Vasudevan
FAST-Lab, Faculty of Engineering and Natural Sciences
Tampere University
Tampere, Finland
saigopal.vasudevan@tuni.fi

Wael M. Mohammed
FAST-Lab, Faculty of Engineering and Natural Sciences
Tampere University
Tampere, Finland
wael.mohammed@tuni.fi

***Abstract*—The automation of processes that handle deformable materials is considered to be a complicated task. Due to their properties, these materials require specialised solutions for their manipulation, using robotic systems and mostly, using specifically developed hardware which limits its use for different deformable objects. To solve this issue, this paper presents an approach for bimanually manipulating Deformable Linear Objects (DLOs) using a dual-arm industrial robot. This approach aims at providing an automatic, generic, and easily reconfigurable solution and is implemented for routing cables in a human-centric platform. The approach consists of a cyber-physical system (CPS) composed by commercial hardware: a robot equipped with two parallel grippers, and a reconfigurable Robot Operating System (ROS) software. In more details, the developed software extracts information about the process, such as the routing path, keypoints of the workstation setup and objects dimensions. Then, it uses the extracted information to generate suitable bimanual trajectories for the robot. Finally, the approach has been tested for three different routing paths.**

***Index Terms*—Bimanual Manipulation, Deformable Object Manipulation, ROS Applications, Knowledge-Based Systems, Cyber-Physical Systems**

I. Introduction

Deformable Linear Objects (DLOs) i.e., cables, ropes, hoses, cloth, etc. are highly relevant products of everyday life and are handled extensively in industrial, household, and medical (to highlight a few) scenarios daily. Much of these manipulations are currently performed manually by a human actor, as there is a wide gap between existing research and the commercial implementation of robotic systems, which could potentially replace humans to perform DLO manipulation and achieve satisfactory results.

Handling of DLOs is still challenging for robots, primarily because of the lack of established cyber-physical systems (CPS) to properly perform and combine the outcomes of sensing, modelling, planning and control, for DLO manipulation [1] [2]. Furthermore, existing work cells where traditional manipulation of DLOs takes place e.g., automotive factories, medical manufacturers, aircraft manufacturing, etc. have work environments designed to be suitable for human workers. The transition from manual to robotic workstations would be easier if the robots were anthropomorphic (dual-arm), as it would enable them to do most tasks which their human counterparts are capable of performing. Furthermore, utilizing dual-arm robots would facilitate the designing of production environments to be human-centric; which allows for mixed working models (humans and robots could work separately in individual shifts or together in parallel), and reduces the need for additional hardware required by conventional (single arm) robot-centric workstations. Additionally, there is an almost inevitable requirement of two robotic arms to handle the greater diversity of behavior and the challenges posed by DLOs and perform well coordinated motions [3].

The proposed manuscript is the documentation of a portion of work done in the project REMODEL[1], which is a 4 year Research Innovation Action (RIA) project. REMODEL aims to bring about advanced handling techniques for DLOs in existing production processes, which are predominantly human-labor intensive. The primary DLOs of interest in project REMODEL include wires, cables, wiring harnesses, laces and flexible tubes/ hoses.

This paper presents an approach to develop a CPS for performing the routing operation of a cable by utilizing a dual-arm industrial robot, in a human centric environment. The dimensional characteristics of the cable are represented mathematically and are utilized by a novel ROS (Robot Operating System) based bimanual planning and control framework to physically manipulate the DLO with the robot.

The following contents of the document are organized as follows. Section II highlights the relevant existing research for hardware, software and DLO manipulation. Section III presents the developed CPS system and explains the methods

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement n° 870133, correspondent to the project entitled REMODEL, Robotic Technologies for the Manipulation of Complex Deformable Linear objects.

[1] https://remodel-project.eu/

used to perform the cable routing operation. Section IV showcases the results and outcomes of the conducted experiments. Finally, Section V documents the conclusions and possible areas for future research and exploration.

II. State of the Art and Theoretical Background

A. Dual-arm manipulation

Using robots in environments originally designed for humans has always been a topic of great interest, as it would allow replacing human workers with robots without the need for major changes to the production environment. However, this is a complex task, and requires robots to implement some human capabilities, such as dual-arm manipulation. This presents new challenges besides the ones that were already present in single-arm manipulation, as new aspects, such as arms synchronization, self-collision avoidance and the high redundancy and internal forces of closed chain systems; have to be taken into account [4]. This increase in the complexity mainly depends on the type of interaction between the arms. D. Surdilovic et al. [5] classified dual-arm manipulation as un-coordinated, when each arm is working on a different task, and coordinated, when both arms work together in the same task. Additionally, a subdivision of this last group was done, differentiating goal-directed, when there is no physical interaction between the arms, and bimanual movements, when both hands interact with the same object.

Due to the additional challenges, motion planning for dual-arm robots requires some new considerations. Lavalle suggested in [6] three strategies for decoupling each arm planning: prioritized planning, where the trajectory of one arm is calculated first and then the second arm trajectory is calculated considering the possible collisions with the first arm path; fixed-path coordination, where the trajectories of both arms are calculated independently and the goal is to find the timings to avoid collisions and obtain the required coordination; and fixed-roadmap coordination, that is an extension of the previous strategy to roadmaps. Other approaches extend techniques traditionally used for single-arm motion planning, such as Randomized Path Planners (RPP) or Rapid-exploring Random Tree (RRT) algorithms, considering the additional problems of collisions and coordination between arms [7], [8]. An example of this is [7], where a RRP algorithm is extended to calculate transit and transfer subpaths for a dual-arm robotic system, in order to move an object without collisions. To achieve this, the arms may have to change the grasp of the object. Thus, during the transit paths the arms can move independently without moving the object and during the transfer paths the arms move the object forming a kinematics closed chain with it.

In this work, a system for performing the bimanual manipulation of a cable is presented, and two different motion planning strategies are followed depending on the situation. When both arms move the cable forming a kinematic closed chain, the trajectory waypoints of one arm are calculated and the second arm moves simultaneously, following the first; whereas for the rest of cases a decoupling fixed-path coordination strategy is followed.

B. Robotic cable sensing and manipulation

Owing to the virtually infinite degrees of freedom (DoF) of DLOs, a planned grasp may deform a section of the object and can cause unpredictable deformation in other sections of the DLO body. This causes form closure failure (failure in applying kinematic constraints) in almost all scenarios, unless in the case of biaxial tension [9]. Additionally, this creates the need for a dynamic updating of the contact points and contact forces applied by the actuator element on the DLO, through an intelligent and iterative feedback loop mechanism, to bring about the desired manipulation result on the object [10]

Vision based monitoring systems are more pervasively utilized to continuously update the deformation model of the DLO in real-time. The obtained point cloud data is analyzed through various Machine Learning (ML) algorithms and the physical DLO are modelled as mathematical splines by utilizing specialized estimators [11] [12]. An alternative to utilizing vision is employing tactile sensing systems utilizing optoelectronic sensors or smart materials, which detect the cable shape and orientation due to the physical deformations caused by the object when they wrap around the DLO [13]. Combining both vision and tactile sensing technologies, improves the accuracy and efficiency of the system by overcoming the issues faced by the individual systems implemented as standalones [14]. Merging multiple sensing technologies add redundancy to the system. A less commonly used method is to purely depend on mathematical models based on the physical properties of the cable and use minimum-energy-based schemes to determine the deviations in the cable geometry and to update the system state [15].

After creating and establishing the system state models, the next stage is to plan for optimum manipulation sequences to bring about the desired effect on the DLO, based on the real-time evolution of the aforementioned conditions. Robot planning is necessary to determine the sequence of acceptable configurations for the manipulator, with respect to the DLO, to perform the tasks. And a control system is required to manage the inputs given to the physical robot, to realize the pre-planned motions [16]. Both robot planning and control can be performed by utilizing ROS. Namely, an existing motion planning framework based in ROS called MoveIt, which builds on ROS concepts and build-systems to create robotic projects for a wide range of applications. The MoveIt environment is capable of handling robot motion planning, manipulation and control. These concepts are covered in more detail in the following section.

C. ROS and MoveIt

ROS is a universal robotic framework, that can be used for creating software for any robotic system that is compatible with it [17]. One of the primary functions of ROS is to serve as an inter-process communication middleware, favoring the system modularity and the integration of systems [18]. This communication is carried out by exchanging data in the form of messages[2], which comprise of simple data structures. ROS enables the utilization of both Asynchronous and synchronous means of data exchange between the various nodes[3] (which represent the simplest components of the ROS systems). The asynchronous communication in ROS is done through a publisher/ subscriber concept, wherein the nodes can communicate with each other, by publishing their messages or subscribing to messages through a common topic[4], respectively. For making Remote Procedure Call (RPC) requests/ reply interactions, ROS utilizes a server/ client concept called services[5]. Services comprise of a two-set message wherein the client node sends a request to the server node and waits for the response. Additionally, for RPCs which require real time monitoring or the ability to pre-empt the process while mid-execution, ROS utilizes actions[6] which essentially provides the user tools to create goal-seeking servers, with means to monitor and kill it, through a client interface which can send the corresponding request to the server.

Regarding the motion planning and manipulation in ROS, MoveIt is the most widely used software [19]. It integrates motion planning, 3D perception, control, kinematic solvers, collision checking and scene information, allowing the fast generation of efficient and collision-free motion plans from a sequence of waypoints [20]. Additionally, MoveIt interfaces the robot controllers in order to send the trajectories to execute. These trajectories are sent in the form of $JointTrajectory$ messages[7], that contains a sequence of $JointTrajectoryPoints$. Each of these $JointTrajectoryPoints$ contains the joint values, speeds, accelerations and efforts of all the joints of a planning group (e.g. a robotic arm), as well as a time value, that is the time at which that point is reached since the beginning of the movement.

III. Cyber-Physical Cable Routing System

A. Robotic platform setup

The test platform where the experiments are conducted is in an enclosed cell following the safety standards for an industrial robot. The robot utilized is the Yaskawa SDA10F, which is a dual-arm industrial robot seven revolute joints in each arm and one in the torso. The robot is fitted with a Weiss WSG50 gripper, which has precise control over the opening/ closing distances of the jaw and the force exerted on the product. The platform also has a platform where the cable (DLO) specimen is initially placed with pre-determined grasp target points for the robot, to perform the initial grasping. The test table where the placing and routing operations are performed is a modular setup, with specialized guides to help support

[2]http://wiki.ros.org/Messages

[3]http://wiki.ros.org/Nodes

[4]http://wiki.ros.org/Topics

[5]http://wiki.ros.org/Services

[6]http://wiki.ros.org/actionlib

[7]http://docs.ros.org/en/noetic/api/trajectory_msgs/html/msg/JointTrajectory.html

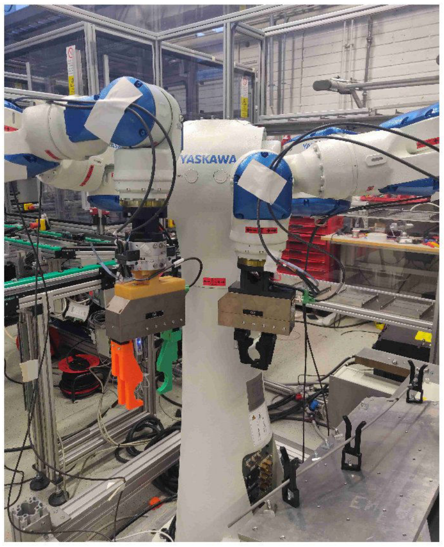
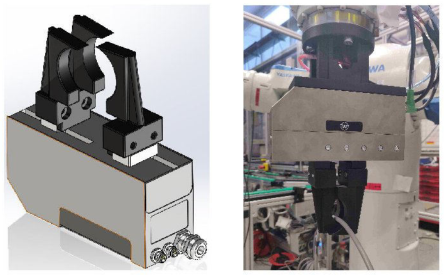


Fig. 1. Robotic platform setup, with the dual-arm industrial robot and the cable routing platform



(a) Fingers CAD model (b) Semi-grasp fingers position

Fig. 2. Gripper and specialized cable routing fingers.

the cables as they are routed by the robot manipulator. The physical hardware setup is shown in Fig.1

The gripper unit has a specialized design to align and center the midsection of the grasped cable. Additionally, the gripping system has three predetermined gripping jaw positions i.e., to perform fixed cable grasping, to slide along the length of grasped cable, and an open position to approach a cable to be grasped or to release an already grasped cable. This was defined based on the physical properties of the cable which includes its diameter, stiffness, and material. The gripper construction model and the semi-grasp position for sliding operation are illustrated in Fig.2.

The cable holder platform (Fig.7(a)) was utilized to preset the initial orientation, direction, and shape of the cable, to establish favorable conditions to grasp and manipulate the cables by eliminating uncertainty to some extent. The cable routing platform (Fig.7(b)) contains generic guides, which serve as insert locations and a support path, along which the robot arms could lay down and route the cable.

The sequence of actions being performed in this experiment is that the robot picks the cable from the cable holder platform, places one end of the cable into the first guide of the routing platform, and then proceeds to route the cable through the remaining guides as per the requirements of the path. The following section of the paper elaborates on the approach for determining the relevant strategy for contextual cable manipulation and a description of the sequence of actions performed in the cable routing experiment.

B. Reconfigurable ROS system

The goal for the development of the cyber-physical system was to make it flexible and easily reconfigurable, requiring little to no manual programming. Additionally, the system aims to be hardware-agnostic, i.e. that it can be implemented with any dual-arm robotic system with enough reachability. Due to this, ROS was the selected framework to develop and integrate the logic of the CPS. The ROS system is composed by two modules (see Fig.3):

1) Information Module: Composed by a set of ROS nodes that extract, process and distribute information about the performed process. This information is sent through ROS services to the trajectory generator module, providing it with the knowledge about the working environment, the manipulated objects and the cable path, that is used to generate the trajectories. The information is extracted from several input files that can be classified in three groups depending on what kind of information they provide. The first group comprises of files that contain information about the working environment, in particular about the cable routing and the cable holder platforms described in Section III-A. Each of these platforms is described by three files: A CAD file in x3d format, from which the relative pose of each element of the assembly with respect to its parent element is obtained; a xml file that describes the geometry of every different element present in the platform, including their dimensions and their keypoints (e.g. the center point of a guide) with respect its own origin frame; and a csv file that relates every element in the CAD file with its model in the xml file. The information of these files is processed and combined, obtaining the keypoints of every element referred to the origin of the platform. This information, as well as the dimensions of each of the elements, can be requested by means of ROS services.

The second group correspond to those files providing information about the manipulated objects. In this case, as the manipulated objects are individual cables, just one file is required, describing their dimensions and color. Finally, the last group is the one that provides information about the process. It is composed by a single file that contains the high-level action plan of the process, specifying the sequence of skills, the platform elements where to perform them and the manipulated objects (e.g. Route Cable1 through Guide1 and Guide2). There are three possible skills: Pick Cable, Insert Connector and Route Cable. As with the first group, the information of these last two groups can be also requested using ROS services.

2) Trajectory Generator Module: ROS node that generates and executes the robotic arms trajectories for the high-level action plan of the process, based on the information about the working environment and the manipulated objects, provided by the information module. The structure of this node is presented in Algorithm 1. First of all, the node request all the required information about the process to the information module. This information includes the high level action plan, the keypoints and dimensions of the working environment elements involved in the process, and the properties of the cables and connectors to manipulate. This data is processed and stored in a dictionary, organizing it initially by object and then by operation. This knowledge about the process is then used to parametrize the different skills.

Algorithm 1 Trajectory Generator Module

```
 1: procedure PICKCABLE($op\_data$)
 2:     Move robot to initial PickCable configuration
 3:     Calculate approach, grasping and retract keypoints
 4:     Calculate approach/retract waypoints by interpolation
 5:     Calculate waypoints for sliding close to connector
 6:     Calculate waypoints for aligning the Z axis of the grippers
        with the Z axis of the routing platform
 7:     Transform all the waypoints to the grippers' frames
 8:     Execute dual-arm movements through the waypoints
 9: procedure PLACECONNECTOR($op\_data$)
10:     Align robot torso with the routing platform's X axis
11:     Calculate approach, insert and retract keypoints
12:     Calculate waypoints to align the cable with the guide
13:     Calculate approach/retract waypoints by interpolation
14:     Transform all the waypoints to the grippers' frames
15:     Execute dual-arm movements through the waypoints
16:     Release and retract connector gripper
17: procedure ROUTECABLE($op\_data$)
18:     Reorient the gripper to face the next guide
19:     Calculate waypoints for approaching the next guide
20:     Calculate waypoints sliding extra cable length
21:     Calculate waypoints for grasping the cable with the other
        gripper
22:     Calculate circular waypoints to align the cable with the guide
23:     Calculate circular waypoints to insert the cable
24:     Update $pull\_distance$
25:     Transform all the waypoints to the grippers' frames
26:     Execute dual-arm movements through the waypoints chang-
        ing the gripper position between pulling and sliding
27: procedure MAIN
28:     $ops\_info \leftarrow$ Request the action plan of the process and
        info about all the involved elements to the Information Module
        through ROS services
29:     for $cable\_i \leftarrow 1$ to length of $ops\_info$ do
30:         call $PickCable(ops\_info[cable\_i][0])$
31:         call $PlaceConnector(ops\_info[cable\_i][1])$
32:         Calculate $pull\_distance$
33:         for $op\_i \leftarrow 3$ to length of $ops\_info[cable\_i]$ do
34:             call $RouteCable(ops\_info[cable\_i][op\_i])$
35:         Release cables and retract arms
```

Once the information is obtained, the node enters in a loop that executes the operations of every cable. For each operation, this information is used to calculate all the necessary keypoints and then, the waypoints of each arm are obtained by linear or circular interpolation between keypoints. The first operation of each cable is always picking it from the cable holder platform. Each extreme of the cable is grasped with one hand with a dual-arm movement. Then, the gripper that is farther to the

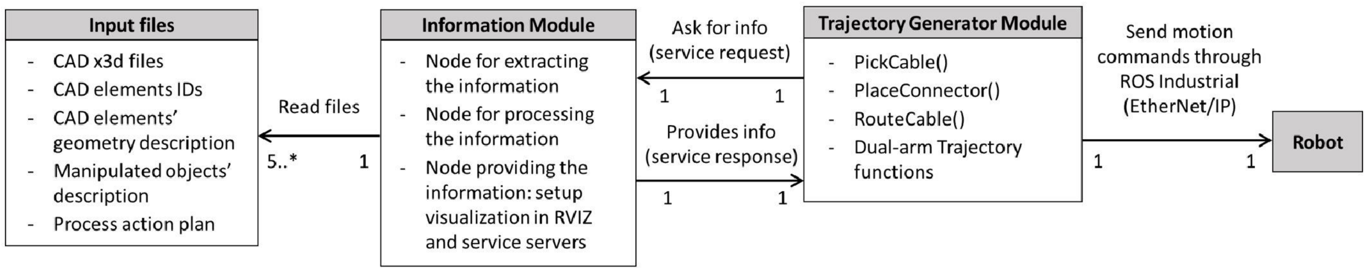

Input files
- CAD x3d files
- CAD elements IDs
- CAD elements' geometry description
- Manipulated objects' description
- Process action plan
Read files
5..*
1
Information Module
- Node for extracting the information
- Node for processing the information
- Node providing the information: setup visualization in RVIZ and service servers
Ask for info (service request)
1
1
Provides info (service response)
1
1
Trajectory Generator Module
- PickCable()
- PlaceConnector()
- RouteCable()
- Dual-arm Trajectory functions
Send motion commands through ROS Industrial (EtherNet/IP)
1
1
Robot

Fig. 3. UML class diagram of the ROS system

connector that will be inserted in the following operation has to slide along the cable, coming closer to the other gripper, achieving the grasp required for the future insertion. Due to the deformable nature of cables, depending on its shape, it can bend and get stuck in the fingers during this process. Therefore, to avoid this, the arm that is moving, moves down initially in order to slide the cable back against gravity so its shape is more predictable. These up/down movements are circular to keep the cable length constant and to not damage it. All these movements are defined with several waypoints that the fingers of the grippers have to go through. However, the node must convert these into waypoints for the wrists (last link of the robotic arms), so they can be sent to MoveIt to calculate the trajectories.

The second operation is always placing one of the cable connectors in a guide of the cable routing platform. For that the torso of the robot must align with the X axis of the platform, the grippers have to align the cable with the guide, insert it, and finally release and retract the hand that is closer to the connector. All this is done, as in the Pick Cable skill, by calculating keypoints from the obtained operation information, and then the waypoints of each arm using different interpolation strategies. After placing the connector, the $pull\_distance$ is calculated, i.e. the distance that the cable must be pulled during the routing operations to adjust the connector to the first guide. This value is calculated considering the fingers thickness, the guides thickness and the offset values.

Finally, once the connector is placed, the remaining operations consist on routing the cable through different guides. As the cable is a deformable object, it must be in tension when being inserted into the guide, such that its shape can be approximately a straight line. As in this approach no sensory feedback is used, such as vision or torque, the arm movements to keep the cable tension are calculated mathematically. First of all, the gripper of the routing arm changes its orientation to point to the next guide and approaches to it sliding the cable. Then, once the gripper is at a certain offset distance to the guide (A in Fig.4(a)), it slides the additional cable length that will be required for inserting the cable. This distance (R_{cable}) is the maximum distance that will be required during the circular movements to insert the cable, that can be at the top of the guide (R_{lift}, before inserting, Fig.4(b)) or in front of the guide (R_{insert}, after inserting, Fig.4(c)), depending on the guide dimensions.

$$R_{lift} = \sqrt{(L + X_{offset})^2 + (H + Z_{offset})^2}$$

$$R_{insert} = L + 2 \cdot X_{offset}$$

$$R_{cable} = max(R_{lift}, R_{insert})$$

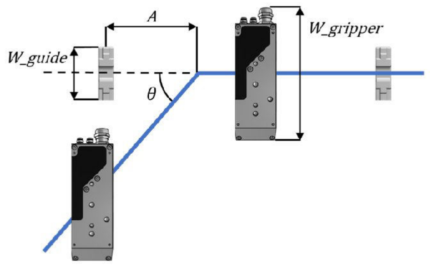
The variables of the previous formulas are graphically explained in Fig. 4. Additionally, in order to avoid collisions when sliding the cable, the minimum angle for the sliding direction (see Fig 4(a)) is calculated according to the following formula:

$$\theta > \arctan \frac{W_{gripper} + W_{guide}}{2A}$$

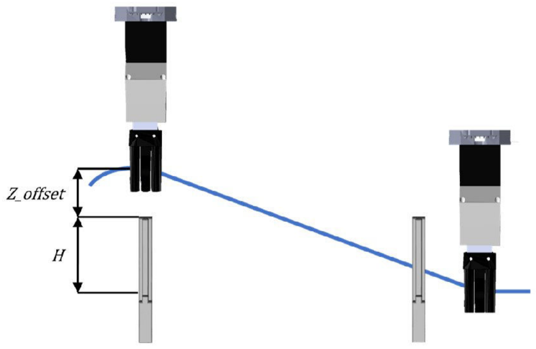
After sliding this extra cable length, the fingers will change to grasping position, and the other arm will move to grasp the cable in the previous guide. Then, the cable is pulled a small distance (function of the distance between grippers and the cable properties) to apply tension to the cable, and this distance between grippers is kept constant, being the radius of the lifting and inserting circular movements, thus keeping the tension of the cable all this time.

All the explained skills, calculate waypoints for each arm in order to generate the dual-arm trajectories. However, the *compute_cartesian_path()* method of the *MoveGroupCommander* MoveIt class just generates single arm trajectories. Hence, in order to generate dual-arm trajectories synchronizing arms and avoiding self-collisions, some functions have been developed (see Algorithm 2).

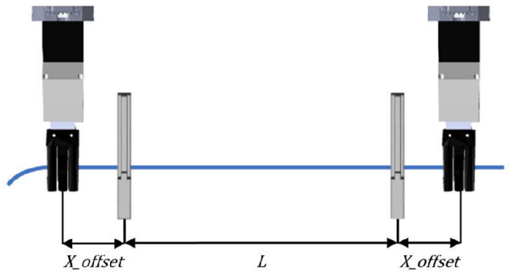
The waypoints of each arm, calculated in each skill, are sent to the $ExecuteDualTraj()$ function, where the potential self-collisions between arms are assessed and, in case of any, the trajectory is split and each part is calculated and executed separately. These are not real collisions, as the waypoints of both end effectors of the robot were calculated to avoid any kind of collisions (with itself or with the elements of the cell). However, this is necessary because the trajectory of each arm is calculated individually and then they are merged. Due to this, when calculating one arm, the other is considered to be static all the time and, even if it is supposed to move at the same time, its position is not updated for the self-collision checking of the MoveIt trajectory planner. Therefore, the MoveIt trajectory calculator could fail or retrieve a weird trajectory if the waypoints of one arm intersect with the initial



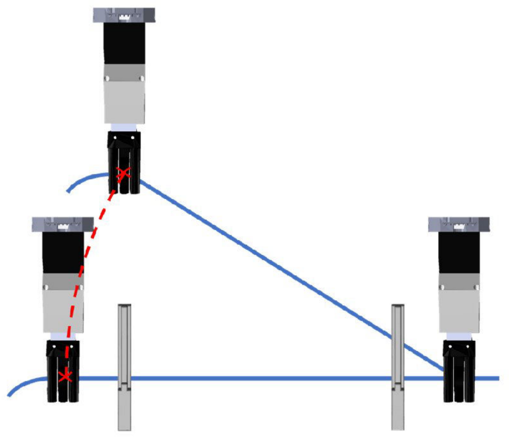
(a) Sliding additional cable length before lifting. Top view



(b) Minimum cable length for lifting the cable (R_{lift})



(c) Minimum cable length for inserting the cable (R_{insert})



(d) Inserting circular movement (R_{cable})

Fig. 4. Cable routing process.

position of the other arm. In this approach, this issue is solved by splitting the trajectory, so it can be executed in different steps, updating the real arms positions before planning the remaining part of the trajectory.

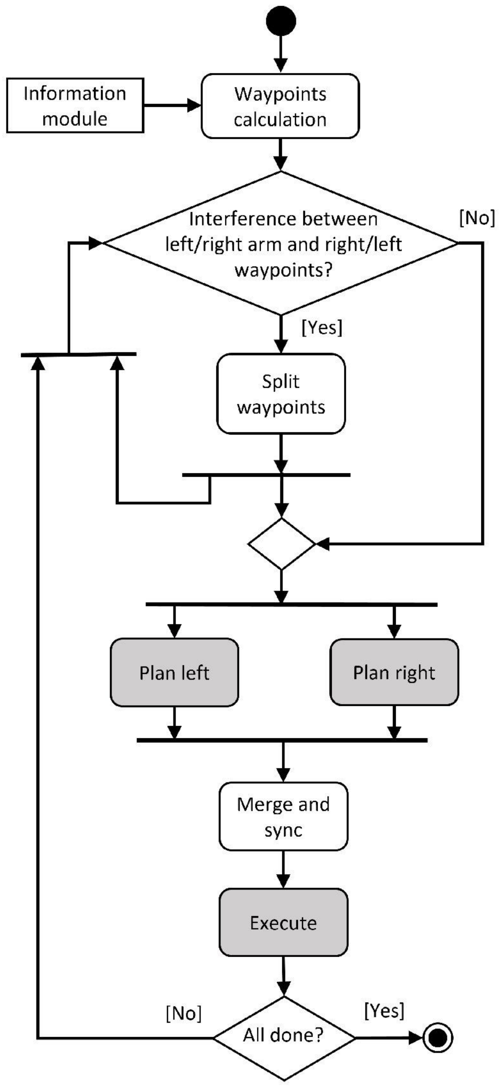


Fig. 5. UML activity diagram of the Trajectory Generator Module and its communication with the Information Module. The action boxes in white were developed in this paper and the ones in grey are existing functions of MoveIt.

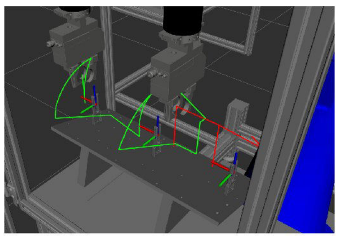
For merging the plans of both arms, they must be composed by the same number of $JointTrajectoryPoints$, as many as the longest trajectory, and they must finish at the same time, the time of the slowest arm. To achieve this, intermediate points are added to the shorter trajectory by interpolation of their joint values, speeds and accelerations. Finally a new trajectory is generated including the joint values of both arms for each time step. In Fig.5 it can be seen how this novel Bimanual Trajectory Generator Module works, representing in grey the existing functions of MoveIt and in white the additional developed functions.

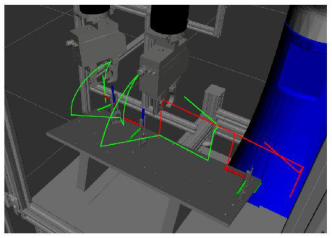
IV. Experimental Evaluation

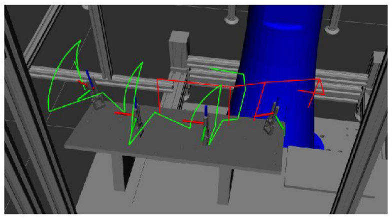
The cable routing system was tested for three different configurations of guides, both in RVIZ (A ROS 3D visualization environment) and with the real robot. In the first scenario, the cable is routed through three guides following a straight line

Algorithm 2 Dual-arm trajectory generation

```
1:  procedure MERGEPLANS($traj\_R, traj\_L$)
2:      $long\_traj \leftarrow$ Determine the trajectory with more $JointTrajectoryPoints$
3:      $short\_traj \leftarrow$ append intermediate points by interpolation to have the same length than $long\_traj$.
4:      $slow\_traj \leftarrow$ Determine the slower trajectory
5:      $times \leftarrow$ times of the $slow\_traj$
6:      $dual\_traj \leftarrow$ Create a new trajectory including the joint values of both arms for each time step
7:      return $dual\_traj$
8:  procedure DUALCARTESIANPATH($wp\_R, wp\_L$)
9:      $traj\_R \leftarrow compute\_cartesian\_path(wp\_R)$
10:     $traj\_L \leftarrow compute\_cartesian\_path(wp\_L)$
11:     $dual\_traj \leftarrow$ call $MergePlans(traj\_R, traj\_L)$
12:     execute($dual\_traj$)
13: procedure EXECUTEDUALTRAJ($wp\_R, wp\_L$)
14:     for $i \leftarrow 1$ to length of $wp\_R$ do
15:         if $|wp\_R[i], L\_pose| \leq collision\_threshold$ then
16:             $j = \frac{len(wp\_L)}{len(wp\_R)} \cdot i$
17:             $wp\_R\_1, wp\_R\_2 \leftarrow$ split $wp\_R$ at index $i$
18:             $wp\_L\_1, wp\_L\_2 \leftarrow$ split $wp\_L$ at index $j$
19:             break
20:     for $i \leftarrow 1$ to length of $wp\_L$ do
21:         if $|wp\_L[i], R\_pose| \leq collision\_threshold$ then
22:             $j = \frac{len(wp\_R)}{len(wp\_L)} \cdot i$
23:             $wp\_L\_1, wp\_L\_2 \leftarrow$ split $wp\_L$ at index $i$
24:             $wp\_R\_1, wp\_R\_2 \leftarrow$ split $wp\_R$ at index $j$
25:             break
26:     call $DualCartesianPath(wp\_R\_1, wp\_L\_1)$
27:     call $ExecuteDualTraj(wp\_R\_2, wp\_L\_2)$
28: procedure MAIN
29:     call $ExecuteDualTraj(wp\_R, wp\_L)$
```


(a) 1st scenario, straight cable


(b) 2nd scenario, one change of direction


(c) 3rd scenario, two changes of direction

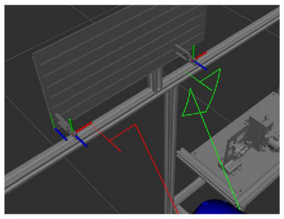
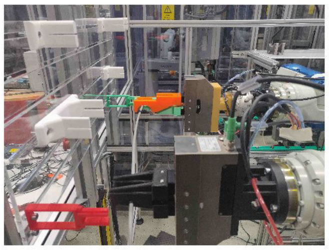

(d) Pick cable

Fig. 6. RVIZ visualization of the generated trajectories (right arm in green and left in red).


(a) Robot picking the cable from the cable holder platform

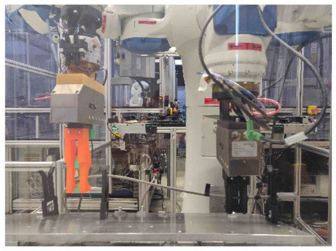

(b) Robot routing the cable along the guides of the cable routing platform

Fig. 7. Real robot during the execution of the cable routing process.

(Fig.6(a)); in the second, the cable is routed again through three guides but with a change of direction (Fig.6(b)); and in the third, four guides are used and the cable path has two changes of direction (Fig.6(c)). In the three experiments a 650 mm long cable with a 6 mm diameter and a VGA connector in one of its ends was used, and it was picked from the cable holder platform, as can be seen in Fig.6(d).

As Fig.6 shows, the trajectories were generated correctly for the three tested scenarios, and the dual-arm movements were executed properly both in RVIZ and with the real robot, routing the cable satisfactorily through the guides (Fig.7(b)). The coordination between arms was good, there were no entanglements when sliding the fingers along the cable, and the cable was not damaged during the manipulation. Additionally, it was proved that the information module works correctly, switching from one scenario to another just by changing its input files, allowing the fast and easy reconfiguration of the system.

V. Conclusions and Future Directions

The manipulation of flexible materials is a challenging task in the robotics and automation field. In most cases, in order to deal with the uncertainties of these objects, the process ends up being too tailored for the specific robotic manipulator. The work done in this paper presents a solution for this issue, developing a CPS for routing a cable in a human-centric workplace using a dual-arm robot. The robot is equipped with two parallel grippers with specialized fingers, designed to either grasp or slide the cable, and the work cell is composed by two platforms, one for holding the cable initially in a known position, and another where the guides for routing the cable are mounted. Regarding the software of the system, it was developed in ROS and it is composed by two modules, one for extracting and providing information about the routing process and the other one in charge of generating and executing the robot trajectories. The system was tested for three different cable routing paths, following a straight line and with one and two changes of direction. The dual-arm trajectories were generated and executed correctly in the three experiments, and the cable was successfully routed in all of them. This shows that the strategy of keeping the cable in tension during the manipulation reduces the uncertainty to some extent, being able to work without sensor data. Additionally, the experiments probed the easy and fast reconfigurability of the system, modifying the process just by updating the information input files.

After the successful results of this first implementation of the system, the next goal is to extend it in order to route and assembly several wire harnesses. This process presents additional challenges, such as separating cable branches and routing multiple cables together. Therefore, the introduction of sensory information in the system will be considered.

REFERENCES

[1] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey,” *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[2] W. M. Mohammed, P. Malvido Fresnillo, S. Vasudevan, Z. Gosar, and J. L. Martinez Lastra, “An approach for modeling grasping configuration using ontology-based taxonomy,” in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, 2020, pp. 507–513.

[3] M. Saha and P. Isto, “Manipulation planning for deformable linear objects,” *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, 2007.

[4] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, “Dual arm manipulation—A survey,” *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340–1353, Oct. 2012.

[5] D. Surdilovic, Y. Yakut, T.-M. Nguyen, X. B. Pham, A. Vick, and R. Martin-Martin, “Compliance control with dual-arm humanoid robots: Design, planning and programming,” in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, Dec. 2010, pp. 275–281, iSSN: 2164-0580.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[7] Y. Koga and J.-C. Latombe, “Experiments in dual-arm manipulation planning,” in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, May 1992, pp. 2238–2245 vol.3.

[8] N. Vahrenkamp, M. Do, T. Asfour, and R. Dillmann, “Integrated Grasp and motion planning,” in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2883–2888, iSSN: 1050-4729.

[9] F. Guo, H. Lin, and Y.-B. Jia, “Squeeze grasping of deformable planar objects with segment contacts and stick/slip transitions,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3736–3741.

[10] Y.-B. Jia, F. Guo, and J. Tian, “On two-finger grasping of deformable planar objects,” in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 5261–5266.

[11] J. Schulman, A. Lee, J. Ho, and P. Abbeel, “Tracking deformable objects with point clouds,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1130–1137.

[12] S. Jin, C. Wang, and M. Tomizuka, “Robust deformation model approximation for robotic cable manipulation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6586–6593.

[13] P. M. Fresnillo, S. Vasudevan, W. M. Mohammed, J. L. M. Lastra, G. Laudante, S. Pirozzi, K. Galassi, and G. Palli, “Deformable objects grasping and shape detection with tactile fingers and industrial grippers,” in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2021, pp. 525–530.

[14] D. De Gregorio, R. Zanella, G. Palli, S. Pirozzi, and C. Melchiorri, “Integration of robotic vision and tactile sensing for wire-terminal insertion tasks,” *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 585–598, 2018.

[15] A. J. Shah and J. A. Shah, “Towards manipulation planning for multiple interlinked deformable linear objects,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3908–3915.

[16] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, X. Li, J. Pan *et al.*, “Challenges and outlook in robotic manipulation of deformable objects,” *arXiv preprint arXiv:2105.01767*, 2021.

[17] “ROS: Home.” [Online]. Available: https://www.ros.org/

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5, issue: 3.2.

[19] “MoveIt Motion Planning Framework.” [Online]. Available: https://moveit.ros.org/

[20] S. Chitta, “MoveIt!: An Introduction,” in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ser. Studies in Computational Intelligence, A. Koubaa, Ed. Cham: Springer International Publishing, 2016, pp. 3–27.