Modeling of a Rigid Tether for ROV Simulation in Underwater Environments.

Quentin Brateau, ENSTA Bretagne, Brest, France, quentin.brateau@ensta-bretagne.org

***Abstract*—Simulation in the field of robotics is a powerful tool. Indeed, it allows to easily and quickly test the robot in different conditions and to have a reproducibility of the results. Then it let us be able to create situations that would be difficult to find in reality, in order to make sure of the robot's behavior. It should be noticed that the simulation of robots does not replace tests in real conditions, but it remains practical during the development phase. However, the simulation of robots in the maritime environment is a field that still has shortcomings, especially when we want to simulate the tethers of submarine robots. This scientific paper presents a method to model this rigid tether in order to build a simulation environment for autonomous underwater robots.**

***Index Terms*—Modeling, Tether, Remotely Operated Underwater Vehicle, Simulation.**

I. Introduction

In the world of marine and underwater robotics we can indentify two categories of elements: mobile marine objects (MMO) and flexible tethers (FT). Mobile marine objects include surface vessels, submarines and remotely operated vehicles. Flexible tethers can represent umbilical cables, traction cables and anchor chains in the marine environment. They constitute all the necessary elements to achieve a mission in this environment which is sometimes quite difficult to explore.

The simulation of moving marine objects is something well known. We are now able to know from state equations the behavior of robots in their environment, and these equations are known for ships, submarines, sailboats, etc...

On the other hand, determining the behavior of a flexible tether becomes more complicated. Indeed, the equation of motion of these objects involves non-linear partial differential equations and the motion between the different objects in the environment are dynamically dependent. It is well illustrated that if the boat moves, it will induce a motion in the flexible tether that will modify the trajectory of the remotely operated vehicle. This is why it is difficult to find behavioral models for flexible tethers in underwater environments.

II. Formalism

The idea proposed in the scientific paper [1] is to solve this problem using finite element simulation. This implies that we need to discretize the tether in order to simulate its global behavior.

Suppose we want to simulate a tether of length L. We will then divide it into a finite number n of nodes connected by links. These links should be of length $l = \frac{L}{n-1}$ as the two nodes at the ends of the tether will not be connected to any other links.

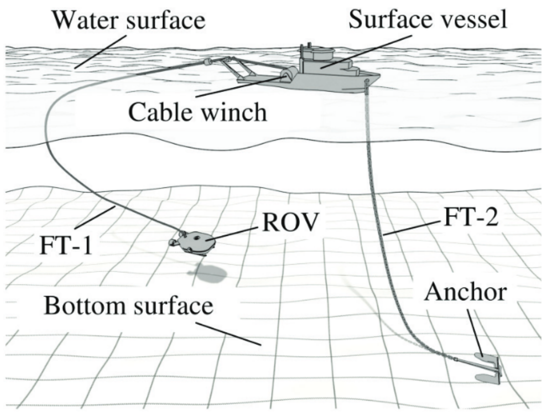

Fig. 1. Example of a complex underwater environment

We will focus here on the case where the first node and the last node are immobile, because otherwise we would have to simulate a mobile marine object that would be attached at the end, which is not the goal of our study.

Next, it is necessary to make a balance of the forces that apply to each tether element. For this simulation, we will take into account the weight, noted F_g, the buoyancy, noted F_b, and the force exerted by the previous element on the considered element, noted $F_{t,previous}$, as well as that of the next element, noted $F_{t,next}$.

These forces will allow us to simply describe the behavior of the tether in its environment. Moreover we could then improve the quality of the simulation by adding other forces such as forces related to a current for example.

A. Weight F_g

Considering that each element has a mass m, we are able to express the weight that applies to this node :

$$\overrightarrow{F_g} = \begin{bmatrix} 0 \\ 0 \\ -m.g \end{bmatrix}$$

B. Buoyancy F_b

If we note the volume of each element V and ρ the density of the fluid in which the tether is immersed, we can also express the buoyancy force of this node:

$$\overrightarrow{F_b} = \begin{bmatrix} 0 \\ 0 \\ \rho.V.g \end{bmatrix}$$

C. Tether force $F_{t,previous}$ and $F_{t,next}$

It is difficult to find an analytical form to describe these two forces. Therefore, we must find a way to describe these forces in order to simulate the tether correctly. This is why we will use a behavioral model here. We know that each node will have to be at a distance l from each of its neighbors. We can assume that the system behaves here as a three-dimensional mass-spring system and we will then consider that these forces are like elastic spring forces.

By noting then p_p the position of the previous node, p_c the position of the current node and p_n the position of the next node, by introducing a coefficient K_p allowing to express the stiffness with which a node will correct its position with respect to its neighbors, we are able to express the behavioral model of these two forces:

$$\overrightarrow{F_{t,previous}} = -K_p.\left(\|p_p - p_c\| - l\right).\frac{\overrightarrow{p_p - p_c}}{\|p_c - p_p\|}$$

$$\overrightarrow{F_{t,next}} = -K_p.\left(\|p_n - p_p\| - l\right).\frac{\overrightarrow{p_n - p_p}}{\|p_n - p_p\|}$$

III. Implementation

As for the implementation, it is available in the same GitHub repository and offers a simulator coded in Python3. The goal of this simulator is to study the viability of such a system, in particular to validate the performance of the tether with this behavioral model.

So we will create a class *TetherElement* which will represent a node. It will have to contain its mass, volume and distance information from its neighbors, but also its position, velocity and acceleration, as well as a pointer to each of its two neighbors.

This will allow us later on to implement a *Tether* class to be able to simulate a tether. This object must have a length, a number of elements and a list containing the different *TetherElement* that compose it. It must also know the mass, the volume of each node in order to correctly instantiate the *TetherElement*, but also the length of each link between nodes.

A diagram of these two classes is visible on the Figure 2. It respects the UML format and allows to see the different class variables and methods associated to each class.

IV. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

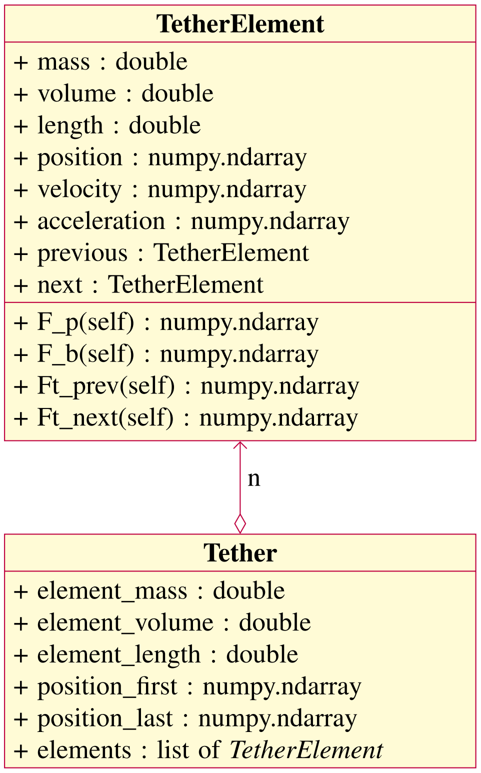
TetherElement
+ mass : double
+ volume : double
+ length : double
+ position : numpy.ndarray
+ velocity : numpy.ndarray
+ acceleration : numpy.ndarray
+ previous : TetherElement
+ next : TetherElement
+ F_p(self) : numpy.ndarray
+ F_b(self) : numpy.ndarray
+ Ft_prev(self) : numpy.ndarray
+ Ft_next(self) : numpy.ndarray
n
Tether
+ element_mass : double
+ element_volume : double
+ element_length : double
+ position_first : numpy.ndarray
+ position_last : numpy.ndarray
+ elements : list of TetherElement

Fig. 2. UML diagram of the implementation

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.