Applying the TPS method to Modeling and Simulation of Biological Systems

Sanaa Lissari[1], Nada Farran[1], Hamel Yigang[1], Rhys Goldstein[2], Gabriel Wainer[3]

[1]Polytech Marseille – Department de Genie Industriel Domaine Universitaire de St Jérôme, 13397 Marseille Cedex 20 France

[2]Autodesk Research 210 King St. Toronto, ON Canada
rhys.goldstein@autodesk.com

[3]Systems and Computer Engineering Carleton University 1125 Colonel By Dr. Ottawa, ON, Canada
gwainer@sce.carleton.ca

Keywords: modeling and simulation, deformable biological structures, presynaptic nerve terminal, actin, mitochondria

Abstract

Deformable biological structures are typically modeled with the finite element method, but we have designed a much simpler impulse-based method called the "tethered particle system" (TPS). The TPS involves the use of discrete-event simulation to track the positions of a large number of particles. Two of these particles approaching to each other may collide and rebound off. The difference is that, provided they are "tethered", two separating particles may also collide and retract inwards. This constrains the distances between pairs of particles, allowing various deformable structures to be represented. In this article, we present the application of the TPS method to the simulation of different biological structures found in nerve cells: actin filaments, cell membranes and synapsin protein that bind with actin, mitochondria, and water particles which influence the motion of other structures. This includes advanced visualization methods.

1. INTRODUCTION

The simulation of deformable structures is used for a wide range of applications in the biomedical field, including the study of physiology, the analysis of joint replacements, and the planning of surgeries. Though the finite element method is likely the most popular method for such applications, the recently-developed tethered particle system (TPS) is a relatively simple alternative [1].

In [2], the TPS was used to model interactions between vesicles and synapsin protein in a presynaptic terminal, the compartment located and the end of a nerve cell where a signal is transmitted to an adjacent nerve cell. As seen in Figure 1, a presynaptic terminal contains a reserve pool of synaptic *vesicles* connected by *synapsins*. Synapsin is a neuron-specific phosphoprotein that binds to small synaptic vesicles and *actin* filaments in a phosphorylation-dependent pattern. Microscopic models have demonstrated that synapsin inhibits neurotransmitter release either by forming a cage around synaptic vesicles or by anchoring them to the F-actin cytoskeleton of the nerve terminal. The goal of the work presented in [2] was to build a detailed model of this nerve terminal, based on the TPS method, with the goal of predicting the number of synaptic vesicles released from the reserve pool as a function of time under the influence of action potentials at differing frequencies.

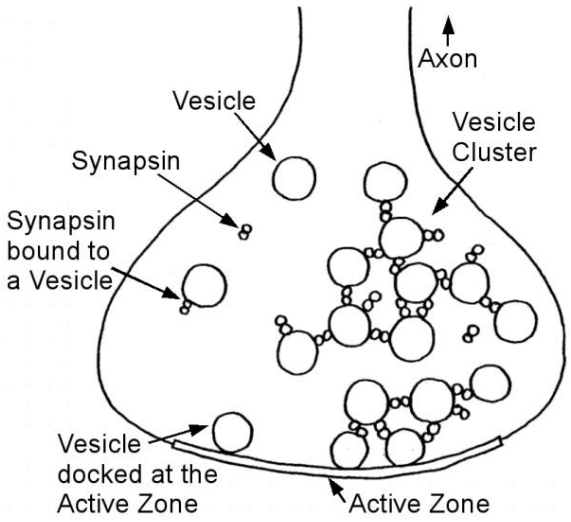

Figure 1. Diagram of a presynaptic terminal

This type of simulation has the potential to help biologists understand the role of different biological entities. For example, they can be used to investigate the theory suggested in [4]: that synapsin regulates the transmission of signals by binding with neurotransmitter-containing vesicles. The idea is that, due to the synapsin, a cluster of vesicles is maintained near a region of the membrane known as the active zone. However, it is also believed that filaments known as actin may play a similar role. Actin bind with both synapsin and the cell membrane. It is also important to account for the presence of larger structures such as mitochondria, and smaller entities like water particles.

While a description of the TPS can be found in [1], here we demonstrate how the method can be used to model

various specific deformable biological structures. The structures presented can all be found in nerve cells.

Section 2 presents related work. After providing a brief overview of the TPS method in Section 3, we explain how it was used to model these other structures found in nerve cells. Section 4 introduces a model of a helical protein representing actin, and demonstrates how it can be tethered to a cell membrane. Section 5 describes an enhanced actin model, and simulates the binding of multiple filaments. Finally, Section 6 applies the TPS to mitochondria and water particles, large and small entities which impede the motion of other objects. The implementation of the simulations is described in Section 7.

2. RELATED WORK

Our interest in *particle-based* simulation methods, where individual particles are tracked. This is in contrast to *population-based* methods like the Gillespie Algorithm [5], where concentrations are tracked instead of particles.

GridCell is an example of a particle-based simulator that operates on cellular lattice [6]. Each cell in a 3D cubic lattice contains at most one particle. At fixed time intervals, each particle may move randomly to one of its 26 neighboring cells, or undergo a reaction that causes itself and possibly a neighboring particle to be replaced with one or two new particles. One of GridCell's strengths is its inherent ability to simulate molecular crowding in a computationally efficient manner. Molecular crowding occurs when the density of particles in a particular region impedes particle motion and reactivity.

GridCell's most obvious weakness is its discretization of space. Other particle-based simulators have been developed for *continuous-space models*, with particle positions described by continuous coordinates. In the MCell program [7], particles move randomly in any direction through 3D space. When a particle encounters a surface, a cell membrane for instance, a reaction may occur.

In [8], the MCell program was used to simulate the reaction and diffusion of chemicals around vesicles docked on a presynaptic membrane. This simulation involved a continuous-space model in which vesicle positions were based on measurements of an actual presynaptic terminal. The vesicles could not move, however, and other entities such as synapsin, actin, and mitochronia were not modeled.

3. TETHERED PARTICLE SYSTEM

The simplicity of the TPS contributes significantly to its appeal. A TPS model consists of a large number of moving particles. Two types of collisions occur between pairs of particles: blocking collisions and tethering collisions.

A blocking collision is the more familiar type. It occurs when two approaching particles reach an inner limiting distance *Δu_blocking*, as illustrated in Figure 2, and causes the particles to rebound outwards.

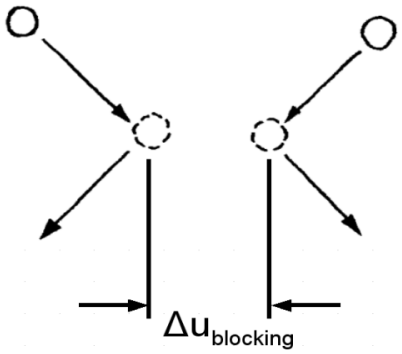

Figure 2. A blocking collision.

When a blocking collision occurs, the two particles involved may become tethered to one another. If two separating tethered particles reach an outer limiting distance *Δu_tethering*, and if the particles remain tethered, then a tethering collision causes the particles to retract. The unraveling cord in Figure 3 illustrates this effect, but is not explicitly modeled by the TPS.

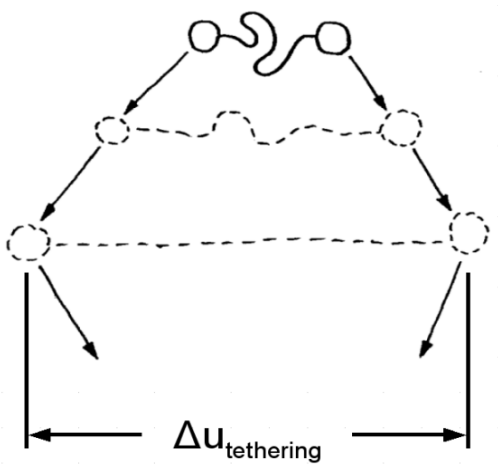

Figure 3. A tethering collision.

Although the particles are shown as circles or spheres in illustrations throughout the paper, only the blocking and tethering distances are explicitly defined in a TPS model. Particles need not be given radii. By constraining the distances between particles with suitably chosen $\Delta u_{blocking}$ and $\Delta u_{tethering}$ parameters, various deformable structures may be represented.

4. MODELING AND SIMULATION OF ACTIN FILAMENTS NEAR VESICLE CLUSTERS

This model consists on actin protein inside the membrane. The actin is a globular protein that form helical filaments (or microfilaments). It has the shape of two helices bound to each other.

To simplify the model, we first built a 'snake-like' structure composed of blocks of triangles, which are adapted to make appear the helices. In each block of the 'snake', there is a triangle of 3 particles or spheres.

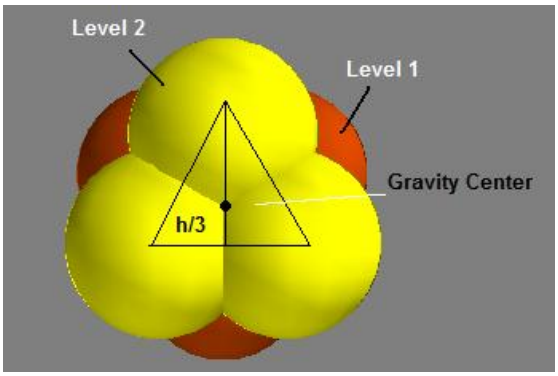

Figure 4. View of one block of the 'snake'

From the first block to the second, we invert the triangle of block 1 and translate it to the position of the block 2 by keeping the same gravity center. We do that every time we go from a block to the next. This gives us an horizontal 'snake' with different blocks. The yellow triangles in Figure 5 are inverted comparing to red triangles.

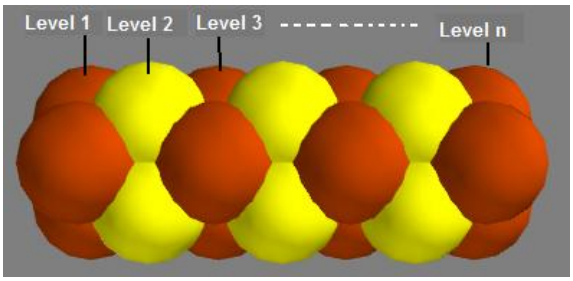

Figure 5. Picture showing levels of the triangles sequence modeling the actin filament

Our 'snake' is composed of blocks of triangles, and we give a different color to each particle in the same block.

Going from one level to other, we rotate colors, so as we can have three helical shapes as shown in Figure 6.

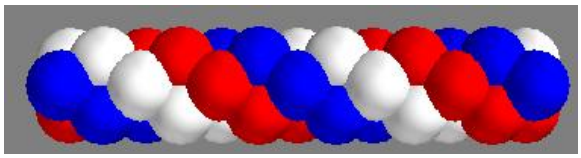

Figure 6. Helical 3-color shape of the Actin

By making the blue particles transparent we end having the simplified shape of actin presented in Figure 7.

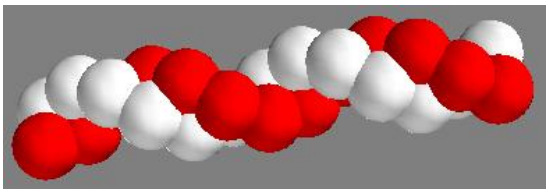

Figure 7. Helical 2-color shape of the Actin

In order to avoid that the actin particles go in different ways during the motion of the filament. we make the particles tethered and we define for them their blocking and tethering distances. For that we used the TPS (Tethered Particle System) principal that determine the rules of collision between particles.

Each particles in 'the level n' is tethered to the other particles of the same level. It's also tethered to two near particles from the level n-1 and to two near particles from the level n+1.

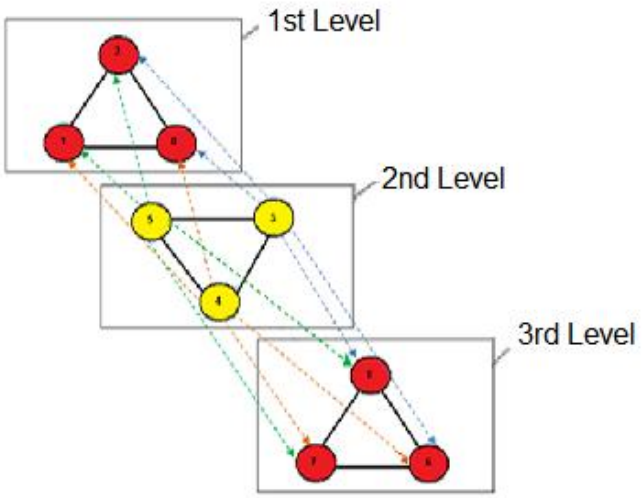

Figure 8. Definition of the tethered particles

In this schema, for example Particle 3 is tethered to particles 4and 5 of the same level. It's tethered particles to 0 and 2 of the first level and particles 6 and 8 of the 3rd level.

- 5 is tethered to the particles 1,2,3,4,7 and 8
- 4 is tethered to the particles 0,1,3,5,6 and 7

The blocking distance between every two tethered particles is: 2*particle_radius – epsilon

The tethering distance between every two tethered particles is: 2*particle_radius + epsilon

Epsilon is a very small value, that helps to have a small difference between the blocking and tethering distance . Consequently, all the particles of the actin stay near to each other when the actin filament moves.

Distance between each two particles is 2 times the particle radius to have this compact shape of the filament.

To set up the motion of the Actin particles inside the membrane, we use random impulses that give every particle a random trajectory. We already have all the particles tethered to each other, which give to the whole actin filament a random motion.

These are the parameters we used to subject actin filaments to random motion:

```
tau_RI_A = 10.0   # average time interval between A-particules random impulses
k_RI_A = 7.0      # shape parameter of A-particules random impulses
v_RI_A = 8.0      # avg velocity of A-particules random impulses
```

Figure 9. Parameters of the random impulse on the actin particles

Each particle is given a random impulse in a random direction at randomized time intervals with an average time interval of tau_RI_A. The k_RI_A and v_RI_A parameters define a gamma distribution from which the magnitude of the impulse is randomly generated.

To put the filament of actin inside the membrane M, we define the following parameters:

- the blocking distance between each sphere of the actin and the big sphere of the membrane M is 0. The actin can move in the center of the membrane without any difficulties.
- the tethering distance between each actin particle and the membrane M is r_M – r_A (r_M : radius of the membrane M , r_A: radius of a sphere of the actin). The spheres of the actin can't go outside the membrane.

But there is an exception:

Actin filament have generally an end that stay tethered to the bottom of the membrane. For this reason and in order to simplify the code and the execution of the simulation, we use an actin filament whose particles have the same species except the particles of the triangle in one end of the actin. These particles need to be different because it will have the property to stick to the membrane.

For that goal we tether the block in the end of the actin to the bottom of the membrane using TPS. The particles of this blocks will have with the membrane the following parameters:

- blocking distance = r_M – r_B – epsilon
- tethering distance = r_M – r_B + epsilon

Here, r_M is the radius of the membrane M, r_B is the radius of of the particles of the end of the actin stuck to the membrane M, and epsilon is a very small value.

As a result, the end of the actin will stay stuck to the perimeter of the membrane M from the inside and will not go far from the perimeter.

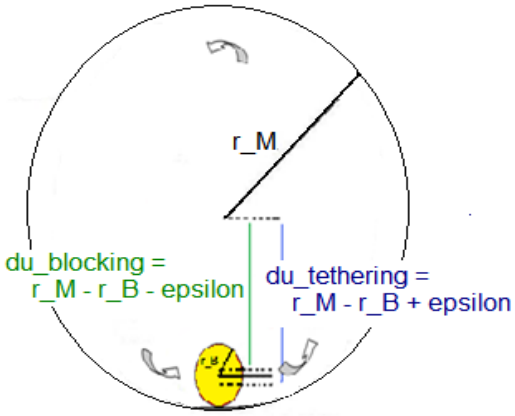

Figure 10. Particles of the bottom of the actin tethered to the plasma of the membrane

As additional information, the actin does not move in the whole perimeter. It stays in the bottom part of the membrane. For this we will use the transparent sphere Z that helps before to determine the zone of the docking sites. The intersection between the sphere Z and the membrane M represent the limits where our actin can go. We know that the particles of the end of the actin are tethered to the membrane M, now they will be tethered to the sphere Z too, respecting these parameters:

- blocking distance = 0
- tethering distance = r_Z – r_B

Here r_Z is the radius of the transparent sphere Z, and r_B is the radius of of the particles of the end of the actin stuck to the membrane M.

The actin is stuck from its end to the membrane M and could not leave the zone delimited by the sphere Z. Consequently, the actin will move only in the zone shown on green in Figure 11.

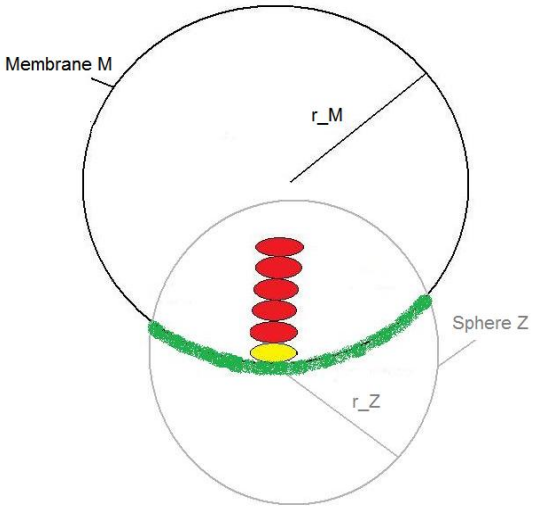

Figure 11. The zone of motion for the actin (green)

The results of the simulation are illustrated in Figure 12.

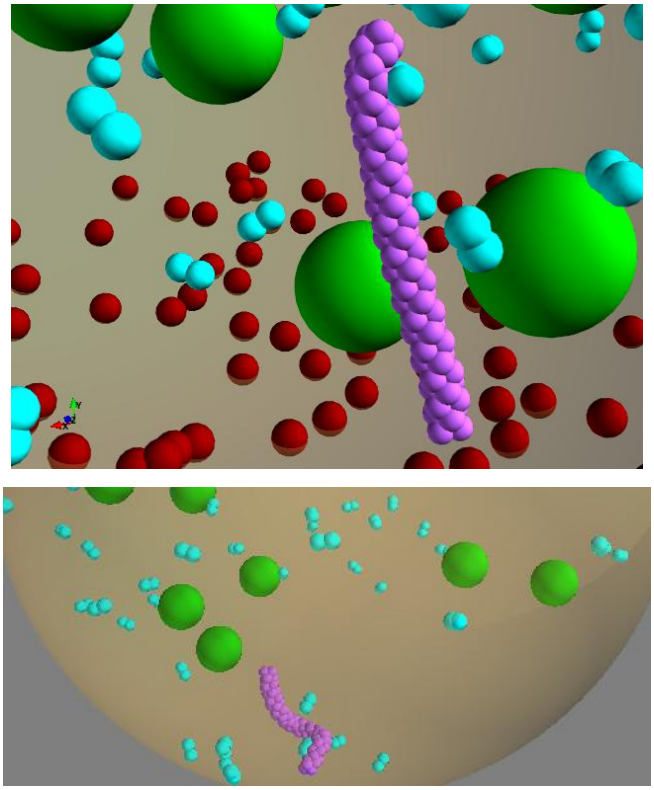

Figure 12. Views of the simulation of the Actin filament inside the membrane

5. SIMULATION OF MULTIPLE ACTIN FILAMENTS BOUND BY SYNAPSIN

In this new model, we focused on the shape of the helix actin filaments; we had to make a real structure with two helix actin filaments turning one around the other. Consequently, we used the same idea of blocks constituted by three particles forming a triangle but in this model, we had in each triangle three types of particles: "A", "B" and "C". The three particles are spheres with the same radius but with different visualization codes:

- We made the particle "B" invisible (we add the opacity = 0).
- We changed the color of the particle "C" into white.
- The particle "A" remained red.

The vector of tethered particles still have the same concept in the first project, as shown below:

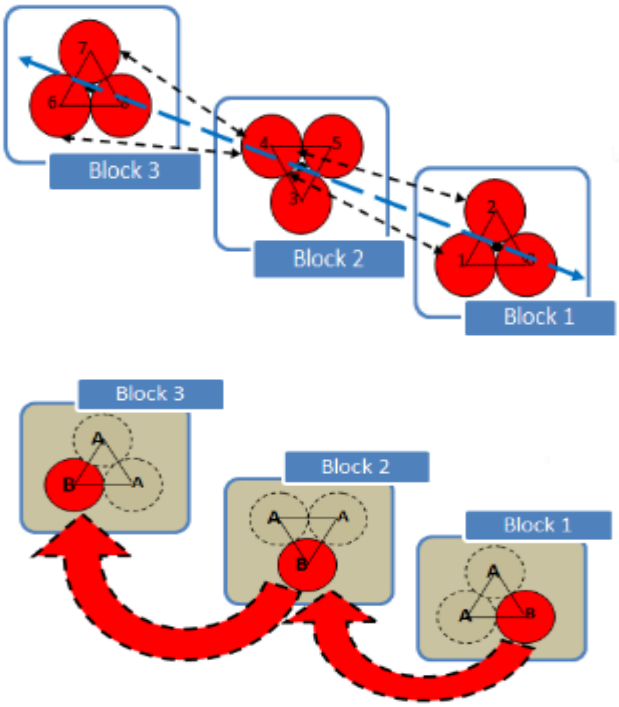

Figure 13. Original positions of the "B" particles

However, what we changed to have the new shape of helix was that the particles "B" would move from one position to the right with every block, and every two block the particle "B" would move from one position to the left. It goes like two positions to the right then back one position to the left (Figure 14).

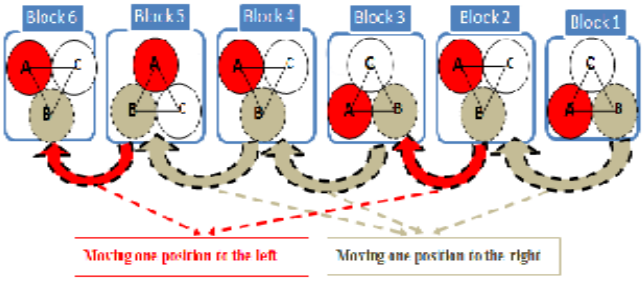

Figure 14. Modeling the positions of the "B" particles

In addition, to make the helix move one around the other we added another particle "D". We gave "D" a velocity to make it move until it hits the helix. On the instant of hitting the particle "D" will transfer some of its energy to the helix and that will make the helix move (Figure 15). The helix will still moving after the hitting seen that all particles have "Delta_u_blocking" and "Delta_u_tethering. For this reason, we adjusted those two parameters:

- The minimum value of the "Delta_u_blocking" between particles A-A, A-B, A-C, A-D, B-B, B-C, C-C, C-D, D-D is:

  "Delta_u_blocking"= 2*Radius - epsilon_A,

- The maximum value of the "Delta_u_tethering" between particles A-A, A-B, A-C, A-D, B-B, B-C, C-C, C-D, D-D is:

"Delta_u_tethering"= 2*Radius + epsilon_A.

The only exception is between particles B-D:

- The "Delta_u_blocking": is 0.0
- And The "Delta_u_tethering" is infty (infinity).

Consequently, the particle "D" can cross through the particle "B" (invisible) till it bits the particles "A" or "C".

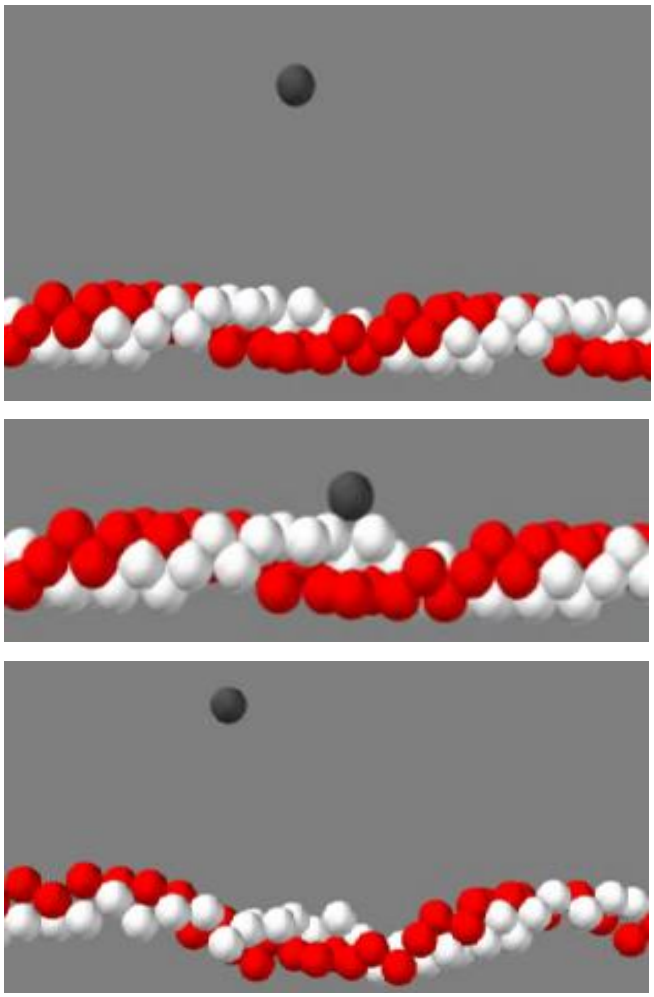

Figure 15. An object (dark "D" particle) impacts an actin filament (red "A" particles + white "C" particles)

The epsilon A is a parameter where we can change its value. In fact, for each time we make it smaller, we can have the shape of the helix straighter when the particle "D" hits the helix and the particles "A", "B", "C" turns one around the other.

To model the multiple helix-shaped actin filaments, we joined two helix actin filaments by a synapsin (Figure 16). This model was put into the membrane and we modeled actin-synapsin bonds during the simulation. Here we changed the dimensions of the actin particles and made the radius equal to 2.5 nm.

A synapsin is formed by two spheres with a radius of 5.0 nm each one, tethered together forming one particle. The two actin filaments are tethered to the synapsin; one tethered to the first sphere of the synapsin and the other one to the second sphere of the synapsin.

In addition, we added random impulses to make the structures move randomly that will make it more natural. Finally, to add it into the membrane, we made a big sphere transparent so we could see the interactions between the particles, where we inserted my two actin filaments connected by a synapsin.

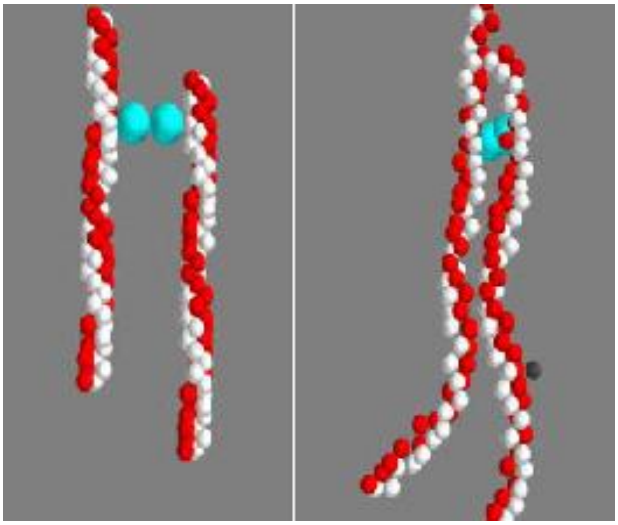

Figure 16. Actin filaments bound by synapsin

We also inserted fifty synapsin moving randomly in the membrane (Figure 17). (We had two different values of parameters for the random impulse: one for the structure and one specially for the fifty synapsin). When the synapsin hits the one of the two actin filaments they should bound on the actin.

Parameters are selected as follows:

- "Delta_u_tethering" between the membrane and any sort of particles should be equal to: radius of the membrane – radius of the particle: R_M - R_P (P can be particle A,B, C or D).
- "Delta_u_blocking" should be equal zero.

- "Delta_u_blocking" between any particle and the synapsin should be equal to: radius of the synapsin radius of the particle : R_S - R_P,
- "Delta_u_tethering between any particle and the synapsin should be equal to: radius of the synapsin + radius of the particle: R_S + R_P (except between the synapsin and the particle B (invisible)
- "Delta_u_blocking"=0.0
- "Delta_u_tethering": infty.

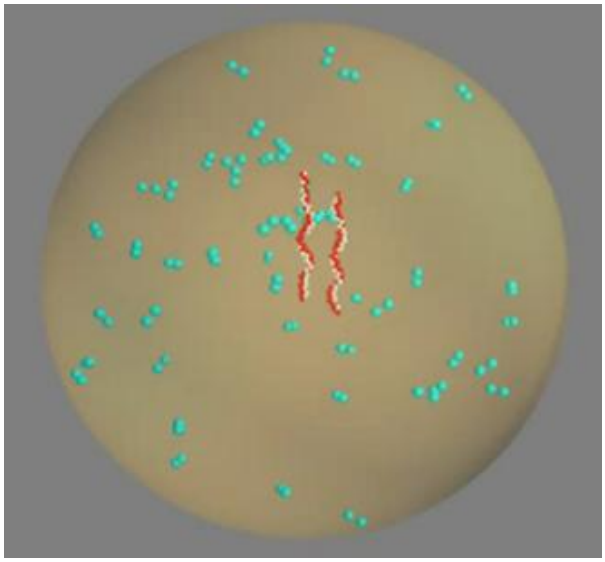

Figure 17. Actin filaments inside a membrane

6. SIMULATION OF MITOCHONDRIA AND WATER PARTICLES

For the simulation of the Mitochondrion, we have tried first to model it using the TPS model by creating what we have called a block of protein tethered together. At the initial state, we can have the picture below.

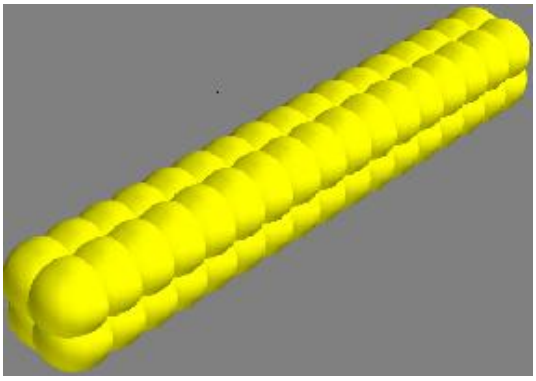

Figure 18. Initial model of the mitochondrion

The concepts used are vector of position, velocity and tethering. By giving to the first vesicle of that picture a positive velocity and to the last vesicle of the same picture a negative velocity, we can obtain a frame like this one.

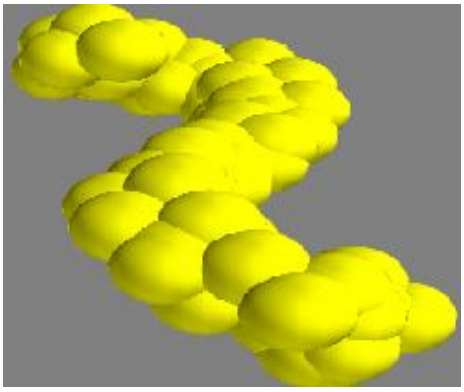

Figure 19. The deformed structure

After having such a picture, what we have decided to do was to reduce the length to make it seem real when putting it inside the membrane. So we have decided to take only the first three blocks tethered together. The results will be shown after the description of the membrane and water particle inside the membrane.

In computer graphics, a variety of techniques have been proposed to model liquids and deformable objects at interactive rates. As important as the plausible animation of these substances is the fast and stable modeling of their interaction.

The idea here was to fill the presynaptic terminal already existing with water and simulate the behavior of the deformable biological structures inside the cell in presence of water.

To model the solid-fluid interaction we use virtual boundary particles. They are placed randomly inside the presynaptic terminal allowing the computation of smooth interactions that yield stable simulations.

For the implementation of the water particles inside the presynaptic terminal already existing, the following steps have been followed:

1. Give the properties of water particle such as radius, random impulse, the color, the mass, and the shape.
2. Specify the relation between the water particles with other structures inside the presynaptic terminal such as tethering distance, blocking distance the rebound , the retract.
3. Add those characteristics inside the code at the right place in order to keep the structure of the existing code.
4. Take in consideration the computer characteristics while defining the number of water particles because to run this simulation it takes long time.
5. Remember that all particles should be tethered to the membrane in order to keep them within the compartment.

After all these steps have been followed, the simulation can be launched and stopped before some time to see if the results are satisfying or not. If not we can modify some parameters. We have to make sure before leaving the simulation running for long periods of time that the preview result was satisfying in order to avoid the lost of time.

The snapshot below shows the mitochondrion surrounded by water particles in a presynaptic terminal.

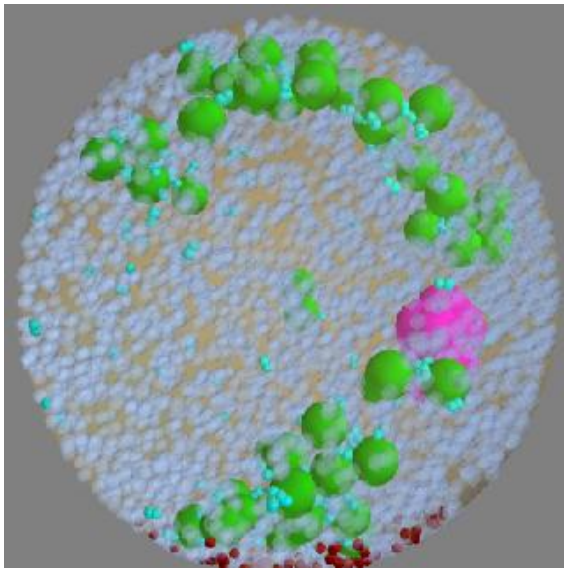

Figure 20. Simulation of a Mitochondrion and water particles in a presynaptic terminal

7. IMPLEMENTATION

We implemented all code in the Python programming language. Typically interpreted, Python is known more for its aesthetic syntax and convenient data structures than for its performance. The language satisfied our need, however, to develop a proof of concept.

Figure 21 shows the Python code used to specify a single level of an actin filament, modeled with three particles arranged in an equilateral triangle.

```
Psi[1] = {"spc": "B", \
            "u": [r_A,-1.5*r_M,0.0],\
            "v": [0.0,10.0,0.0],\
         "tethered":[0,2,3,4,5,79]}
Psi[2] = {"spc": "B", \
            "u": [- r_A,-1.5*r_M,0.0], \
            "v": [0.0,10.0,0.0],\
        "tethered":[0,1,3,5,6,79]}
Psi[3] = {"spc": "B", \
          "u": [0.0, -1.5*r_M,sqrt(3)*(r_A)], \
          "v": [0.0,10.0,0.0],\
         "tethered":[0,1,2,4,6,79]}
```

Figure 21. Sample Python code

8. CONCLUSION

We have demonstrated how TPS can be used to model various deformable biological structures. The techniques used to model actin filaments could be applied to other relatively long and narrow biological entities, whereas the basic arrangement of particles adopted for the mitochondrion model could be used for other relatively round biological objects. A set of inert particles, like the water particles presented in this paper, could be used to represent a variety of different small objects that impede the motion of other structures. Future work includes refining various model parameters, improving the performance of the simulation code, and integrating all the structures modeled in this paper into a single realistic model of a presynaptic terminal.

REFERENCES

1. Goldstein, R., G. Wainer. 2009, "Simulation of Deformable Biological Structures with a Tethered Particle System Model." In Proceedings of the 32nd Canadian Medical and Biological Engineering Conference, CMBEC (Calgary, AB, Canada).
2. Goldstein, R., G. Wainer. 2009, "Simulation of a Presynaptic Nerve Terminal with a Tethered Particle System Model." In Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC (Minneapolis, MN, USA).
3. Benfenati, F., F. Valtorta, P. Greengard, 1991, "Computer Modelling of Synapsin I Binding to Synaptic Vesicles and F-actin: Implications for Regulation of Neurotransmitter Release," Proceedings of the National Academy of Sciences, USA.; 88(2): 575–579.
4. De Camilli, P., 1995, Keeping synapses up to speed, Nature, 375: 450–451.
5. Gillespie, D. T., 1977, "Exact Stochastic Simulation of Coupled Chemical Reactions," Journal of Physical Chemistry, 81(25): 2340–2361.
6. Boulianne, L., S. Al Assaad, M. Dumontier, W. Gross, 2008, "Grid-Cell: a stochastic particle-based biological system simulator," BMC Systems Biology, 2(1): 66–74.
7. Stiles, J. R.; T. M. Bartol, 2001, "Monte Carlo Methods for Simulating Realistic Synaptic Microphysiology Using Mcell," In Computational Neuroscience: Realistic Modeling for Experimentalists (Edited by Erik De Schutter; published by CRC Press), 87–127.
8. Coggan, J. S., T. M. Bartol, E. Esquenazi, J. R. Stiles, S. Lamont, M. E. Martone, D. K. Berg, M. H. Ellisman, T. J. Sejnowski, 2005, "Evidence for Ectopic Neurotransmission at a Neuronal Synapse," Science, 309(5733): 446–451.