An Interactive Simulator for Deformable Linear Objects Manipulation Planning

Nahum Alvarez and Kimitoshi Yamazaki

Abstract— **Manipulation of Deformable Linear Objects is a difficult task for robots in real environments due to requiring high computational power in order of predict accurately the object's deformation. Taking advantage on the proliferation of cheap and easy to use 3D environment simulation engines, in this paper we propose a method for manipulation planning of deformable linear objects which uses a physics simulation engine predicting the object behavior, and calculate possible outcomes for the manipulation. We implemented a system designed to generate a plan with the necessary actions to carry out in order to bring a deformable linear object from a configuration to another. In our system, a planner simulates the object's behavior when being manipulated and choses the most suitable action to perform. All the processing is done in an interactive way, being possible to interact in real time with the simulated object to correct the manipulations or insert additional objects to the environment. Our method enables such complex manipulation in order to manipulate the objects efficiently and generate accurate plans. Also, it offers other secondary contributions like fast sample database creation for automatic training methods and fast initial gripping point selection.**

I. Introduction

In our society, deformable linear objects (from here onwards, we will refer to them as DLO) are found in a wide number of industrial products, and their manipulation is a challenge for autonomous robots that intervene in the process. This is mainly due to the behavior of the object after each interaction, requiring the robot to adapt to its changing shape continuously, often being necessary to build complex reasoners using high amounts of computational power and time [1]. Thus, these objects have special characteristics to take in account, like a high probability to self-collide when manipulated, make knots or become entangled, requiring specific prediction methods [2]. To fulfill these requirements, we can find a number of approaches, like using approximation rules or customized physics calculations [3] [4], being another suitable solution to use a third-party simulation engine [5]. This option allows generating 3D objects, manipulate them and calculate the outcomes of such manipulations in a dynamic way, saving costly efforts in terms of development time and computing requirements. This is the option we chose due to being a powerful solution but also an affordable one, and easily extendable to other similar research.

In this paper we present a feature often overlooked: an interactive mode capable of modifying the simulation in the middle of the planning process. The motivation for focusing in this capability is to allow a wider range of adaptability in the task manipulating DLO, correcting potential plans or modifying on real time the environment. This feature would improve the usefulness of the planning system for training robots or assisting them in learning tasks.

The concrete contribution of this work consists in an interactive simulation system capable of generate short plans for bringing an input DLO to a certain goal configuration being a fast and simple method, capable of predict accurately the behavior of deformable linear objects and create plans for their manipulation. The system does this task by performing different movements on the object, and selecting the most suitable for each plan step. The main contribution of our work is this interactive planning method for DLO manipulation with the novel aspect of allowing real time interactive modifications in order to adapt to a dynamic environment or correct previous plan actions. The system allows to obtain initial and subsequent grasping points, short manipulation plans, or training robots. The paper is structured as follows: section 2 reviews related works in the literature, highlighting studies on DLO manipulation and interactive systems for autonomous robot's support. Section 3 describes in detail our method and architecture, and section 4 shows the experiment we carried out. Finally, we discuss in section 5 the results we obtained in the experiment and present our conclusions and future plans on this research in section 6.

II. Related work

Manipulation of DLOs is a field with a number of complex aspects: recognition, action planning, gripping method and gripping point selection, for example [6]. Recognizing a DLO can be done using a representation consisting in a series of connected points, or Point Chain Model. This model simplifies the processes of recognizing and simulating it, as it is enough to detect the line it "draws" in the space and then generate a virtual one without losing too much information: even if the DLO will be able only to bend and move through a set of aligned points, but with enough of them, it will allow an acceptable realistic simulation, minimizing the computational time required to do it [7], and we use it in our system as well for those reasons. Other models use it in combination with a list of crossing points [8], make use of deep learning techniques in order to label

*Research supported by New Energy and Industrial Technology Development Organization (NEDO).

Nahum Alvarez is with the Shinshu University, Nagano, Japan (e-mail: nahum@shinshu-u.ac.jp)

Kimitoshi Yamazaki is with the Shinshu University, Nagano, Japan (e-mail: kyamazaki@shinshu-u.ac.jp).

the key features of the object [9], or even the complete definition of a curved cylinder [2]. Additionally, in order to allow an accurate physical simulation of the object the point configuration of the DLO is not enough, and it is necessary to provide additional parameters like its friction, flexibility or elasticity degrees among others, but few models capture those parameters, leaving their tuning to the hands of a domain expert [10].

Even the most basic form of manipulation -Picking a DLO- is a non-trivial task: its initial gripping point is important because the depending of where and how the manipulator pick the DLO, it will behave differently and determines its subsequent manipulation so is necessary to create some prediction model for its behavior once picked [11]. Once the DLO has been grabbed by a manipulator, the next objective is to bring it to a goal configuration by performing a plan of ordered transformations. We can find a number of different approaches in the literature for the plan generation. Techniques derived from knot theory has been used to solve unraveling problems or knot strings with a state based planner [12], using a roadmap based method [2], or learning by observation techniques [7]. Other techniques deal with finding a minimum path of stable configurations of the DLO between the starting and goal states, with a specific made physics model [13] or calculating the deformation needed through approximations and proceeding with the appropriate manipulation [14]. However many of these methods do not have in account the DLO's behavior and its interactions with the physical world, or are prepared only for tightly controlled environments (even ignoring gravity interactions). Calculating the DLO's future state and its reaction is important because while manipulating it we have to calculate potential collisions with other objects or itself, or even taking in account other issues related with its nature, like compensating vibration or hanging [15]. This issue becomes more important if we want to include some form of interaction that cannot be predicted initially, requiring a planner advanced enough to adapt to dynamically changing situations. This feature has been subject of deep research in planning in general, but was applied to robots scarcely until recent years, when human-robot collaboration applications identified the issue as a need to be developed further [16]. In the literature, we can find instances with variable degree of interactivity and manipulation planning like trajectory planning [17] or learning by observation, but systems that react actively to changes in the environment are scarcer. An examples of such planners is ReAct [18], a system that implements a complete logic framework that is able to plan interactively allowing user's modifications in real time, but requires a very low level and strict definition of the objects introduced and its behavior. Other examples can be seen in [19], with a system designed to collaborate with humans in disaster situations, and [20], which proposes a theoretical model for robot task assistance requiring collaboration with humans in order to correct or confirm the plan at some points. However, the interaction in there is merely done by giving orders and not with actions or trough external objects. Our system avoid to use the interaction as a means of direct control of the robot and instead treats it as an environment changing force. In our system, we use a 3D virtual environment simulation engine for this task. Previous research on automatic robot manipulation has taken advantage of simulation engines, saving time and complexity [21], using this software for different purposes, having different degrees of physics simulation: building a sensor simulator and taking care of ray casting physics [22], using only the graphic simulation for manipulator arms but using a different physics engine, due to the specific physics calculations needed for motion planning [23], or simulating robot models and their sensors [24]. However, using such engines for simulating the physical behavior of the manipulated objects has been somewhat sparse.

III. DLO Simulation and planning

In this work, we present an interactive system that performs DLO simulation and manipulation planning with adaptation to real-time modifications in the environment. The system receives the information related to the initial configuration of a DLO and a goal configuration of the same object, then as a result of its process, it generates a plan containing the necessary steps for transforming the initial configuration into the goal configuration. The initial and goal configurations of the DLO are given in the Point Chain Model format, containing an ordered list of the three-dimensional coordinates of points. This model gives fast computing times but at the same time enables an acceptable level of realistic simulation [5][6], allowing us to simplify the way in which the system will manipulate the DLO, as the points composing it are the possible places where the potential actions that modify its configuration will be performed, and the simulation engine will bend the DLO by those points after calculating its physical behavior.

The use of a physics engine for the simulation tasks enables us to simulate objects in an interactive way, allowing "on the fly" modifications or corrections if the planner detects significant deviations in the outcome. The system is prepared to load an initial environment with the DLOs, including the manipulator gripper hands and basic obstacles. Figure 1 depicts an example of the environment with a DLO and a gripper hand. Once started the planning process, it is possible to modify the existing objects or introduce new ones. Then, the planner adapts to these changing situations

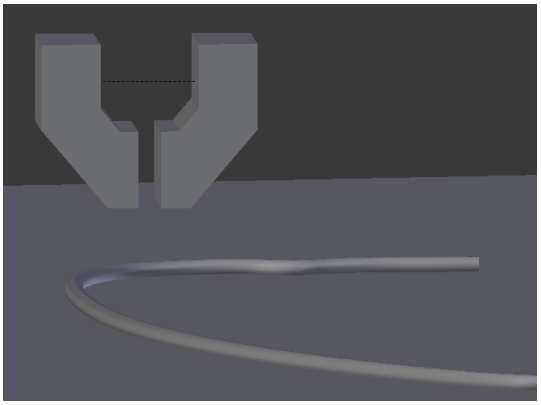

Fig. 1: A DLO and a gripper hand in the virtual environment. The hand is provided with a set of actions for gripping and moving, allowing interaction with the DLOs

and calculates the appropriate actions according to the environment. The resulting output of the planner is a list of ordered steps; each step is composed of a number of movement actions carried out by a set manipulator hands, with the limitation of one action per hand. Each action has a set of parameters like the point of the DLO on which the action is performed.

The movement actions available in the system are:

- Lift: the gripper hand takes the DLO on a certain point and lifts it to a certain height.
- Lower: the manipulator lowers is height and open the gripper hand, releasing the DLO if it was gripping it.
- Cross: the gripper hand takes the DLO on a certain point and pushes it, until it crosses below the other gripper hand if present, or until a certain distance is reached.
- Move: the gripper hand takes the DLO on a certain point, lifts it and moves to the coordinates relative to its center of the equivalent point in the goal DLO.
- Release: the gripper hand opens, dropping the DLO if it was gripping it.

The whole planning process can be divided in two stages: setup, and planning: the setup stage prepares the virtual environment and the necessary data for running the planner, the planning stage being the proper system in which the potential actions are tested, generating a plan containing the steps for bringing the DLO to its goal configuration. Also in this stage, the system executes each plan step as it selects the best action to perform, finally showing the resulting behavior and configuration of the DLO. Each stage is described in detail in the next subsections.

A. Setup Stage

In the setup stage, both initial and goal DLOs are loaded from their description and simulated in the virtual environment, which initially contains just a floor; the goal DLO is simulated for reference, and as we will see later, for backwards simulation planning. Gripper hands and objects present in the initial or goal setup are also simulated if necessary, for example in case the goal configuration needs to be gripped in some way by a hand. Finally, the system is provided with a set of the available manipulator gripping hands. Each gripper hand is a generic object with a 3D component that also implements a set of functions for its creation and action performing. An example of a simple hand can be seen in Figure 2.

B. Interactive Planning Stage

Once this setup is ready, the system starts with the planning stage and searches which steps are necessary to put the DLO into the goal configuration. The planner uses a backwards simulation method where the goal configuration object is manipulated at the same time as the initial one, trying to obtain intermediate points for the plan. The idea is that after manipulating the initial DLO, the best result is not necessarily the most similar to the goal configuration, but to an intermediate configuration obtained by manipulating the goal DLO. This intermediate configuration is obtained by performing the same action over a copy of the goal DLO, on a point equivalent to the one where the action is performed at the initial DLO; this way the copy of the goal will end in a configuration where part of its points are in a similar position as the initial configuration. A general depiction of this search strategy is shown in Figure 2. The flow of the planner is depicted in a graph in Figure 3 and in detail works as follows:

1. The system begins by performing sequentially each action over each one of the DLO's points.
2. Once the action finishes, a modified initial DLO is obtained, and the system checks if it is stable. For the system, "stability" means that the DLO remains in that configuration without falling or slipping. Unstable

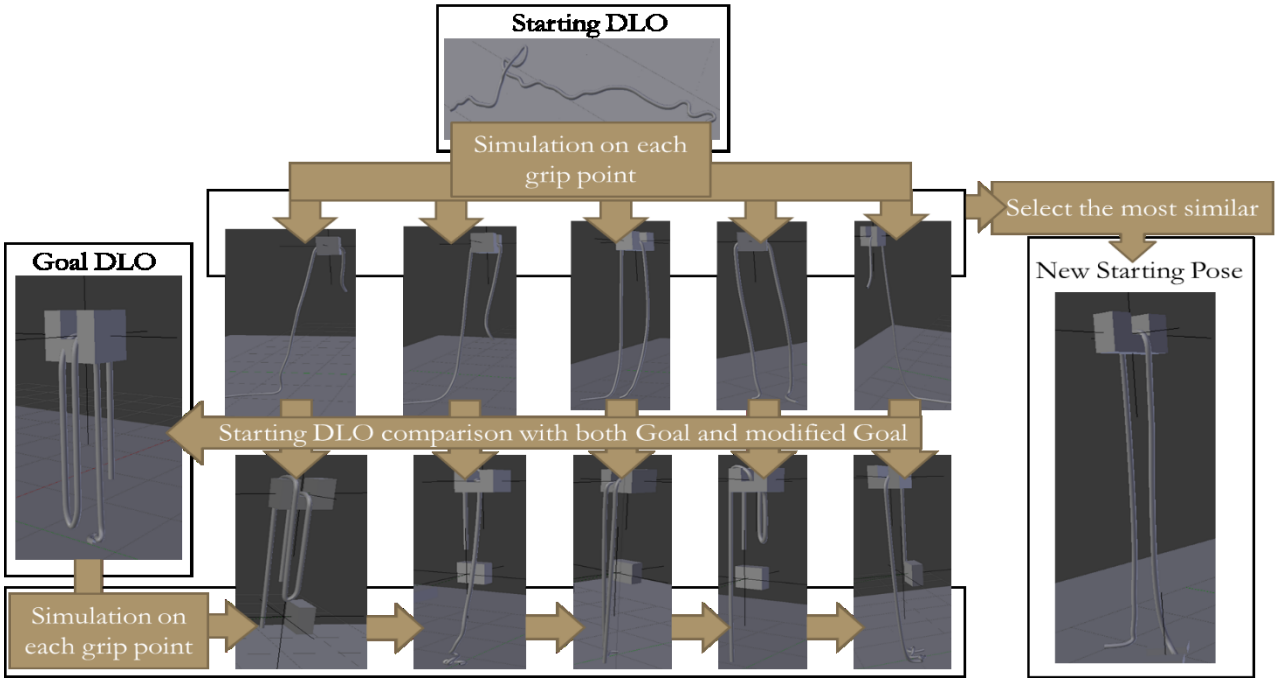

Fig. 2: The system's simulation process. A movement action is performed on each one of the DLO's possible grip points and the system stores the outcome. It also simulates backwards from the goal object and when finishes, it selects the most similar outcome to the goal.

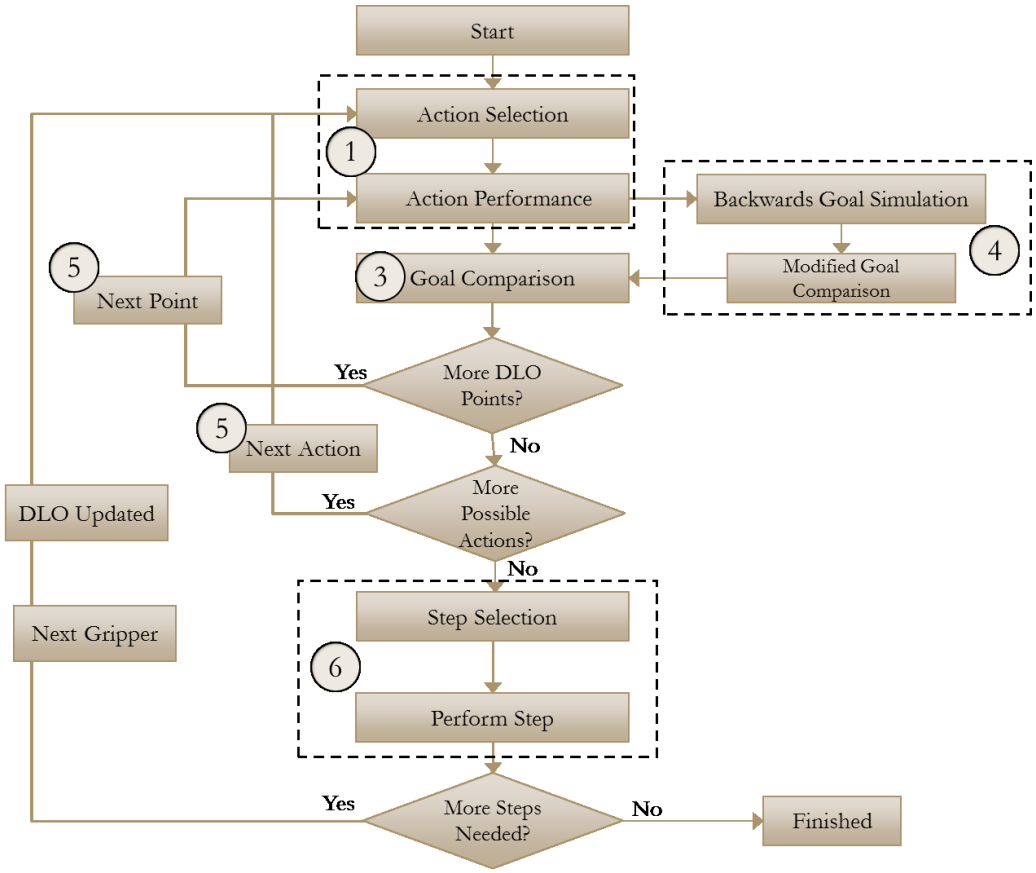

Fig. 3: The planner's workflow. The planner is prepared to work with a set of movement actions and manipulator hands, and checks if backwards planning is required. The numbers in the diagram reference the corresponding number in the text.

configurations are discarded by the planner.

3. If the resulting configuration is considered stable, the system compares it with the DLO goal configuration, generating a difference score; the more similar both DLOs are, the lower this number will be. Thus, the score is stored along with the gripping point, performed action, and action's parameters as comparison data. The process continues with the next DLO point in the Point Chain Model until all the points have been tested.
4. At the same time as the action simulation goes through each DLO's point, the system also makes a copy of the goal object and the same action is performed on it at the same point. The result is a modified goal DLO, which is also compared with the modified initial one and its comparison result is stored as well. The reason for creating a modified goal DLO is to check if the current action fits as an intermediate action necessary to reach the goal configuration. If this comparison yields a lower difference score than the obtained for the same action in the previous step, the configuration is stored replacing the previous data related to the action, and if is greater, is discarded like the unstable configurations.
5. After storing the comparison data, the initial and goal DLO are returned to their original configurations and the process is repeated for the next point in the initial DLO. Once the system finishes the simulation over all the DLO's points, it starts again with another action, repeating this cycle until no more actions are remaining.
6. Once finished, the planner executes the action from the comparison data with the lowest difference score.

There are two instances where the planner will start over again the process, but this time using the resulting object as a new initial DLO:

- In the case of obtaining the resulting DLO from a comparison with a modified goal DLO, meaning that the current configuration of the DLO is intermediate and requires further planning.
- If the difference score of the resulting DLO is greater than a determined threshold, meaning that the DLO still needs to be manipulated.

In these cases, the system will follow the same process than in the search of the first step, performing actions on each point of the new initial DLO, obtaining a new resulting DLO, and checking again if more steps are necessary.

Also, the user has the capability of interact with the environment at any time, including when the planner is in the middle of a step calculation. This interaction can be done by inserting, modifying or deleting objects or changing the configuration of the DLO. This is allowed in order to reflect real situations when the robot manipulates a DLO and unexpected variables may appear in the middle of the manipulation process influencing the operation, like other entities (robots or humans) or naturally occurring events (wind, objects falling). As the system allows to insert these new circumstances on real time in the virtual environment, when this happens the planner detects it and re-starts the current step's planning process.

Finally, when the system decides that no more steps are needed, mainly because it considers that the DLO is enough

similar to the goal object, it finish the planning process and generates the plan containing the details and parameters of each one of the selected step's actions.

IV. Validation Experiment

We conducted an experiment in order to test if our planner generates viable plans in situations where unpredicted events can hinder the plan success, trying to identify possible future issues. The experiment consisted on selecting a DLO from a set of virtual objects previously generated from different photographs of cables, and generating a plan for bringing it to a goal configuration, selected from the same set. In the middle of the planning process, an interactive event would be randomly triggered that could disrupt the original plan, making the system to re-plan the actions in order to reach the goal configuration in this new situation. The set of pre-generated DLOs contains 9 objects with different configurations and same length and number of points in their Point Chain Model. The starting configuration of the DLO could be initially interacting with other objects, like gripper hands or obstacles, which would be simulated as well when the object is selected.

The experiment consisted of running the planner for each combination of DLO, with the only exception of not choosing the same initial and goal DLO. We configured the planner to stop when the resulting DLO configuration has a difference score compared with the goal DLO lower than 100. Then, from all the plans we obtained, we selected 40 plans that had more than one step. Then, we did a second run for each one of the combination of DLO related to the selected 40 plans, but after the planner selected and executed the first step of the plan, we introduced interactively an event modifying the simulation environment. This event was selected randomly from a set of three: placing a solid cube block near the DLO (randomly placed inside a distance range), inserting wind blowing in a random direction (strong enough to move the DLO continuously), and modifying the DLO configuration by pushing it (this modification is moderate, not pushing the DLO more than a fixed distance in a random direction and pushing it on a random DLO point). Finally, the system was configured to operate with up to two manipulator gripper hands, and all of the available actions.

V. Results and Discussion

We classified the obtained plans by their number of steps and the difference score between the resulting DLO and the goal DLO. The results comparing both runs of the system can be seen in Table 1.

The numbers are similar in both runs, and even though the plan without external interaction obtained slightly better average values, after conducting a T test for two populations assuming equal variances with an alpha value of 0.05 we observed that there was no significant difference between them ($t = 0.13$ for the number of steps and $t = 0.14$ for the difference score). This means that our system is equally efficient whether the original plan is performed or if unexpected events happen in the environment and the planner has to recalculate.

Table 1: Average values for the number of steps taken and the difference score obtained by the planner.

Round 1		Round 2	
Steps	Difference Score	Steps	Difference Score
2.45	64.15	2.675	59.225

If we focus in the individual results, we can see that the average number of steps in the second round is slightly greater than in the first one. Intuitively, after any unpredictable event, the previously executed step and its potential plan won't match perfectly with the current situation; in some of the observed cases, the changes caused by the event were not enough to foil the original plan, only being necessary to adapt the current step calculation, but in other instances, the plan had to be extended with an extra action to bring the DLO to a more suitable configuration, thus increasing the average number of steps. On the contrary, we can see that some of the cases generated a difference score for the second run slightly lower, which can be explained by some instances of the experiment benefiting from the random events by chance, allowing a closer final configuration to the goal. Nevertheless, as we stated before those differences are not statistically significant and represent only anecdotal events; we can affirm that the planner we developed is robust enough to generate viable plans under unpredicted circumstances. There were two more interesting issues we observed. First, in the case of inserting a solid block near the DLO, in some of the actions the planner tried to move the DLO trough the block. Naturally, the simulator would detect the collisions, but as the gripper is being moved forcefully both objects overlapped. However, the DLO as is being controlled only by the simulator, reacted against the force and was dropped, being detected by the planner as an "unstable" action. This is the intended (and correct) behavior, but it inspired us to add as a future improvement the capability to detect an overlapping and avoid the obstacle with the gripper hand that is carrying the DLO. Secondly, we observed that adding wind influenced greatly to the outcome of the planner, being an important nuisance if it blows "against" the selected movement. However, if by chance the wind blows to the advantage of the action, it would help reducing the difference score or even the number of steps. The reason for this is that depending of its direction, the wind could help the DLO fit better to the gripper, improving its stability, or hinder the action, reducing it. We observed both these effects in the experiment and we think that adapting to the wind would help the planner in optimizing its actions.

VI. Conclusion

In this paper we presented a system that makes use of a simulation and physics engine to generate plans for DLO manipulation able to receive interactive modifications. The goal of our research is the development of an interactive and cheap physics simulation tool capable of inputting plans to robots for DLO manipulation that adapts to external events

that change the operation environment. The system generates a plan of actions to be performed on a DLO in order to bring it to a goal configuration, and allows interacting with the planning process and the environment at any time, changing the simulation parameters or introducing new elements. The system uses a license free simulation engine in order to simulate the physics of the process, and the planner adapts to the new situations on real time.

We conducted an experiment in order to test the if the planner is able to generate valid plans when external events meddle with the process, where we simulated a number of DLO and used them as initial and goal DLO, randomly generating plans to bring the initial DLO configuration to the goal one, introducing a random event that should disrupt the original plan. The results we obtained indicate that the planner adapts to the events effectively and generates satisfactory plans statistically similar to the one it would have generated without the interactive events. We also identified a number of features we can improve, mainly adapting better to certain events, having in account their particularities. In the short term, we will improve the system by adding more interactive events in order to do a larger scale experiment, and see how the planner reacts to different external stimuli and if there are more ways to counteract them like in the case of inserting obstacles or wind. As for our future plans, we aim to add those countermeasures to different interactions, improving the adaptability of the planner. Also, we see the necessity of testing our system with a real robot in a real situation so we are planning to develop a bridge interface between our system and the robot, and perform an experiment similar to the one we did in the virtual environment.

In conclusion, we think that in the case of action planning for autonomous robots, is very important not only to be able to generate plans a priori from a determined situation, but also allow to interactively modify the environment and react on consequence, as adaptability is a required characteristic of robots interacting with humans in real life situations, such as collaborative tasks or natural environments with a high level of unpredictability. As a consequence of this necessity, we developed a planning system for DLO manipulation robust enough to allow interactive modifications of the environment on real time and adapt to them, generating effectively plans for bringing the DLO from a configuration to another.

REFERENCES

[1] Frank, B., Stachniss, C., Abdo, N., & Burgard, W. (2011, September). Efficient motion planning for manipulation robots in environments with deformable objects. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on (pp. 2180-2185). IEEE.

[2] Saha, M., Isto, P., & Latombe, J. C. (2008). Motion planning for robotic manipulation of deformable linear objects. In Experimental Robotics (pp. 23-32). Springer Berlin Heidelberg.

[3] Henrich, D., Ogasawara, T., & Wörn, H. (1999). Manipulating deformable linear objects-Contact states and point contacts. In Assembly and Task Planning,(ISATP'99) Proceedings of the 1999 IEEE International Symposium on (pp. 198-204). IEEE.

[4] Yue, S., & Henrich, D. (2002). Manipulating deformable linear objects: sensor-based fast manipulation during vibration. In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on (Vol. 3, pp. 2467-2472). IEEE.

[5] Juang, J. R., Hung, W. H., & Kang, S. C. (2011). Using game engines for physical–based simulations–a forklift.

[6] Henrich, D., & Wörn, H. (Eds.). (2012). Robot manipulation of deformable objects. Springer Science & Business Media.

[7] Takamatsu, J., Morita, T., Ogawara, K., Kimura, H., & Ikeuchi, K. (2005). Representation of knot tying tasks for robot execution. JOURNAL-ROBOTICS SOCIETY OF JAPAN, 23(5), 66.

[8] Shirakawa, Tomoya; Matsuno, Takayuki; Yanou, Akira; Minami, Mamoru, "String shape recognition using enhanced matching method from 3D point cloud data," in System Integration (SII), 2015 IEEE/SICE International Symposium on , vol., no., pp.449-454, 11-13 Dec. 2015. doi: 10.1109/SII.2015.7405021

[9] Huang, S. H., Pan, J., Mulcaire, G., & Abbeel, P. (2015). Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects. In Intelligent Robots and Systems (IROS), 2015 RSJ International Conference on (pp. 878-885). IEEE.

[10] James, D. L., & Pai, D. K. (1999, July). ArtDefo: accurate real time deformable objects. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (pp. 65-72). ACM Press/Addison-Wesley Publishing Co..

[11] Remde, A., & Henrich, D. (1999). Picking-up deformable linear objects with industrial robots.

[12] Wakamatsu, H., Tsumaya, A., Arai, E., & Hirai, S. (2006, May). Manipulation planning for unraveling linear objects. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on (pp. 2485-2490). IEEE.

[13] Moll, M., & Kavraki, L. E. (2006). Path planning for deformable linear objects. Robotics, IEEE Transactions on, 22(4), 625-636.

[14] Berenson, D. (2013, November). Manipulation of deformable objects without modeling and simulating deformation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4525-4532). IEEE.

[15] Yue, S., & Henrich, D. (2001). Manipulating deformable linear objects: model-based adjustment-motion for vibration reduction. Technische Universität Kaiserslautern, Fachbereich Informatik.

[16] Sisbot, E. A., Ros, R., & Alami, R. (2011, July). Situation assessment for human-robot interactive object manipulation. In *2011 RO-MAN* (pp. 15-20). IEEE.

[17] Masone, C., Franchi, A., Bülthoff, H. H., & Giordano, P. R. (2012, October). Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In 2012 IEEE/RSJ international conference on intelligent robots and systems (pp. 2641-2648). IEEE.

[18] Dogmus, Z., Erdem, E., & Patoglu, V. (2013). REACT! An Interactive Tool for Hybrid Planning in Robotics. arXiv preprint arXiv:1307.7494.

[19] Luo, Z. (2014, November). Interactive Model Fitting for Human Robot Collaboration. In *e-Business Engineering (ICEBE), 2014 IEEE 11th International Conference on* (pp. 151-156). IEEE.

[20] Galindo, C., Fernández-Madrigal, J. A., & González, J. (2008). Multihierarchical interactive task planning: application to mobile robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(3), 785-798.

[21] León, B., Ulbrich, S., Diankov, R., Puche, G., Przybylski, M., Morales, A., Asfour, T., Moisio, S., Bohg, J., Kuffner, J. and Dillmann, R., 2010. Opengrasp: a toolkit for robot grasping simulation. In Simulation, Modeling, and Programming for Autonomous Robots (pp. 109-120). Springer Berlin Heidelberg.

[22] Gschwandtner, M., Kwitt, R., Uhl, A., & Pree, W. (2011). BlenSor: blender sensor simulation toolbox. In Advances in Visual Computing (pp. 199-208). Springer Berlin Heidelberg.

[23] Ferraguti, F., Golinelli, N., Secchi, C., Preda, N., & Bonfe, M. (2013, September). A component-based software architecture for control and simulation of robotic manipulators. In Emerging Technologies & Factory Automation (ETFA), 18th Conference on (pp. 1-5). IEEE.

[24] Echeverria, G., Lassabe, N., Degroote, A., & Lemaignan, S. (2011, May). Modular open robots simulation engine: Morse. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 46-51). IEEE.