Simulated Needle Insertion With KUKA

Peter Cook Bulukin

February 2020

Matricola: 1839826
Course of Study: Erasmus

1 Objective

The objective of this project was to simulate the force-feedback during a simulation of a needle insertion in V-REP, and map this to a haptic interface. This part of the project focuses on the calculation of the force-feedback during simulation given that the KUKA robot is able to move.

2 Scene objects

We received a scene in V-REP that contained a phantom with several detailed tissues represented as organs. We also received a model of the KUKA robot. We put these to models together in one scene so that they could interact. The model we received had the needle set to static, and since we need the needle to be dynamic to interact with the physics engine of V-REP, I set it to dynamic and added a force sensor connection between the last link and the needle, *Link_needle_connection*, so that the needle wouldn't fall off the robot.

3 Scene properties

Only one of two objects has to be dynamic for them to interact through the physics engine of V-REP. Since the phantom is a cross section of the body, most of the organs are floating mid air, and thus have to be static. From before, all the tissues were set to non-respondable. For two objects to interact with each other through the physics engine, they have to be respondable. All tissues were thus set to respondable. This will change during simulation, but a simulation should always start with all tissues being respondable. The needle has to be able to interact with all the tissue, but the tissues are not supposed to interact with each other. This is not a problem, since two objects that are both set to static, will not interact with each other in the physics engine.

4 Moving the KUKA Robot

The full project is intending to use the Geomagic haptic interface to move the KUKA robot. Since my part of the project was dealing with the needle-tissue interaction, the robot just had to move. To do this, the lwr_tip of the KUKA robot, which is a dummy attached to the needle tip, were linked to Dummy_tool_tip. This dummy tool tip could then be moved, and the KUKA robot would move trying to follow the Dummy_tool_tip. The Dummy_tool_tip were then set to follow a path, inserting the needle into the phantom. This made it easy to develop and test the implementation, without having to directly control the KUKA robot.

5 V-REP Plugins

Plugins in V-REP has a specific structure and is in our case, written in C++. A plugin in V-REP includes the v_repLib.h, that gives access to the API of V-REP. Some versions of plugins also include luaFunctionData.h, to aid the process of generating Lua functions. Generating Lua functions is a convenient way of making the functionality of the plugin not depend on the object names in the V-REP scene, but requires child-scripts inside of LUA to connect to the plugin using callbacks. This has not been widely used in our projects, and we assume that the object names stay the same for the plugin to work as intended. Our plugin is thus tightly connected to the scene it was created for. Other parts of V-REP plugins include the three functions v_repStart, v_repEnd and v_repMessage. v_repStart is just called when the plugin is loaded and v_repEnd just when the plugin is unloaded, and is thus used for setup and cleanup tasks respectively. v_repMessage is the messaging routine of the plugin, and V-REP calls this function very often with different callback messages, that can thus trigger functionality in the plugin. The most important messages in our project is simulationabouttostart, simulationended and modulehandle. The first two is used to implement setup functionality and cleanup tasks respectively. Modulehandle is called during simulation and can be used to create the continuous modelling of the environment. To make beginning of writing a plugin easier, V-REP provides a plugin skeleton, that can be used to write out a plugin from. This is what was done in the beginning, for the functionality to be easily tested and later transferred to the project plugin.

6 Chai3D

Chai3D is a framework for connecting the Geomagic haptic interface to V-REP. It generates a plugin for V-REP, with automatically generated code, intended to create interactions between the Geomagic and V-

REP. In our project we were supposed to edit a Chai3D plugin.

7 Code base former projects

We received earlier projects that contained old code in the framework of Chai3D, that we were supposed to use to connect our simulation to the Geomagic haptic interface. This contained code to control the KUKA robot as well. Since I was supposed to write the logic of the simulation of the interactions between the needle and the phantom, and were firstly working in the plugin skeleton, this code base is more related to the other part of the project. Unfortunately, there were several issues with controlling the robot, and preferably, we should have been able to control it with the haptic interface from the start using the old projects. Even tough my part of the project weren't directly linked to the haptic interface, I still took part in the integration work. I integrated my code into the Chai3D plugin and changed out the old external force computations with my newly written external force computations.

8 Code project structure

8.1 Pre-rupture modelling

The phantom and its tissues are complex shapes that are not easily modelled. Because of that, the force calculations should be divided into two parts. Tissue is regarded as a viscoelastic material, acting with elasticity before rupture and viscosity after. This means that the interactions forces before a rupture takes places are flexible, while the needle is still outside of the tissue, and another model is needed where the needle is inside of the tissue. Because of the complexity of the surface shapes, we used the dynamic engine of V-REP to model the interaction forces before a rupture, until a force threshold were reached, and a rupture were simulated. V-REP contains the function simGetContactInfo() this can be used to check the contact info between two objects. It includes both the position, normal direction and forces that happens in the contact. This gives us the forces needed to calculate the force interaction on the outside of the tissue.

8.2 Rupture modelling

When a certain force threshold is reached, the needle is supposed to rupture the skin and penetrate the tissue. V-REP's physics engine doesn't support rupture functionality, so this had to be modeled in our code. When a certain force threshold is reached between the needle and a tissue, a rupture takes place. The object of the ruptured tissue is set to non-respondable in V-REP to allow the needle to penetrate the tissue. The rupture is added to a vector of ruptures, to be used later.

8.3 After-rupture modelling

After the rupture the needle is inside of the tissue. Here I used to different methods during the development. The first method takes more computational power, and would give a little bit more precise estimate of the penetration distance, because it would notice when the needle exited on the other side of the tissue. This would take the pockets of air inside of the phantom into consideration. The second method assumes that the penetration of one tissue continues all the way either to the needle tip or a new tissue is recorded. The implementation of the first method requires CGAL. This comes native in later versions of V-REP but is not included in earlier versions. I therefore chose to go over to Eigen, which was used throughout the other projects to do the geometric calculations. The other advantage of the first method is that it doesn't assume that the needle stays in the same place as the puncture happened, but calculates the needle penetration live. The assumption that the point of puncture stays the same during the whole procedure is a central assumption, and makes this advantage of the first method unnecessary. The only scenario where this advantage could be useful, is if the phantom and tissue were dynamic, and could actually move during the procedure. Then this method would be updated on the movements, while the second method would have wrong estimates. Both methods has the goal of estimating the length of the different segments of the needle, that is currently inside of the different tissues. This length can then be used to calculate the total interaction forces while the needle is inside of the tissue. The models that were used will be issued later.

8.3.1 Method 1

The first method for calculating the penetration lengths were based upon getting the extended contact info from the interaction between the needle and all punctured tissues. Even after the tissue is non-respondable, contact info can still be retrieved. simGetContactInfoEx gives contact info on all contact segments that is part of the contact. The first method was based upon, for each tissue, to get all the segments of the contact between the whole needle and the tissue. From this we could calculate the two points that were furthest apart. By brute force, this has computational complexity $O(n_t \cdot n_p^2)$, where n_t is the number of tissues, and n_p is the number of points in the contact segments associated with each tissue. To reduce this, an option is to compute the convex hull using Jarvis March (Gift Wrapping), Graham Scan, Quickhull or similar methods.[2] They have sub polynomial expected computation time, and some of them i.e. Graham Scan can reach $O(n_t \cdot n_p)$ if some criteria are met. After finding the convex hull, rotating calipers could be used to calculate the farthest points in the convex hull.[4] The original algorithm is designed for two dimensions. O'Rourke generalized it to three dimensions, but in practice estimation algorithms are used because they are faster.[3] After some experimentation, I found that because of the shape of the needle, the convex hull reduced the number of points to just a few, and I could thus just search through them for the longest distance.

8.3.2 Method 2

The second method is a much simpler method. It records the point of rupture for each rupture. Then it calculates the distance from this rupture, either to the needle tip or until the point of the next rupture. If the needle tip exits the tissue on the other side without entering a new tissue, the air penetrated will still be considered as penetration. This could be corrected by checking the contact between the needle tip dummy object and the tissue, to record when the needle exited the tissue. The implementation increments and decrements the full length of penetration and the penetration length of only the last punctured tissue iteratively. This is because if the needle is not inside of a certain tissue, we know that the penetration length will not change.

9 Force interaction model

The forces created by the interaction between a needle and soft tissue is not just possible to model with a simple friction model. Several papers have been written about this, and empiric experiments have been used to create a fitting model. The behaviour of the fibers in the tissue results in different interaction forces on insertion and retraction of the needle.[6] I have thus included two abstracted interaction models. The first model uses first, a capsule stiffness, a nonlinear spring model, before puncture, friction during penetration based on a modified Karnopp model, given in equation 1 and a constant for a given tissue to model the cutting force.[6] In our project I only implemented the Karnopp friction part, before changing to a Kelvin Voigt model to model the viscoelastic properties of tissue during needle insertion. This model is valid during low velocities. Some research were done into a more precise modified Kelvin-Voigt model, that takes into consideration the needle tip and the depth of penetration.[5] The kelvin-voigt model of our project[1] is given by

$$\sum_{t \in T} \mu_t V A(x_t)$$

where μ is the viscosity constant for each tissue, x is the length of the segment of the needle penetrating the tissue and $A(x) = \pi r x$ is the area of the needle in contact with the tissue. Here r is the diameter of the needle. The original function is split in two, depending on if the needle penetration is thicker than the tissue. This is not needed in our case, since a different angle of the needle will give different thicknesses. The distance recorded will depend on the accuracy of the penetration length estimate commented above. The Kelvin-Voigt model used is a viscoelastic model. In our project the elasticity of the model is relevant before puncture. Since the physics engine models this, our code does only incorporate the viscosity. The total external force will be the force given by the physics engine, acting on the needle, and the forces modeled by the Kelvin-Voigt model during penetration. External forces from the physics engine will still be prevalent whenever the needle meets a new tissue. For now I have only added customized viscosity constants for fat, muscle, lung and bone. There is also no difference between the puncture threshold of different tissues.

10 V-REP Version

Because of a lot of compatibility issues, we have had to change our V-REP version multiple times. I started using CoppeliaSim, because this is the newest version and thus all updated documentation and Github repositories are linked to this. This resulted in having the change the code a lot when changing to V-REP. It appears that the choice of V-REP version greatly impacts the stability of the simulation. Differences in naming conventions and plugin conventions has required me to remake the plugin for every version we have tried using. It has been a problem, that the older project that we were supposed to build upon uses a V-REP version that is not any longer downloadable because of errors. The current version I am working in is 3.2.2, and some weird behaviour from the KUKA robot is apparent in some simulation scenarios. Some of this weird behaviour happens to the KUKA robot in certain poses. I have decided to overlook this, and just try to avoid these poses, since that is not my part of the project.

11 Further work

I have made sure that the Chai3D plugin calculates the external forces on each iteration. The global force computation was kept from the former project and has to be revised. The external force calculation is inserted into the global force calculation and is from there inserted into the haptic interface. The plugin will now calculate the external forces on the needle, independently of how the robot is controlled. I have used the plain velocities of the needle tip in the V-REP scene for my calculations. If this should go through a low pass filter, this has to be included in future work. I would propose that the speed of the dummy of the haptic interface tip should be filtered and then have the actual needle tip just follow this. Thus, the calculations of the forces should not depend directly on any objects directly linked to the speed of the haptic interface, but the actual needle tip of the KUKA robot in the simulation. That is, as long as the KUKA robot moves smoothly, there is no point in filtering the needle tip speed.

12 Conclusion

I have in my part of the project implemented force interactions between the needle of the KUKA robot and a phantom. The forces are calculated from a combination of the physics engine of V-REP and modeling of viscosity after rupture of tissue modeled after a Kelvin-Voigt Model.

13 Equations

$$F_{\text{friction}}(z, F_a) = \begin{cases} C_n \text{sgn}(z) + b_n z, & z \leq -\Delta v/2 \\ \max(D_n, F_a), & -\Delta v/2 < z \leq 0 \\ \min(D_p, F_a), & 0 < z < \Delta v/2 \\ C_p \text{sgn}(z) + b_p z, & z \geq \Delta v/2 \end{cases} \tag{1}$$

References

[1] P N Brett et al. "Simulation of resistance forces acting on surgical needles". In: *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine* 211.4 (1997). PMID: 9330545, pp. 335–347. DOI: 10.1243/0954411971534467. eprint: https://doi.org/10.1243/0954411971534467. URL: https://doi.org/10.1243/0954411971534467.

[2] *Convex hull algorithms*. 2019. URL: https://en.wikipedia.org/wiki/Convex_hull_algorithms (visited on 02/19/2020).

[3] David. *Computing oriented minimum bounding boxes in 2D*. 2014. URL: https://geidav.wordpress.com/2014/01/23/computing-oriented-minimum-bounding-boxes-in-2d/ (visited on 02/19/2020).

[4] H. Freeman and R. Shapira. "Determining the Minimum-Area Encasing Rectangle for an Arbitrary Closed Curve". In: *Commun. ACM* 18.7 (July 1975), pp. 409–413. ISSN: 0001-0782. DOI: 10.1145/360881.360919. URL: https://doi.org/10.1145/360881.360919.

[5] Mohsen Mahvash and Pierre Dupont. "Mechanics of Dynamic Needle Insertion into a Biological Material". In: *IEEE transactions on bio-medical engineering* 57 (Nov. 2009), pp. 934–43. DOI: 10.1109/TBME.2009.2036856.

[6] A. M. Okamura, C. Simone, and M. D. O'Leary. "Force modeling for needle insertion into soft tissue". In: *IEEE Transactions on Biomedical Engineering* 51.10 (Oct. 2004), pp. 1707–1716. ISSN: 1558-2531. DOI: 10.1109/TBME.2004.831542.