Impulse-Based Dynamic Simulation of Deformable Biological Structures

Rhys Goldstein and Gabriel Wainer

Carleton University, Ottawa ON K1S5B6, Canada

Abstract. We present a new impulse-based method, called the Tethered Particle System (TPS), for the dynamic simulation of deformable biological structures. The TPS is unusual in that it may capture a gradual process of deformation using only instantaneous impulses that occur in response to particle collisions. This paper describes the method and its application to synaptic vesicle clusters and deformable biological membranes. Unlike many alternative methods, which require solutions to systems of equations or inequalities, the calculations in a TPS simulation are all analytic. The TPS also alleviates the need to choose regular time intervals appropriate for biological entities that may differ in size by orders of magnitude. The method is promising for simulations of small-scale self-assembling deformable biological structures exhibiting random motion.

1 Introduction

Simulation is becoming an increasingly common tool among biologists and medical researchers, complementing traditional experimental techniques. As Kitano explains in [11], experimental data is first used to form a hypothesis, and that hypothesis may be investigated with a simulation. Predictions made by the simulation can then be tested using *in vitro* and *in vivo* studies, and the new experimental data may lead to new hypotheses. This iterative process can be applied to basic research on biological systems, as well the development of drugs and other treatments.

Modeling and simulation methods that capture the dynamics of deformable biological structures are frequently targeted at surgical planning and training [3], as well as the analysis of prosthetics [8]. Models of smaller-scale deformable biological structures are rarer, but examples include the simulated deformation of 8-μm red blood cells [19], and that of membrane-sculpting proteins on the 10-nm scale [12].

The most common methods for simulating the dynamics of deformable structures are mass-spring-damper systems and the finite element method [7]. Our method, the Tethered Particle System (TPS), differs in that it uses only impulses to alter motion. Impulse-based methods have previously been used to simulate rigid bodies, but are generally neglected or considered unsuitable for objects that deform. It is counterintuitive to model deformable structures with impulses, as

C. Priami et al. (Eds.): Trans. on Comput. Syst. Biol. XIII, LNBI 6575, pp. 39–60, 2011.

impulses are instantaneous whereas the deformation of an object is a continuous process that may require a significant length of time. Nevertheless, if one represents a deformable structure as a network of a large number of particles, then numerous collisions and impulses between those particles may produce the effect of a gradual deformation of the overall structure.

We demonstrate that impulse-based methods provide a relatively simple way to allow deformable biological structures to assemble themselves from rigid particles representing proteins and other biological entities. Also, with an impulse-based simulation, it is easy to incorporate the random motion exhibited by these small biological objects. The TPS proved useful for the simulation of deformable vesicle clusters in a presynaptic nerve terminal, which form from interactions between randomly-moving synaptic vesicles and synapsin protein.

Section 2 provides an overview of existing dynamic simulation methods for both rigid bodies and deformable structures. Section 3 describes the TPS method in detail, and Section 4 presents TPS models of synaptic vesicle clusters and various deformable membranes. Strengths and weaknesses of the new method are discussed in Section 5.

2 Dynamic Simulation Methods

We use the phrase "dynamic simulation" to indicate the simulation of motion using laws of classical dynamics. Here we describe several pre-existing dynamic simulation methods, some intended for rigid bodies and others designed for deformable structures.

2.1 Dynamic Simulation of Rigid Bodies

This section reviews methods for the dynamic simulation of rigid bodies. In these methods, object deformation may be represented by a loss of kinetic energy, for example, or an overlapping of objects. If an object's changing shape is modeled, however, then we classify it as a deformable structure instead of a rigid body.

"Impulse-based" methods are perhaps the most obvious approach to the dynamic simulation of rigid bodies. An impulse-based method involves two tasks: collision detection (the task of calculating the time at which any two objects come into contact) and collision response (the task of computing the new trajectories of two colliding objects). In response to a collision, the trajectory of an object changes instantaneously in simulated time. The instantaneous change in the momentum of the object is referred to as an "impulse" [13].

Because impulse-based methods assume instantaneous contacts, the approach seems inappropriate for the modeling of stable contacts. If a ball is rolling across a table, for example, it remains in contact with the table for a length of time. In his 1996 Ph.D. thesis, Brian Mirtich demonstrated that stable contacts could be modeled as sequences of independent collisions [14]. Consider an impulse-based simulation of an object bouncing along a horizontal surface. Provided each bounce was sufficiently short in height and duration, the model could accurately represent a ball rolling across a table.

Another well-known drawback to impulse-based methods is possibility of simultaneous or nearly-simultaneous collisions. Consider a situation in which a small object is directly between two much larger approaching objects. After the first large object hits it, the small object may end up travelling back and forth between the larger objects in a long sequence of nearly-simultaneous collisions. This might require considerable computational effort. If kinetic energy is lost in each collision, then it is possible for the sequence of collisions to become infinite, slowing the simulation to a halt.

One advantage to impulse-based simulation is the simplicity of the method. If one is to implement an algorithm to detect collisions between pairs of objects, which is necessary in perhaps all of the competing methods, it is a simple matter to apply the law of conservation of momentum to give the two objects new trajectories. The constraint-based method of [1], by contrast, may compute new trajectories for more than two objects simultaneously. This is done by minimizing a linear function constrained by a system of linear inequalities. Depending on the model, this problem may be NP-hard, may have no solutions, or may have multiple different solutions.

Both impulse-based methods and the constraint-based method of [1] prevent the penetration of objects. "Penalty methods" differ in that they allow approaching objects to overlap slightly upon colliding. Typically, a spring is temporarily inserted between colliding objects; the more the objects overlap, the stronger the restoring force of the compressed spring [15].

2.2 Dynamic Simulation of Deformable Structures

We now review methods for the dynamic simulation of deformable structures, as opposed to rigid bodies. Note that phrase "dynamic simulation" excludes a wide range of methods for modeling deformable structures. An algorithm that fits a spline to a cross-sectional image of a human lung, for example, certainly models a deformable structure. But unless the motion of the lung is predicted from laws of physics, we would not consider it a dynamic simulation.

One way to model a deformable structure is with a set of point masses. Each mass is connected to its neighbors with a spring and possibly a damper. A spring applies a force that, depending on its present length, attracts or repels the masses on either end. A damper applies a force that decreases the relative speed of the masses on either end. These "mass-spring-damper" systems can be used to simulate the dynamics of deformable structures by predicting the acceleration of each mass, at regular time intervals, according to spring, damper, and external forces [16]. The mass-spring-damper method is essentially a penalty method like those described in Section 2.1 for rigid bodies. The difference is that the springs in Section 2.1 are inserted temporarily between detached colliding objects, whereas in this case the springs tend to be permanent and the point masses do not necessarily represent distinct objects.

Some mass-spring-damper models use spherical particles instead of point masses. This technique is used in [5] to simplify the detection of collisions between deformable objects. Each object is composed of several overlapping

spherical particles. Instead of detecting collisions between the possibly-concave surfaces of these objects, only collisions between particles are considered. A similar approach is taken in [9], which incorporates friction, viscous forces, and the fracture of deformable objects.

Mass-spring-damper methods are used extensively in computer graphics. They are considered computationally efficient, but not particularly accurate. Incompressible deformable objects and nearly-rigid thin membranes are difficult to model, and appropriate spring parameters may be difficult to determine. Stiff objects, modeled using springs with large restoring forces, threaten the stability of mass-spring-damper simulations. Techniques have been developed to address the stiffness problem. The simplest solution is to decrease the time step, though this increases computational costs.

A popular alternative to mass-spring-damper systems is the "finite element method" (FEM) [7]. FEM actually refers to a more general mathematical technique, but we will refer to it as a dynamic simulation method for deformable structures. In an FEM model, a deformable object is represented as a set of adjacent polyhedra. Each polyhedron, or "element", has a set of vertices, or "nodes". Although recorded attributes are associated with each node, material properties can be obtained at every point in each element by interpolating the attributes of each node. Positions of each node may change at each time step. FEM simulations are considered to be more accurate, but also more computationally intensive, than those based on mass-spring-damper models. The FEM is most efficient with metals and materials that exhibit relatively little deformation. Highly deformable materials, like soft biological tissues, require frequent re-calculation of large mass and stiffness matrices that depend on the positions of the nodes.

Impulse-based methods, like those used for rigid bodies, tend to be either neglected or avoided for the dynamic simulation of deformable structures. A literature search revealed an "impulse response deformation model" [18], which does simulate the dynamics of deformable structures, but is not an impulse-based method despite its name. In this case the term "impulse" refers to an initial perturbation in an object's shape. Convolution integrals are used to track the object's shape after the perturbation.

The possibility of applying impulse-based methods to deformable objects is acknowledged in [10], but quickly dismissed with the assertion that "impulse-based methods assume short contacts only, and therefore they are not suitable for soft objects". The argument is intuitive: impulses are instantaneous changes in momentum, whereas the deformation of an object is a gradual process that takes place over time. In [14], Mirtich states that the strongest restriction of impulse-based methods is that models are comprised of only rigid bodies.

The pre-existing method that most closely fits the phrase "impulse-based dynamic simulation of deformable structures" was developed recently to simulate inextensible cloth [2], as well as volume-conserving deformable objects [6]. In both cases, impulses are applied simultaneously to all particles in a structure at regular intervals. Because the purpose of these impulses is to constrain either

the distances or volumes between the particles, the method can be classified as constraint-based as well as impulse-based. The simulations of [2] and [6] differ from impulse-based rigid body simulations in that, in the latter, impulses occur in response to collisions and not at regular intervals.

3 Tethered Particle System Method

A TPS model tracks the positions and velocities of numerous particles, each with a fixed mass, that interact with one another via collisions. The TPS method is unusual in that it is an impulse-based method, meaning that any change in a particle's velocity is instantaneous, yet the method is designed for representing structures that deform over a length of time. The key idea is that a deformable structure may be represented by a group of particles; even though each individual particle changes velocity in a sequence of instantaneous impulses, the configuration of the particles in the group changes in a seemingly gradual process over time. This section provides a detailed description of the TPS method, including key equations.

3.1 Blocking and Tethering Collisions

In order for a group of particles to exhibit any structure at all, the distances between certain pairs of particles in the group must be regulated or restricted. In a TPS model, the distance between a pair of particles is constrained by two types of collisions: "blocking collisions" and "tethering collisions".

What we refer to as a "blocking collision" is what one normally associates with the word "collision". As illustrated in Figure 1a, a blocking collision occurs when two approaching particles reach an inner limiting distance. This "blocking distance" is represented by $\Delta u_{blocking}$.

Note that although we will frequently depict a particle as a circle or sphere of some radius, neither the shape nor the size of a particle is explicitly defined in a TPS model. The particles in Figure 1 could have been drawn as larger circles, for instance, perhaps overlapping with one another at the time of collision.

The particles in Figure 1a are shown rebounding at a similar angle to that at which they had been approaching. This indicates that the collision is elastic, meaning that no kinetic energy is lost. When real-world objects collide, they deform and absorb kinetic energy. Although individual particles in a TPS have no shape and do not explicitly deform, we may wish to account for energy loss in particle collisions. The loss of kinetic energy in the inelastic collision of Figure 1b causes the particles to rebound at a smaller angle.

Energy loss due to collisions is generally modeled with a parameter called the "coefficient of restitution", which expresses the ratio of the post-collision relative speed of two particles to the pre-collision relative speed [1]. In a TPS model, different coefficients are used for different types of collisions. In a blocking collision, the coefficient of restitution is referred to as the "rebounding coefficient", and is represented by $c_{rebound}$. We will assume $0 \leq c_{rebound} \leq 1$, with $c_{rebound} = 0$

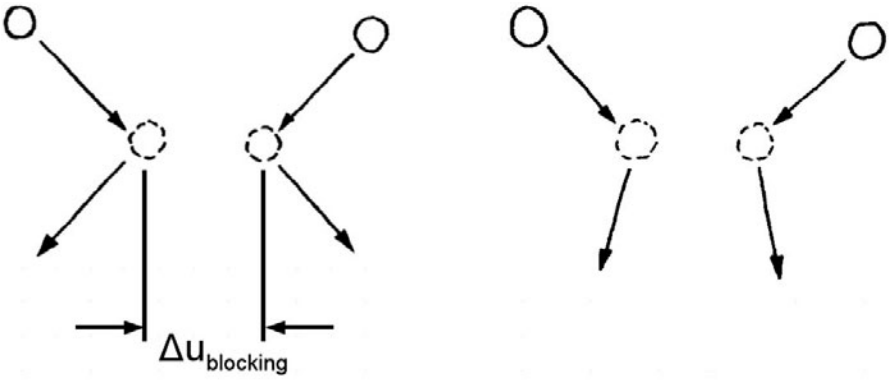


Fig. 1. Illustrations of blocking collisions, in which pairs of approaching particles reach the blocking distance $\Delta u_{blocking}$ and rebound. In (a) the collision is elastic, whereas (b) depicts an inelastic collision.

indicating maximum energy loss, and $c_{rebound} = 1$ indicating a perfectly elastic collision.

Any two specific particles may be tethered together at the start of a simulation. Also, when a blocking collision occurs between two particles, they may become tethered. If two particles are tethered and moving away from one another, and if they reach the "tethering distance" $\Delta u_{tethering}$, then one of two things may happen. The particles may become untethered, in which case they continue moving away from one another. Otherwise the particles remain tethered, undergo a tethering collision, and retract inwards. The phrase "tethering collision" is unintuitive because one normally expects a collision to occur only between approaching objects. We use the word "collision" for separating particles as well so that we can apply the phrases "collision detection" and "collision response" to either type of particle-particle interaction.

A tethering collision may be envisioned as a situation in which a cord has completely unravelled, and therefore delivers an inward impulse to the particles attached to it on either end. Such a cord is illustrated in Figure 2a. Initially, the cord is slack (solid line). As the particles move apart, the cord unravels and eventually becomes taut (dotted lines). At that point, the particles change direction and move inwards. Unlike a spring in a mass-spring-damper model, a cord in a TPS model has no effect on either particle before reaching its maximum length.

Figure 2a is meant to portray an elastic tethering collision, as the particles end up approaching one another at the same relative angle at which that had been separating. Figure 2b, by contrast, illustrates an inelastic tethering collision in which energy is lost. The particles end up approaching at a smaller angle. To address energy loss in tethering collisions, we introduce another type of coefficient of restitution. We refer to it as the "retraction coefficient", and represent it with $c_{retract}$ satisfying $0 \leq c_{retract} \leq 1$.

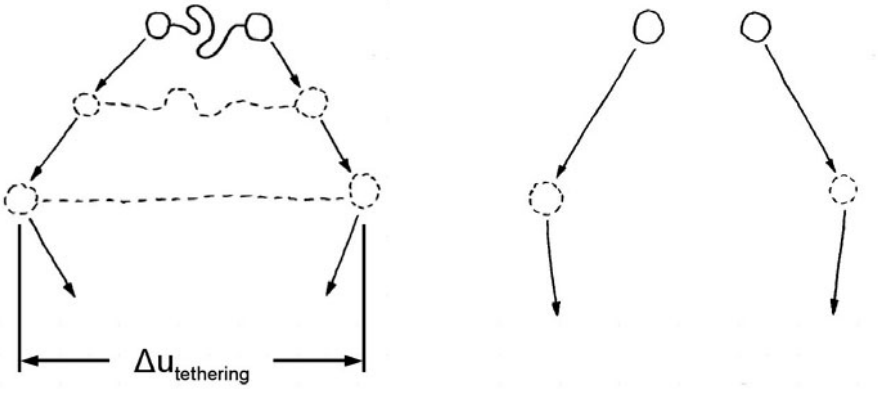


Fig. 2. Illustrations of tethering collisions, in which pairs of separating tethered particles reach the tethering distance $\Delta u_{tethering}$ and retract. In (a) the tethering collision is elastic, whereas (b) depicts an inelastic collision.

3.2 Basic Simulation Procedure

Suppose we have two particles, A and B. Their masses are m_A and m_B respectively, their current positions are $\boldsymbol{u}_A$ and $\boldsymbol{u}_B$, and their velocities are $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$. The distance Δu between the particles can be expressed as a function of the time Δt.

$$\Delta u = \sqrt{\sum \left(((\boldsymbol{u}_B + \boldsymbol{v}_B \cdot \Delta t) - (\boldsymbol{u}_A + \boldsymbol{v}_A \cdot \Delta t))^2 \right)} \tag{1}$$

The basic procedure in a TPS simulation is to repeatedly solve (1) for Δt for all pairs of relatively close particles. Solving (1) with $\Delta u = \Delta u_{blocking}$ yields the time remaining before a blocking collision, whereas $\Delta u = \Delta u_{tethering}$ gives the time of a tethering collision. Time is advanced by the smallest calculated value of Δt, the time remaining before the next collision. When that collision occurs, the new velocities of the two particles involved are calculated from (2), and the process repeats.

$$\begin{aligned} \boldsymbol{v}_A{}' &= \boldsymbol{v}_A + \frac{\Delta \boldsymbol{p}}{m_A} \\ \boldsymbol{v}_B{}' &= \boldsymbol{v}_B - \frac{\Delta \boldsymbol{p}}{m_B} \end{aligned} \tag{2}$$

The vector $\Delta \boldsymbol{p}$ above is the impulse, the change in momentum of particle A as a result of the collision. To obtain its value, it is useful to calculate the following vectors. Note that $\boldsymbol{v}_{\hat{u}}$ is the relative velocity of the particles projected onto the axis between them.

$$\begin{aligned} \hat{u} &= \frac{\hat{u}_B - \hat{u}_A}{\sqrt{\sum (\hat{u}_B - \hat{u}_A)^2}} \\ \boldsymbol{v}_{AB} &= \boldsymbol{v}_B - \boldsymbol{v}_A \\ \boldsymbol{v}_{\hat{u}} &= \sum (\boldsymbol{v}_{AB} \cdot \hat{u}) \cdot \hat{u} \end{aligned}$$

If the particles rebound or retract, then $\Delta \boldsymbol{p}$ can be calculated from (3) below with $c_{restitute}$ being either $c_{rebound}$ or $c_{retract}$.

$$\Delta \boldsymbol{p} = \left(\frac{1}{m_A} + \frac{1}{m_B}\right)^{-1} \cdot (1 + c_{restitute}) \cdot \boldsymbol{v}_{\hat{u}} \tag{3}$$

The actual computations performed are complicated by the possibility of revolution, simultaneous and nearly-simultaneous collisions, and random impulses. These concepts are described in Sections 3.3, 3.4, and 3.6. The TPS remains simpler than most deformable structure simulation methods in that all unknown variables can be calculated analytically from explicit formulas. There are no systems of equations or inequalities that need to be solved simultaneously or iteratively.

3.3 Revolution

Note that it is the tethering collisions that distinguish the TPS from more traditional particle collision algorithms. They place potentially-useful outer limits on the distances between certain pairs of particles, but introduce a performance problem that must be addressed. The problem is illustrated in Figure 3. Particle A remains stationary because it has an infinite mass, whereas particle B is in motion with a finite mass. The two particles remain tethered, and undergo a sequence of elastic tethering collisions. Immediately after each collision, particle B approaches A at a relatively small angle θ.

Note that at any stage in a TPS simulation, time is advanced by an irregular interval to that of the next collision. The greater the frequency with which collisions occur, the slower the simulation progresses. The problem with the above scenario is that, if θ is small, the tethering collisions become extremely

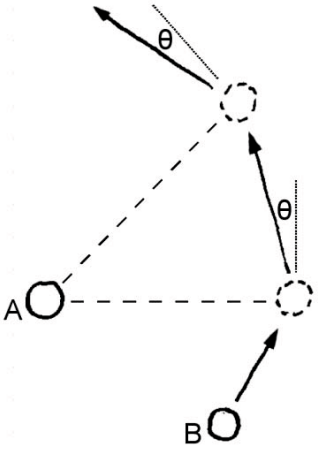


Fig. 3. A scenario in which particle B revolves around A in a sequence of tethering collisions

frequent and the simulation may become impractically slow. Worse, if $\theta = 0$, then time cannot be advanced at all without violating the constraint of $\Delta u_{tethering}$ on the distance between A and B. The problem still exists if the mass of A is finite, and is exacerbated by small values of $c_{retract}$.

To address the problem, we place a lower limit $\theta_{revolve}$ on the angle at which particles can approach after a tethering collision. In a collision where this restriction takes effect, we say that the particles "revolve" instead of "retract". We also introduce a "revolution coefficient" $c_{revolve}$ that expresses the ratio of the new relative velocity to the old one after one complete revolution of the particles, allowing energy to be lost. We require $0 \leq c_{revolve} \leq 1$.

Calculations pertaining to revolution require us to obtain $\boldsymbol{v}_{\hat{w}}$, the component of the relative velocity perpendicular to the axis between them.

$$\boldsymbol{v}_{\hat{w}} = \boldsymbol{v}_{AB} - \boldsymbol{v}_{\hat{u}}$$

If the condition in (4) is satisfied, then the retraction impulse calculated from (3) is sufficient.

$$\sqrt{\sum \left(\left(c_{retract} \cdot \boldsymbol{v}_{\hat{u}} \right)^2 \right)} > tan\left(\theta_{revolve}\right) \cdot \sqrt{\sum \left(\boldsymbol{v}_{\hat{w}}{}^2 \right)} \tag{4}$$

If (4) is not satisfied, we abandon (3) and use the more general equation in (5).

$$\Delta \boldsymbol{p} = \left(\frac{1}{m_A} + \frac{1}{m_B} \right)^{-1} \cdot (\boldsymbol{v}_{AB} - \boldsymbol{v}_{AB}{}') \tag{5}$$

Here $\boldsymbol{v}_{AB}{}'$, the post-collision relative velocity, is obtained as follows.

$$\begin{aligned}
\boldsymbol{v}_{revolve} &= \boldsymbol{v}_{\hat{w}} - tan\left(\theta_{revolve}\right) \cdot \sqrt{\sum \left(\boldsymbol{v}_{\hat{w}}{}^2 \right)} \cdot \hat{u} \\
\hat{u}_{revolve} &= \frac{\boldsymbol{v}_{revolve}}{\sqrt{\sum \left(\boldsymbol{v}_{revolve} \right)^2}} \\
\boldsymbol{v}_{AB}{}' &= c_{revolve}{}^{\frac{\theta_{revolve}}{\pi}} \cdot \sqrt{\sum \left(\boldsymbol{v}_{AB}{}^2 \right)} \cdot \hat{u}_{revolve}
\end{aligned}$$

If $\theta_{revolve} = \frac{\pi}{n}$, then it takes n collisions for the two particles to achieve a complete revolution, and the relative speed decreases by a factor of $c_{revolve}{}^{\frac{1}{n}}$ after each collision.

3.4 Loading and Restitution

It is widely known that simultaneous and nearly-simultaneous collisions threaten the efficiency of impulse-based methods, potentially slowing simulations to a halt. We now describe this problem followed by our solution.

Consider the scenario in Figure 4. Assume particles A, B, and C are all of the same mass, and that collisions are elastic. At time t_{AB}, moving particle A collides with stationary particle B, transferring all of its momentum. Then at time t_{BC}, particle B collides with stationary C and transfers all of its momentum.

An efficiency problem arises if the mass of particle B is reduced to a fraction of that of A and C, as illustrated in Figure 5. At time t_{AB_0}, particle A transfers

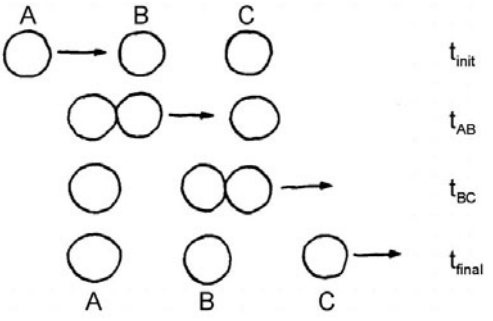


Fig. 4. A scenario in which 2 collisions occur between three particles of equal mass

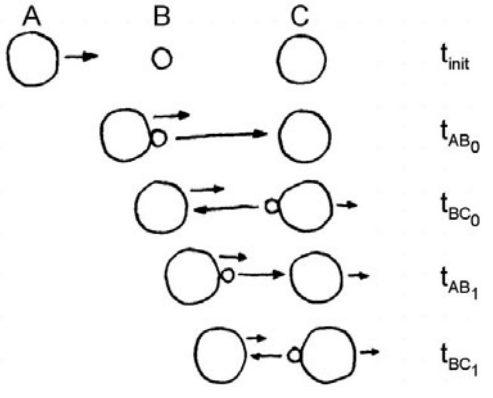


Fig. 5. The same scenario as in Figure 4, but the center particle is now less massive. Numerous nearly-simultaneous collisions result.

only some of its momentum to B. Particle B reaches C and rebounds at t_{BC_0}, then meets A again at time t_{AB_1}. Because only a small amount of momentum is transferred in each collision, particle B must rebound back and forth in a sequence of nearly-simultaneous collisions. If the mass of B is one thousandth that of A and C, roughly 70 elastic collisions occur before enough momentum has been transferred to separate all three particles.

The processing of 70 collisions is in itself a significant computational cost for such a simple scenario, but there are many situations in which the simulation will halt completely. When a simulation was performed with the Figure 5 scenario and a rebounding coefficient of 0.9, the momentum transferred on each collision eventually rounded to zero and the simulation stalled.

We propose a novel approximation that addresses the threat of simultaneous and nearly-simultaneous collisions. The idea is to separate each collision into a loading phase and a restitution phase, and to allow restitution to take place at a later time. When particles collide (the loading phase), they form loaded groups. A "loaded group" acts as a single body with the combined mass of all the particles

in the group. A restitution delay time $\Delta t_{restitute}$ is introduced, after which the loaded particles separate (the restitution phase). Loaded particles may remain together longer than $\Delta t_{restitute}$ if necessary to ensure that the order in which particles separate is opposite that in which they loaded. Figure 6 illustrates the approximation.

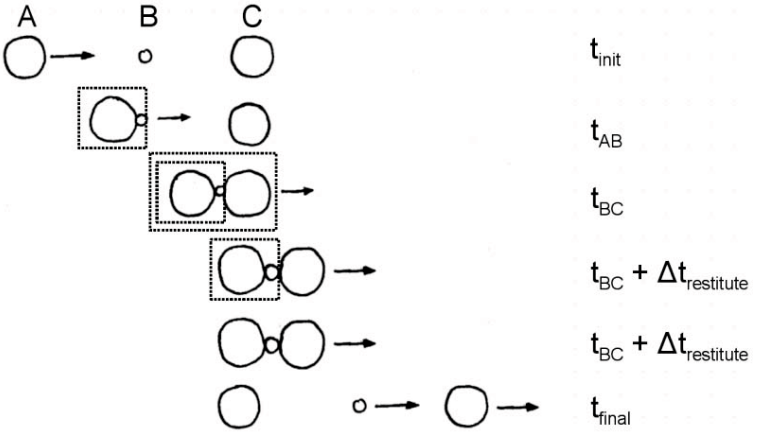


Fig. 6. A scenario demonstrating an approximation that addresses the problem of simultaneous and nearly-simultaneous collisions. Particles form loaded groups for durations of $\Delta t_{restitute}$, during which time they act as single bodies.

At time t_{AB} in Figure 6, particles A and B collide, form a loaded group, and proceed with matching velocities. Suppose that this loaded group did not encounter any other particle. In that case, at time $t_{AB} + \Delta t_{restitute}$, particles A and B would separate or "restitute". But that does not happen, as at time t_{BC} while A and B are still loaded, they encounter particle C. The impulse delivered to C depends not on the mass of B, but rather on the mass of A and B added together. It is the temporary accumulation of mass that tends to increase the momentum transferred per collision, and thus reduces the number of collisions.

It is necessary that particles in a loaded group restitute in the opposite order from that in which they loaded. After all three particles form a loaded group at t_{BC}, particles A and B may no longer separate at time $t_{AB} + \Delta t_{restitute}$. The loaded group remains intact until $t_{BC} + \Delta t_{restitute}$ instead, at which point particle B separates from C. The result of the B-C restitution is calculated with the masses of A and B still combined. After the B-C restitution is complete, but also at the simulated time $t_{BC} + \Delta t_{restitute}$, particles A and B finally separate.

Calculations pertaining to loading and restitution are relatively simple. Suppose particle A, in a loaded group with mass M_A, collides with particle B, in a loaded group with mass M_B. After loading, which takes place immediately, the new velocity of every particle in both groups is $\boldsymbol{v}_{load}$.

$$\boldsymbol{v}_{load} = \left(1 + \frac{M_B}{M_A}\right)^{-1} \cdot \boldsymbol{v}_A + \left(1 + \frac{M_A}{M_B}\right)^{-1} \cdot \boldsymbol{v}_B$$

At a later time, when the two loaded groups separate in the restitution phase of the collision, an impulse $\Delta \boldsymbol{p}_{AB}$ is applied between the groups. It value is obtained by taking the impulse calculated from either (3) or (5), and subtracting the impulse that was effectively applied when the velocities where changed to $\boldsymbol{v}_{load}$.

$$\Delta \boldsymbol{p}_{AB} = \Delta \boldsymbol{p} - \left(\frac{1}{M_A} + \frac{1}{M_B}\right)^{-1} \cdot \boldsymbol{v}_{AB}$$

The proposed approximation can dramatically reduce the number of collisions in a simulation, even if $\Delta t_{restitute}$ is very small. For the scenario involving three particles in a line, with the outside two particles being a thousand times more massive than the middle particle, the approximation reduced 70 elastic collisions to only four. If $\Delta t_{restitute} = 0$, loading and restitution occur back-to-back and the approximation is effectively canceled.

3.5 External Impulses

The impulses described in Sections 3.1 through 3.4 above arise from interactions between at least two particles. The novelty of the TPS method lies in the fact that these impulses alone are capable of predicting processes of deformation. However, if the particles in a TPS model are influenced by no other factor whatsoever, then one is limited to simulating objects moving deterministically in a gravity-free vacuum. In order to capture Brownian motion, drag forces, and other effects influencing the motion of biological objects, it is necessary to introduce external impulses into a TPS model. An "external impulse" is an instantaneous change in momentum that may be applied to a single particle any point in time during a simulation.

Different methods and models may be used to calculate the timing, the directions, and the magnitudes of external impulses. A realistic TPS model of a biological system may combine external impulses of many different types. We recommend that modelers at least incorporate a type of external impulse we refer to as a "random impulse": a momentum change of randomized magnitude and randomized direction applied to a particle each time a randomized time interval expires. From a practical perspective, random impulses prevent the kinetic energy in a TPS model from converging to zero due to the energy losses associated with particle collisions. From a physical perspective, random impulses may represent Brownian motion, variability in electric potential fields or fluid pressure, or interactions with otherwise unrepresented biological entities.

An external force can be represented by a sequence of external impulses. To incorporate a gravitational acceleration of g, for example, one may apply downward impulses of magnitude $m_A \cdot g \cdot \Delta t_g$ to each particle of mass m_A at regular time intervals of Δt_g. One can also use external impulses to model the drag force exerted on a particle by the surrounding fluid. Suppose that a particle of radius r_A and velocity $\boldsymbol{v}_A$ is immersed in a fluid with a dynamic viscosity of μ_{fluid} and a velocity of $\boldsymbol{v}_{fluid}$. One could approxiate the drag force on the particle using Stoke's law, and apply external impulses of $6 \cdot \pi \cdot \mu_{fluid} \cdot r_A \cdot (\boldsymbol{v}_{fluid} - \boldsymbol{v}_A) \cdot \Delta t_d$ at

time intervals of Δt_d. In small-scale biological models, it would in many instances be reasonable to assume $\boldsymbol{v}_{fluid}$ to be zero.

When an external impulse $\boldsymbol{\Delta p}$ is applied to particle of mass m_A and velocity $\boldsymbol{v}_A$, the new velocity $\boldsymbol{v}_A{}'$ is calculated as follows.

$$\boldsymbol{v}_A{}' = \boldsymbol{v}_A + \frac{\boldsymbol{\Delta p}}{m_A}$$

For the sake of simplicity, we recommend that each external impulse be associated with a single particle, and that $\boldsymbol{\Delta p}$ be calculated without accounting for surrounding particles that may be temporarily in the same loaded group (see Section 3.4). Once $\boldsymbol{\Delta p}$ is calculated, then if the particle happens to be in a loaded group of mass M_A, the velocity of every particle in the group is changed from $\boldsymbol{v}_{load}$ to $\boldsymbol{v}_{load}{}'$.

$$\boldsymbol{v}_{load}{}' = \boldsymbol{v}_{load} + \frac{\boldsymbol{\Delta p}}{M_A}$$

3.6 Particle Species

In a TPS model, it is useful to define several distinct species of particles. Certain properties are chosen for each species individually, and certain properties are associated with each combination of two species.

Associated with each species is the mass m of every particle of that species. We also include several parameters pertaining to the random impulses described in Section 3.5. For each species that we wish to exhibit random motion, we select an average time interval τ_{RI} between random impulses. The actual intervals are sampled from an exponential distribution during the simulation. We obtain the magnitude of each impulse from a gamma distribution, which requires a shape parameter k_{RI} and an average momentum value μ_{RI}.

There are five parameters associated with each pair of species: the blocking distance $\Delta u_{blocking}$, the tethering distance $\Delta u_{tethering}$, and the coefficients $c_{rebound}$, $c_{retract}$, and $c_{revolve}$. Suppose there are three species A and B and C, for example. There are then six combinations of two species: A-A, B-B, C-C, A-B, B-C, and C-A. Thus we would have a total of 30 parameters. In practice, the selection of most of these parameters turns out to be trivial, for we will likely want most pairs of species to remain untethered. If two A particles cannot become tethered, then the A-A tethering distance is ∞ and the coefficients $c_{retract}$ and $c_{revolve}$ are irrelevant. Also, instead of choosing blocking distances for each pair of species, we can choose radii for each individual species and add them together to obtain the blocking distances.

4 Tethered Particle System Models

Here we present TPS models of two types of deformable biological structures: vesicle clusters and membranes.

4.1 Vesicle Clusters

An action potential, a signal that propagates along the axon of a nerve cell, will ultimately arrive at a presynaptic nerve terminal. Inside this compartment are tens or hundreds of neurotransmitter-containing sacs called synaptic vesicles [17]. These vesicles bind with a certain type of protein, called synapsin, to form clusters. Vesicles can also become docked at the active zone on the membrane of the compartment. When an action potential arrives, these docked vesicles may release neurotransmitters and trigger an action potential in an adjacent neuron. Our focus in this section is on simulations that capture the dynamics of vesicle clusters as deformable structures, the formation of these clusters, and the manner in which they congregate at the active zone.

Consider a TPS model consisting of particles of three different species: V, S, and D. Each V particle represents a vesicle. Synapsins, being dimers, are represented by pairs of tethered S particles. A D particle is a docking site, a mobile location in the active zone of the membrane on which a vesicle may become docked. Such a model was used in the simulation of Figure 7, which shows two vesicles (large spheres) surrounded by synapsins (dimers) and docking sites (small spheres). The vesicles are both tethered to opposite ends of a synapsin.

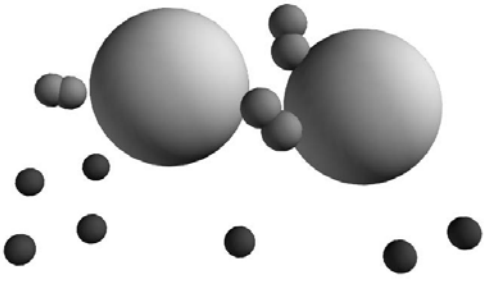


Fig. 7. A snapshot of a simulation showing two vesicles (V particles), three synapsins (pairs of S particles) and seven docking sites (D particles)

The tethering of particles is governed by the following rules.

- A V particle and another V particle may never be tethered (vesicles do not bind to one another directly).
- An S particle and another S particle may be tethered at the start of a simulation; if they are not tethered at the beginning, they will never become tethered, and if they are initially tethered, they will never detach (an S particle and its tethered counterpart represent one synapsin).
- An S particle and a V particle will become tethered if they collide, if the S particle is not already tethered to a vesicle (at most two vesicles may bind to a two-particle synapsin), and if the V particle is not already tethered to the S particle's counterpart (we do not allow both ends of a synapsin to bind to the same vesicle).

- A D particle and another D particle may never be tethered.
- A D particle and a V particle will become tethered if they collide, if the D particle is not already tethered to another V particle, and if the V particle is not already tethered to another D particle (vesicles and docking sites pair up).
- A D particle and an S particle may never be tethered.

Table 1 lists the blocking and tethering distances we selected for V, S, and D particles. As indicated in the table, approaching docking site and vesicle particles collide and rebound at 25 nm. If tethered and separating, they retract at 30 nm. The distances are chosen to reflect the sizes of actual structures. The diameter of a vesicle is roughly 40 nm, for example, the value used for the vesicle-vesicle blocking distance. Note that a blocking distance of zero indicates that blocking collisions never occur between those species, whereas a tethering distance of ∞ indicates that tethering collisions never occur.

Table 1. Blocking and tethering distances for particles representing vesicles, synapsins, and docking sites

Particle Species Pair	Blocking Distance $\Delta u_{blocking}$ (nm)	Tethering Distance $\Delta u_{tethering}$ (nm)
V-*V*	40	∞
S-*S*	2.5	7.5
S-*V*	22.5	25
D-*D*	10	∞
D-*V*	25	30
D-*S*	0	∞

The masses of V and S particles are chosen to be roughly proportional to their volumes, whereas each D particle is assigned a relatively high mass for its size to account for resistance in the membrane. The rebounding, retraction, and revolution coefficients are selected such that a considerable amount of kinetic energy is lost when vesicles, synapsins, and docking sites collide. Random impulses are applied to all three of these types of particles to maintain a certain level of kinetic energy in the entire system.

In order to model the formation, disruption, and motion of vesicle clusters, it is necessary to constrain the V, S, and D particles to a region representing the presynaptic nerve terminal compartment. The simplest way to achieve this is to model the nerve cell membrane as a rigid sphere. This is done by adding two particles to the model, one with species M and one with species Z. Both of these particles are given infinite mass, which ensures that they remain stationary. The particle of species M, representing the membrane, is tethered to all V (vesicle), S (synapsin), and D (docking site) particles.

For the sake of convenience, we introduce the parameter r_M to approximate the radius of the compartment, r_V to approximate that of a vesicle, and r_S

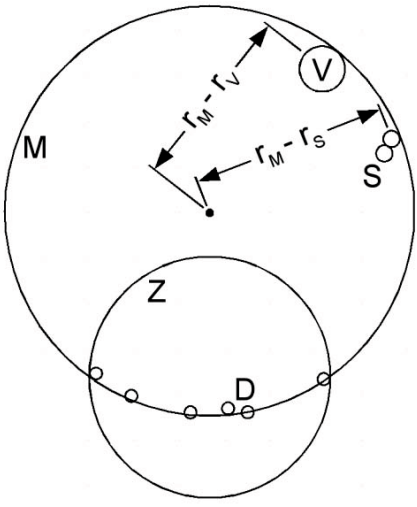


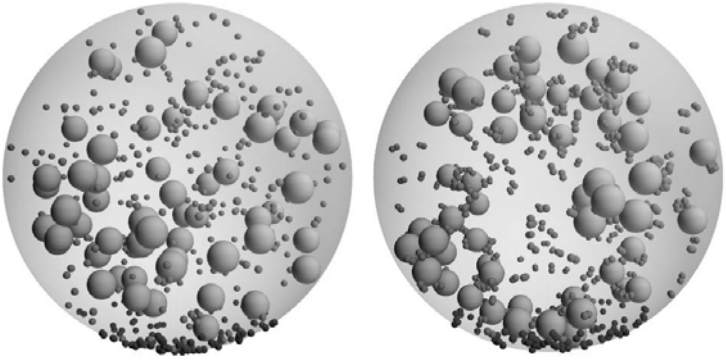
Fig. 8. A diagram illustrating the relationships between five particle species

to approximate the radius of half of a synapsin. As illustrated in Figure 8, an M-V (membrane-vesicle) tethering distance of $r_M - r_V$ keeps vesicles in the compartment, and an M-S (membrane-synapsin) tethering distance of $r_M - r_S$ does the same for synapsins. Because vesicles and synapsins move freely within the compartment, the M-V and M-S blocking distances are both zero.

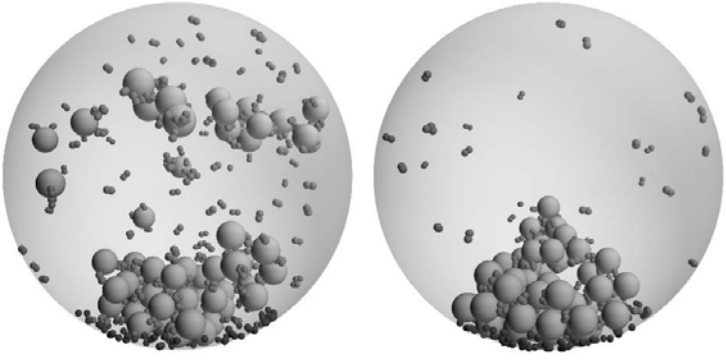
The D particles, representing docking sites, must be constrained to the spherical surface representing the cell membrane. The M-D blocking and tethering distances are therefore both chosen to be near r_M, with $\Delta u_{tethering}$ slightly greater than $\Delta u_{blocking}$. Another constraint on the docking sites is that they must all be located in that region of the membrane known as the active zone. Hence, all D particles are tethered to the Z particle shown in Figure 8. With the exception of this tethering, the Z particle has no influence on any other particle.

Figure 9 shows four snapshots of a simulation of a presynaptic terminal of a nerve cell. Vesicles and smaller synapsins move inside the semi-transparent M particle, while docking sites move slowly along the bottom of the membrane. The Z particle that constrains the docking sites is invisible.

Initially, the location of each vesicle and synapsin is randomized within the spherical compartment. None of the vesicles are initially tethered to synapsin. After the simulation begins, the tethering of colliding V and S particles leads to the formation of vesicle clusters. These clusters, which begin to take shape in Figures 9b, and 9c, grow fewer in number but larger in size as the simulation progresses. The tethering of V and D particles constrains some of these clusters to the membrane. In Figure 9d, all of the vesicles have gathered in a single cluster at the active zone. Synaptic vesicles are typically observed in similar membrane-bound clusters in real presynaptic nerve terminals.



(a) $t = 0$ (b) $t = 100$



(c) $t = 200$ (d) $t = 600$

Fig. 9. Snapshots of a simulation of a presynaptic nerve terminal with a rigid spherical membrane. With a randomized initial distribution, vesicles form clusters that eventually congregate at the active zone at the bottom of the membrane.

Experimental results presented in [4] suggest that synapsin helps to maintain a number of vesicles in the vicinity of the active zone, which in turn increases the chance that a sequence of action potentials will be transmitted from one neuron to the next. Dr. James J. Cheetham, a biologist at Carleton University, uses TPS models like the one in Figure 9 to investigate this theory. By performing numerous simulations with different numbers of vesicles and synapsins, the availablility of vesicles at the active zone can be quantified as a function of synapsin concentration. The research involves an iterative process in which the TPS model is repeatedly improved, and successive sets of simulation results are compared with experimental data. One improvement made to date is the inclusion of action potentials, which cause certain tethered vesicles and synapsins to separate from one another over a period of time. New vesicle-synapsin bonds form after each action potential subsides.

4.2 Deformable Membranes

Although the rigid spherical membrane of Section 4.1 is likely adequate for a number of investigations involving vesicle clusters, the representation of deformable membranes may help capture the dynamics of a presynaptic nerve terminal on a larger scale. Deformable membranes may also prove useful for models of entire nerve cells, networks of nerve cells, tissues, blood vessels, and possibly even large organs.

A simple way to represent a membrane with a TPS is illustrated in Figure 10. Particles are positioned on a surface, and each particle is tethered the nearest neighboring particles. To avoid excessively-sharp folds and other anomalous features, a membrane should have at least two layers; that is, there should be two or more parallel surfaces of particles, and corresponding particles on adjacent surfaces should be tethered together.

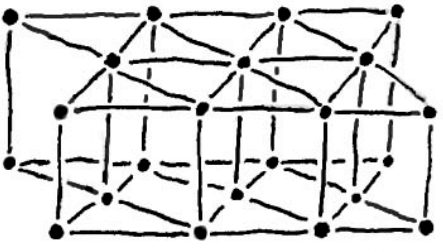
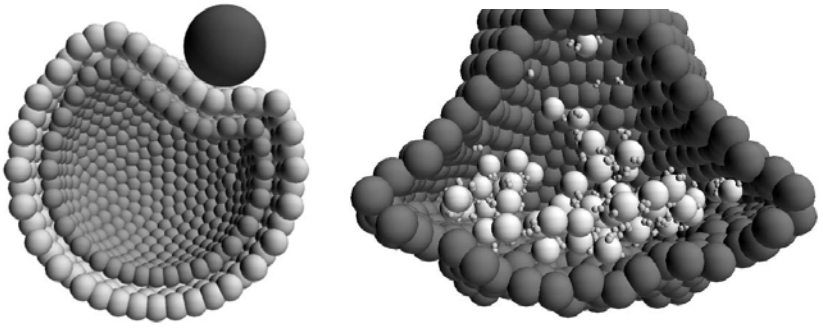


Fig. 10. An illustration of how deformable membranes may be represented. Dots are particles, and lines indicate pairs of tethered particles.

Particles on a membrane surface need not be coplanar, and need not be arranged in the triangular grid pattern shown in Figure 10. One alternative is demonstrated in Figure 11a, which shows an initially spherical membrane deforming in response to an impact with an initially downward-moving particle. The particles in the membrane were arranged in two concentric icosahedral grids, each constructed by iteratively interpolating the edges of a 20-sided regular polyhedron. In Figure 11b, this deformable icosahedral structure is used as a nerve cell membrane enclosing a presynaptic compartment. The membrane is coerced into a pear-like shape through the selection of initial particle velocities.

Several challenges can be identified by observing the results of the Figure 11b simulation. First, the edges of the underlying 20-sided polyhedron tend to protrude from the membrane as it deforms. A more random distribution of particles would reduce this effect. Another challenge pertains to resolution. The nerve cell membrane in the figure is too thick, but reducing its width would require greater numbers of particles and computations. Because intracellular fluids are essentially incompressible, maintaining the interior volume of a closed deformable membrane in a TPS model is yet another challenge. It might be possible to correct a changing interior volume at regular time intervals using external



(a) (b)

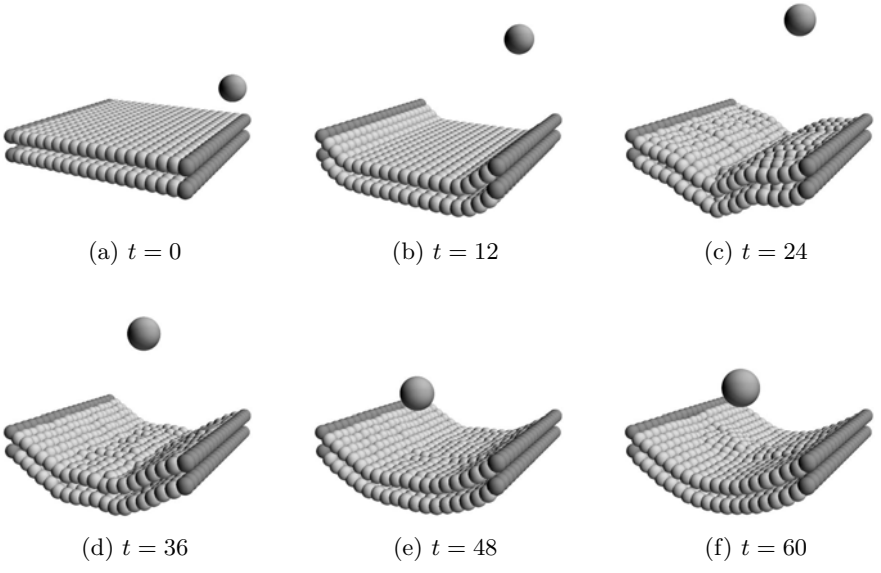
Fig. 11. On the left, an initially-spherical deformable membrane suffers an impact. On the right, a presynaptic nerve terminal is simulated with the same deformable membrane. The front half of the membrane is not shown in either snapshot, and the outer membrane layer is not shown on the right.

impulses, though the calculation of those impulses would require the use of a fluid dynamics algorithm in conjunction with the TPS.

Figure 12 shows an effort to simulate a square biological membrane or soft tissue clamped along two opposing edges. The membrane has two layers of particles, and each particle on the inside is tethered to the four adjacent particles in the same layer and one particle in the opposite layer. All particles along the two clamped edges are assigned a mass of ∞ and an initially-zero velocity, rendering them immobile.

Gravity is incorporated in the Figure 12 model via downward external impulses applied to each mobile particle at regular time intervals. As a result of these impulses, the initially flat membrane in Figure 12a is starting to sag in Figure 12b. In Figure 12c the membrane exhibits a wave-like pattern as it responds to internal tethering collisions triggered by the initial fall. Small non-deterministic ripples appear in the membrane as a consequence of two sources of randomness; the order in which particles receive gravitational impulses is randomized, as is the order in which simultaneous collisions are resolved. The gravity-induced waves have mostly subsided after 48 time units, as shown in Figure 12e. Shortly after, a falling particle impacts the membrane and produces the small ridge in Figure 12f.

The simulation of Figure 12 reveals a case for which the TPS method should not be used: a structure composed of numerous tethered particles subject to sustained opposing external forces. The opposing forces in the Figure 12 model include the downward force of gravity and the upward reaction force acting on the membrane along the stationary edges. While a macroscopic soft tissue will lengthen as the applied force increases, a TPS deformable membrane will reach its maximum length and stop stretching. If, for whatever reason, the force of gravity were to increase by an order of magnitude, the membrane in Figure 12 would sag faster but no further. The Figure 12 simulation is also computationally



(a) $t = 0$ (b) $t = 12$ (c) $t = 24$

(d) $t = 36$ (e) $t = 48$ (f) $t = 60$

Fig. 12. A square membrane clamped along two edges responds first to gravity, then to an impact with a projectile. Note that the projectile is moving upwards in Figures 12a, 12b, and 12c, then downwards in 12d and 12e. In 12f, it is moving up after colliding with the membrane.

inefficient. After the membrane has sagged, many pairs of particles remain near or at their tethering distances. These stable contacts cause collisions to occur at an extremely high rate, slowing the simulation to a crawl.

5 Conclusion

The TPS method described and demonstrated in this paper provides convincing evidence that impulse-based methods can be used to simulate the dynamics of deformable structures. The new method is very similar to that of [14], yet contradicts the assertion that such impulse-based methods require models to be comprised of only rigid bodies. The TPS requires only analytic calculations, alleviates the need for regular time intervals, and is particularly promising for simulations of small-scale self-assembling deformable biological structures. External impulses may be added to a TPS model to produce random motion, to apply drag forces, or to account for other factors influencing the dynamics of biological objects. We have demonstrated the application of the TPS to vesicle clusters and membranes.

As mentioned in Section 4.2, the method as currently defined does not appear to be useful for models of complex macroscopic structures subject to sustained and opposing external forces. The simulation of a clamped membrane subject to

gravity, shown in Section 4.2's Figure 12, is a good example of an application requiring either an alternative method, or perhaps some future enhanced version of the TPS. In the case of small-scale self-assembling biological structures subject to random motion, like the vesicle clusters of Figure 9, stable contacts are far less likely to pose a problem.

A detailed comparison of the TPS with alternative dynamic simulation methods remains important future work. Here we speculate that the FEM would be difficult to apply to small-scale self-assembling biological structures, as rapid deformation and re-structuring would require frequent re-calculation of mass and stiffness matrices. Mass-spring-damper systems suffer from the threat of instability, though it is possible to address this problem with constraints as done in the method of [2] and [6]. Recall from Section 2.2 that, although it is described as "impulse-based", the [2]/[6] method requires new trajectories to be computed for each node in a deformable structure at regular time intervals. The TPS simulates deformation with impulses applied not at regular intervals, but rather in response to collisions. The [2]/[6] method seems to be the more computationally efficient, as the number of collisions in a TPS simulation can be extremely high. The TPS is appealing in that all calculations are analytic; there is no need for the iterative algorithm of [2]/[6] that repeatedly re-calculates trajectories until all constraints are satisfied within an arbitrary tolerance level.

Impulse-based methods like the TPS are compelling in large part because they alleviate the need to choose regular time intervals. In the case of a biological system, the selection of an appropriate time interval would be complicated by the fact that interacting biological entities may differ in size and momentum by many orders of magnitude. A suitable interval for one entity may be too large or too small for another.

Acknowledgments

We thank Dr. James J. Cheetham, from the Department of Biology at Carleton University, for providing expertise in the biology of presynaptic nerve terminals and numerous suggestions on how to model them.

References

1. Baraff, D.: Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. Computer Graphics 23(3), 223–232 (1989)
2. Bender, J., Bayer, D.: Impulse-based simulation of inextensible cloth. In: Proceedings of The International Conference on Computer Graphics and Visualization (IADIS), Amsterdam, Netherlands (2008)
3. Brown, J., Sorkin, S., Bruyns, C., Latombe, J.-C., Stephanides, M., Montgomery, K.: Real-time simulation of deformable objects: Tools and application. Computer Animation, 228–236 (2001)
4. Camilli, P.D.: Keeping synapses up to speed. Nature 375, 450–451 (1995)
5. Conti, F., Khatib, O., Baur, C.: Interactive rendering of deformable objects based on a filling sphere modeling approach. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan (2003)

6. Diziol, R., Bender, J., Bayer, D.: Volume Conserving Simulation of Deformable Bodies. In: Proceedings of Eurographics, Munich, Germany (2009)
7. Gibson, S.F.F., Mirtich, B.: A Survey of Deformable Modeling in Computer Graphics. Mitsubishi Electric Reasearch Laboratories (1997), `www.merl.com`
8. Guess, T.M., Maletsky, L.P.: Computational modelling of a total knee prosthetic loaded in a dynamic knee simulator. Medical Engineering & Physics 27(5), 357–367 (2005)
9. Jansson, J., Vergeest, J.S.M.: A discrete mechanics model for deformable bodies. Computer-Aided Design 34(12), 913–928 (2002)
10. Keiser, R., Müller, M., Heidelberger, B., Teschner, M., Gross, M.: Contact Handling for Deformable Point-Based Objects. In: Proceedings of the Vision, Modeling, and Visualization Conference (VMV), Stanford, CA, USA (2004)
11. Kitano, H.: Computational systems biology. Nature 420, 206–210 (2002)
12. Klein, M.L., Shinoda, W.: Large-Scale Molecular Dynamics Simulations of Self-Assembling Systems. Science 321(5890), 798–800 (2008)
13. Mirtich, B., Canny, J.: Impulse-based Simulation of Rigid Bodies. In: Proceedings of the 1995 Symposium on Interactive 3D Graphics (SI3D), Monterey, California, United States (1995)
14. Mirtich, B.V.: Impulse-based Dynamic Simulation of Rigid Body Systems. PhD. University of California at Berkeley, Berkeley (1996)
15. Moore, M., Wilhelms, J.: Collision Detection and Response for Computer Animation. Computer Graphics 22(4), 289–298 (1988)
16. Moore, P., Molloy, D.: A Survey of Computer-Based Deformable Models. In: Proceedings of the International Machine Vision and Image Processing Conference (IMVIP), Maynooth, Ireland (2007)
17. Südhof, T.C., Starke, K.: Pharmacology of Neurotransmitter Release. Springer, Heidelberg (2008)
18. Tagawa, K., Hirota, K., Hirose, M.: Impulse Response Deformation Model: an Approach to Haptic Interaction with Dynamically Deformable Object. In: Proceedings of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS), Alexandria, VA, USA (2006)
19. Wang, T., Pan, T.-W., Xing, Z.W., Glowinski, R.: Numerical simulation of rheology of red blood cell rouleaux in microchannels. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 79(4), 041916+ (2009)