EPiC Series in Computing

Volume 81, 2022, Pages 12–23

Proceedings of 11th International Congress on Advanced Applied Informatics

Make Simulatable 3D Cable Model from Single RGB Image

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

[1] Kyushu Institute of Technology, fanzhemingjisuanji@yahoo.co.jp

[2] Kyushu Institute of Technology, toyohiro@isc.kyutech.ac.jp

[2] Kyushu Institute of Technology, ohashi@isc.kyutech.ac.jp

Abstract

With the development of the times, use the electrical devices is essential for our daily live. For using these devices, we need cables to charge or connect them. So, for people, the cables can be found almost everywhere. The cables also bring new problems, like the cables always appear in a messy form with crosses and knots. We have to tidy up the cables before using them and this is a time-consuming and tedious task. And in colleges and companies where have a large number of cables, when we clean the room or laboratory, we can always find these cables are annoying. For this reason we think that it would make our daily live more convenient to use robots to manipulate and untie the cables. Therefore, for manipulating and untying the cables, this paper proposes a method which can convert the 2D cable data from image into 3D cable data in Unity3D, where the 3D cable model is movable and can simulate the real cable, we call this 3D cable model the "Simulatable Cable Model". In our approach, we use 2D and 3D neural networks to recover the 3D position information of the cable from the input image, then adjust this 3D position information to increase it's accuracy, and finally create the "simulatable cable model" in Unity3D. The "Simulatable Cable Model" provides a new way to manipulate the cables, that is, to simulate the actions in the virtual environment and then apply it in the real world. Such a method can be used not only to support people daily life with robots, but also can be used to arrange cables in the workplace like factories and so on. We believe that our research is applicable and helpful to the recognition and manipulation of all cord-like objects, and will also be useful in the field of recognizing and manipulating soft objects.

Keywords: Cable, 3D Simulatable model, Deep learning, Computer Graphics

1 Introduction

In the information age, people's lives are filled with all kinds of electrical appliances. Cables are indispensable for using these appliances. But, the bendable nature of cables make them always appear in a messy form, before using the cables we have to untie them, which is troublesome and inefficient. So we have the idea of using a robotic arm to perform the untying operation. The core idea is to create a 3D cable model that can simulate a realistic cable (We call this model the "Simulatable Cable Model".), use reinforcement learning techniques to try out the untying operation on this model to get the correct sequence of actions to untie the cable, and finally apply this sequence of actions to a physical robotic arm to untie the realistic cable. Therefore, in this paper we will introduce the first part of this method that is how to make the "Simulatable Cable Model". Our approach requires only one RGB image as input then we can create a cable model in Unity3D. We use a 2D neural network to recognize the position of the cables in the input image and use a 3D neural network to make a 3D voxel model of the cables. Then, we fix the position errors of the voxel model to get more accurate 3D position information, and finally we use this 3D information to create a "simulatable cable model" in Unity3D. We believe that make accurate 3D model of cables is important and useful for untying operation and recognizing cord-like objects. Finding the correct action sequence of untying operation will be our main topic in the next phase. We believe that our "simulatable cable model" provides a new way of thinking about manipulating cables, i.e., trying to operate cables in a virtual environment and then applying the successful actions in the real world. Although our goal is to untie the cables, but our results can be used in other ways as well. For example, use robots to do the daily life support (e.g. to help people with physical disabilities to charge their mobile phones), and arrange cables in hazardous areas (e.g. high voltage power lines at height).

2 Related Works

Before introduce our method there are some methods about the cord-like object recognition and modeling have been published, we will briefly introduce them.

Paper[1], [2] and [3] represent traditional recognition methods that use images and point cloud data to recognize cables by analyzing features such as edge lines, pixels, voxels etc. These methods have some drawbacks, one of which is that the features like edge line which are obtained by using the canny filter cannot be used directly and require some processing to remove the noise. Secondly, the success rate of recognition is generally not high. Thirdly, a lot of programming is required to process the image and point cloud data. In paper[4], YOLO (a method use deep learning technology to recognize the objects in image) is used for cable recognition, the result are good, but only is two dimensional data, and the lack of depth information makes subsequent untying operations difficult. In paper [5], they used the mass-spring method to complete the modeling of a simple cord-like object. However, their results are too simple and do not have the crossed and knotted state of the cord-like object. Secondly, their result is to recognize the cord-like object with a fixed end point in the vertical plane, therefore this method has some prerequisites that are not fully satisfied in daily life. In paper[6], 2D and 3D neural networks are used to create the voxel model of the cable, but this still has it's limitations. Firstly, the amount of training data they prepared is too small. Secondly, the final result of this method has some position errors. However, the method in paper[6] is able to make the 3D model that have the general shape of the cable in input image.

Although the existing methods have some drawbacks, it is undeniable that they all in some extent can recognize the cables from the pictures and confirm the cable's location. The 3D model mentioned in paper[6] was interesting to us, so we proposed an improved method based on method introduced in paper[6]. This paper is only have a brief introduction about paper[6], so view paper[6] is very helpful

for the reader to understand our method. The core idea of paper[6] is to use a 2D neural network to recognize the cables in the input image and cut the input image into a series of small images based on the position of the cable in the picture, then use a 3D neural network to convert each small image to a corresponding small 3D model, and finally assemble all the small 3D models together to get a complete 3D model. Figure1 shows the flowchart and the variation of the data of the method in the paper[6]. We have made some improvements to this method. First, we slightly change the structure of the 2D network to make it faster. Secondly, we increased the training data of the 3D network to improve the accuracy of the 3D network. Third, we adjusted the position of voxels in the result of the method in paper[6], and fixed the position errors to make it more correct. Finally we built a new model on Unity3D (Simulatable Cable Model), this new 3D model can simulate real-world, which allows our results can simulate realistic cable in some extent.

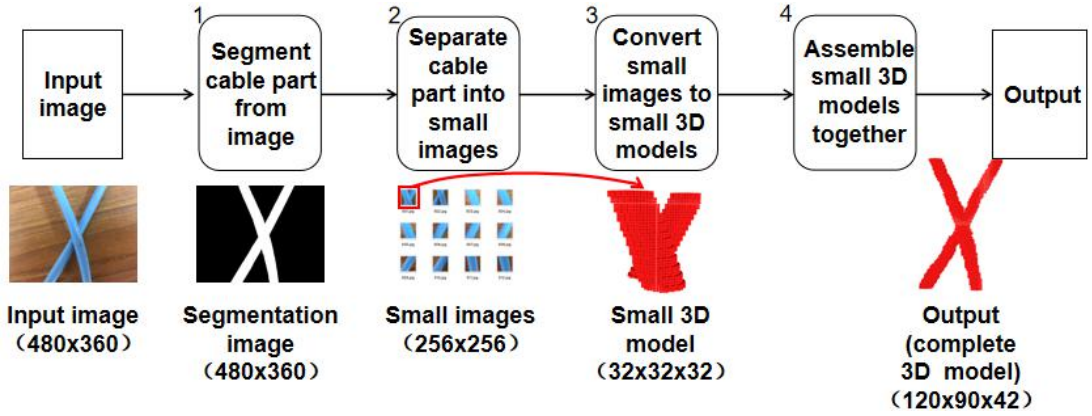

Figure 1: The flowchart and the variation of the data of the method in the paper[6]. The 2D and 3D neural networks are used in the parts marked with the numbers 1 and 3, respectively.

3 The Improvement Point

This paper introduce an improved method based on the method in paper[6]. In this paper we will focus on the problems of the method in Paper[6], the reasons for having these problems and the improvements we give. We carry over and slightly change the part1, part2 and part3 in the Figure 1. Figure 2 shows the flowchart and the variation of the data of our method. Combine Figure 2 with Figure 1, we can see these improvements more intuitive.

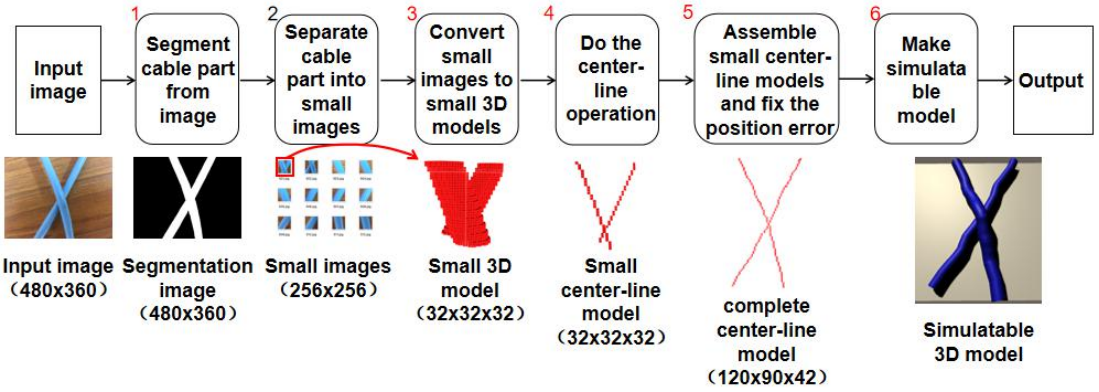

Figure 2: The flowchart and the variation of the data of the improvement method we proposed in this paper. The parts marked with red numbers are the parts that been improved.

3.1 Reduce the Classes of the result of 2D Neural Network

In the original method they divide the input image into 3 classes, the background (the position where have no cable), the normal cable part (cable position without crosses and tangles) and the complex cable part (cable position with crosses and tangles). The reason for the division of cable part is that they believe that the complex cable part makes more sense than the normal cable part when actually manipulate the cable, i.e. it is easier to untie the cable by manipulating the complex cable part than by manipulating the normal cable part. Therefore, they choose the complex cable part as the grasp point of cables and confirm it's position in the recognition process. In our method, reinforcement learning will be used to determine the strategy for untying the cable. So that giving the choice of the grasp point of the cable to the computer will make the method more intelligent and use less manpower. Then, the distinction between normal and complex cable parts in the recognition process would be meaningless. Therefore, in our method, the output of the 2D network is reduced to two classes: background and foreground (where the cables are located). Figure3 shows the training data, including the input image, the original label divided into three classes and the modified label divided into two classes. 500 sets of data were made, 400 for training and 100 for testing, all of which were 480x360 in size. After training, we plotted Figure4, which shows **the accuracy of the 2D neural network that can maintain at over 95 percent.** Compared to the original method, although there is no significant change in accuracy, but the running time is reduced by a small amount. Although the amount of reduction in running time for a single image is imperceptibly small for humans, it is helpful in somewhat when training the neural network in batches over a long period of time. On average, the processing time for each image is reduced about 0.0108 seconds. The batch size we used is 6 (because our GPU performance is relatively ordinary) and train this 2D network about 30,000 times. **So the total time saved can be calculated to be 1620 seconds (about 27 minutes).**

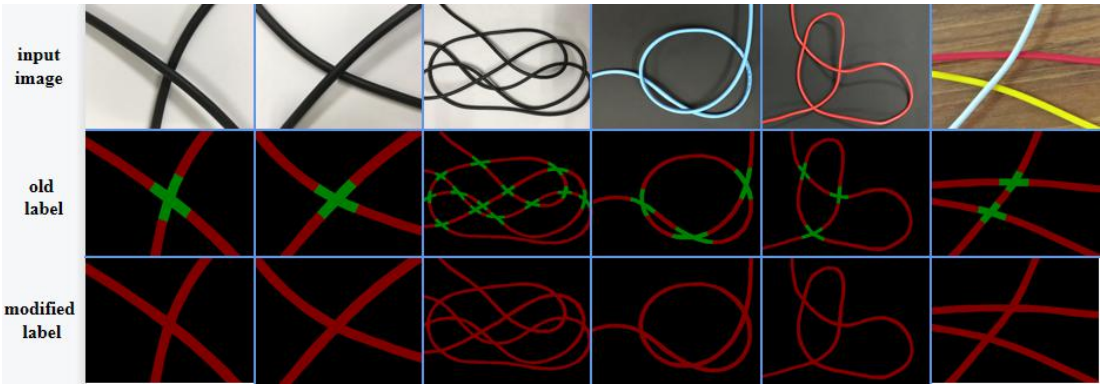

Figure 3: Modified training data of 2D neural network

Figure 4: Introduce graph of the accuracy of the test data of 2D network. The horizontal coordinate is steps and the vertical coordinate is the accuracy value

3.2 Add training data for 3D neural networks

As mentioned in paper[6], the 3D network has the disadvantage of lacking training data and training times. We increased the training data to 1000 sets, of which 800 sets were used for training 200 sets were used for testing. We didn't change the method of make the small 3D voxel model in the label (this making method is wrote in paper[6]). We set 96 as the batch size and trained this 3D network in a total of 11000 times. Figure5 shows the results of the training and on average **the 3D neural network was able to maintain the accuracy of over 90 percent, which is higher than the accuracy of 80 percent in paper[6].**

Figure 5: Introduce graph of the IoU, accuracy and loss of the test data of 3D network

3.3 Fix the position errors of the complete 3D model

In paper[6], we call their result the "complete 3D model", their result have a disadvantage that some positions are not correctly connected. This disadvantage is illustrated in Figure6, where picture1 is the input image, picture2 is the complete 3D model produced by the method in the paper[6] and picture3 is the same model with picture2 but viewed from different angles. In picture3, we use green and blue boxes to mark incorrect connection parts. The green box marks the incorrect connection in the horizontal plane and the blue boxes mark the incorrect connection in the depth direction. The reason for the incorrect connection in the horizontal plane is that there are some errors in the assemble process. The reason for the incorrect connection in the depth direction is that the height of the cable in the upper part of the cross part was incorrectly estimated when they made small 3D model of training data. This problem is illustrated in Figure7, which shows a set of training data, the upper half of the cable at the cross part in the label should be a curved shape with a high center and low ends due to gravity (shown by the green line in the figure7), whereas they set the voxels to a straight shape with the same height when made the label (shown by the blue line in the figure7). This resulted in the cross part will have a height difference that like a staircase step, rather than a smooth curve when all the small 3D models were assembled.

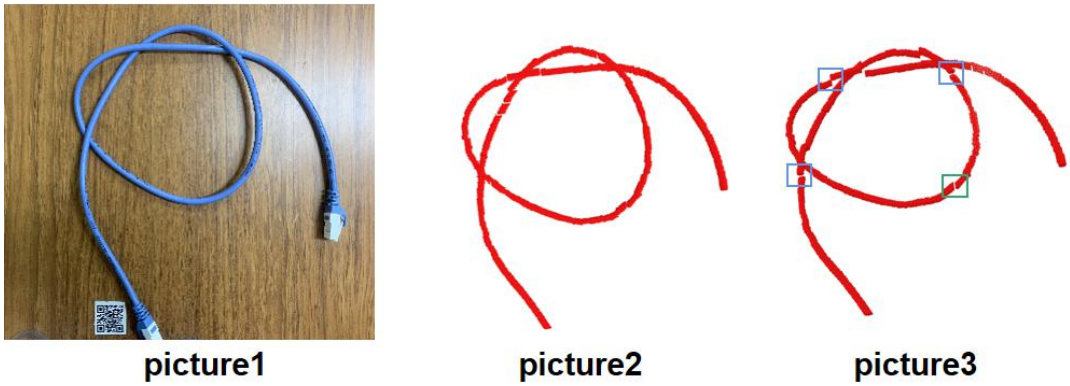

Figure 6: The incorrect connection in paper[6] method

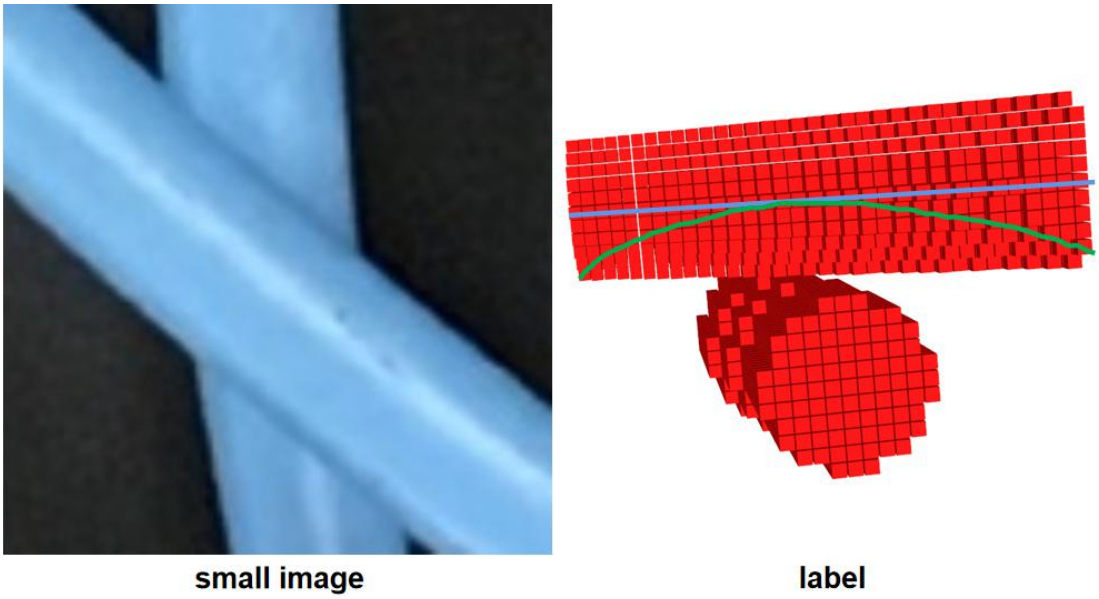

Figure 7: The reason of incorrect connection in depth direction.

Because of the amount of training data is so large that it would be too much work for modify all the label which has cross parts in it. So, the solution we used to solve this problem is to adjust the position of the incorrect connection parts in the complete 3D model. Since it is difficult to make adjustments directly on the complete 3D model, we propose the center-line model for adjusting operation. Center-line is a line that consists of only the most central voxel of the 3D model. Center-line model is a model that use the center-line and the width of the cable to represent the shape of the cable in input image (The width of the cable can be obtained from the complete 3D model, we use the number of voxels as a unit of measurement). First, we get the small center-line models by doing the center-line operation in the small 3D models which are got from 3D network and then assemble all the small center-line models together to obtain the complete center-line model. Figure8 shows how to do the center-line operation in small 3D model, all the small 3D model are made up of 32 circular or oval slices, by finding and preserving the center voxel of each slice, we can do the center-line operation in the small 3D model. Figure9 shows some small center-line models. Figure10 shows the complete 3D voxel model and the complete center-line model.

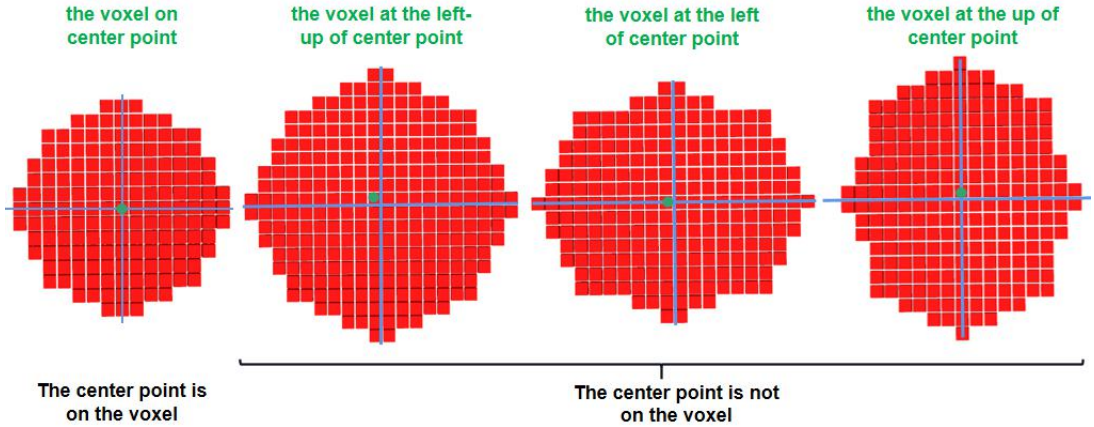

Figure 8: The figure about how to get the center-line of small 3D model. The small 3D model is made up of 32

circular or oval slices. The center line of the small 3D model can be found by finding the center point of each slice. As each slice is circular or oval it is only necessary to find the horizontal and vertical diameters of the slice (blue lines in the diagram) and the intersection of these two diameters is the center of the slice. When this intersection is not located on voxel, the voxel to the left or up or up-left of the intersection will be chosen as the center point of this slice (green dots in the diagram). The slices of small 3D models made by the 3D network may not appear perfectly circular or oval but may have a few extra or missing voxels, but as long as the slices have the general shape of the circular or oval, then we can do the center-line operation with the above method.

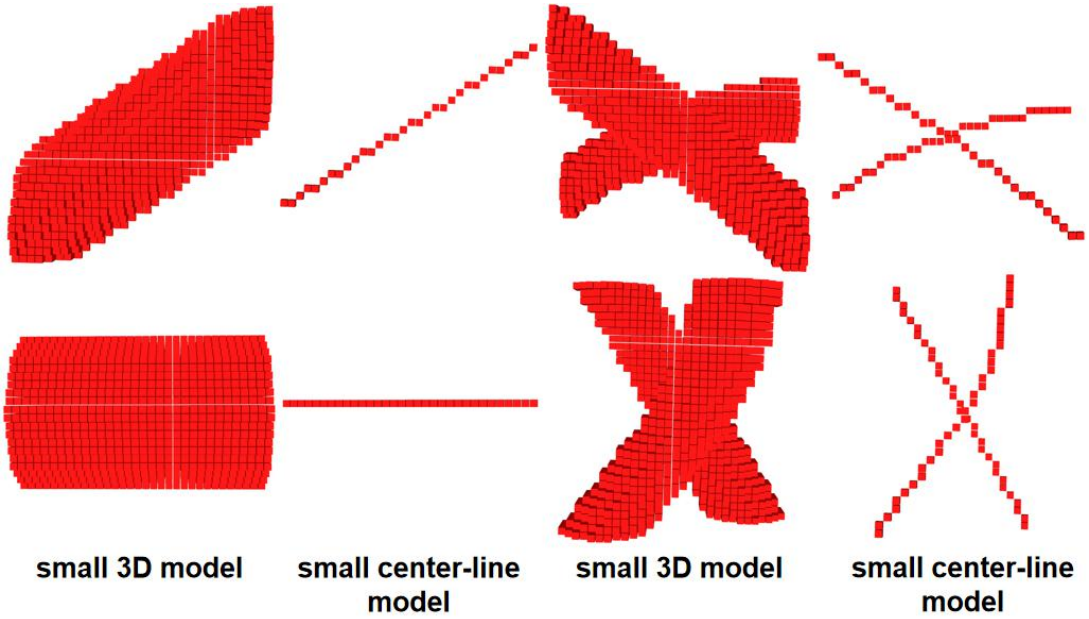

Figure 9: The example of the small 3D model and small center-line model

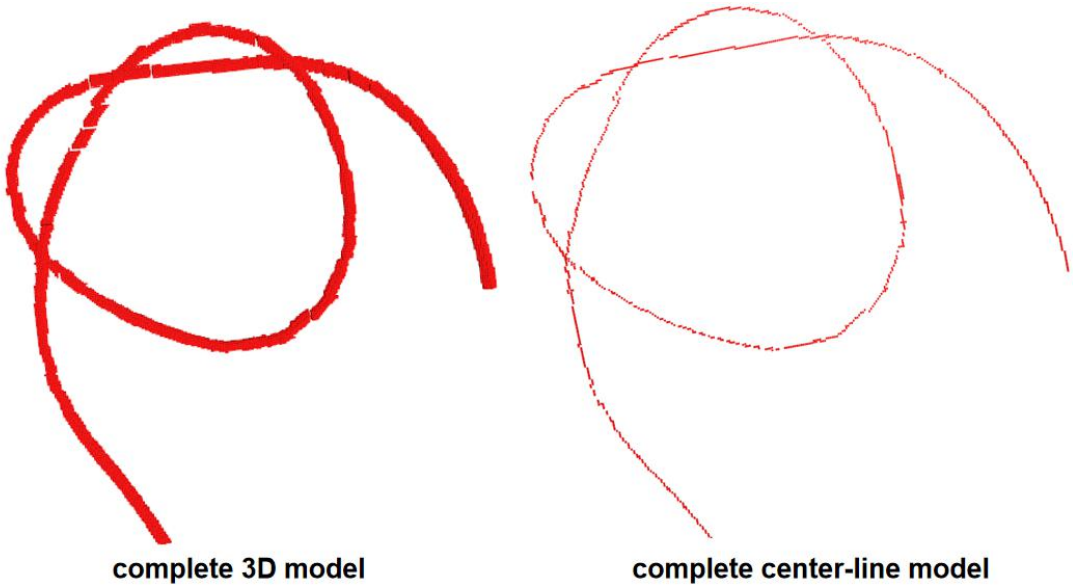

Figure 10: The example of the complete 3D model and the complete center-line model

Once the complete center-line model has been obtained, we can adjust the incorrect connection part. In the depth direction, we average the height make the voxels can be connected smoothly to fix the incorrect connection. In the horizontal plane, we fill voxels between the breakpoints (the voxels that should be connected but are not) to make them connected. Figure11 illustrates these two adjust methods. Figure12 shows the center-line model before the adjust and the center-line model after the adjust. In Figure12, picture1 is the unadjusted complete center-line model, picture2 is the same model with picture1 from a different perspective which have blue and green boxes to mark the incorrect connection parts in the depth direction and horizontal plane respectively, and picture3 is the adjusted complete center-line model and we can see that the incorrect connection parts have been fixed both in horizontal plane and depth direction.

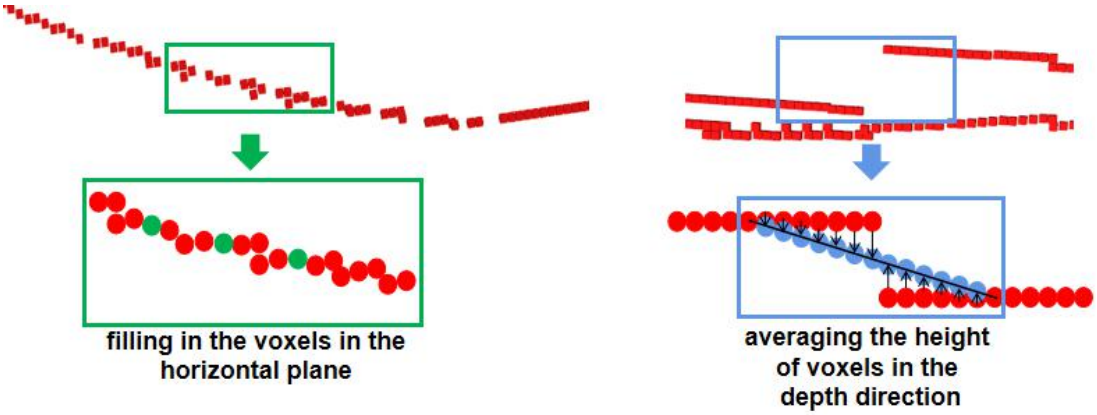

Figure 11: Illustrative diagram of the two adjust methods

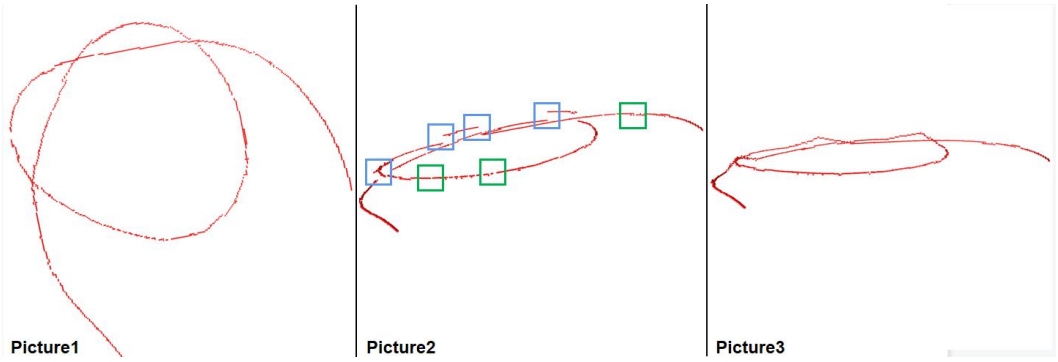

Figure 12: Illustrative diagram of the complete center-line model before and after adjusted

3.4 Make Simulatable 3D model in Unity3D

For simulating the real world, it is essential to simulate the physical laws of the real world, such as gravity, friction, etc. We use Unity3D[7] to implement the simulation of the laws of physics. Unity3D is a world-renowned physics engine that can perform the simulation part very well. Other than that, it is also necessary to simulate the properties of the cable (recovery forces when the cable is bent and stretched). Now, the PBD [8][9] method is almost the best approach to simulate the soft objects in virtual environment. The core idea of PBD method is to use a large number of vertexes to represent the shape of soft object, and set some constraints among those vertexes so that they can interact with each other, so that when the position of one or several vertexes changes, the remaining

vertexes will be displaced according to these constraints, thus simulating the deformation of the soft object during motion. Those constraints are set based on the laws of real-world physics and the characteristic of soft objects to ensure that the model can simulate the motion of soft objects correctly to the maximum extent possible. Paper[10] introduce a unified particles application that based on PBD method and use particles to simulate soft objects in computer. Figure13 shows a simple example, the example shows how to simulate a fabric with computer. The red spheres in the left image represent the particles, and the blue lines connected between the particles represent the constraints. The image on the right shows how the model looks like after rendering. Paper[10] only briefly mentioned that the bending and stretch constraints are needed for simulating rope, but that's not enough. Paper[11] proposed that on the basis of PBD method also need to define stiffness, contact, friction and torsion constraints in order to better simulate the rope. In paper [12], the "material frame" of "Cosserat Rods" is proposed to be defined as a new constraint in the form of quaternions thus integrating the Cosserat Rods and PBD methods to get better simulation result of the bending and rotation of elastic rods. The above are some examples of PBD-based methods to simulate ropes, where the key point is to set the constraints correctly.

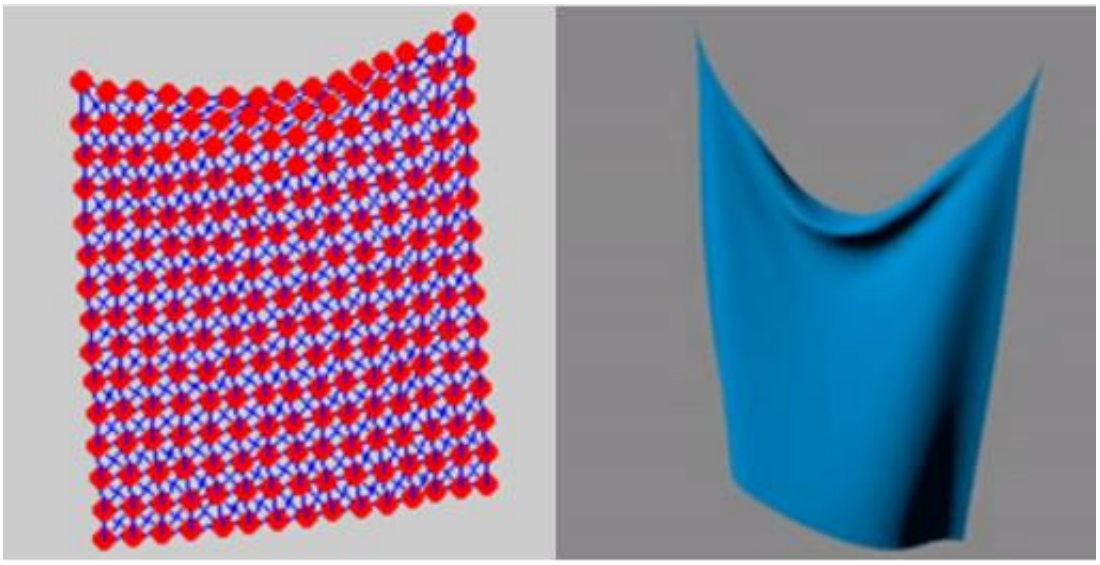

Figure 13: The example of PBD method

The PBD method has it's own drawbacks is that it will consumes a lot of computational power when the number of particles is too large, because computer has to calculate the current position of all particles for each frame according to the constrains when Unity3D refreshes. So for us, we want to reduce the number of particles as can as possible to make our system more easier and faster. So we propose the sphere model, where only one voxel is kept at every other "distance" based on the complete center-line model. This "distance" is set to the width of the cable, and then the voxels that are retained will be used as the center of the spheres and the width of the cable will be used as the diameter, to draw a sphere at the location of the voxel so that we can represent the shape of the cable with a series of spheres, which we call the sphere model. Figure14 shows an example of a sphere model. In Figure14, picture1 is the adjusted complete center-line model, picture2 and picture3 are the sphere models got by doing some process on the center-line model, which are views of the sphere model from different angles. The number of voxels required in the complete 3D model is in the tens of thousands, whereas in the sphere model, the number of spheres which will be used as particles can be limited to not over 150. This significantly reduces the number of particles.

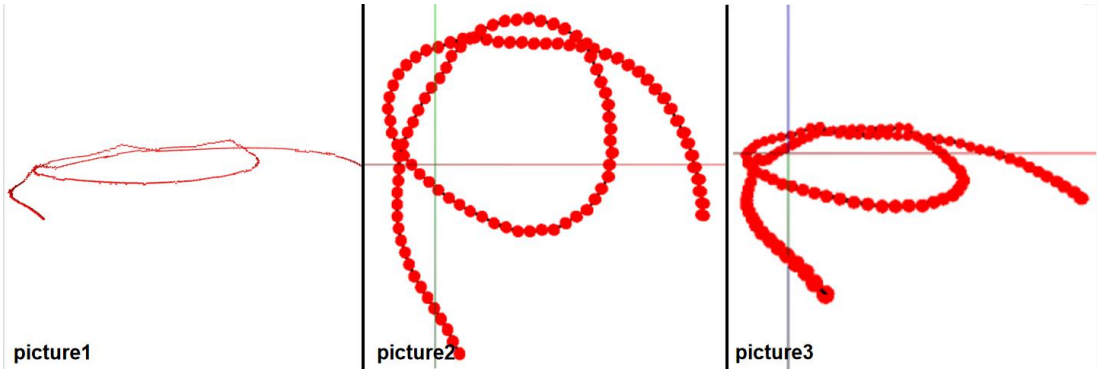

Figure 14: The example of sphere model

We already introduced that simulating soft objects in the virtual environment is a very difficult and time-consuming task. Fortunately, there are some open source systems based on the PBD method have been developed. Many of them are also available for the Unity3D platform. We choose a system called OBI Rope [13], which provides a programmable interface for users to call the classes and functions that defined by OBI system for making cord-like objects in user own programs. OBI Rope divides particles into basic particles and filled particles. Basic particles are the particles that are inputted by the user to determine the basic shape of the cable. Filled particles are the particles that are automatically filled in the target object by OBI system to ensure that there are no gaps between particles to ensure that the collisions between objects can be correctly detected. Figure15 shows the basic particles and the filled particles. In practice, we use spheres in the sphere model as basic particles and input their 3D coordinates (which can be obtained from the sphere model) and their diameter (which is equal to the width of the cable). Then, when the simulation starts, the OBI system can generates a cable model with the same shape as the cables in the input image.

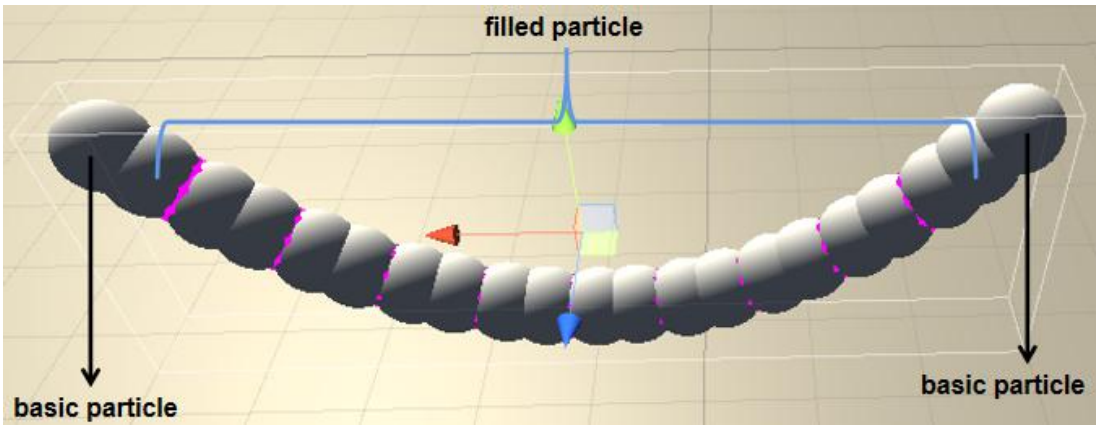

Figure 15: The example of particles model of the cable by OBI system.

4 Result

In Figure16 we prepared several sets of examples to present the results of our proposed method, each set consisting of the input image and a top view of the resulting 3D model i.e. “Simulatable cable

model". Looking at each set of examples it can be seen that for different backgrounds (wood, black, white, etc.), and different colors and shapes of cable in the input images, our proposed method is able to make it's 3D model successfully. Moreover, the connectivity of these 3D models is correct. Figure17 shows the deformation of the cable when the 3D model is in motion. We can see that the cable model behaves somewhat stiffly when it in the motion, but we think that the model will move more smoothly and naturally when the friction factor and the elasticity factor of the cable are carefully adjusted.

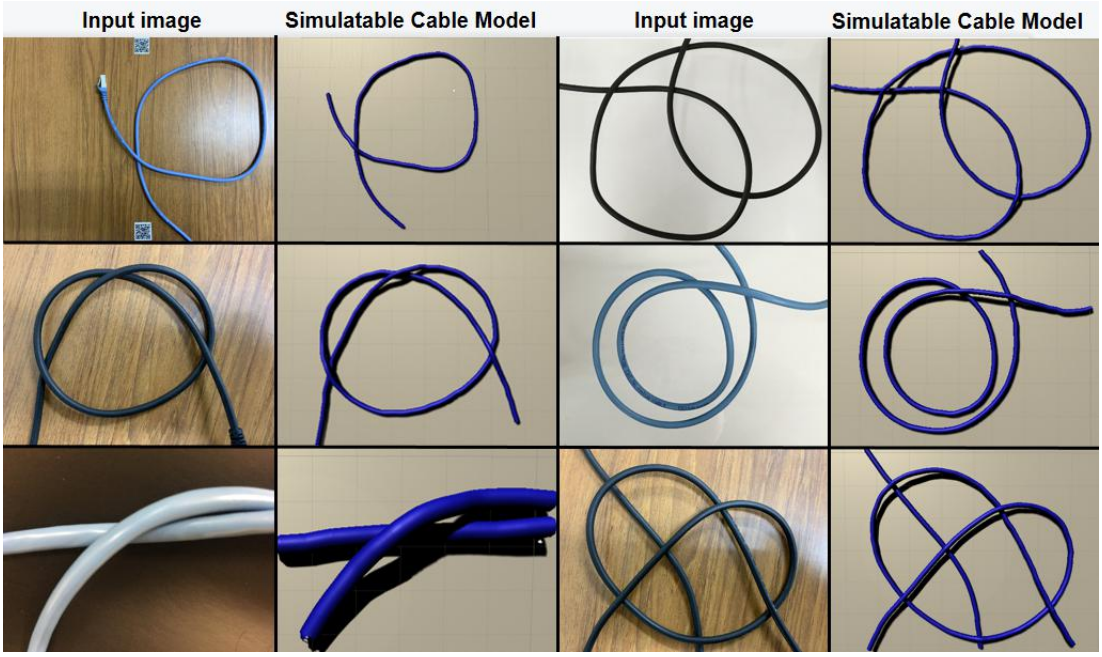

Figure 16: The results of our method

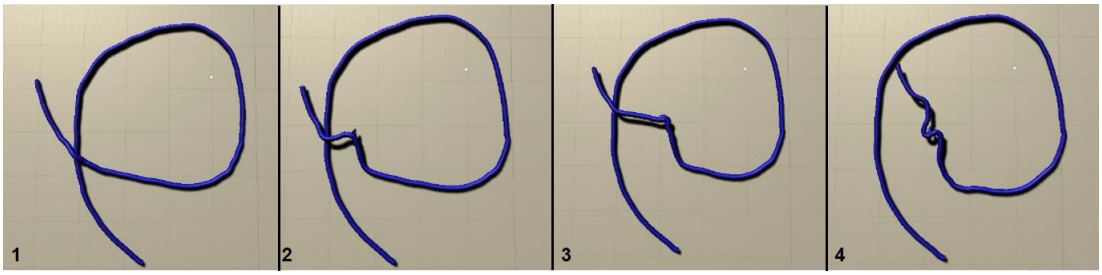

Figure 17: The deformation when the cables in "Simulatable Cable Model" are moving.

5 Conclusion

We propose an improved method based on the method in the paper[6]. Our method rises the success rate and efficiency of the model making and solves the problem of incorrect connections of voxels in the complete 3D model that existed in the original method. Our method produces a 3D cable model that is more accurate and more closely matches the cables in the input image. In contrast to the static model produced by the original method, our model has the function of simulation, this is a huge

improvement. So that when we make an untying attempt we can do this without affecting the real world, and we can do more attempts in the virtual world in the same time than in the real world, because the model can be returned to the original state with a single keystroke after the untying action, which is obviously not possible in the real world. Our "Simulatable Cable Model" can interact with the computer in real time, so that we can leave the exploration of the untying operation to the computer (using reinforcement learning techniques), which is also the future main subject for us. **Regarding to the success rate of the system, the success rate of our method depends on the success rate of the 2D and 3D networks, and according to Figures 4 and 5 our method has a success rate of over 80%, which is at least 10 percentage points higher than the original method. On average, it takes 2 minutes to convert a 480x360 image into a simulatable model in Unity3D, the exact time taken is proportional to the length of the cable in the image i.e. the number of particles required for the 3D model.** We believe that although the technology of wireless transmission is developing, it will take time for it to become widespread and not everything can be transmitted wirelessly. Therefore, we do not see the demand for cables decreasing any time soon, so the need for robotic cable manipulating will still exist. And our research will help in this area, both in the lives and work of individuals and in the workplaces such as factories or colleges where a lot of cables are needed.

References

[1] Kenji Sato, Kei Okada, and Masayuki Inaba.: "Humanoid's daily life support actions using image processing that recognizes string-like objects," Robomec, 2A1-D26:1-4, January 2006. (In Japanese)
[2] Yusuke Chikanari, Yui Takabayashi,, and Kohei Asai.: "Three- Dimensional Roller Coaster Simulator Using the String Recognition," IPSJ Interaction 2016, 163A04, March 2016. (In Japanese)
[3] 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).
[4] Zheming Fan, Takeshi Ohashi, and Toyohiro Hayashi.: "Research on cord-like object recognition system," IPSJ SIG Technical Report, 2019. (In Japanese)
[5] Katsuhito Sudoh, Koh Kakusho, Michihiko Minoh: "Shape Modeling for String-like Deformable Objects in Manipulation by Observing Real Objects" IPSJ Journal, vol. 43, No.12, pp. 3632–3642. (2002). (In Japanese)
[6] FAN ZHEMING, OHASHI TAKESHI, HAYASHI TOYOHIRO.: "Make 3D cable model from RGB image" IIAI AAI conference, SCAI 2021,2021/7/11～7/16 . (2021).
[7] Unity3D home page: https://unity.com/
[8] Matthias Muller, Bruno Heidelberger, Marcus Hennix, John Ratcliff.: "Position Based Dynamics" 3rdWorkshop in VRIPHYS (2006).
[9] Mile Macklin, Matthias Muller, Nuttapong Chentanez.:"XPBD: position-based simulation of compliant constrained dynamics" MIG '16: Proceedings of the 9th International Conference on Motion in Games, pages49-54.(2016)
[10] Mile Macklin, Matthias Muller, Nuttapong Chentanez.:"Unified particle physics for real-time applications" ACM Transactions on Graphics, Volume 33, Issue 4, Article No.: 153, pp 1–12 (2014)
[11] Blazej Kubiak, Nico Pietroni, Fabio Ganovelli.:"A Robust Method for Real-Time Thread Simulation" Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 2007, Newport Beach, California, USA, November 5-7 (2007)
[12] T.Kugelstadt, E.Schomer.: "Position and Orientation Based Cosserat Rods" ACM SIGGRAPH Symposium on Computer Animation(2016)
[13] OBI system home page: http://obi.virtualmethodstudio.com