Geometry Based Rope Knot Tying Simulation

Mianlun Zheng
School of Computer
Wuhan University
Wuhan, China
mianlunz@gmail.com

Huasong Han
School of Computer
Wuhan University
Wuhan, China
hhs19931014@gmail.com

Qing Guo
School of Computer
Wuhan University
Wuhan, China
guoqing123256@gmail.com

Zhiquan Hu
School of Computer
Wuhan University
Wuhan, China
whu2012zhiquanhu@gmail.com

Qi Wang
School of Computer
Wuhan University
Wuhan, China
whu120enjoywq@gmail.com

Zhiyong Yuan*
School of Computer
Wuhan University
Wuhan, China
zhiyongyuan@whu.edu.cn

***Abstract*: Geometry based knot tying simulation of rope was implemented. The rope was modeled as a series of rigid links and FTL (Follow the Leader) algorithm was used for its deformation. Cylinders were drawn as the rigid links and also as detecting units for the rope's collision detection. The radius of cylinders could be changed to model different ropes. In our collision detection, an improved AABB bounding box algorithm was adopted to accelerate collision detection. The method proposed can process rope's self collision and collision between rope and other rigid objects. The collision distance parameter was set to implement the equivalent friction. Users could grab a head node of the rope and control it moving by using haptic device. And all the other nodes' positions were calculated according to their current ones and neighbors'. Finally the results of the simulation were given and discussed. The model is a real time one and has equivalent physical properties.**

Keywords: Knot tying ; Geometry Based Simulation ; Rigid links ; FTL algorithm ; Collision detection

I. INTRODUCTION

Virtual Reality is one of the fastest developing technologies of information technology in recent years. It is regarded as the three best-prospect computer technologies with multimedia technology and network technique[1].Virtual Reality takes advantage of computer to simulate a three-dimensional virtual world, providing users with simulated sense of vision, hearing, touch and so on. In this way, feelers can observe the objects in any time and perspective, without any restrictions.

In the background of Virtual Reality, how to simulate ropes and knot tying can be used in industrial machinery design. For example, automatic strapping machines of bank notes, automatic knot-tying machine of kelp, automatic suture knot-tying machines for minimally invasive surgery and the like. Moreover, in 3D games, slender lines can also be modeled to make graphs more true to life, such as hair fluttering and fabric sewing[2].

The study of knot-tying simulation can be divided into two types: physically-based simulation and geometrical features based simulation.

The former one includes kinematics simulation and dynamics simulation, both using physical approaches to describe the rope's attributes and model it. As a result, these models are all with high veracity. For example, in 2002 Dinesh K. Pai. proposed a model to establish a thin and flexible body based on Cosserat Rod theory[3]. He represented the thread by a space curve and described its shape after torsion and bend by solving dynamics equations. In 2007, Spillmann, J. add FEM into Cosserat theory[4], thus making the model more realistic. Besides, Moll Mon et al raised a path planning strategy of flexible lines according to the least energy loss principle[5]. It gets the positions of the line by defining its rotation stress equations. In 2008, Fuhan Shi et al used FEM models to simulate sutures[6]. Phillips et al used a spring-mass model with adaptive segmentation capabilities to simulate rope knotting in 2002[7]. However, these models have a problem that real time was insufficient because of large cost of computation and solving complex equations.

Geometry based methods model the rope by using mathematical equations, which have good characteristic of real-time but lack physical properties. Joel Brown put forward a method based on geometry in 2004, in which a string of continuous cylinders are used to represent a rope, using a algorithm called FTL(Follow The Leader) [8]. Besides, it also applies a series of constraint conditions to hold characteristics of the rope. In addition, a multipoint line is used on behalf of the linear shape by Zhang Hui[9].They controlled the line's shape by a series of discrete points, with an aim of obtaining the length of the rope and its path information. Huang Mingji et al applied B-Spline into their model and created a realistic model[10].

But this method can only simulate certain levitating ropes whose two endpoints are fixed and without a length constraint. Jian Huang et al solved problems involving rope's deformations by differential geometry methods[11], getting the starting point through the global coordinate system. Next, acquire the relationship between every two neighbor points on the rope in local coordinate system and finally the whole rope is gotten.

In this paper, we mainly pay attention to the rope's movement, so the real-time performance is considered. FTL geometric algorithms can be easily implemented without large amount of computations which also has the benefits of high real-time performance. What's more, it can describe the movement of the rope. So FTL algorithm is used to achieve a rope knot simulation, while adding the equivalent friction and constraints into the model to solve the disadvantages of a lack of physical characteristics.

II. MODEL OF THE ROPE

In this paper, the rope with a length of L is separated into a plurality of discrete particles. A rigid rod with length of l connects each two particles .Therefore, the complete rope is represented by all connected rigid rods. Each particle is numbered as $N_i (i = 1...n)$, and the i 'th rod links i 'th and $i+1$ 'th particle. However, we can't reflect the attribute of rope's width only using some linked segments.In order to make it up, we draw a cylinder outside every rigid rod as seen in "Fig.1", where the cylinder has a radius of r.

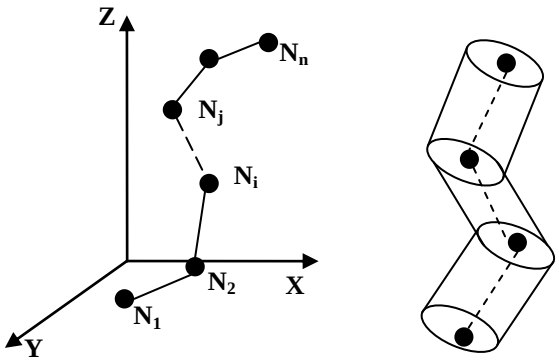

Figure 1. The rope model and its width attribute.

When the position of rope's particles change, redrawn their cylinders accordingly. In this way, the model has a better sense of reality. Moreover, it is possible to simulate different types in other applications with a changeable radius of cylinders that is the width of the rope. The rope modeling process can be divided into following steps:

- Grab a particle of the rope as a head node.
- Use FTL algorithm to update particles' positions.
- Collision detection and correction of particles' positions.

A. *Grab the head node*

From micro perspective, rope's movement is exactly the changing positions of particles in our model. We grab a node in the rope model as the head node of rope's movement and control it so as to move all the nodes in the model which represent the string position.

In this experiment, a certain particle's coordinates are obtained in three-dimensional space through the mouse. In fact we can only get two-dimensional coordinates of the mouse on the screen, so we have to calculate its three-dimensional ones. After that, take the node which is nearest to the mouse's coordinates as our head node, then drag the rope to move.

B. *The implementation of FTL algorithm*

The article utilize FTL algorithm to calculate the rope's movement. Suppose that a particle N_i is grabbed as the head node, while N_i controls the whole line's motion. The mouse drag Ni to a new position, then we compute the rest particles' new location by using the algorithm. Each particle comes toward the current position of the previous particle, but keep a distance of l.

Assume that the particle N_i 's last position is $X_{i.old}$, while its current position is $X_{i.\text{new}}$. Calculate particles' positions in uplink and downlink directions respectively from the head node Ni. The upper computation starts from the node whose $k = i$, with k increasing 1 each time.

The particle N_{k+1} 's current position moves a unit distance along the direction that is from N_k 's current position to N_{k+1} 's last position. The calculation formula is as follows:

$$X_{k+1.new} = X_{k.new} + \frac{(X_{k+1.old} - X_{k.new})}{|X_{k+1.old} - X_{k.new}|} \quad (1)$$

Similarly, in the other direction that is downward direction, each time k decreases 1, while these particles' positions are calculated by the same method. The calculation formula is as follows:

$$X_{k-1.new} = X_{k.new} + \frac{(X_{k-1.old} - X_{k.new})}{|X_{k-1.old} - X_{k.new}|} \quad (2)$$

We can see the change process in "Fig.2":

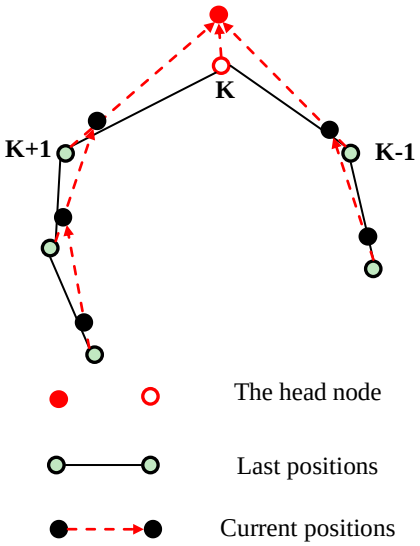

Figure 2. How FTL algorithm works.

The calculation of the rope in two directions will stop when encountering the end of the rope or one another point captured by mouse(there can be multiple head nodes).Between two adjacent head nodes i and j, the algorithm will be implemented twice, respectively from $i+1$ to j and $j-1$ to i, and the new positions of the other particles between the two head nodes equal to the average of two calculation results of particles' positions.

C. *Collision detection*

In this paper the rope model will draw cylinders outside of the line segment. As a result, the phenomenon occurs that cylinders penetrate each other or the rope penetrates other objects. Therefore, self-collision detections need to be done for the rope as well as collision detections for the rope and other objects around, in case that the rope will penetrate thus leading to model distortion.

In our detection, the collision detection unit is a long strip regular cylinder, whose radius and height are already known. So we can use AABB bounding box method to perform the detection. For the cylinder i and j, within each time step calculate the minimum distance between line segment N_iN_{i+1} and N_jN_{j+1} as d. If d is less than $2r$, it's obviously that the two cylinders collide with each other. At this moment the solution is pushing the two cylinders outward to a certain distance to make d exceed $2r$.

In order to improve the detection rate of collision, we proposed using an improved AABB collision detection algorithm which was put forwarded by Gao Yuqin[12]. After the exclusion of the impossible colliding objects , the rest objects will be built for an AABB tree in specific depth , and then traverse the all points of AABB's sub box ,finally we rebuild a bounding box which is closer to the objects. This method can do contribution in narrowing the range of detection, reducing the computational complexity and improving the detection rate.

First, traverse each vertex of the rope, then get all their coordinate values X , Y , Z . Construct the original bounding box according to X_{max} , $X_{\min}$, Y_{max} , $Y_{\min}$, Z_{max} , $Z_{\min}$. Actually this box is the root node of the binary tree. Next, subdivide the original bounding box and construct an AABB bounding box. That is making a space division of the box at father node according to the particles on the rope, recursively build two AABB trees from top to bottom.

As we can see in "Fig.3,in each recursive process, the number of particles of the rope surrounded by AABB box bounding divide the bounding box into two halves, and then traverse the points belonging to two bounding box ,finally reconstruct the bounding box to make it closer to objects. After structuring binary tree, we detect two child nodes of the tree recursively. If collision is not found on a node, then stop detecting the child nodes of this node, because it has been determined at this time there is no collision on this node. If you find this node collision exists, it must continue to do the detection recursively until the leaf node is detected. When a collision is detected two cylinders were obtained segments N_iN_{i+1} and distance N_jN_{j+1} nearest two points P_i and P_j.

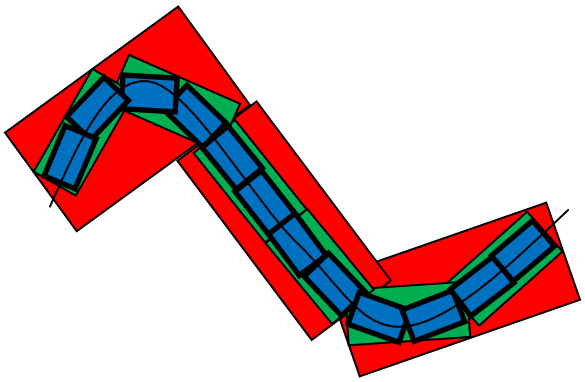

Figure 3. Construct an AABB bounding box tree.

Calculate the unit vector that points from P_j to P_i :

$$I=\frac{(P_i-P_j)}{|P_i-P_j|} \qquad (3)$$

Then adjust the positions of N_i 、 N_{i+1} 、 N_j and N_{j+1} as follows:

$$\begin{cases} N_i = N_i + \Delta I \\ N_{i+1} = N_{i+1} + \Delta I \end{cases} \qquad (4)$$

$$\begin{cases} N_j = N_j - \Delta I \\ N_{j+1} = N_{j+1} - \Delta I \end{cases} \qquad (5)$$

Δ is the displacement distance, it can be obtained by the following equation:

$$\Delta = 2R - d + \varepsilon \qquad (6)$$

And ε is a non-negative parameter.

In fact, there will be friction generated between the colliding ropes, but the FTL algorithm is based on geometry instead of the mechanics. So we set constraints in the vertex of the segment which is covered by cylinder to simulate the effect of friction. It can be adjusted by the parameters ε in equation (6) to control the volume of equivalent friction[13]. The greater ε is the smaller friction becomes. If a vertex collides with a number of vertices, then the final position of the vertex is equal to the average adjusted distance of all its collision vertices.

The collision detection of rope and other rigid objects are similar to the above procedure. It's also started traversing from the root node of the AABB level tree. If the node at present doesn't intersect with the rigid body, then stop traversing. If not, continue traversing the two child nodes of the current node recursively and the intersection test until encounter the leaf nodes. Finally, checkout if the rope vertex of the leaf node collide with the rigid body[14]. Adjust rope's location when collision occurs.

III. EXPERIMENTAL RESULTS

We write the program in C++ language and construct a circumstance of OpenGL. FTL algorithm and collision detection were well combined in our program. We set a right parameter ε in collision detection to simulate friction, which can be reflected in our program. When we drag the rope, if collision occurs, ε decides how far rope's particles move.

A. *Knot tying of the rope*

We implemented the knot tying simulation without any another rigid bodies. In this experiment, rope's collision detection of itself was used. The first node of the rope was grabbed by the mouse and leaded it to move and knot. When moving, the rope can't pass through itself. In short, the program runs well as we can see in "Fig. 4":

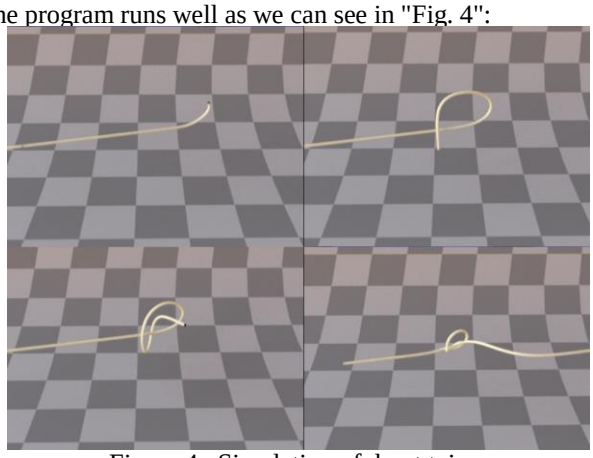

Figure 4. Simulation of knot tying.

B. *Revolve around the pillar*

In this program, a pillar was added, and we grabbed the first node to make it move around the pillar. While moving, we detected rope's self collision and that between the rope and the pillar.

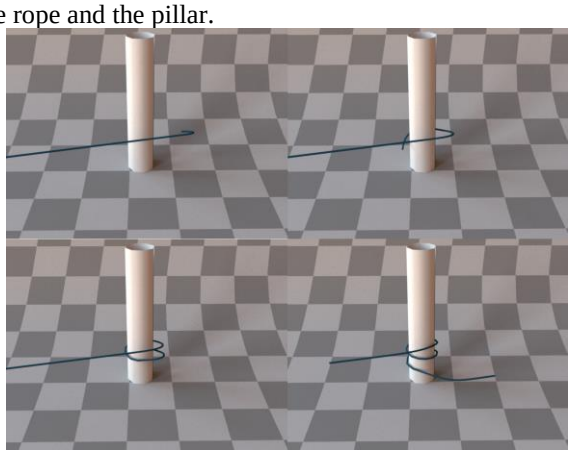

Figure 5. Simulation of rope's revolving around the pillar.

C. *Knot tying around the pillar*

Some changes were made based on the simulation above. The rope moved and also knotted around the rope.

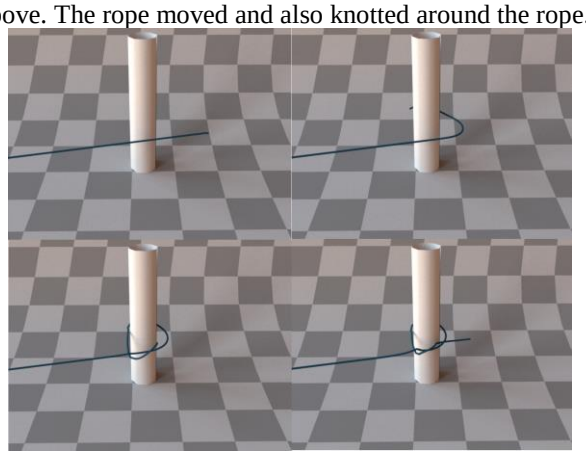

Figure 6. Simulation of rope's revolving around the pillar and knot tying.

IV. CONCLUSIONS

In this paper, we successfully simulated knot tying model. A geometry based algorithm (FTL) was applied and an improved collision detection method was executed. And the model has a strong real-time property. Besides, we set a collision parameter to achieve equivalent friction and it plays a very good effect. However, the model still lacks physical characteristics. In future studies, physical models combined with geometry methods will be considered to enhance the rope's physical reality.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (61372107, 61272276) and the Open Funding Project of State Key Laboratory of Virtual Technology and Systems, Beihang University (BUAA-VR-13KF-15).

REFERENCES

[1] 赵沁平. 虚拟现实综述[J]. **中国科学 (F 辑: 信息科学)**, 2009, 39(1): 2-46.

Zhao Qinping. Virtual reality review[J]. Science in China (F serise: Information science), 2009, 39(1): 2-46.

[2] Bergou M, Wardetzky M, Robinson S, et al. Discrete elastic rods[J]. **ACM Transactions on Graphics (TOG)**, 2008, 27(3): 63

[3] Pai D K. Strands: Interactive simulation of thin solids using cosserat models[C]//**Computer Graphics Forum. Blackwell Publishing**, Inc, 2002, 21(3): 347-352.

[4] Spillmann J, Teschner M. C o R d E: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects[C]//**Procee dings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation**. Eurographics Association, 2007: 63-72.

[5] Moll M, Kavraki L E. Path planning for minimal energy curves of constant length[C]//**Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. IEEE**, 2004, 3: 2826-2831.

[6] Shi F, Payandeh S. On suturing simulation with haptic feedback[M]//**Haptics: Perception, Devices and Scenarios. Springer Berlin Heidelberg**, 2008: 599-608.

[7] Phillips J, Ladd A, Kavraki E E. Simulated knot tying[C]//**Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. IEEE**, 2002, 1: 841-846.

[8] Brown J, Latombe J C, Montgomery K. Real-time knot-tying simulation[J]. **The Visual Computer**, 2004, 20(2-3): 165-179.

[9] 张会. 虚拟群体组织协同行为建模方法研究[D]. 长沙: 国防科学技术大学, 2006.

Zhang Hui. Organizational coordinate behaviors modeling of virtual entity group[D]. Changsha: National University of Defense Technology, 2006.

[10] Huang Mingji, Dong Xiuping. Research on Flexible Cable Geometric Modeling Technology in Virtual Maintenance Based on VRML [C]// **Mechanic Automation and Control Engineering.** Wuhan, China. USA: IEEE, June, 2010: 772-775.

[11] Jian Huang, Pei Di, Toshio Fukuda, Takayuki Matsuno. Dynamic Modeling and Simulation of Manipulating Deformable Linear Objects [C]// **International Conference on Mechatronics and Automation, Takamatsu, Japan. USA: IEEE**, 2008: 858-863.

[12] 高玉琴, 何云峰,于俊清. 改进的基于 AABB 包围盒的碰撞检测算法[J]. **计算机工程与设计**, 2007,28(16):3815-3817.

Gao Yuqin,He Yunfeng,Yu Junqing. Improved collision detection algorithm based on AABB[J] , **Computer Engineering and Design** , 2007,28(16):3815-3817.

[13] 韩育芳. 质点弹簧构造的皮肤缝合算法研究[D], 秦皇岛: 燕山大学, 2012.

Han Yufang. The Algorithm study of skin suturing procedure simulation based on mass-spring[D]. Qinhuangdao: Yanshan University, 2012.

[14] 贾世宇,潘振宽. 具备触觉反馈的实时绳子打结仿真[J]. **系统仿真学报**, 2009,21(20): 6519-6523.

Jia Shiyu,Pan Zhenkuan. Real-time knot tying simulation of ropes with haptics[J]. **Journal of System Simulation**, 2009,21(20): 6519-6523.