A Fast Score-Based Method for Robotic Task-Free Point-to-Point Path Learning

A Fast Score-Based Method for Robotic Task-Free Point-to-Point Path Learning

Alex Pasquali, Kevin Galassi and Gianluca Palli

***Abstract*—The manipulation of deformable objects represents an open research topic because of the difficulties in accurately modeling the object behavior in real-world scenarios. This paper presents a trajectory planning framework for the assembly of wiring harnesses for the automotive and aerospace sector, reducing the learning time and simultaneously presenting suitable performance and reliability. A genetic algorithm is used to generate new trajectories according to application constraints. Those trajectories are then executed by the robot and evaluated by means of proper sensor feedback. The proposed framework enable to learn and autonomously improve the task execution, while mantaining a significantly low programming time. Experimental results are reported showing how the robot is capable of optimizing the manipulation of the DLOs gaining experience along the task repetition, while showing high success rate from the very beginning of the learning phase.**

***Index Terms*—Manipulation, Deformable Objects, Machine Learning, Genetic Algorithm, Industrial Manufacturing**

I. Introduction

Despite the recent advancements in automatic production systems, the industrial manufacturing tasks involving manipulation of deformable objects, like the electrical harness for domestic appliances or automotive industries, have still very limited automation. The robotic manipulation of Deformable Linear Objects (DLOs) is a complex task [1]–[3]. In [4] an algorithm to automatically generate trajectories for the manipulation of electrical cables based on a limited a set of instruction is proposed.

It results that industrial manufacturing tasks involving deformable object manipulation are still characterized by intense human labor. On other hand, the introduction of easy-to-program collaborative robots, enabling the possibility to register the desired trajectory through the technique called kinesthetic teaching, can surely produce a benefit for the automation for these complex manipulation tasks. Recent research, in fact, shows how human workers are starting to accept more easily to be aided by robots [5], while others focus more on how to interface the robot with humans in a safe environment [6]. EMGs signals can be used as well in order to further improved the user experience during kinesthetic teaching [7].

Nowadays, another discussed trend in production plants is the inclusion of learning techniques to improve the execution of a given task. In robotics trajectory planning, these methods can be divided into supervised learning or Reinforcement Learning (RL). In the first category, the autonomous system uses a set of labeled data to learn a desired behavior or trajectory [8]. However, in many applications, an adequate annotated dataset cannot be easily created, and the problem is even worsened when system reconfiguration is required or a change of the execution path is needed. On the contrary, RL does not require any knowledge of the desired trajectory and utilizes a reward signal obtained from the interaction between the robot and the environment to train and learn a policy that describes the sequence of action to take based on the state. Several attempts based on these techniques been proposed, each of them having the following main limitations. Firstly, it has been outlined how is more difficult to learn policy when robots are involved, this is given by the high dimensionality of the system and the possible control action [9], [10]. Another key aspect is the safety of reinforcement learning, in fact during the exploration of the possible solutions the robot behavior may turn to be unsafe. For this reason, recent research aims creating a safer environment for testing proposing an intermediate solution between supervised and reinforce [11]. Even in this case, changing the product and the trajectory requires the complete re-training of the policy or the network, increasing the time to deploy a robust solution [12] and in general, the deployment of model-free RL techniques on real robots is highly related to the hyper-parameters [13]. An assessment of reinforcement learning algorithm applied to DLO manipulation were made in [14], while in [15] the problems related to hyper-parametrization and training time for a DLO manipulation system are discussed. The benefit that RL can provide to improve trajectories recorded by kinesthetic teaching are discussed in [16], [17].

In this context, the present work presents a framework specifically adapted to the robotic manipulation of DLOs, maintaining a significantly low programming time and, at the same time, able to improve the task itself during execution. The proposed framework can be used to improve the robot trajectories specified by the user through waypoint definition or kinesthetic teaching, optimizing the path and further reducing the stress on the cable and the execution time.

II. Methodology

The proposed method, illustrated in Fig. 1, make uses of genetic algorithms and is based on the usage of a user-specified path as initial *father* (Sec. II-A), obtained either from kinesthetic teaching or just from pre-defined points,

Alex Pasquali, Kevin Galssi and Gianluca Palli are with DEI - Department of Electrical, Electronic and Information Engineering, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.

This work was supported by the European Commission's Horizon 2020 Framework Programme with the project REMODEL - Robotic technologies for the manipulation of complex deformable linear objects - under grant agreement No 870133.

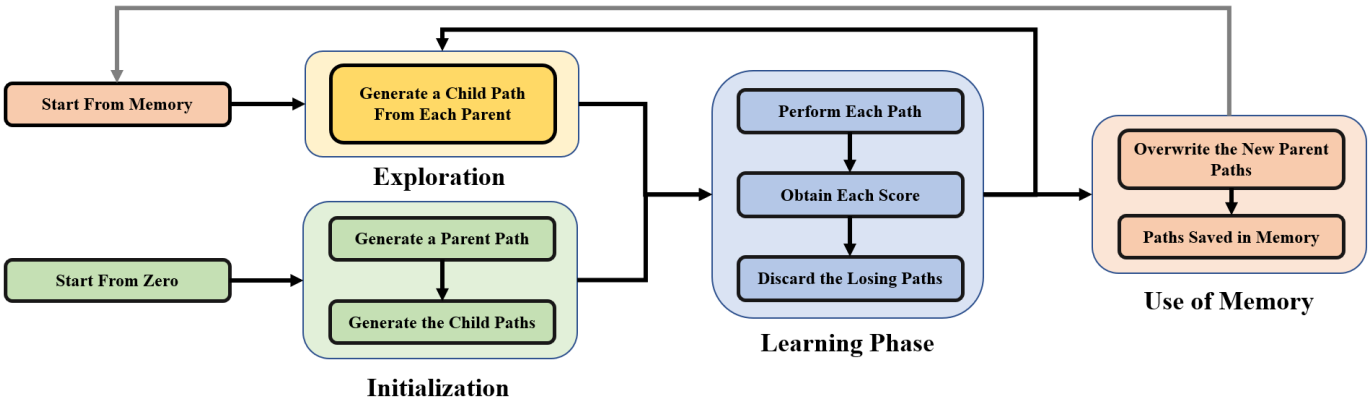


Fig. 1: This diagram illustrates the main components of this path learning algorithm, which consists of 4 main parts: *Initialization*, *Exploration*, *Learning Phase* and *Use of Memory*.

characterized by the capability of achieving the task successfully. Despite being able to fulfill the task, this path may not be optimal in terms of performance. Therefore, alternative paths are generated through the application of the concept of mutation, as detailed in Sec. II-B, enabling competition among the solutions. During each iteration of the algorithm, all paths are executed and evaluated using a predefined score function. The winning paths are chosen as the breeding parents for the generation of a new set of solutions (Sec. II-C). Through this process, only the best paths are propagated, ensuring that only a better result replaces the previous one.

A path is defined as a sequence of ordered three-dimensional points connected by linear motion with predefined tools orientation, as depicted in Fig. 2, ensuring a fully deterministic behavior of a robot. A point P is defined as $P = [x, y, z]^T \in \mathbb{R}^3$ and a path $\mathcal{T}$ as $\mathcal{T} = \{P_s, P_{w,1}, ..., P_{w,n}, P_e\}$ where n is the number of ordered waypoints and $P_s, P_{w,i}, P_e \in \mathbb{R}^3$ are three-dimensional points that represent respectively the start-point, the waypoint i and the end-point. Each $P_{w,i}$ can be designated as either mutable or non-mutable. The number n and the locations of points $P_s, P_{w,i}, P_e$, used to characterize a path, are considered to be user-defined variables. The number n of waypoints $P_{w,i}$ that characterize a path needs to be balanced: too many points would lengthen the learning time, while too few points may not be sufficient to characterize the task properly, harming the performances. These waypoints can be set, by the user, as mutable or non-mutable for the learning phase.

A. Initialization

The path initialization is performed by the user either by means of kinesthetic teaching or by using conventional robot path design tools. As previously said, the user must define the starting P_s and ending point P_e of the path, which are already considered as definitive for the solution, and the desired number n of intermediate ordered waypoints $P_{w,i}$ with a specific location number i. In case less than n waypoints are fixed by the user, the remaining not-fixed

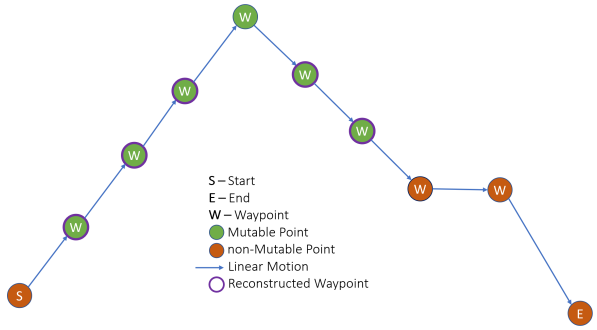


Fig. 2: Characterization of a path using both mutable and non-mutable waypoints and the process of reconstructing a waypoint in case the user does not provide all the necessary points for the path.

waypoints are automatically set as mutable and reconstructed by using a linear interpolation among the fixed ones (Fig. 2). An advantage of the characterization of the points between mutable and non-mutable is the possibility to specify areas in which, for example due to possible collision, the optimization should be limited. In this way, a complete first path is obtained, that if it is set up correctly is already able to perform the task, which will allow the learning phase to take place while the application is running properly.

B. Exploration

The process of obtaining a new path is achieved through mutation. A mutation refers to a random modification of a mutable waypoint $P_{w,i}$ of an existing path, referred to as the *father* path. Through this modification, a new path, referred to as the *child* path, is generated. In order to improve the overall quality of the path, the waypoint $P_{w,i}$ to be mutated is chosen based on a probability p_i that is proportional to its impact on reducing the path's score $R(\mathcal{T})$ (Fig. 3). A Score function

$R(\mathcal{T})$ of a path $\mathcal{T}$ can be seen as:

$$R(\mathcal{T}) = \sum_{k=k_i}^{k_f} f(k) \tag{1}$$

where k_i is the number of the first time iteration when the path begins to be executed, k_f the time iteration when it ends, and $f(k)$ a generic always negative user-definable function ($f(k) < 0 \ \forall k$) that defines the contribution of the score at each time iteration. The sum of these contributions defines the score $R(\mathcal{T})$.

Note that the use of a generic score function $R(\mathcal{T})$, the algorithm can be applied to a large variety of task like a DLO manipulation described in this application, however with the rearrange of the function it is possible to apply the same method to an interaction task.

By subdivide the values provided by the score function $f(k)$ (Eq. (1)), it is possible to assign weights w_i to the mutable waypoints $P_{w,i}$. Through normalization of these weights w_i, probabilities of selection p_i can be derived. In order to do that it is possible to consider $n_T = n + 2$ as the number of points in a path, $n_k = k_f - k_i$ the number of iterations that a path spends to be performed and a defined function $f(k) < 0$ $\forall k$ for the score function $R(\mathcal{T})$ which refers to Eq. (1). To subdivide the value of the score function $R(\mathcal{T})$ it is possible to define a base B and an offset O:

$$B = \lfloor \frac{n_k}{n_T} \rfloor; \qquad O = \lfloor \frac{n_k}{2n_T} \rfloor$$

The i-th weights w_i, which will then be converted to probabilities p_i, are calculated as:

$$w_i = -\sum_{k=iB-O}^{iB+O} f(k)$$

$$\forall i \in [1, n] : P_{w,i} \textit{ is mutable}$$

The probabilities p_i associated to the waypoint $P_{w,i}$:

$$p_i = \frac{w_i}{\sum w_k}$$

$$\forall i \in [1, n] : P_{w,i} \textit{ is mutable} \quad \forall k \in [1, n] : P_{w,k} \textit{ is mutable}$$

When a point is selected for the mutation, it is spatially displaced along one of its principal directions (x, y or z) chosen at random to change its position. The displacement d is also randomly chosen between two user-defined maximum displacement d_M and minimum displacement d_L. This allows the user to control the degree of exploration of the search space and can be adjusted to balance the trade-off between exploration and exploitation.

C. Learning Phase

The proposed strategy exploits the concept of *father* and *child*. The mutation of a waypoint from in a *father* path F generates a new *child* path C as detailed in Sec. II-B. The approach therefore provides that, starting from the first path, a number of user-defined *child* paths are generated, in such a way that the number of overall paths n_p is even. This is because at each stage S_i, that represent the set of paths to be executed at the evolution number i, the robot will perform the paths that have been defined and the algorithm will discard half of them based on the scores obtained. The first stage S_1 can be represented as:

$$S_1 = \{F, C_1, ..., C_{n_p-1}\}$$

In this first stage the *Children* $C_1, ..., C_{n_p-1}$ are generated using a totally random mutation since it is not available a score value $R(F)$ for the initializing path F. On the next stage the winning trajectories will become the *fathers* $F_1, ..., F_{\frac{n_p}{2}}$; each of these *father* paths generate a single *child* path $C_1, ..., C_{\frac{1}{n_p}}$, using the mutation method explained in Sec. II-B, in such a way at each stage there are the same number of paths competing n_p. The i-th stage S_i:

$$S_i = \{F_1, ..., F_{\frac{n_p}{2}}, C_1, ..., C_{\frac{n_p}{2}}\}$$

In this way, only the best trajectories will be allowed to move forward. Only a better result can replace an old one. A redundant methodology was selected, in which the same trajectory is repeated multiple times, in order to ensure the validity of the results under the challenging conditions in which the learning process must operate. This allows to minimize the occurrence of false positive results. The results on the same path may be different, as these scores, which are generated through a user-chosen function $f(k)$ that quantifies the goodness of a path, often depend on physical parameters such as contact forces or friction. An example of this can be seen in the next section,

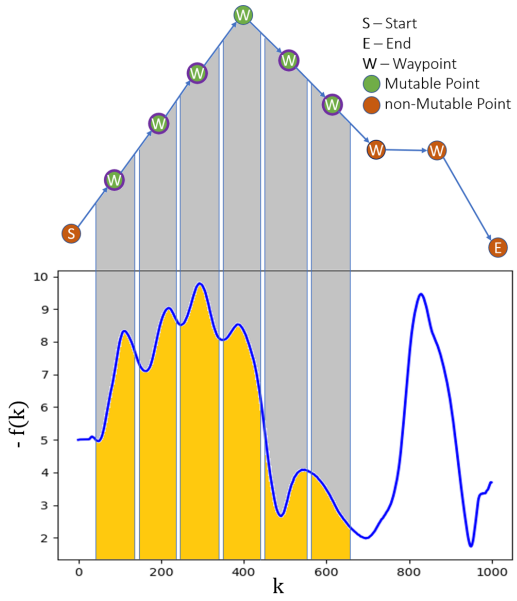


Fig. 3: The illustration shows graphically the influence that the waypoints of a path $\mathcal{T}$ have on the $-f(k)$ score values, highlighting the score areas of interest for each point. This example shows the negative of the score, then the $-f(k)$ function to enhance the visualization of the example.

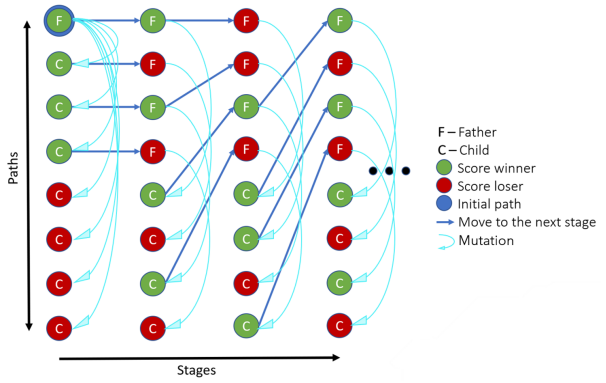


Fig. 4: Progression that occurs during the learning phase between $n_p = 8$ paths, with the advancement of winning paths. Also highlighted is the first stage $S_1 = \{F, C_1, ..., C_{N_p-1}\}$ where there is the initializing path F (*Father*) and a $n_p - 1$ number of paths generated C_i (*Children*) with $i \in [1, n_p - 1]$

Algorithm 1 Score-Based Path Learning

if (Start from Memory) **then**
  $S \leftarrow$ Last Stage in Memory
else
  $S[0] \leftarrow F$
  $S[1 : N_p - 1] \leftarrow$ Generation of $n_p - 1$ Children
end if
while (Stop Criterion) **do**
  for (T in S) **do**
    Perform The Path T
    $R(T) \leftarrow$ Collection of the Scores values $f(k)$
  end for
  $S[0 : \frac{N_p}{2} - 1] \leftarrow$ Best Half Paths (new Fathers)
  $S[\frac{N_p}{2} : N_p - 1] \leftarrow$ Generation of a Children for Each Father
  Saving The Current Stage in Memory
end while

focused on manipulating objects such as DLOs (Deformable Linear Objects). The learning process, shown in Fig. 4, can be considered to have reached its termination point when the scores obtained exhibit saturation, indicating that the path for maximizing the score has been identified and a viable solution to the task has been obtained. However, it is also possible to continually run the algorithm in an effort to further improve task execution while still successfully completing the task.

D. Use of Memory

The use of memory in a learning path method is important for the ability to pause and restart the execution of the algorithm. By using memory to store the stages ($S_i = \{F_1, ..., F_{\frac{n_p}{2}}, C_1, ..., C_{\frac{n_p}{2}}\}$) of the learning phase, the learning process can be resumed from where it left off, rather than having to start over from the beginning, saving time and computational resources. It allows monitoring the progress, and if needed, to adjust the parameters of the algorithm or the system.

III. Experiments

To evaluate the algorithm, it has been realized an experimental setup to reproduce the operation of cable routing for automotive harness manufacturing. The task consist in the laying down of a wire or a sub-harness in a given configuration to permit a taping gun to fix the multiple cable together forming a harness ready to be used. Normally, the aforementioned operations are performed by manual labor, however there is a particular interested in the automatization of such process and offer the possibility to test the algorithm in a real industrial scenario. The robot used for the operation is a collaborative 7DoF Panda Robot equipped with an external force/torque sensor mounted on the end-effector. Additionally, the gripper mounts specifically designed fingers with the tips' surface covered by a material that permit the sliding of the cable where the pressure is not exercised.

A. DLO Manipulation test

The application necessitates the proper organization of three distinct groupings of cables in a specific harness configuration, which are:

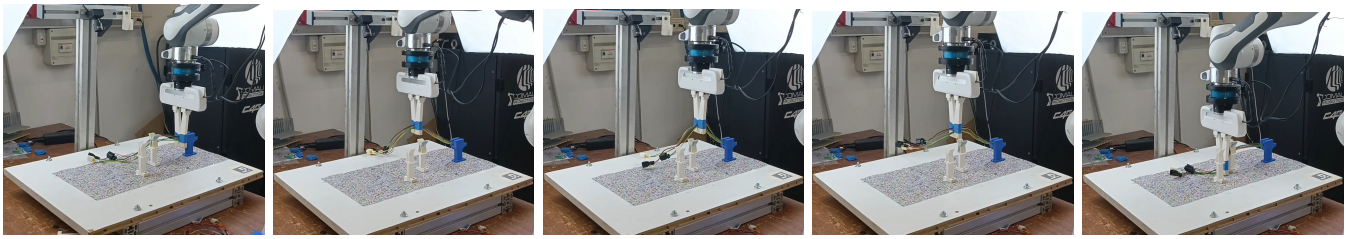
- Six-Poles Connector (SPC) cable group;
- Ten-Poles Connector (TPC) cable group;
- Eleven-Poles Connector (EPC) cable group.

The environment was specifically designed to accommodate this requirement, and the implementation of two clips and an appropriate connector block arrangement was deemed necessary, as can be seen from the Fig. 6. Each cable group movement is decomposed into subsequent tasks, at first the cable fixed on one end should be inserted in a clip, and after that the movement occur between two clips.

The goal of this experiment is to develop a robot capable of performing the manipulation tasks while maintaining balanced the module of the forces $|F|_k = \sqrt{F_{x,k}^2 + F_{y,k}^2 + F_{z,k}^2}$ obtained by the sensor placed between robot's gripper and the robot's flange. The scoring function described in 1 is chosen to consider the variance $\sigma_{|F|}^2 = \frac{1}{n_k}\sum_{k=k_i}^{k_f}(|F|_k - \mu_{|F|})^2$, mean $\mu_{|F|} = \frac{1}{n_k}\sum_{k=k_i}^{k_f}|F|_k$, and peak values $|F|_{max}$ of the modules of the forces $|F|_k$, with the aim of maintaining balanced the forces and minimizing any force peaks as much as possible, as we are dealing with fragile components that could be damaged if handled improperly. The always negative score function $R(\mathcal{T})$ is:

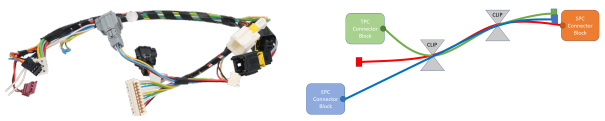
$$R(\mathcal{T}) = -\sigma_{|F|}^2 \mu_{|F|} |F|_{max}$$

Considering SPC cable group as the first subject of the experiment, the final user defines the path points providing the start-point P_s, end-point P_s and the intermediate waypoints $P_{w,i}$ that can be set to be mutable or immutable. In this specific case, it was decided to impose two non-mutable way-points ($P_{w,8}$, $P_{w,9}$) near the end point (P_e) and one mutable way-point ($P_{w,5}$) in the middle of the path. The starting



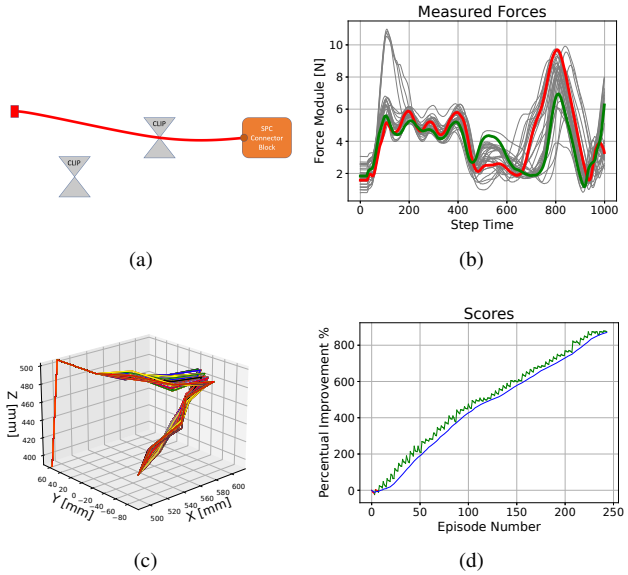
(a) P_s (b) $P_{w,5}$ (c) $P_{w,8}$ (d) $P_{w,9}$ (e) P_e

Fig. 5: Sequence of points chosen for the first experiment concerning the SPC cable group.



(a) (b)

Fig. 6: Fig. 6a shows the end result of the proper organization of the three distinct cable group. Fig. 6b shows the top view implementation idea of the set-up by highlighting the arrangement of the clips and connector blocks.



(a) (b) (c) (d)

Fig. 7: First experiment on the SPC cable group. Fig. 7a shows the desired result for the task in the top view. Fig. 7b graphs the exchanged forces obtained during the learning phase. Fig. 7c graphs the paths explored during the learning phase. Fig. 7d shows the trend of the improvements function I during the execution of the algorithm.

point (P_s) was chosen to ensure correct cable grip in the proximity of the connector block. The intermediate way-point $P_{w,5}$ chosen is used to strategically indicate which direction of motion is favorable for performing the task, based on the user's experience (Fig. 5). The final sequence of points ($P_{w,8}$, $P_{w,9}$) was chosen to ensure successful cable insertion, as the endpoint P_e of the path is strictly defined by the user. The orientation of the robot's end-effector is kept constant to maintain the gripping fingers orthogonal to the direction of cable exit from the connector block and at the same time parallel to the clips to facilitate the cable insertion. A total of $n_T = 11$ user defined path points were recorded, while $n_P = 8$ were the number of competing paths at each stage S.

The improvement value $I(F_{i,k})$ at stage S_k of a winner path F_i can be seen as:

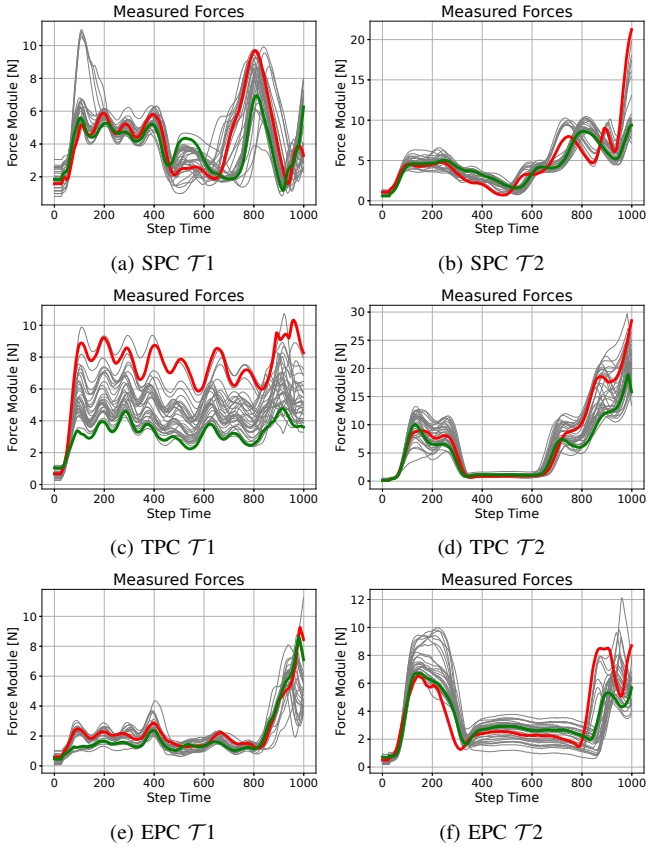
$$I(F_{i,k}) = I(F_{1,k-1}) + 100\frac{\left|\frac{1}{R(F_{i,k})}\right| - \left|\frac{1}{R(F_{1,k-1})}\right|}{\left|\frac{1}{R(F_{1,k-1})}\right|} \qquad (2)$$

$$\forall k \in [2, +\infty); \qquad \forall i \in [1, \frac{n_p}{2}]$$

where $I(F_{1,1}) = 0$ in order to give the initial conditions to the equation. By utilizing this expression, the function is able to depict the improvement in a relative manner, thereby providing a clear illustration of how the current solutions surpass those of the predecessors. Regarding the experiment, in Fig.7 the result obtained from the execution of the SPC. As it is possible to see in Fig.7d, from the very first iteration, the algorithm is capable to sensibly improve the improvement function I seen in eq. 2

B. Trajectory improvement

The data in Fig. 8, shown how the algorithm is capable to improve the trajectory along all the given path. In all the six sample trajectories taken from the real case setup, the algorithm was capable to sensibly improve the score, reducing the force exchange, by approximately 9.13% up to 2934.17% following the desired reduction offered by the cost function. The table also presents data indicating relatively high peak values for the application, such as a value of $28.47N$ in the TPC $\mathcal{T}2$ I. These peaks are observed during the process of inserting cables into clips. The clips are mechanically rigid and require a considerable amount of force to be opened. The best result were reported in the trajectory corresponding to the TPC $\mathcal{T}1$ trait. The worst performance instead in TPC $\mathcal{T}2$ however, the result obtained are still better than the original taught trajectory.



(a) SPC $\mathcal{T}1$ (b) SPC $\mathcal{T}2$

(c) TPC $\mathcal{T}1$ (d) TPC $\mathcal{T}2$

(e) EPC $\mathcal{T}1$ (f) EPC $\mathcal{T}2$

Fig. 8: Red: Initial Path, Green: Last Path, Grey: Tested trajectories. The figure shows snapshots taken during the execution of the algorithm.

Path	Paths Performed	$\lvert F\rvert_{max}$ [N]	$\mu_{\lvert F\rvert}$ [N]	$\sigma^2_{\lvert F\rvert}$	I_{abs}
SPC $\mathcal{T}1$ I SPC $\mathcal{T}1$ L	488	9.70 6.95	4.33 3.83	4.06 1.81	+254.52%
SPC $\mathcal{T}2$ I SPC $\mathcal{T}2$ L	336	21.25 9.38	4.65 4.42	11.75 4.3	+555.38%
TPC $\mathcal{T}1$ I TPC $\mathcal{T}1$ L	416	10.32 4.77	7.31 3.20	3.57 0.58	+2934.17%
TPC $\mathcal{T}2$ I TPC $\mathcal{T}2$ L	208	28.47 18.93	6.76 5.24	45.94 20.43	+335.92%
EPC $\mathcal{T}1$ I EPC $\mathcal{T}1$ L	504	9.27 8.61	2.29 1.96	2.35 2.68	+9.13%
EPC $\mathcal{T}2$ I EPC $\mathcal{T}2$ L	312	8.69 6.72	3.57 3.40	4.98 2.49	+172.75%

TABLE I: Learning Results (I: Initial Path, L: Last path). I_{abs} is the improvement in absolute terms between the I and L paths of the same task.

IV. Conclusions

In this work, a framework for continuous learning of a point-to-point trajectory has been implemented. The algorithm takes as input a trajectory in the workspace, which can be easily obtained through kinesthetic teaching or programming. During task execution, the algorithm improves the trajectory in order to minimize the effort required by the robot. The results obtained from manipulation of deformable linear objects were promising, and the framework is planned to be extended to other industrial tasks, such as interaction tasks, in future work. The focus on minimizing the effort required by the robot is important for increasing efficiency and reducing operational costs in industrial settings. By continuously learning and refining the trajectory, the framework has the potential to significantly improve the performance of robotic systems in a variety of applications. As future application, the framework will be tested in an interaction task in addition to the 'grasp and drag' approach proposed.

References

[1] N. Lv, J. Liu, X. Ding, J. Liu, H. Lin, and J. Ma, "Physically based real-time interactive assembly simulation of cable harness," *Journal of Manufacturing Systems*, 2017.

[2] N. Lv, J. Liu, and Y. Jia, "Dynamic modeling and control of deformable linear objects for single-arm and dual-arm robot manipulations," *IEEE Transactions on Robotics*, 2022.

[3] S. Iwamura, Y. Mizukami, T. Endo, and F. Matsuno, "Cable-path optimization method for industrial robot arms," *Robotics and Computer-Integrated Manufacturing*, 2022.

[4] K. Galassi and G. Palli, "Robotic wires manipulation for switchgear cabling and wiring harness manufacturing," in *IEEE Int. Conf. on Industrial Cyber-Physical Systems*, 2021.

[5] C. Bröhl, J. Nelles, C. Brandl, A. Mertens, and V. Nitsch, "Human–robot collaboration acceptance model: Development and comparison for germany, japan, china and the usa," *International Journal of Social Robotics*, 2019.

[6] M. Pantano, D. Regulin, B. Lutz, and D. Lee, "A human-cyber-physical system approach to lean automation using an industrie 4.0 reference architecture," *Procedia Manufacturing*, 2020, 30th International Conference on Flexible Automation and Intelligent Manufacturing.

[7] R. Meattini, D. Chiaravalli, L. Biagiotti, G. Palli, and C. Melchiorri, "Combining unsupervised muscle co-contraction estimation with bio-feedback allows augmented kinesthetic teaching," *IEEE Robotics and Automation Letters*, 2021.

[8] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[9] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[10] R. Calandra, A. Seyfarth, J. Peters, and M. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence (AMAI)*, 2015.

[11] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," *arXiv preprint arXiv:1707.05173*, 2017.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[13] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," *10.48550/ARXIV.1809.07731*, 2018.

[14] M. Bednarek and K. Walas, "Comparative assessment of reinforcement learning algorithms in the taskof robotic manipulation of deformable linear objects," in *2019 4th International Conference on Robotics and Automation Engineering (ICRAE)*.

[15] R. Zanella and G. Palli, "Robot learning-based pipeline for autonomous reshaping of a deformable linear object in cluttered backgrounds," *IEEE Access*, vol. 9, 2021.

[16] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, 2011.

[17] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008.