Planning Method of Untying Cable with Reinforcement Learning Technology

Fan Zheming[1] Hayashi Toyohiro[2] and Ohashi Takeshi[2]

[1] Fan Zheming, Kyushu Institute of Technology, fanzhemingjisuanji@yahoo.co.jp
[2] Hayashi Toyohiro, Kyushu Institute of Technology, toyohiro@isc.kyutech.ac.jp
[2] Ohashi Takeshi, Kyushu Institute of Technology, ohashi@isc.kyutech.ac.jp

Abstract. Since entered information age, we use many devices in our daily live which need cables to connect and charge them. When we want to clean the room or maintain the equipment, we have to spend a lot of time to handle the cables, which is very tedious and inefficient. Therefore, we think that if we can handle cables with robot, it will make our life more convenient. Therefore, in this paper, we propose a new method of planning how to untie cable with reinforcement learning technology. The main idea of our method is to use a 3D simulated model of the cable to determine the action process of untying the cable and use the action sequence we get from the simulation to control a physical robot arm to untie the cable. We made a prototype system with 3 parts to verify the feasibility of our method. First, we recognize the cable in the image and convert that image into a 3D simulated cable model. Second, we use reinforcement learning technology to determine the action sequence in the 3D simulated cable model. Finally, we apply this action sequence to a physical robotic arm that will untie the cable in the real world. We believe that our approach is very useful for using robots to manipulate cord-like objects.

Keywords: Cable, Untying operation, Reinforcement Learning, Simulated Model.

1 Background

In our daily life we need use many smart devices like mobile phone or tablet and so on. For using those devices we need cables to connect and charge them, so we can find many cables around us. And when you want to do something with cables you always find them in a mess. But, arrange cables is a boring task and will take a lot of time. Especially in university or company where have a large number of cables, arrange them is a huge workload. So, we want to find a new method can manipulate cables with robot, and this will make our life more efficiency. In this paper, we proposal a method to determine the actions of untying the cross parts of cables and achieve it with a robot arm. In our method we define cables into two situations, easy(without knot and cross) and complicated(with one or more crosses), and our purpose is transform complicated state cables to easy state. In our prototype system we use deep learning technology to recognize cables and reconstruct 3D cables

model, use reinforcement learning technology to analyse cables model and get a action sequence for robot arm, use Niryo One(one kind of robot arm) to manipulate cables in real world. We will introduce all the details of our method in after chapters.

In RoboCup competitions, there are many sessions where robots are used to manipulate objects, and I think that in the future, manipulating soft objects will be part of the assessment. For example, in the @Home competition, the theme of @Home is to develop service and assistive robot technology with high relevance for future personal domestic applications. Cord-like objects are very common in life, and the manipulation of cord-like objects will be important to assist our live. We think our research is helpful in advance the technology of intelligent robots.

2 Previous Research

There are some researches about manipulate cord-like objects has been published. We choose two of them as representation to introduce those method.

In paper [1], they proposal a method to manipulate easy state cables(without cross or knot). They put cables on a white plane and use canny filter to get edge information of cables. Use those information to confirm the 2D position of cables and use binocular camera to get depth information. At last they let robot arm to hold cable's end point and move along cable. There are some problems. First, they only can manipulate easy state cables but that is not enough to solve the problem we meet in daily life. Second, the result of canny filter have many noises need to be deal with. Third, they need put cables in a white plane to make sure canny filter can get edge information of cables but in daily life there is no such ideal situation for recognizing.

In paper [2], they use point cloud data to get rope's 3D position and use Knot Theory to analyse rope state and decide how to manipulate rope. At result they can untie complicate state ropes. But, there are problems too. Point cloud data is huge and will take more time to deal with then image. And point cloud data also have noises need a pre-processing before recognize. Using knot theory for action decision making, which requires human to summarize the characteristics of knots and develop algorithms to analyze the situation, this part requires human to do and cannot be generalized in all cases.

In our method, we use only one RGB image to get all the information we need, which allows us to process a small amount of data. We use deep learning technology to recognize cables which allows our method can work in different environment. We use simulated model to planning the actions that let us can do more tests than the real world in the same amount of time, this make our method more efficiency. The biggest difference between our method and previous methods is that we use neural network to process the data, making our method more accurate. And our planning approach uses reinforcement learning technology so that the decisions are made by computers rather than humans, which makes our approach more intelligent and can reduces human workload.

3 Proposed Approach

3.1 System Structure

In this paper, we approach a new planning method of untying cables and made a prototype system to verify it's feasibility. You can see the flowchart of our system in Fig.1. We will introduce the details of each part in subsequent chapters.

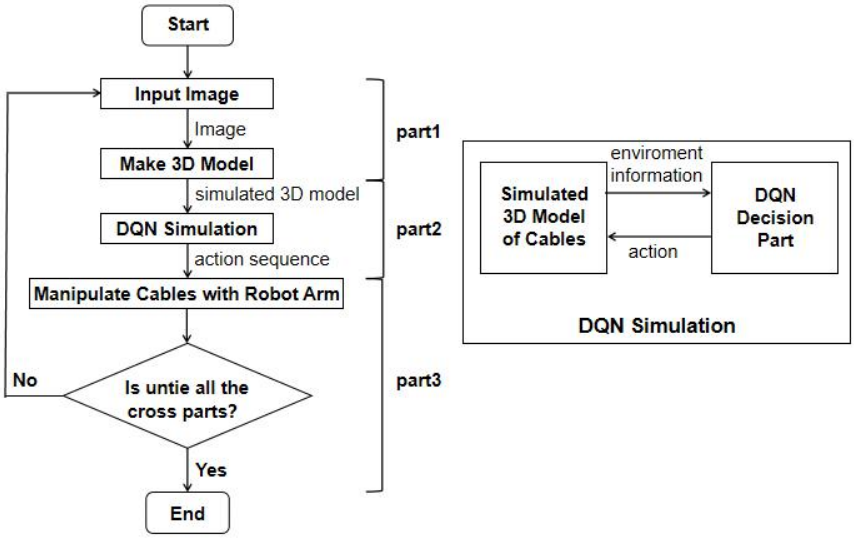


Fig. 1. A flowchart of our system. We show the structure of our system and the detail of DQN simulation part. DQN requires two parts: simulation and decision. The decision part decides the action based on the information of the simulation part, then the simulation part executes this action. This two parts will keep exchanging information until the cable is untied.

3.2 Make Simulated 3D Model

In our system, first thing we need to do is get cables 3D position information because we need to analyse those information to decide how to manipulate cables. In paper [3], they proposal a method use a 3D convolutional network to transform a object from 2D image to 3D voxels model. We use this method to get information from image. But this is not enough, in the decision part we need use DQN[6], DQN need a virtual environment which can simulate real world environment. Paper[3] method only can make a static 3D model so we need to make this model movable. To simulate real world, not only need cable position but also physical attributes (gravity, friction force, air resistance and so on) are needed, so we choose Unity3D to achieve this simulation. We also need to simulate cord-like objects characteristic(Recovering force of bending and stretching), paper[4] introduce a method named PBD(Position Based Dynamics) which can simulate soft object in virtual world. PBD method use a large number of particles to represent the shape of objects and set a series of limits among particles to simulate the deformation of soft object when it be moved. Now

there are some mature systems have been developed based on PBD method, we choose one of them named OBI Rope[5] to make 3D model. In Fig.2 we show how we made 3D model and how the 3D simulated model look like.

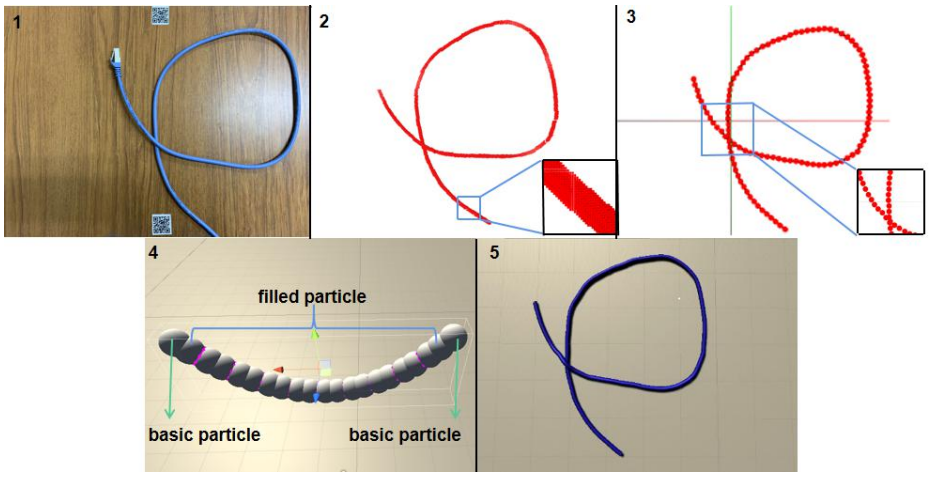


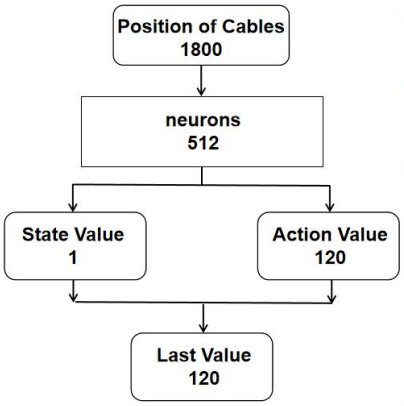
Fig. 2. Illustrative diagram about how to use OBI to make cable model. Picture_1 is input image. Picture_2 is a 3D voxel model of the cable made from picture_1 by the method of paper[3]. Picture_3 shows all the basic particles, the basic particles position are obtained by processing on the voxel model of picture_2. Picture_4 shows an example of how to make a cable model with OBI. Only the leftmost and rightmost particles are basic particle, and the rest of the particles are filled automatically by the OBI system. We only need to input basic particles position and then the OBI system can make the cable model. Picture_5 shows the cable model after rendering, which will be used in DQN planning part.

3.3 DQN Action Decision

DQN Setting Details

In our system, we use DQN to make action decision. DQN(Deep Q Network) is a kind of Reinforcement learning algorithm which is combine Q-learning method and neural network. Paper[6][7][8][9][10] introduce the prototype and improvement version of DQN, in this paper we use the improvement version. DQN, like Q-learning, define **States**, **Actions**, **Reward** and **Value**. In our system, **States** is the data which can represent the shape of cables. We set **Actions** as 4 options: front, back, left and right, it means we will hold one point of cable and move it in this direction. **Reward** is set like below: **1.**If after one move the number of cross part of cables is reduce then we will give a positive reward(+1). **2.**If after one move the number of cross part is increase then we will give a negative reward(-1). **3.** If we untie one cross part with too many steps(over 100 steps) or the holding part is out of the reach robot arm then we also give a negative reward(-1). **Value** is the evaluation value of an action when it is chosen in a certain state. If the **Value** is high, then this choice of action is beneficial to our success in untying the cross part. On the contrary, if it is low, it is detrimental to untie cross part. We save the **Values** in neural network and

update the **Values** by a formula(introduced in paper[6]), after training we can untying the cross parts by following the actions which have high **Value**. In Fig.3 we show the structure of neural network and the execution steps of DQN.



- 1. Initialize neural network and make a memory to save train data.
- 2. Initialize 3D model, get **cables position data**.
- 3. Input cables position data to network and get **values** and choose the **action** with **ε-greedy policy**.
- 4. Do the action in Simulated 3D model and get return information of environment, set **new cable position**.
- 5. Save **training data** and update network parameters.
- 6. Judge if **task over(untie one cross)**. If yes, go step 7, if not, go step 3 (this time input is new cable position data).
- 7. Judge if for this 3D model we already have over **80 percent** success rate. If yes, end the DQN, if no, go step 2.

Fig. 3. Structure of DQN neural network and the execution steps of DQN.

The length of the input and output of the network is 1800 and 120. In Fig4 we show the details of input and output. All the spheres in the **Basic Particle Image** are basic particles (maximum is 150, if the number of basic particles in the model is less than 150, the rest will be filled with 0) and their coordinates can represent the shape information of the whole cable. So we can use a **[150x3]** list represent the current 3D position information of the cable(**time t**). The network also needs to consider the continuity of the cable position changes at the time level so we input the cable position information at the moment **[t-3,t-2,t-1,t]** (**t is the current state**). So we set the length of the network input to **1800 ([150x3x4])**. The green spheres in the **Basic Particle Image** are alternates of the manipulate particle (maximum is 30, We will choose one of them for the robot arm grip), each alternate will have 4 different actions (up, down, left, right), so the output of the network is set as **120([30x4])**. Another option regarding the selection of the manipulate point is to leave it up to the human. For example, I personally prefer to set the manipulate point near the cross part, but this requires human analysis of the current cable state. In the case of reinforcement learning algorithms, it is preferred to leave all the judgments to the computer which can make the decision-making method more intelligent. The method we used is let computer to make all the decision and we think this is the best for our research. Each basic particle is given a number when initialize the model, and every fourth basic particle is set as an alternate of the manipulate point (**numbers 0, 5, 10, ... , 145**). These points represent the parts of the cable that we can move, and after tests the system can decide the manipulate point and it's direction of movement to untie the cross part of cables according to the current state.

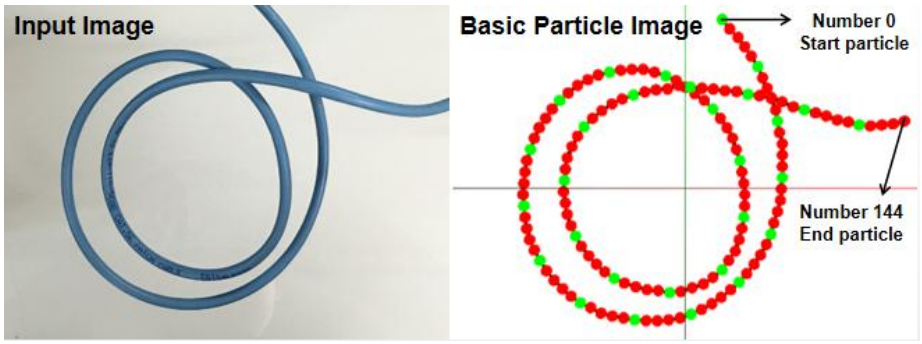


Fig. 4. Detail of input and output of neural network.

Training Data

Due to the time reason, we only made 26 models for training. We set the categories of training data according to the shape and complexity of the cable, the complexity being proportional to the number of movements required to untie the cross part. So we use the number of cross parts in the cable model to distinguish between categories. We divide them into 4 categories. Among them, the number of training data with 1,2,3,4 cross parts is 12,4,5,5 pieces, respectively. Fig.5 shows some examples which has the number of cross parts wrote in the top-left corner. Our system can untie all training models and use the previous untying experience in subsequent training. This will make the time needed for training shorter and shorter.

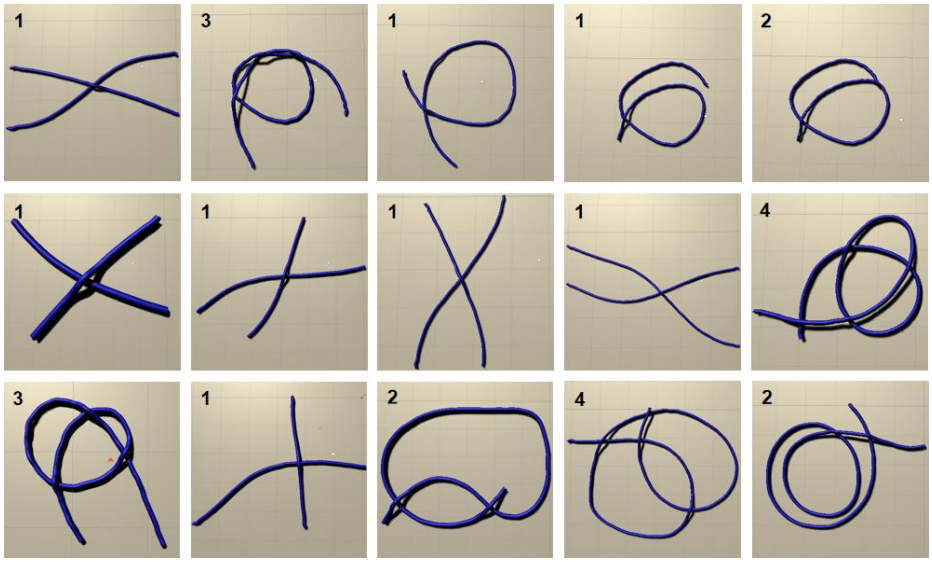


Fig. 5. Some example of training data.

3.4 Robot Arm Manipulation

About Niryo One

In this part we want to control robot arm according to action sequence. The robot arm we used is Niryo One[11] which is a collaborative and open source 6-axis robot. In Fig.6 we show the picture of Niryo One(left), user can control it by a Python API. For control robot we need transform coordinate first, in Fig.6 we show the front space of Niryo One(right picture), there are two QR-codes are fixed in a given coordinate of robot coordinate system, and we can use a QR-code detector to get those QR-codes coordinate in image coordinate system. Use those coordinate information we can transform coordinate from image coordinate system to robot coordinate system.

Fig. 6. The picture of Niryo One and the front place of Niryo One.

About Two Hands Operation

When we tested the robot arm, we found that when we move a part of the cable a force occurs that makes the whole of the cable move. And this force is only related to the material, it is difficult to accurately simulate this force in the simulated 3D model. Therefore, for untying cable we want the cable to move only locally. Moving locally means that when the cable is moved, the target part is moving and the rest of the cable remains unmoved. So, we perform a two-handed operation, but we only have one robot arm. The solution is that we assume that there are two robot arms, one of which moves according to the action sequence and the other just grabs a point of the cable and does not move. We replace the second robot arm with a tape to keep the cable available for local movement. Fixing different places will have different effects, especially when the target model has multiple cross parts. We choose the middle of the entire cable as fixation point which can divide the whole cable into two parts. This position ensures that when the cable is moved, whether move the first or second part, the rest of the cable is always immobile. In other words, half of the cable is always immobile, which maximizes ensure the cables are moving locally.

3.5 Result

We prepared a execution as example with 3 cross parts and our aim is to untie all the cross parts. After untying the first cross part, we found a problem that the virtual

simulation could not perfectly simulate the real world. Fig.7 shows the details of this problem. We prepared 4 pictures, picture_1 is the input picture, picture_3 is the simulated 3D model made from picture_1, picture_4 is the result after simulation in the virtual world, and picture_2 is the result of the operation in the real world with the exact same action sequence which is used in the simulation. We find that the shape of picture_2 and picture_4 are not similar, so we cannot use the model of picture_4 to continue the simulation. Our solution is re-creating a simulated 3D model after untie one cross part, which can minimum the error between the simulated world and the real world.

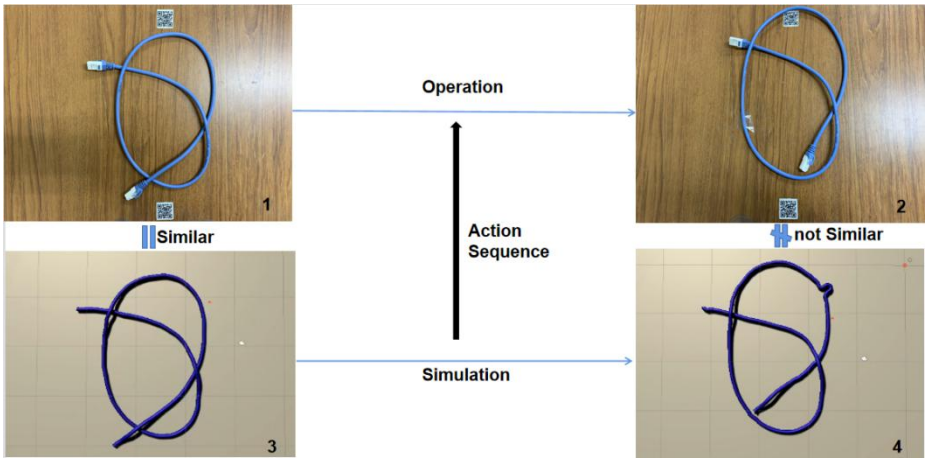


Fig. 7. The details about the problem we meet in simulation.

In Fig.8, **Form1** is the input picture of the system the cable in this picture has three cross parts, **Form2** is the form of the cable when we untie the first cross part, **Form3** is the form of the cable when we untie the second cross part and **Form4** is the form of the cable when we untie all the cross parts, this is also the output of our system. Fig.9, 10, and 11 show the details of the operation when each cross part is untied. The first row in Fig.9, 10, and 11 shows two images, the left one is the form before untie cross and we marked target cross part with a red box, the right one is the form after untie cross, and below the two pictures are a series of small pictures which can dynamically show the process of untying.

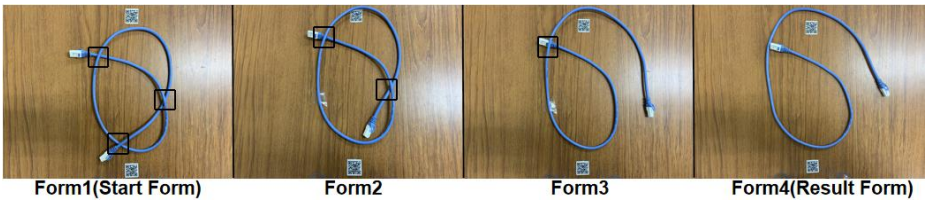


Fig. 8. The details about deformation of cable.

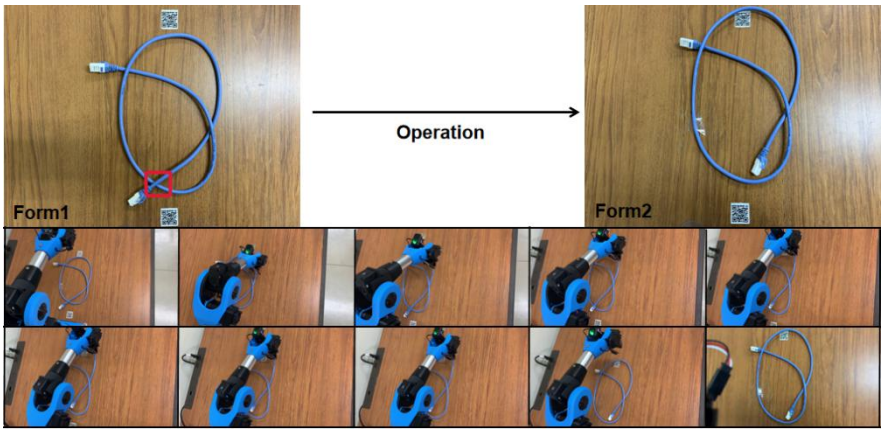


Fig. 9. The details about operation from form1 to form2.

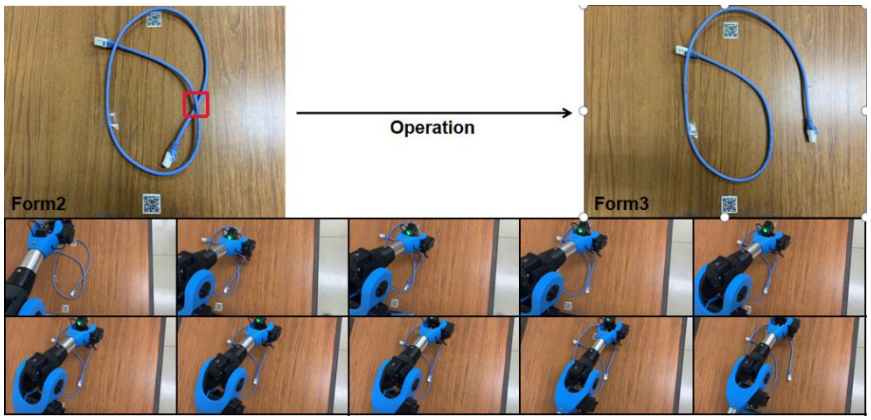


Fig. 10. The details about operation from form2 to form3.

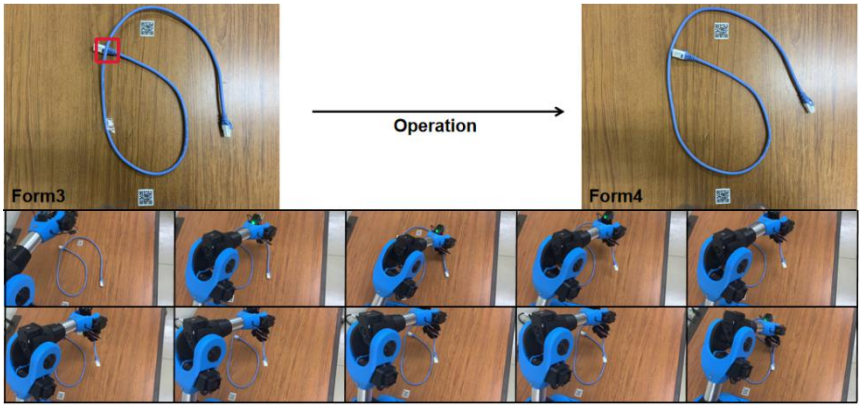


Fig. 11. The details about operation from form3 to form4.

4 Conclusion

We propose a new planning method to use robot arm to manipulate cables to untie them in a cross-twisted state. Our proposal aims to make people's lives more convenient by handing over the manipulation of cables and other cord-like objects that we see in daily life to a robot arm. In this paper, we build a prototype system to verify the feasibility of the proposed method. Although the system is only a prototype, it has been tested to successfully untie cables with multiple cross parts, which proves the feasibility of our theory. After this we will challenge the cable to more complex states, such as the knotted state. Finding and developing methods and systems that use two robot arms to collaborate with each other to untie knots will be our main topic in the future.

Regarding the **running time of the system**, in Fig.1, **the average time of part1 is around 45 seconds, the average time of part3 is around 20 seconds**, and the problem is the time of part2, which we need to introduce in detail. The time to planning the action sequence is directly related to the complexity of the cable and the number of steps needed to move to untie cross part. In our system **the time to solve one cross is about 1 minute average**, in the example there are 3 crosses so **the total time of part2 is about 3 minutes.** Regarding the **success rate** of the system, the success rate of part2 is very high, for complex cases, it only takes a long time to explore and there is no case where the action sequence cannot be obtained. Part3 success rate is also very high, occasionally it happens that the robot arm is not sensitive to the subtle operation response, but as long as the action sequence can be obtained correctly, this part is also don't need worry. Therefore, the success rate of our system is more dependent on the success rate of the part1, which is described in paper [3] with a success rate of over 70%. **In summary, the success rate of our system can also be maintained at more than 70%.**

We also prepare five graphs based on the changes of the data during the training process. Fig.12 plots the curve of the change in **loss values**, we can see the loss value decrease gradually through the training. Fig.13 plots the **how many movements** are taken from the beginning to the end of each **episode**(An episode means one training. For a model, it requires many times training to find the solution to untie cross). The number of movements in each episode is slowly becoming smaller compared to the beginning, which indicates that the system is optimizing the action sequence. Fig.14 plots the **manipulate point** in each **step**(One step means moving the cable once, and there will be many steps in an episode.), and we can see that manipulate points always converges to the a certain basic particle, it means the system have the function of selecting the suitable manipulate point according to current state (Take form2 to form3 as an example, the system tends to choose the 85th particle as the manipulate point at the beginning, and when steps exceed 200, the system tends to choose the 20th particle as the manipulate point). Fig15 plots the **selection of the action** in each step, in the late training obviously converges to a certain action (Take form2 to form3 as an example too, we can see that when the steps exceed 200, the system tends to choose action0 as the execution action. Combining with Fig14, we can see that in the current state(form2), choose the 20th particle as the manipulate point and action0 as

executed action, which is good for untying the cross part.). Fig16 plots **the results of each episode**, it can be seen through training the successful results has increased significantly.

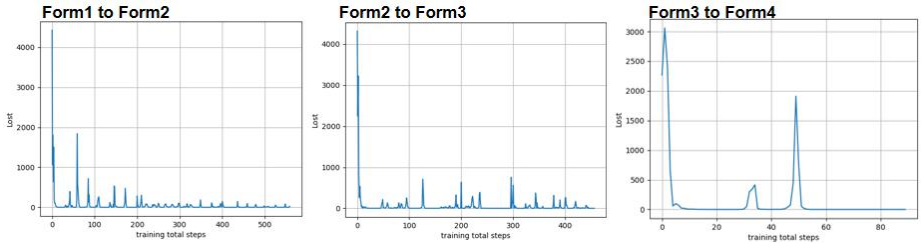


Fig. 12. The graph of change in loss value. Horizontal coordinate is the number of steps and vertical coordinate is the loss value

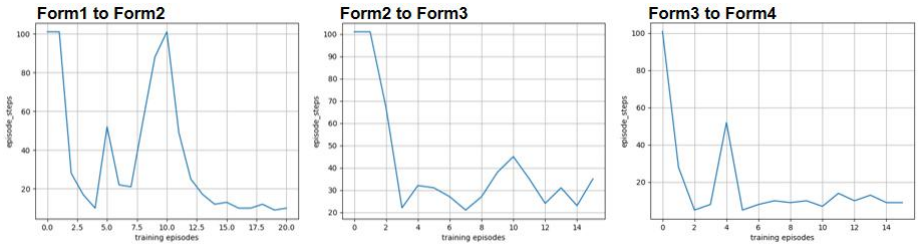


Fig. 13. The graph of change in episode_steps. Horizontal coordinate is the number of episodes and vertical coordinate is episode_steps.

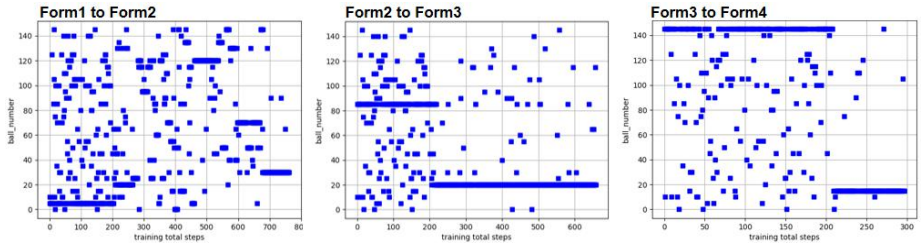


Fig. 14. The graph of change in manipulate point. Horizontal coordinate is steps and vertical coordinate is the number of particle which is selected as manipulate point.

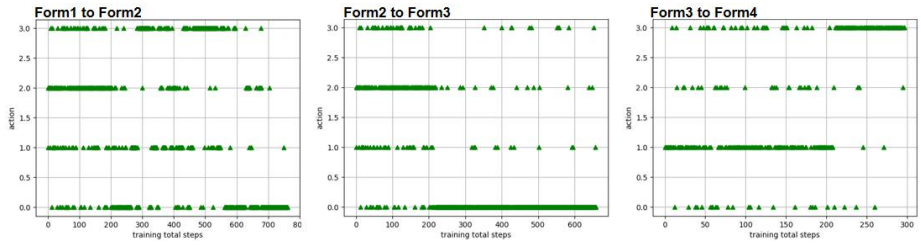


Fig. 15. The graph of change in action. Horizontal coordinate is steps and vertical coordinate is the number of action which is selected.

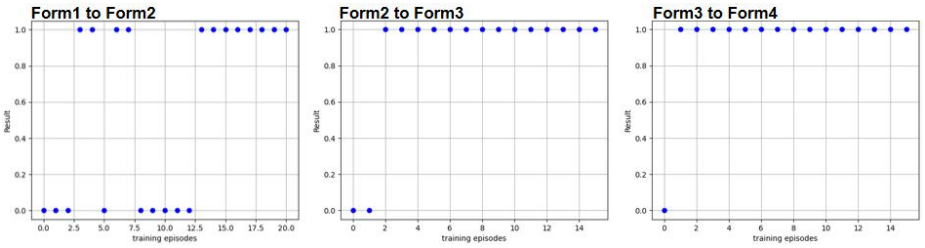


Fig. 16. The graph of change in result. Horizontal coordinate is episodes and vertical coordinate is the result(0 means fail and 1 means success) .

References

1. Kenji SATO, Kei OKADA, Masayuki INABA.: Humanoid's daily life support actions using image processing that recognizes string-like objects (in Japanese). Robomec, 2A1-D26:1-4, January (2006).
2. Takayuki Matsuno, Tomoya Shirakawa, Tomotoshi Watanabe, Mamoru Minami.: String Untying Planning Based on Kont Theory and Algorithms to Generate the Motion of a Manipulator (in Japanese). Journal of the Robotics Society of Japan 2018, vol. 36, No.6, pp. 429–440. (2018).
3. FAN ZHEMING, OHASHI TAKESHI, HAYASHI TOYOHIRO.: Make 3D cable model from RGB image. IIAI AAI congress . (2021).
4. Matthias Muller, Bruno Heidelberger, Marcus Hennix, John Ratcliff.: Position Based Dynamics. 3rdWorkshop in VRIPHYS (2006).
5. OBI home page: http://obi.virtualmethodstudio.com/.
6. Volodymyr Minh, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller.: Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop (2013).
7. Volodymyr Minh, Koray Kavukcuoglu, David Silver.: Human-level control through deep reinforcement learning. Nature volume 518, pages 529–533 (2015).
8. Hade van Hasselt, Arthur Guez, David Silver.: Deep Reinforcement Learning with Double Q_learning. AAAI-16 (2016).
9. Ziyu wang, Tom Schaul, Matteo Hessel, Hade van Hasselt.: Dueling Network Architectures for Deep Reinforcement Learning. PMLR 48:1995-2003 (2016).
10. Tom Schaul, John Quan, Ioannis Antonoglou.: Prioritized Experience Replay. ICLR (2016).
11. Niryo One homepage: https://niryo.com/product/niryo-one/.