Simulating the cell internal structure using Delaunay triangulation

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Physik
der Johann Wolfgang Goethe – Universität
in Frankfurt am Main

von

Graziela Grise
aus Belo Horizonte – Brasilien

Frankfurt 2010
(D 30)

vom Fachbereich Physik der Goethe-Universität
Frankfurt am Main als Dissertation angenommen.

Dekan: Prof. Dr. Dirk-Hermann Rischke
1. Gutachter: Prof. Dr. Michael Meyer-Hermann
2. Gutachter: Prof. Dr. Horst Stöcker

Datum der Disputation: 6te Oktober 2010

Zusammenfassung

Es gibt nahezu unendliche viele Daten über biologische Systeme, aber bis heute nur wenige Versuche, diese Daten in ein schlüssiges Gesamtbild zu fügen. Obwohl biologische Systeme eine höhere Komplexität aufweisen als die meisten physikalischen Systeme, können auch sie auf vielen Ebenen von theoretischen Modellen profitieren.

Das biologische System, dem sich diese Arbeit widmet, ist die Zelle, die fundamentale strukturelle und funktionale Einheit aller lebenden Organismen. Zellen verfügen über eine komplexe interne Struktur und viele verschiedene Organellen, die durch eine Plasmamembran von der Umgebung abgegrenzt sind.

Neue Entwicklungen moderner experimenteller Techniken zur Beobachtung von einzelnen Zellen und deren Bewegungen (insbesondere das "two-photon imaging") schufen großartige Möglichkeiten für die Theoretische Biologie, da diese die Aufzeichnung von Zellformen und -migrationen im Inneren lebender Tiere erlauben. Darüber hinaus wurden Techniken entwickelt, die es erlauben die Position von bis zu 100 verschiedenen Molekülen im Zellinneren zu jedem beliebigen Zeitpunkt aufzuzeichnen.

Diese Entwicklungen erhöhen den Bedarf für ein theoretisches Zellmodell, mit dem auch das Zellinnere ortsaufgelöst beschrieben werden kann. Herkömmliche Methoden wie zum Beispiel Differenzialgleichungen eignen sich nur zur Beschreibung großer Systeme, in denen das Verhalten einzelner Einheiten alleine das Verhalten des Gesamtsystems nicht relevant beeinflusst und in denen Messgrößen als Durchschnitt über das gesamte System bestimmt werden können. Zur Beschreibung kleiner Systeme, in denen jede Einheit individuell identifiziert und ihr Verhalten untersucht werde muss, werden Agenten-basierte Modelle benötigt.

Die Migration von Zellen ist intrinsisch mit ihrer Form verbunden. Aus diesem Grund müssen theoretische Modelle zur Beschreibung der Migration auch die Form der Zellen berücksichtigen. Dies wieder erfordert Kenntnisse über die interne Struktur der Zelle. In diesem Fall genügt es nicht, Zellen als einzelne Punkte im Raum zu betrachten, sondern es ist notwendig, diese durch eine Gruppe auf der Grundlage bestimmter Eigenschaften interagierender Unterteilchen zu beschreiben. Die gleichen Anforderungen gelten für Modelle, die versuchen die Bewegung von Molekülen im Zellinneren zu beschreiben: das Modell muss in der Lage sein, jede Zelle individuell zu beschreiben und zusätzlich mit einer gewissen Auflösung das Zellinnere abzubilden.

Im letzten Jahrzehnt ermöglichte die Evolution der Computersysteme immer mächtigere Simulationen von Zellen und Gewebe zum Verständnis von biologischen Systemen. Allerdings muss immer ein Kompromiss zwischen der Größe eines Systems und dem Detailgrad der Simulation gefunden werden. Eine weitere Komplexitätsstufe kommt hinzu, wenn die innere Struktur der Zellen ebenfalls aufgelöst werden soll. Je höher der Detailgrad des Modells, desto kleiner sind die Systeme, die mittels Computersimulationen abbildbar sind. Deshalb ist es notwendig, Modelle zu entwi-

ckeln, die eine variable Auflösung der internen Zellstruktur ermöglichen, so dass man sowohl kleine Systeme bei hoher Auflösung wie auch große Systeme mit geringerer Auflösung simulieren kann.

Das Ziel dieser Arbeit ist die Entwicklung eines Modells für eine Zelle, welches die Eigenschaften der internen Struktur berücksichtigen kann. Hierzu wird ein Agenten-basierter Ansatz verwendet, in dem die Zelle durch eine Vielzahl miteinander wechselwirkender Teilchen simuliert wird. Prinzipiell können diese Teilchen jede interne Struktur der Zelle repräsentieren. In dem hier besprochenen Modell werden zwei Arten verwendet: Membranteilchen und zytosolische Elemente. Das entwickelte Modell eignet sich damit sowohl zur Simulation von einzelnen Zellen als auch zur Beschreibung multi-zellulären Systemen.

Modell

Eine kinetische und dynamische Delaunay-Triangulation wurde zur Definition der Nachbarschaftsbeziehungen zwischen den Teilchen verwendet. Allerdings eignen sich die Eigenschaften der Delaunay-Triangulation nicht, die Wechselwirkungen zwischen Membranteilchen zu beschreiben. Die Zellmembran ist eine Lipid-Doppelschicht in der keine langreichweitigen Wechselwirkungen existieren. Aus diesem Grund sollten die Teilchen, die die Membran bilden, keine derartigen Wechselwirkungen aufweisen, sondern ihre Interaktionen lediglich auf die zweidimensionale Fläche beschränken, die die Membran repräsentiert.

Zur Lösung dieses Problems wurde eine Methode entwickelt, die aus der ursprünglich dreidimensionalen Triangulation nur diejenigen Verbindungen auswählt, die die Oberfläche der Zelle, also die Membran, bilden. Diese Methode wird im Detail im Abschnitt *Oberflächenrekonstruktion* erläutert.

Die Wechselwirkungen zwischen den zytosolischen Elementen ist durch ein Lennard-Jones-Potenzial gegebenen, wie auch die Interaktionen zwischen den zytosolischen Elementen und den Membranteilchen. Die Parameter für beide Wechselwirkungen waren jedoch unterschiedlich. An der Oberfläche, d.h. zwischen den Membranteilchen, wurde eine rein elastische Wechselwirkung angenommen.

Die Bewegung jedes Teilchens wurde durch die zugehörige Bewegungsgleichung beschrieben. Dabei wurde der Verlet-Algorithmus zur Lösung der Bewegungsgleichungen gewählt. Da das Zytosol durch ein zähflüssiges Gel approximiert werden kann, ist es eine gültige Vereinfachung, die Bewegung der Zytosolteilchen als stark gedämpft anzusehen. Aus diesem Grund wurde für alle Wechselwirkungen die Näherung der starken Dämpfung (“overdamped approximation”) gewählt.

Zusätzlich wurde eine adaptiver Algorithmus verwendet, um die Größe des Zeitschritts für die Berechnung jeder Interaktion zu bestimmen. Dieses Vorgehen ist besonders zu Beginn der Simulation hilfreich, da die nicht thermalisierten Teilchen in künstlichen Konstellationen beliebig nah bei einander sein können und somit starke unphysiologische Kräfte zwischen ihnen wirken.

Oberflächenrekonstruktion

Wie im vorangegangenen Abschnitt erklärt, beschränken die biologischen Eigenschaften der Zellmembran die Wechselwirkungen zwischen den Membranteilchen auf die Oberfläche, die sie bilden. Hierdurch ist die ursprünglich dreidimensionale Delaunay-Triangulation nicht geeignet, die Interaktionen zwischen den Membranteilchen zu beschreiben, und es ist notwendig, eine auf die Oberfläche eingeschränkte Delaunay-Triangulation zur Abbildung dieser Wechselwirkungen zu konstruieren.

Die Methode hierzu geht von einer Punktmenge P (Membranteilchen), die zu einer Oberfläche $\mathbb{R}^3$ gehören, sowie einer zweiten Punktmenge Q aus, die die interne Struktur der Zelle (zytosolische Elemente) beschreibt. Hieraus wird die dreidimensionale Delaunay-Triangulation von $P \cup Q$ berechnet, die ohnehin benötig wird, um die Wechselwirkungen zwischen den zytosolischen Elementen zu beschreiben. Zusätzlich ist die dreidimensionale Delaunay-Triangulation der Ausgangspunkt zur Definition der eingeschränkten Delaunay-Triangulation.

Die Methode funktioniert wie folgt: ausgehend von der dreidimensionalen Delaunay-Triangulation von $P{\cup}Q$ wird eine Struktur *Quasi-Oberfläche* genannt bestimmt. Die Quasi-Oberfläche ist eine Untermeng der Delaunay-Triangulation und beinhaltet alle Verbindungen eingeschränkten Oberfläche sowie einige weitere. Aus diesem Grund beinhaltet die Quasi-Oberfläche unerwünschte dreidimensionale Strukturen (Simplexe oder Gruppen von Simplexen).

Die verbleibenden drei-dimensionalen Strukturen, die zur Quasi-Oberfläche gehören, müssen aus dieser entfernt werden. Hierzu wird jede Gruppe verbundener Simplexe (d.h. Simplexe, die eine gemeinsame Fläche haben) in der Quasi-Oberfläche ausgewählt. Diese Simplex-Gruppen werden im Folgenden als *Cluster* bezeichnet.

Im Anschluss wird der Rand der ausgewählten Cluster identifiziert und die Vertices geordnet. Dies geschieht mit Hilfe einer Prozedur, die für jede Verbindung eines Clusters überprüft, ob sie zum Rand des Clusters gehört oder nicht. Nachdem jede Verbindung, die zur Membran gehört identifiziert ist, können die Vertices anhand ihrer Verbindungen geordnet werden. Beispiel: Falls bei einem Cluster mit vier Punkten die Verbindungen des Randes (A, B), (B, C), (C, D) und (D, E) sind, dann ist der geordnete Rand (A, B, C, D, E).

Nachdem der Rand des Clusters identifiziert und geordnet ist, muss ein Retriangulations-Algorithmus für jeden Cluster angewandt werden. Hierzu wurden zwei Algorithmen entwickelt: der eine für Cluster mit internen Punkten und der andere zur Lösung von Clustern ohne interne Punkte.

Die Retriangulationsmethode für Cluster ohne interne Punkte ähnelt dem Problem der Triangulation eines Polygons. Allerdings können im speziellen Fall hier nur Verbindungen für die Triangulation ausgewählt werden, die in der originalen Triangulation über alle Punkte enthalten waren.

Cluster mit vielen internen Punkten stellen besondere Herausforderungen an die Retriangulation. Ihre Eigenschaften hängen stark von der Zahl der internen Punkte und ihrer räumlichen Anordnung ab. Zur Entwicklung eines erfolgreichen Algorithmus' zur Retriangulation solcher Cluster ist in einer sorgfältigen Abwägung die Auswahl der Verbindungen einzuschränken, so dass die finale Triangulation die gewünschte Oberfläche retrianguliert, und genügend Flexibilität zuzulassen, so dass eine solche Konfiguration gefunden werden kann.

Einige Einschränkungen wurden bei der Auswahl der Verbindungen für die finale Triangulation eines Clusters mit internen Punkten eingeführt, hauptsächlich mit dem Ziel eine möglichst glatte Retriangulation zu erzeugen. Zuerst werden kürzere Verbindungen bevorzugt – lange Verbindungen sind selten und die Auswahl einer langen Verbindung führt mit großer Wahrscheinlichkeit zu einer Situation, in der die Retriangulation des Clusters nicht abgeschlossen werden kann. Zusätzlich ist es notwendig sicherzustellen, dass alle Dreiecke, die durch die Annahme zweier Verbindungen erzeugt werden, die gleiche Orientierung habe. Falsch orientierte Dreiecke machen es notwendig, lange Verbindungen auszuwählen, die wiederum eine erfolgreiche Retriangulation erschweren. Weiterhin ist es erforderlich, dass die Projektion aller Punkte eines Clusters, die nicht Teil des Dreiecks sind, auf die Ebene des Dreiecks außerhalb des Dreiecks liegen.

Ausgangskonfigurationen

Die Erzeugung einer Ausgangskonfiguration beginnt ausschließlich mit Membranteilchen. Diese Teilchen werden zufällig in einer künstlichen sphärischen Hülle erzeugt. Um sicherzustellen, dass die Teilchen in dieser sphärischen Hülle bleiben, wurde eine harte Begrenzung (“hard sphere”) eingeführt, die die Membranteilchen nicht durchdringen können. Zusätzlich wurde ein einzelnes künstliches Teilchen in das Zentrum der sphärischen Hülle eingefügt. Dieses Teilchen übt anziehende Wechselwirkungen ähnlich einem Zentralpotenzial auf die Membranteilchen aus.

In der sphärischen Hülle wechselwirken die Membranteilchen elastisch mit einander. Bis zu diesem Zeitpunkt sind in dem System noch keine zytosolischen Teilchen enthalten. Die Membranteilchen interagieren in dieser Anordnung bis sie gleichförmig in der sphärischen Hülle verteilt sind. Die Ausgangskonfiguration wird als thermalisiert angesehen, sobald eine bestimmte Genauigkeit im Abstand der Teilchen erreicht wurde.

Eine Delaunay-Triangulation aller interagierender Teilchen wird zur Bestimmung der Nachbarschaftsbeziehungen verwendet und in jedem Zeitschritt gemäß der veränderten Teilchenpositionen angepasst.

Sobald die Membranteilchen in der sphärischen Hülle thermalisiert sind, wird das künstliche Teilchen in der Mitte entfernt und die sphärische Hülle zufällig mit zytosolischen Teilchen gefüllt. Diese interagieren miteinander und mit den Membranteilchen, die sie umschließen, durch ein Lennard-Jones-Potential. Zu diesem Zeitpunkt sind die Membranteilchen allerdings fixiert und unbeeinflusst von der Wechselwirkung mit den zytosolischen Teilchen. In dieser Konfiguration wechselwirken die zytosolischen Teilchen bis sie thermalisiert sind.

Test der Oberflächenrekonstruktion

Nach der Implementation der Methode zur Retriangulation der Membrane wurde der Zeitbedarf zur Retriangulation eines einzelner Clusters studiert, gefolgt von einer Analyse wie dieser Zeitbedarf mit der Anzahl der Punkte in einem Cluster variiert. Die Häufigkeit, mit der jede Clustergröße im Gesamtsystem vorkommt, wurde ebenfalls analysiert, da diese Information benötigt wird, um zu garantieren, dass die Gesamtzeit, die benötigt wird eine Zelle zu retriangulieren, konvergent ist. Zuletzt wurde die Gesamtzeit zur Retriangulation aufgezeichnet und das Skalenverhalten mit der Variation der Membran bestimmt.

Die Zeit, die benötigt wird, einen Cluster zu retriangulieren wächst mit der Größe der Cluster. Das beobachtete Verhalten ist ungefähr quadratisch, weicht jedoch für kleine Clustergrößen ab. Dies kann dadurch erklärt werden, dass die Retriangulations-Methode mit der Zahl der internen Punkte nicht jedoch mit der Gesamtzahl der Punkte in einem Cluster quadratisch skaliert. Es konnte gezeigt werden, dass die durchschnittliche Zeit, die benötigt wird, einen Cluster zu lösen, weder von der Gesamtgröße des Systems noch von der Störung, die an das System angelegt wurde, abhängt.

Ebenso interessant ist die durchschnittliche Zeit, die benötigt wird, alle Punkte in einem Cluster zu retriangulieren. Je größer der Cluster, desto größer ist die Komplexität und dadurch auch die Zeit, die benötigt wird, den kompletten Cluster zu retriangulieren. Für große Clustergrößen wird eine lineare Skalierung gezeigt.

Eine weitere Größe von Bedeutung ist die Häufigkeit mit der Cluster einer bestimmten Größe in einer Simulation auftauchen. Da, wie oben erwähnt, die Retriangulationszeit für große Cluster ungefähr quadratisch mit der Größe skaliert, könnte die Gesamtzeit der Simulation explodieren, falls die Häufigkeit des Auftauchens großer Cluster nicht ebenfalls mindestens quadratisch abnimmt. Glücklicherweise nimmt die Häufigkeit exponentiell mit der Clustergröße ab und garantiert somit eine kontrolliebare Simulationszeit auch für große Systeme.

Um dies zu bestätigen, wurde die Zeit bestimmt, die benötigt wird alle Cluster einer Größe zu retriangulieren. Diese Zeit ist das Ergebnis der Multiplikation der Häufigkeit mit der Cluster einer Größe auftauchen mit der durchschnittlichen Zeit, die benötigt wird diese Cluster zu retriangulieren. Da die Häufigkeit exponentiell abnimmt und die Retriangulationszeit quadratisch wächst, ist zu erwarten, dass die Zeit alle Cluster einer Größe zu retriangulieren ebenfalls einen exponentiellen Abfall mit der Clustergröße zeigt. Dies wurde für verschiedene Systemgrößen und Pertubationen bestätigt.

Zuletzt wird gezeigt, dass die Gesamtzeit zur Retriangulation aller Cluster mit dem Grad der Pertubation, d.h. die Abweichung der Membranpunkte von einer perfekten Kugel, zunimmt.

Diskussion und Schlußfolgerung

Das Problem der Rekonstruktion einer auf eine zwei-dimensionale gekrümmte Fläche eingeschränkten Punktmenge ist sehr komplex und nicht vollständig gelöst. Der hier vorgestellte Algorithmus ist stark an die spezifische Anwendung innerhalb dieser Arbeit angepasst, aber nicht vollständig fehlerfrei. Die entwickelte Methode hat zwei Schwachpunkte, an denen es zu Fehlern kommen kann: die Bestimmung und Auswahl der Ränder der Cluster und zu strikte Einschränkungen bei der Retriangulation für Cluster mit vielen internen Punkten.

Obwohl noch einige Erweiterungen möglich sind, ist die Autorin überzeugt, dass die hier vorgestellte Arbeit einen wichtigen Schritt auf dem Weg zu einem Agenten-basierten Modell ist, welches nicht nur die Simulation von sub-zellulären Strukturen erlaubt sondern auch sinnvolle Wechselwirkungen zwischen Membranteilchen berücksichtigt. Dieses Modell ist für eine große Anzahl potenzieller Anwendungen interessant: von der Studie von Zellformen und Migration zur Studie der intrazellulären Dynamik, einschließlich der Möglichkeit einzelne Moleküle in der Zelle mit räumlicher Auflösung darzustellen.

Die Autorin hofft mit dieser Arbeit in einer sowohl zum Feld der sub-zellulären Modelle wie auch zum Gebiet der Oberflächenrekonstruktion beigetragen zu haben. Wenn nicht durch neue Anwendungen, so mit Sicherheit durch neue Ideen und einen frischen Ansatz durch die Verbindung verschiedener Überlegungen aus unterschiedlichen Bereichen der Theorie.

Abstract

The goal of this project is to develop a framework for a cell that takes in consideration its internal structure, using an agent-based approach. In this framework, a cell was simulated as many sub-particles interacting to each other. This sub-particles can, in principle, represent any internal structure from the cell (organelles, etc). In the model discussed here, two types of sub-particles were used: membrane sub-particles and cytosolic elements.

A kinetic and dynamic Delaunay triangulation was used in order to define the neighborhood relations between the sub-particles. However, it was soon noted that the relations defined by the Delaunay triangulation were not suitable to define the interactions between membrane sub-particles. The cell membrane is a lipid bilayer, and does not present any long range interactions between their sub-particles. This means that the membrane particles should not be able to interact in a long range. Instead, their interactions should be confined to the two-dimensional surface supposedly formed by the membrane.

A method to select, from the original three-dimensional triangulations, connections restricted to the two-dimensional surface formed by the cell membrane was then developed. The algorithm uses as starting point the three-dimensional Delaunay triangulation involving both internal and membrane sub-particles. From this triangulation, only the subset of connections between membrane sub-particles was considered. Since the cell is full of internal particles, the collection of the membrane particles' connections will resemble the surface to be obtained, even though it will still have many connections that do not belong to the restricted triangulation on the surface. This "thick surface" was called a *quasi-surface*.

The following step was to refine the quasi-surface, cutting out some of the connections so that the ones left made a proper surface triangulation with the membrane points. For that, the quasi-surface was separated in clusters. Clusters are defined as areas on the quasi-surface that are not yet properly triangulated on a two-dimensional surface. Each of the clusters was then re-triangulated independently, using re-triangulation methods also developed during this work.

The interactions between cytosolic elements was given by a Lennard-Jones potential, as well as the interactions between cytosolic elements and membrane particles. Between only membrane particles, the interactions were given by an elastic interaction.

For each particle, the equation of motion was written. The algorithm chosen to solve the equations of motion was the Verlet algorithm. Since the cytosol can be approximated as a gel, it is reasonable to suppose that the sub-cellular particles are moving in an overdamped environment. Therefore, an overdamped approximation was used for all interactions. Additionally, an adaptive algorithm was used in order to define the size of the time step used in each interaction.

After the method to re-triangulate the membrane points was implemented, the

time needed to re-triangulate a single cluster was studied, followed by an analysis on how the time needed to re-triangulate each point in a cluster varied with the cluster size. The frequency of appearance for each cluster size was also compared, as this information is necessary to guarantee that the total time needed by to re-triangulate a cell is convergent. At last, the total time spent re-triangulating a surface was plotted, as well as a scaling for the total re-triangulation time with the variation.

Even though there is still a lot to be done, the work presented here is an important step on the way to the main goal of this project: to create an agent-based framework that not only allows the simulation of any sub-cellular structure of interest but also provides meaningful interaction relations to particles belonging to the cell membrane.

Contents

CHAPTER 1

Motivation

Contents

1.1 Theory and experiments in physics and biology

In physics, theoretical models are considered to be a powerful tool, and extremely successful theoretical models evolve towards becoming theories. Classical and quantum mechanics, electromagnetism, or quantum chromodynamics are just some examples of such theoretical models. The aim of these theories is not to explain every physical phenomena in an unified way (though this is the aim of some others – albeit none successfull so far), but just to describe a specific subset as efficiently and accurately as possible.

Physical theories are not always right, even though some times it looks like this is the case. Classical mechanics fails when sizes are too small or speeds are too high, but it is still unbeatable for describing mechanical systems whose properties fall in between. Using quantum mechanics or relativity to solve problems that fall within the range of classical mechanics, although possible, is extremely cumbersome and inefficient.

In biology, on the other side, the development of theoretical models is still fairly recent, and sometimes confined to phenomenological use (fitting experimental data but not allowing any deeper understanding of the system). Systems in biology are extremely complex and, unlike in physics, do not behave always in the same way, i.e., the laws that govern biological systems are not as rigid as the ones governing physical systems. This makes the trajectory from theoretical models to theories unlikely, since it is improbable that a single model can explain a large range of biological phenomena.

The lack of rigid laws may explain why theoretical models were not widespread in biology until recently, but it does not mean that biological systems cannot profit from models. There exist a virtually infinite amount of data about biological systems, but really few people trying to make sense of it inside a bigger picture. In order for biology to really gain from a tighter integration between theoretical models and experiments, theoretical biologists should focus in creating models that are coherent with the available experimental data and that have some predictive power, which in turn can lead to new experiments and better models. In their paper "A framework for consciousness" [Crick 2003], Christof Koch and Francis Crick say:

> "A framework is not a detailed hypothesis or set of hypotheses; rather, it is a suggested point of view for an attack on a scientific problem, often suggesting testable hypotheses. Biological frameworks differ from frameworks in physics and chemistry because of the nature of evolution. Biological systems do not have rigid laws, as physics has. Evolution produces mechanisms, and often sub-mechanisms, so that there are few 'rules' in biology which do not have occasional exceptions.
>
> A good framework is one that sounds reasonably plausible relative to available scientific data and that turns out to be largely correct. It is unlikely to be correct in all the details. A framework often contains unstated (and often unrecognized) assumptions, but this is unavoidable."

1.2 Goals of this study

The aim of this work was to develop a framework for the simulation of cells (including internal resolution) in an agent-based fashion.

The recent development of advanced experimental techniques in cell tracking (specially with the advent of two-photon imaging [Denk 1990, So 2001, Miller 2002]) opened up great opportunities for theoretical biologists interested in modeling cell shape and migration. With these techniques the shape and movement of individual cells can be tracked inside living animals, generating data supposed to mirror the real behavior of cells in vivo. Additionally, experimental techniques capable of making snapshots of a cell with up to a hundred molecules at a time emerged [Schubert 2003, Schubert 2006], calling for a cell model on which molecules can be followed in a space-resolved manner.

Traditional methods like differential equations are only adequate to describe big systems, where the behavior of one individual entity alone is not relevant to the systems' overall outcome, and where the properties one is interested in can be averaged. In small systems, where particles must be individually identified and their behaviors independently studied, agent-based models are needed.

The migration of cells is intrinsically connected to their shape [Alt 1995]. Therefore, a theoretical model aiming in describing cell migration has to take cell shape in consideration, which in turn requires knowledge about the internal structure of the cell. Cells, in this case, cannot be simulated as a single point in space, but more as a set of sub-particles interacting according to given properties. At the same time, a model intending to follow molecules inside a cell requires the same properties: it must treat cells individually and it needs to include some sort of internal resolution of the cell.

In the last decade the evolution of computer systems has allowed the simulation of cells and tissues to become a powerful tool towards the understanding of biological systems. However, researchers developing theoretical models have to find a compromise between the size of the system to be simulated and the level of details included in the simulations. This becomes even more complicated once the inclusion of resolution of the cell's internal structure is taken into consideration. The more detailed the model, the smaller the system one will be able to study with. Therefore, a successful model to simulate cells should allow a variable internal resolution, being flexible enough to simulate both small systems with high internal resolution and big systems with low internal resolution.

The developed framework takes in consideration the cell's internal structure and is suitable to simulate single cells as well as multi-cellular systems. The cell is represented as a set of discrete interacting particles, and the neighborhood relations between the sub-cellular particles are obtained via a kinetic and dynamic three-dimensional Delaunay triangulation [Schaller 2004, Beyer 2005, Meyer-Hermann 2008]. A set of interaction potentials is used to differentiate the types of particles and to define their roles inside the cell. A model with two distinguished types of particles will be described in this thesis. Particles will belong either to the membrane (membrane particles) or to the internal structure (cytosolic elements) of the cell.

Even though the initial idea when developing this framework was that all particles should interact via the neighborhood relations provided by a Delaunay triangulation, it was soon clear that this set of connections was not appropriated to mimic the interactions between membrane particles. The cell membrane is mainly composed by a lipid bilayer that can be approximated by a two-dimensional bended surface. Therefore, membrane particles should only interact vial local connections restricted to the two-dimensional bended surface formed by these membrane particles. In this case, the algorithm to define the neighborhood relations between membrane particles should enforce that the membrane interactions only occur restricted to this surface. This points to the usage of a two-dimensional Delaunay triangulation restricted to the membrane surface for the connections between membrane particles, instead of the classical three-dimensional Delaunay triangulation.

The problem of defining a Delaunay triangulation restricted to the surface represented by the membrane particles has some similarities with problems in surface reconstruction: starting from a set of sample points, how can the surface that originally generated these points be reconstructed? Indeed, many of the methods used for surface reconstruction consist in obtaining a restricted Delaunay triangulation from a set of sample points, having as starting point its regular three-dimensional Delaunay triangulation [Cazals 2006]. The restricted Delaunay triangulation will, in this case, represent the desired surface.

Unfortunately, traditional surface reconstruction methods cannot be readily used in the developed framework, as they allow the deletion of points in the original triangulation [Amenta 1998, Amenta 1999]. In the problem of defining neighborhood relations for membrane particles, the number of particles should not fluctuate, unless the size of the system (the membrane) is intentionally being changed.

However, a method that defines this restricted triangulation as a subset of the three-dimensional Delaunay triangulation was still highly desired, as the original Delaunay triangulation is needed to define the neighborhood relations between cytosolic elements as well as the cross interactions between membrane particles and cytosolic elements.

The development of a computational method to achieve this objective, providing meaningful interaction relations to membrane particles, was a central goal in this thesis.

1.3 Structure

This thesis is structured as following: chapter 2 starts with a description of the state of the art of mathematical models in theoretical biology. This is followed by an introduction of the relevant properties of the cell and its main structures, with a special focus on the features most relevant to this work. A brief presentation of two new experimental techniques that allow a deeper understanding of the shape and internal structure of cells (two-photon imaging and multiple epitope ligand cartography) is then given, followed by a review of the already available theoretical models used to simulate the sub-cellular structure.

In chapter 3, an introduction to the sub-cellular model developed during this work is presented. It starts with an introduction to the relevant properties of the Delaunay triangulation and its dual, the Voronoi tessellation, followed by a presentation of the interaction potentials chosen for the interactions between the cell's sub-particles. The equations of motion and the method chosen to solve the dynamics of the system are then given.

From chapter 3 it gets clear that a method to define the neighborhood relations between the membrane particles is needed, so that the membrane particles are re-triangulated in a surface. The introduction and explanation of this method is given in chapter 4. The method identifies a rough surface (called quasi-surface), that contains all connections belonging to the surface to be re-triangulated plus some that do not. From that surface, smaller independent structures (called clusters) are selected. Each of the clusters is then treated and re-triangulated independently. To re-triangulate a cluster it is necessary to identify and order the vertices belonging its boundary. At last, two re-triangulation algorithms are proposed, each one adequate to a different class of clusters.

Chapter 5 contains examples and explanations about the initial configurations generated using the potential and interactions described in chapter 3, as well as the exact parameters chosen for these interactions for the results presented in this work.

Chapter 6 present the tests performed with the model and the results obtained with it. The setup used for performing the simulations is given, and the performance of the model with relation to time and the involved parameters is discussed. Additionally, examples of re-triangulations are presented.

Finally, in chapter 7, the advantages and disadvantages of the proposed model are discussed, as well as its range of validity. At last, an outlook is given, where potential developments and improvements for the method are proposed.

Chapter 2

Introduction

Contents

2.1 Modeling techniques in biology

Even when restricted solely to the field of Immunology, biological systems vary by orders of magnitude in size and time scale. Typical behaviors of the systems can also vary widely. Movement, for example, can be active or passive, random or directed. In order to account for the extremely different, complex behaviors, many different mathematical models were developed along the time.

These mathematical models vary depending on the degree of representation of both internal and physical space of the system in interest, and can be separated into two large groups: continuum models and discrete models (see figure 2.1). Continuum models have low internal representation of the system, while discrete models have high representation of the system's internal state. In continuum models the entire system can be characterized by using continuous variables, while in discrete models the internal state of the simulated system is at least partially characterized by discrete variables. The higher the representation of internal or physical state, the more complex the model.

The evolution of continuum models is usually given by differential equations. At the bottom left of figure 2.1 are ordinary differential equations, with low representation of both internal and physical state. Ordinary differential equations are used, for example, to model population dynamics. At the bottom right of figure

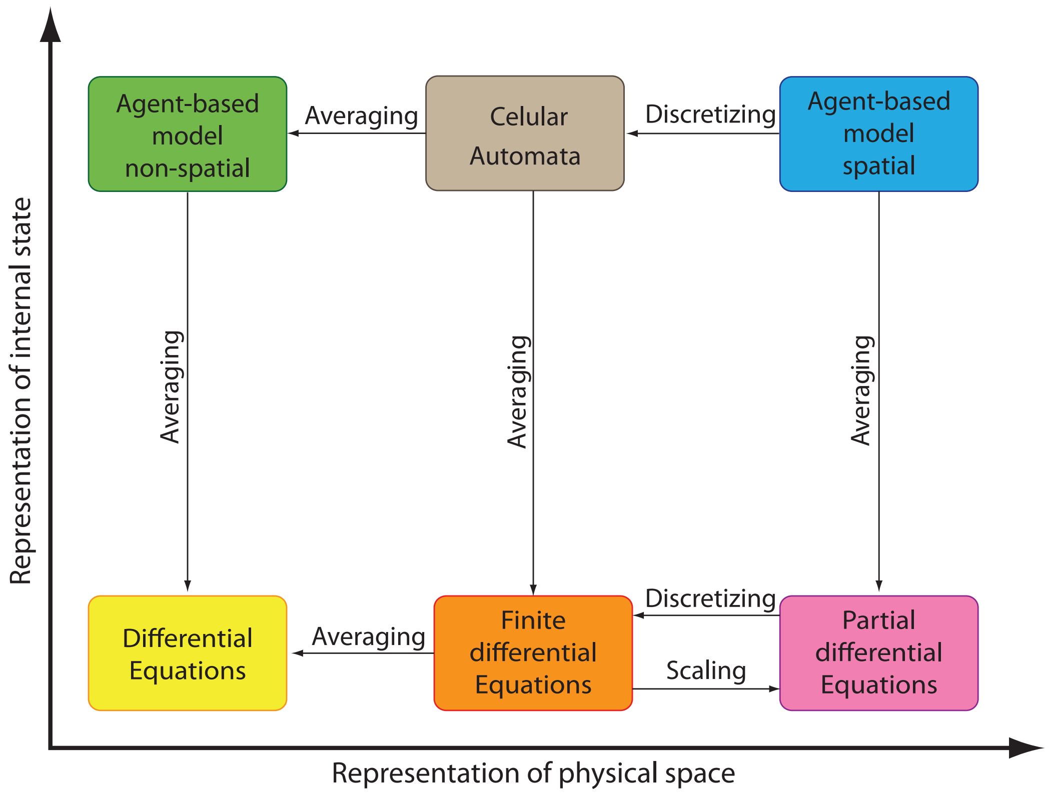

Agent-based model non-spatial
Averaging
Celular Automata
Discretizing
Agent-based model spatial
Representation of internal state
Averaging
Averaging
Averaging
Differential Equations
Averaging
Finite differential Equations
Discretizing
Scaling
Partial differential Equations
Representation of physical space

Figure 2.1: Mathematical models in theoretical biology.

2.1 are partial differential equations, used mostly for the study of density dynamics. They offer a high representation of physical space, but still only a low representation of internal state. A good review of continuum models can be found in [Murray 2002, Murray 2003].

When a high representation of the internal state of a system is needed (for example when cells are to be spatially followed) one must use agent-based models. In these models, each agent in the system is a discrete particle. If the model has no physical representation, it is called non-spatial. If the spatial representation is needed, the agent-based model is called off-lattice (see top of figure 2.1).

A special class of agent-based models is the so called cellular automata. Cellular automata models have both a representation of internal and physical state, but some of the system variables are discretized in a lattice. Because of the use of a lattice, space is also discretized in these models. The concept of cellular automata was first introduced by John von Neumann [von Neumann 1966].

The interaction between objects in a cellular automata model is usually given by a set of rules. However, it can be quite difficult to draw connections between these rules and physically measurable quantities. Additionally, cellular automata models give sometimes rise to artifacts coming from the use of a lattice, and the stochastic counter-strategies often employed make this connection even more difficult. A more recent review can be found at [Wolfram 1994].

Off-lattice agent-based models are often computationally more demanding than other models. However, they are extremely useful when cells or other agents in a system must be followed individually and spatially. Additionally, the interactions between objects is usually motivated by physical interactions, making off-lattice agent-based models more easily connected with physical variables than the cellular automata.

2.2 The cell

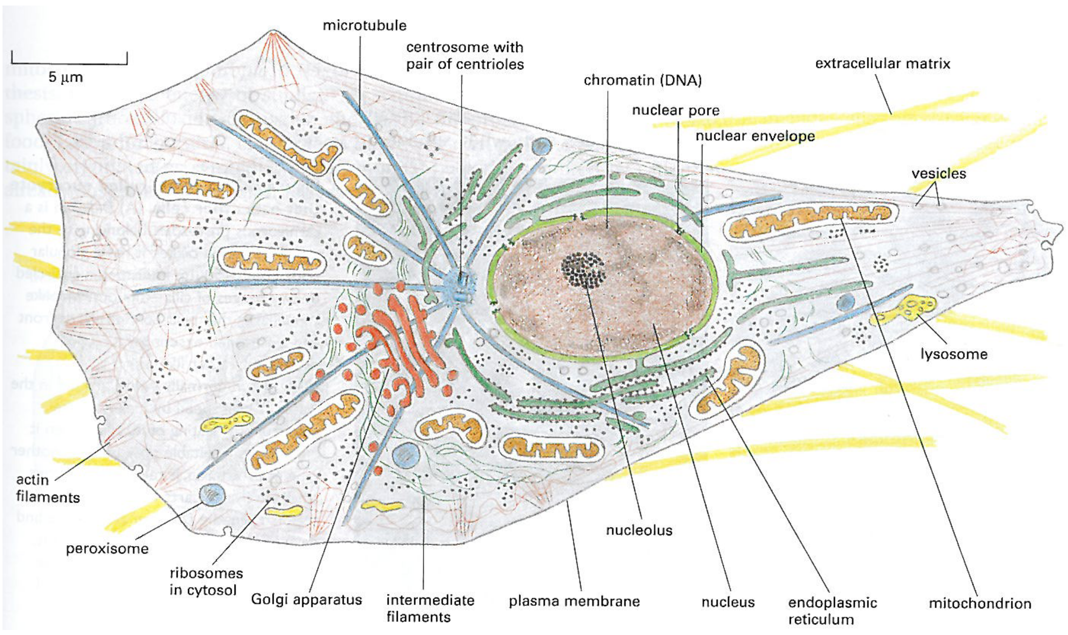


Figure 2.2: A schematic picture of a cell and its internal structure. Picture from [Alberts 2002]

The system of interest in this work is a cell. Cells are the fundamental structural and functional units of living organisms, and had been extensively studied experimentally. Cells can be divided into two main groups: eukaryotic and prokaryotic. Prokaryotic cells have a simpler structure and are usually independent, while eukaryotic cells are often part of a multicellular organism. Eukaryotic cells are, in general, bigger and more elaborate than prokaryotic cells. As the immune system is the main subject of study in the Systems Immunology group at FIAS, where this work was developed, emphasis will be given to the description of eukaryotic cells.

By definition, eukaryotic cells keep their DNA in an internal compartment, the nucleus, separated from the cytoplasm by a double layer of membrane. Eukaryotic cells are, typically, 10 times bigger in linear dimension and 1000 times larger in volume than a prokaryotic cell. Additionally, they have a cytoskeleton, i.e., a system of protein filaments that function as a scaffold to the cell, forming a system of girders, ropes, and motors that gives the cell mechanical strength, controls its shape and drives and guides its movements.

Apart from the nuclear envelope, there exist many other organelles inside the cell, separated from the cytoplasm by internal membranes structurally similar to the plasma membrane. Many of these organelles are involved in processes related to digestion and secretion. Additionally, animal cells (and the free-living eukaryotic cells called protozoa) can change their shape rapidly and engulf other cells and small objects by phagocytosis.

In the next subsections a small overview about the cell membrane, the cytoplasm and the cytoskeleton will be given. This overview is mainly based on information from [Alberts 2002], one of the main reference books in cell biology.

2.2.1 The cell membrane

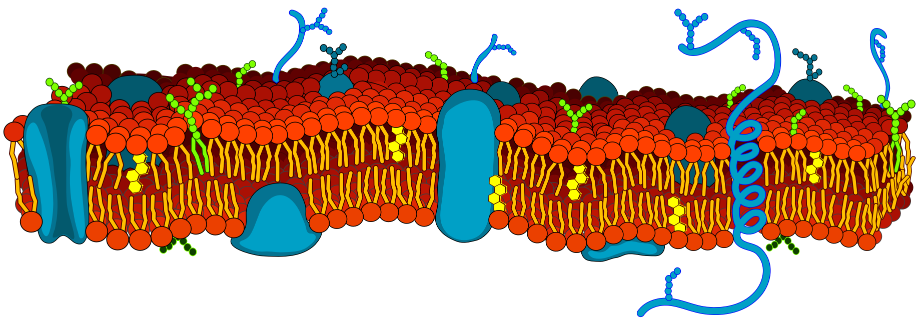


Figure 2.3: A schematic picture of the cell membrane. The lipid bilayer is drawn in yellow and red, the blue and green objects are proteins (the figure is on the public domain).

The cell membrane is a fundamental structure to the life of a cell. The plasma membrane encloses and defines the cell boundary, allowing the cytosol to have characteristics essentially different from the extracellular environment.

The cell membrane consists of a very thin film of lipid and proteins. The lipid molecules are structured as a continuous double layer approximately 5 nm thick. The lipid bilayer behaves as a two-dimensional liquid, and individual lipid molecules are able to diffuse rapidly laterally within their monolayer. The lipid bilayer serves also as an impermeable barrier for the cell, impeding the passage of most water soluble molecules.

There are three main classes of lipid molecules: phospholipids, cholesterol and glycolipids, with the phospholipids being the most numerous. The lipid composition of the two monolayers of the lipid bilayer can vary significantly. This asymmetry reflects the different functions between the internal and external layers of the cell.

There are approximately 5×10^6 lipid molecules in a 1 μm $\times$ 1 μm area of the lipid bilayer, or around 10^9 lipid molecules in the plasma membrane of a small animal cell. All lipid molecules in the cell membrane have a hydrophilic (or polar) and a hydrophobic (or nonpolar) end.

The mass of the cell membrane is about 50% constituted of lipid molecules. The other half is almost entirely constituted of proteins. Proteins are responsible for almost all the membrane functions, from the transport of specific molecules to the catalysis of membrane-associated reactions. They serve as specific receptors, enzymes, transport proteins and etc. Many proteins extend across the lipid bilayer. Some proteins also serve as structural links, connecting the cytoskeleton through the lipid bilayer to the extracellular matrix or to an adjacent cell. Others function as receptors to detect and transduce chemical signals in the cell's environment.

Many membrane proteins are able to diffuse rapidly in the plane of the membrane, much like the lipid molecules. However, cells have mechanisms to immobilize specific proteins and to confine both membrane proteins and lipid molecules to particular domains in the lipid bilayer. Proteins in the cell membrane can also form clusters, or a cross-linked array of proteins (for example, the complex of proteins that make the connections between actin filaments and the cell membrane).

Additionally, lipid bilayers are impermeable to water soluble molecules. Therefore, the cell membrane must contain transport proteins, in order for these molecules to be transported into and out of a cell. Transport proteins can belong to two different groups: carriers and channels.

2.2.2 The cytoplasm

The cytoplasm is everything in a cell that is enclosed by the cell membrane, with the exception of the nucleus and its contents. It consists of cytosol and the cytoplasmic organelles suspended in it. While prokaryotic cells generically consist of a single intracellular compartment surrounded by the plasma membrane, eukaryotic cells are subdivided into functionally distinct, membrane-enclosed compartments.

2.2.2.1 The cytosol

The term cytosol is used to refer to the liquid phase of the cytoplasm in an intact cell. It is a complex mixture of cytoskeleton filaments, dissolved molecules, and water that fills a little more than 50% of the volume of a cell. The cytosol is a gel, with a network of fibers dispersed through water. Additionally, most of the cell's intermediary metabolism, i.e., the many reactions by which some small molecules are degraded and others synthesized to provide the building blocks for macromolecules, happens on the cytosol.

An animal cell contains about 10 billion proteins of 10000–20000 different kinds, and the synthesis of almost all of them begins in the cytosol. Each protein is then delivered specifically to the cell compartment that needs it.

2.2.2.2 Organelles in eukaryotic cells

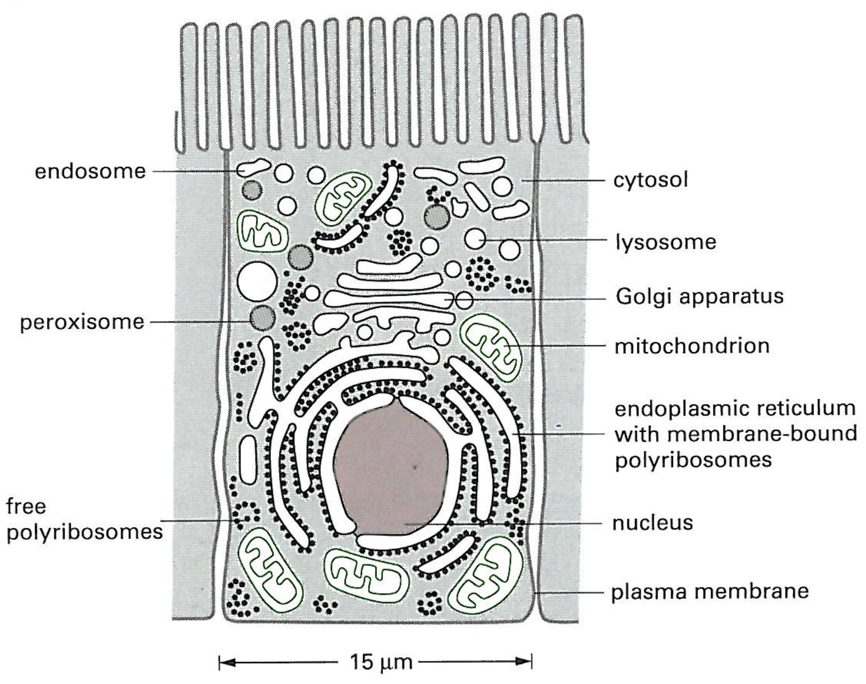


Figure 2.4: Schematics of an eukaryotic cell showing its main organelles. These organelles are briefly described in 2.2.2.2. Picture from [Alberts 2002]

The organelles that are part of the cytoplasm are usually separated from the cytosol via a membrane not unlike the cell membrane. All eukaryotic cells have the same basic set of membrane-enclosed organelles.

Although many of the biochemical processes in a cell take place in or on the membrane surfaces, intracellular membrane systems do more for the cell than just provide increased membrane area. They create enclosed compartments that are separate from the cytosol, providing the cell with functionally specialized aqueous spaces.

Nucleus: The nucleus contains the main genome and is the principal site of DNA and RNA synthesis.

Endoplasmic reticulum About half the total area of membrane in a eukaryotic cell encloses the endoplasmic reticulum (ER). The ER has many ribosomes bound to its cytosolic surface. The ribosomes are engaged in the synthesis of membrane proteins, that will either be secreted to the cell exterior or sent to other organelles. The ER also produces most of the lipid for the rest of the cell, and functions as a store for Ca^{2+} ions.

The Golgi apparatus The Golgi apparatus consists of organized stacks of compartments called *Golgi cisternae.* It receives lipids and proteins from the ER and dispatches them to a variety of destinations.

Mitochondria Mitochondria take up oxygen and harness energy from the oxidation of food molecules to generate most of the ATP used by cells to drive reactions that require an input of free energy. They contain their own genetic material, and probably evolved from bacteria that were taken up into the cytoplasm of the eukaryotic cells and survived as symbionts.

Lysosomes and endosomes Lysosomes contain digestive enzymes, used to degrade dead intracellular organelles, macromolecules and particles taken in from outside the cell by endocytosis. Before arriving at the lysosomes, endocytosed material must first pass through a series of organelles called endosomes.

Peroxisomes Peroxisomes are small vesicular compartments that contain enzymes utilized in a variety of oxidative reactions.

In general, each membrane-enclosed organelle performs the same set of basic functions in all cell types. Depending of the specialized functions of the cells, however, organelles can vary in quantity and might obtain additional properties specific to that cell type.

2.2.3 The cytoskeleton

The cytoskeleton is a network of protein filaments that spatially organizes the cytoplasm. It is the structure responsible to keep cells correctly shaped and physically robust, and allow cells to organize themselves in space and to interact mechanically with their environment. It is also responsible for the ability of some cells to change their shape and move from a place to another. Additionally, the cytoskeleton is behind the rearrangement of the cells' internal components as they grow, divide and adapt to a changing environment. It drives and guides the intracellular traffic of organelles from one part of the cell to another, gives support to the cell membrane and provides the mechanical linkages that let the cell bear stresses and strains without being ripped apart as the environment shifts and changes. An important function of the cytoskeleton not discussed here is its role in mitosis. During mitosis, the cytoskeleton is responsible to pull the chromosomes apart and to split the cell in two.

The many different functions of the cytoskeleton come from the behavior of three families of proteins, which assemble to form three main types of filaments: microtubules, actin filaments and intermediate filaments. Each type of filament has distinct mechanical properties and dynamics.

Microtubules: Strong, rigid, hollow tubes. They determine the positions of membrane-enclosed organelles and direct intracellular transport.

Actin filaments: Determine the shape of the cell's surface and are fundamental for cell movement. They are very thin, hard to stretch and easy to break.

Intermediate filaments: Provide the cell with mechanical strength and resistance to shear stress. They are easy to bend but hard to break.

All three types of filaments form as helical assemblies of subunits that self-associate using a combination of end-to-end and side-to-side protein contacts, and exist as a result of a flow-equilibrium of the assembling and de-assembling of these subunits. The differences in the structures of the subunits, as well as the way they self-assemble, are responsible by the variety of mechanical properties of the filaments. The combination of the properties of these three groups of filaments with a large number of accessory proteins linking the filaments to each other and to other cell components is what makes the cytoskeleton so useful and important to a cell.

The large-scale structures in the cytoskeleton are very dynamic, and can change or persist according to the cells need. Since the individual components that form the filaments are constantly moving and changing, a structural rearrangement in a cell does not require a lot of extra energy. The time scale for the duration of a large-scale structure in a cell ranges from less than a minute up to the cell's lifetime.

2.3 Experimental landscape

Two relatively recent developments in experimental methods are of special interest in the context of this work. The first, two- (or multi-) photon imaging, is a new microscopy method that allows tracking of cell shape and movement inside living animals, generating data supposed to mirror the real behavior of cells in vivo [Denk 1990, So 2001, Helmchen 2005]. The second, named multi epitope ligand cartography (MELC), is a robotic immunofluorescence microscopy system able to identify up to hundred molecules inside a cell at a determined point in time [Schubert 2003, Schubert 2006].

The framework developed during this work has the potential to be used to simulate both cell shape derived from two-photon imaging data and the tracking of molecules inside a cell as obtained by multi epitope ligand cartography.

2.3.1 Two-photon image data

Two-photon fluorescence microscopy (combined with in vivo fluorescence labeling techniques) allows three-dimensional imaging of biological specimens and tracking of cell movement in living organs with only minimum disturbance (see figure 2.5). Therefore, the behavior of the cells can be considered to be representative of real in vivo behavior. Two-photon microscopy can also be used to study the progression of diseases as Alzheimer and the development of tumors. The method was invented by Denk et al. [Denk 1990], and revolutionized three-dimensional in vivo imaging of cell and tissues. A good introduction and a great review in two-photon microscopy can be found respectively in [So 2001] and in [Helmchen 2005].

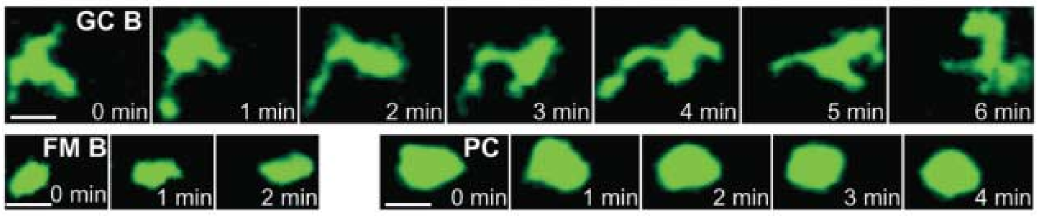


Figure 2.5: Time-lapse images from two-photon microscopy showing the morphology of a B cell from the germinal center, a B cell from the follicular mantle, and a plasma cell. The scale bars are 10 μm. Image from [Allen 2007].

Two-photon microscopy relies on a non-linear process called fluorescence excitation by two-photon absorption. This process occurs when two photons arrive simultaneously (within ~ 0.5 fs) at a molecule, combining their energies and bringing the molecule to an excited state. The sum of the energy carried by the two photons must, of course, be at least equal to the energy gap between the molecule's ground and excited states. The excited molecule then proceeds along the normal fluorescence emission.

The transition probability to an excited state via two-photon absorption is extremely low at normal light intensities. To make the transition probability higher, the light source has to be concentrated in space and time, i.e., many photons must arrive at the same place at the same moment. In order to achieve an efficient excitation, a light source with intensity on the order of 10^{10} to 10^{12} Wcm^{-2} is needed. This required radiance level can be obtained by a light from a 1W continuous-wave laser focused to a 10^{-9} cm^2 diffraction-limited focal volume. The most common light sources are lasers that provide ultrafast (width around 100 fs) pulses at the high repetition rate of about 100 MHz. These lasers can be quite expensive.

Despite its high price, two-photon microscopy has several advantages over traditional confocal microscopy. First, since the quantum energy of two photons is combined, the transitions excited have higher energies than the excitation light. This means that, even though the multi-photon absorption occurs in the near-infrared wavelength range (700 to 1000 nm), the photon emission occurs in the visible spectral range. This is an advantage because the ability of light to penetrate scattering tissue increases with its wavelength and, in general, near-infrared light is less phototoxic, since most tissues lack significant endogenous absorbers.

Additionally, a fundamental property of two-photon microscopes is depth discrimination. This means that, when the laser beam is focused through the microscope objective, the absorption of multiphotons is spatially confined to a region about 1 μm thick around the focal point (perifocal region), where the photon density is high. Depth discrimination occurs because the signal depends supralinearly on the density of photons. The fluorescence that is excited outside the perifocal region is virtually negligible, further reducing photodamage in the tissue.

In figure 2.6 a schematic view of a two-photon microscope is shown. A laser source is used to generate near-infrared ultrashort pulses, followed by systems that allow the adjust of intensity and bean size. The laser is then coupled to the microscope. The laser beam passes then by scan lens (fS), tube lens (fT) and by an objective (fO), being then focused in the specimen. Two-photon excited fluorescence is isotropically emitted, and can be collected in epi- and/or trans-collection mode. In in vivo experiments epicollection is used exclusively. High-sensitivity detection electronics are used to ensure maximal detection efficiency and signal dynamic range.

2.3.1.1 The use of two-photon microscopy in biological systems

The ability of the two-photon microscopy technique to do deep imaging while reducing photodamage makes it a great tool to the study of biological systems. The method is nowadays often used for high-resolution imaging in various organs of living animals.

In order to be able to study a system with the technique, though, the structure (cells, molecules) of interest must be labeled. In order to do so, synthetic dyes must be inserted into the system. There exist a variety of methods that allow the introduction of these synthetic labels into a system: injection into the blood stream to label vasculature, filling of individual cells via recording electrodes, high-affinity binding of certain dyes to protein aggregates and so on. In immunology, the "design" of transgenic animals (mostly mice) expressing anatomical markers that have a widespread cell-specific labeling is the focus of dedicated experimental research.

The most common applications of two-photon imaging in the field of immunology are in the study of T cell dynamics inside the lymph nodes [Miller 2002, Stoll 2002,

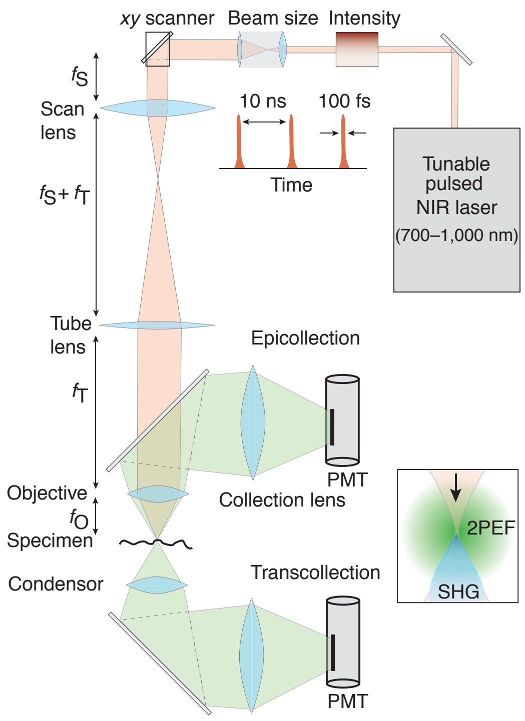


Figure 2.6: A schematic drawing of typical components in a two-photon microscope. This system consists of a high-peak-power pulsed laser, a high-throughput scanning microscope and high-sensitivity detection circuitry. Figure from [Helmchen 2005].

von Andrian 2002] and in the study of B cell dynamics inside germinal centers [Allen 2007, Hauser 2007, Schwickert 2007].

2.3.2 Multi epitope ligand cartography

The multi epitope ligand cartography (MELC) was first described by Schubert et al. [Schubert 2003, Schubert 2006]. It is an approach that allows the mapping of the sub-cellular location of hundreds of proteins in a single sample. A single fluorophore is used (with different antibodies) by repeated rounds of staining, imaging and photobleaching. Images are registered in a sequence, that later on will be combined in a single image of up to hundred proteins in the same tissue or cell (see figure 2.7).

The method is specially interesting because it gives access to information about the system studied that is complementary to the information obtained by traditional methods. While usual fluorescence microscopy methods provide temporal information and high spatial resolution for a maximum of five to ten proteins at a time, this method can provide the simultaneous observation of several layers of proteins and molecules, albeit without temporal evolution.

The procedure to map the dozens of different species or classes of molecules in morphologically intact cells is completely automated and requires a robotic immunofluorescence microscopy system. The technology uses large molecular libraries to detect and localize the many individual molecular species in cells using fluorescence. For that, a slide with a given specimen is placed on a inverted wide-field fluorescence microscope equipped with fluorescence filters. Fluorochrome-labeled antibodies and wash solutions are then added and removed from the sample robotically under temperature control, without any displacement of the sample or of the objective. In each cycle a pair of antibodies is added and phase contrast and fluorescence images acquired by a high-sensitivity cooled CCD camera. The sample is washed with PBS, bleached at the excitation wavelengths and post-bleaching phase contrast and fluorescence images are acquired. All data acquisition is fully automated. Each individual molecule in a cell is detected and registered as a spatial signal map, and aligned relative to other molecular signals in the same cell.

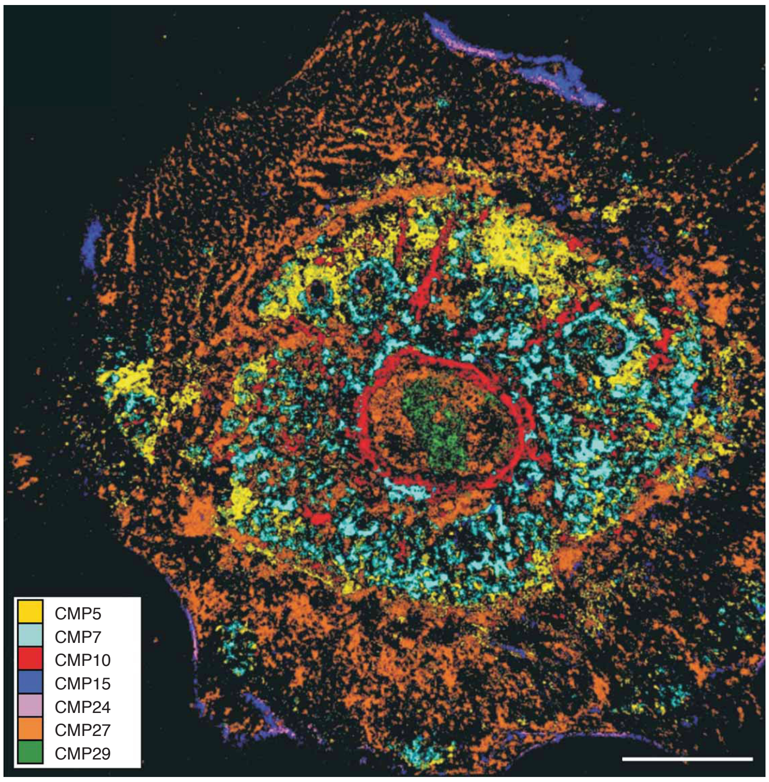


Figure 2.7: Toponome map of a primary human hepatocyte produced by mapping 15 molecules, defining a set of sub-cellular combinatorial molecular phenotypes and representing seven of them by different colors to display a set of mutually exclusive compartments. Figure from [Schubert 2006].

MELC analyses the topological order, i.e., the cellular organization, of the major molecular classes of a cell – proteins, carbohydrates, lipids, nucleic acids. It therefore addresses the organizational equivalent of genome and proteome in a cell, referred to as a toponome. A toponome contains all functional protein networks of a cell. The higher-complexity, higher-resolution information coming from automated microscopy can potentially allow training of substantially improved prediction systems, which in turn can be used to guide new experiments.

2.4 Simulating the internal structure of a cell

Although cells have been extensively studied, there are still many open questions about even their most basic behaviors. Additionally, until recently, there were only few theoretical works trying to make sense of biological systems. The field of theoretical biology is still very young.

The long term goal of this work (that certainly surpasses the scope of this thesis) is to provide a theoretical model for a cell that can help clarify basic questions related to the mechanisms behind cell shape and movement. How do cells change shape? How do cells move? These questions are definitely connected with each other, as cells need to change shape in order to move [Alt 1995].

The cell's sub-cellular structure is certainly fundamental for its ability to change shape and move. Particularly important is the cell's cytoskeleton. The cytoskeleton works as a scaffold for the cell.

In order to make a model for a cell capable of giving some insight on its shaping fundamental properties, the cell's internal structure has to be taken in consideration. Models that offer no resolution of the cell's internal structure (continuum models, for example, or even agent-based models that represent each cell as a single object) will certainly fall short off this task.

Left are agent-based models where each cell is represented by a group of objects. In this case, each of these objects can carry different properties and the interaction between these objects can give rise to emerging behaviors that, in the best of the hypothesis, mimic the behaviors observed in cells.

2.4.1 Sub-cellular models

There exist a few theoretical models aiming at describing cell shape and migration taking into consideration its sub-cellular level. A brief review of these models will be given in this section.

2.4.1.1 The extended Potts model

The extended Potts model [Graner 1992, Glazier 1993] (also known as cellular Potts model) was developed by James Glazier and François Graner in 1992, as an extension of large-Q Potts model simulations of coarsening in metallic grains and soap froths, with the objective to simulate the sorting of a mixture of two types of biological cells (see figure 2.8). Nowadays this is the most traditional model for the description of cell shape.

The model is a generalization of the Ising model with multiple spin states. It discretizes the continuous cellular pattern on a lattice, and each lattice site has a spin associated to it. A separate spin is associated to each cell in the pattern. Each cell is then defined as all volume elements that have the same spin state, and therefore need not to be simply connected. Additionally, each cell has also a cell type associated to it.

In order to obtain the Hamiltonian used, the following properties have to be taken into consideration: bonds between lattice sites with a similar spin have energy zero (which means that the energy inside a cell is zero). At cell boundaries, there is a cell-type-dependent surface energy. In addition, since biological cells generically have a fixed range of sizes, an elastic term and a fixed target area (that may depend on cell type) must also be included.

Changes in shape and, therefore, the movement of cells are governed by thermodynamic interactions. The following Monte Carlo algorithm is used: at each time step in the simulation, a lattice site is randomly selected. The spin value of the selected site is then converted to the spin value of a (also randomly chosen) neighboring site with a probability dependent on the variation on the energy of the system. The spin is always changed when the system is driven to a configuration with smaller energy. However, if the configuration ends up having a bigger energy, the exchange is done according to the probability $exp(-\Delta\mathcal{H}/kT)$, where $\Delta\mathcal{H}$ is the energy variation, k is the Boltzmann constant and T is the temperature of the system.

The Potts model is very simple and yet to a certain extent realistic, in that the position and diffusion of the membrane determine the dynamics of the system, as they do for real loosely aggregated cells. Relative contact energies and boundary curvatures drive all motion, and thus vertices are always close to their equilibrium conditions.

However, a change of the cells' shape is always correlated to a change of the cell's volume, since a subunit on the lattice has to change its spin state to the one of its neighbor, if the cell is to move. These volume fluctuations are averaged out if longer time scales are considered and a volume conserving potential is included in the energy entering the Boltzmann law. Therefore, these unphysical changes in cell volume associated with cell movement are negligible in the limit of many subcellular nodes per cell. On the other side, in order to apply the Potts model to cells with a small number of subunits, additional rules are needed.

2.4.1.2 The Hyphasma

An alternative lattice model architecture which intrinsically includes the one- and the multi-subunit limit was developed by Michael Meyer-Hermann and Philip Maini in 2005 [Meyer-Hermann 2005]. The model was originally used to describe lymphocyte migration in secondary lymphoid organs, in order to interpret the results found with the experimental method of two-photon imaging.

The intrinsic inclusion of the one- and multi-subunit limit on the model allows

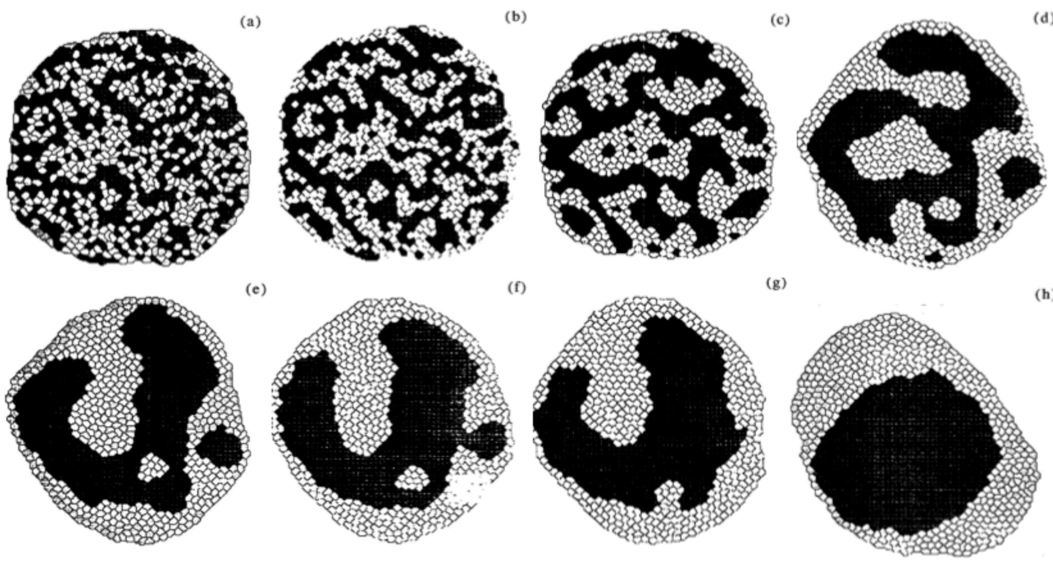


Figure 2.8: Example of cell sorting simulated using the Potts model. Different stages of the simulation are shown. Figure from [Glazier 1993].

the possibility of determining the lattice constant by the size of the smallest cell in the studied system, that then can be described as a single volume element. This provides faster computation when one is interested only in differences in volume (and not in the substructure of the cells).

The method uses a mechanism of cell movement that conserves the total volume of the cell when no growth or shrinking is intended. All reaction kinetics are formulated as reaction rates and actions are taken according to probabilistic decisions. Additionally, the physical movement of the cells is formulated in terms of forces acting on subunits of the cell. Cell objects are represented by the cell volume, the cell polarity, a list of cell subunits, and internal velocity states. As only the movement of free cells was examined, the cell velocity from the experiments was used directly as input for the model. Additionally, an overdamped approximation was used in order to calculate the cell displacement.

The kinetics of the cell subunits is based in two types of velocity: undirected active movement with persistence of orientation and cell reshaping. The direction of the active movement is determined by the polarity of the cell. This polarity is assumed to change randomly with a probability per time step that represents the persistence time.

For the active movement of the cell, the barycenter of the cell is virtually shifted in the direction of the polarity to the border of the cell. Then every subunit rep-

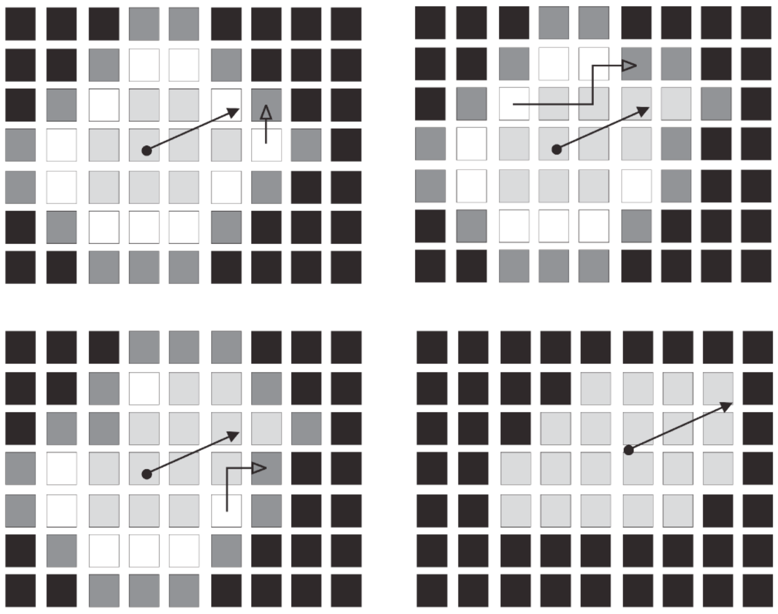


Figure 2.9: A schematic description of how the cell movement works in the model. The current barycenter is denoted by the dot, and the cell polarity by the arrow attached to it, that also points to the position of the virtual barycenter at the border of the cell. The cells subunits in light grey are immobile, the ones in white belong to the border of the cell and may be moved. Free target points are shown in dark grey and other points in black. On the top left the starting position is shown. Random white cells are chosen and moved towards the target point nearest to the virtual barycenter, one by one, until all white cells have been moved or are not allowed to move anymore. A possible final configuration is shown on the lower right panel. Figure from [Meyer-Hermann 2005].

resenting a border point of the cell is randomly moved towards free lattice points near the new barycenter. The movement is suppressed in case the movement of a border subunit would cause the subunits of the cell to disconnect from each other. The displacement of subunits is stopped when either no border subunit remains to be moved or the barycenter of the cell has been displaced by one lattice constant (see figure 2.9).

All forces that reshape the cell towards a sphere, i.e., hydrostatic pressure, reduced actin filament assembly, actomyosin contraction, or membrane surface tension, are included in a single reshaping force. This force is implemented as an overall elastic force that drives the subunits of an elongated cell back to the current barycenter and promotes a spherical shape.

A nice feature of this model is that it generically couples cell deformation and cell displacement in a way that is quantitatively consistent with two-photon imaging data of lymphocytes.

2.4.1.3 The sub-cellular element model

Off-lattice models, widely used in the simulation of multi-cellular systems [Meineke 2001, Beyer 2008, Bock 2009, Galle 2008], are also often applied to sub-cellular dynamics. In 2005, a model designed to allow the simulation of large numbers of cells, but still taking in consideration cell-shape dynamics with the use of sub-cellular particles, was introduced by Timothy Newman [Newman 2005]. In this model (named Sub-cellular Element Model) cells are not confined to a lattice, being allowed to occupy any position on space.

In the model, each cell is composed of a determined number of sub-cellular particles (see figure 2.10). Chemical signaling is not included in the system, and the cells interact to each other via local biochemical interactions. The position of each sub-cellular element changes according to three processes: a weak stochastic component, and elastic response to intracellular biomechanical forces and an elastic response to intercellular biomechanical forces. Additionally, the movement is considered to be overdamped. The dynamics of the elements is described by Langevin equations.

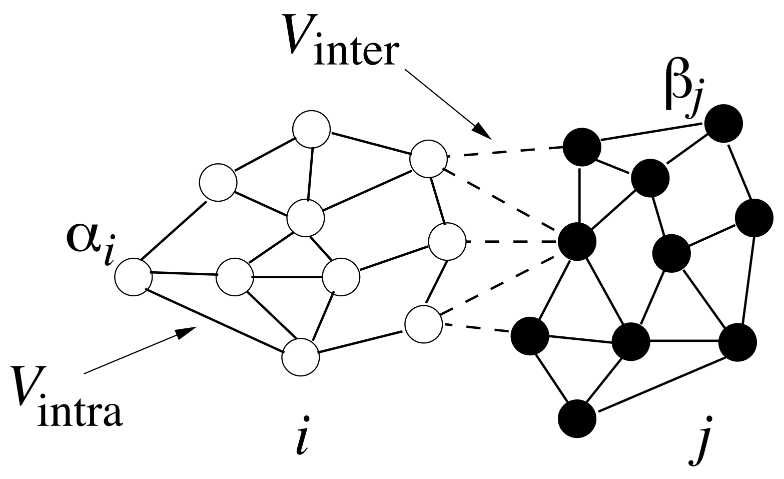


Figure 2.10: Two cells (i and j) interacting according to the sub-cellular element model. White elements belong to cell i and black to cell j. Intracellular interactions are represented by solid lines, while intercellular interactions are represented by dashed lines. α_i is an element of cell i, β_j an element of cell j, V_{intra} the intracellular potential and V_{inter} the intercellular potential. Figure from [Newman 2005].

The intra- and inter- cellular interactions are characterized by phenomenological potentials. All relevant biological details have to be encoded into these two potentials. Since the choice of the potentials is purely phenomenological, some discussion of the biological motivation for the choice is necessary.

The method provide efficient procedures to update neighborhood relations. An interesting application in cell rheology of this model was recently published [Sandersius 2008].

2.4.1.4 The tensegrity model

An alternative approach to include the cells internal structure in a model to describe cell shape and properties is given in [Ingber 2003a, Ingber 2003b]. In this approach a cell is approximated by a tensegrity structure, where tensegrity structures are defined as systems that stabilize their shape by continuous tension ("tensional integrity") rather than by continuous compression.

The idea was to create a cell model that relates mechanics to chemistry at the molecular level, and that can be translated into mathematical terms. Tensegrity includes two broad structural classes: prestressed and geodesic. Both fail to maintain shape stability when mechanically stressed without continuous transmission of tensional forces. These tensional forces are supposed to be sustained by cytoskeletal microfilaments and intermediate filaments, and balanced by internal microtubules struts and extracellular matrix (ECM) adhesions.

Experimental validation of the cellular tensegrity model requires demonstration of three major behaviors of living cells: that they behave mechanically as discrete networks composed of different interconnected cytoskeletal filaments, that cytoskeletal prestress is a major determinant of cell deformability, and that microtubules function as compression struts, and act in a complementary manner with ECM anchors to resist cytoskeletal tensional forces.

In an article by Patrick Cañadas at al. from 2002 [Canadas 2002], this mathematical model is used for an analysis of the structural viscoelasticity of the cytoskeleton. For the simulation, a 30-element tensegrity structure was modified in order to consider the viscoelastic properties of cables (figure 2.11). Cables were assumed to behave like viscoelastic Voigt bodies (elastic element in parallel with viscous dashpot). Additionally, the Young modulus and cross-sectional area that characterize the cables and bars were kept constant, while the viscosity modulus of cables and the length of bars were allowed to vary.

The viscoelastic response of the system was studied over a large range of initial states of overall deformation by solving a system of differential equations where the external-forces vector was related to both the nodal-displacement vector associated with a global-rigidity matrix and the rate of nodal-displacement vector associated with the global-damping matrix. This equilibrium equation system was solved for small variations of force and displacement hypothesis, considering a linear incremental method.

Overall, the results supported the idea of a structural origin to cellular viscoelasticity, and confirmed the biological relevance of the author's model in this specific problem.

2.4.1.5 Towards a new sub-cellular model

The model presented in the next chapters of this thesis embraces the strong features of the aforementioned models into a single modeling framework. It is an off-lattice agent-based model and can be run with any sub-cellular resolution. Many particle interactions are used to define a cell. Neighborhood relations are defined via a

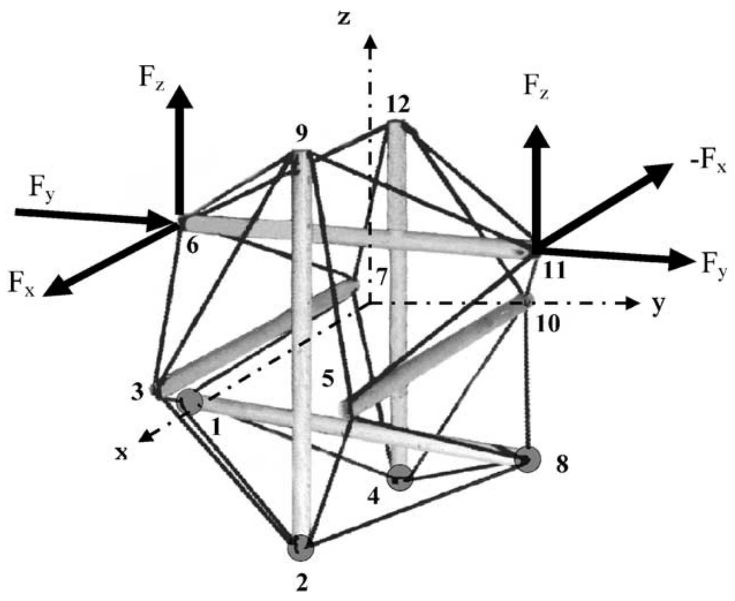


Figure 2.11: 30 element tensegrity structure (6 bars and 24 viscoelastic cables) with the external forces to be applied. Figure from [Canadas 2002].

Delaunay triangulation and the interacting particles have the possibility to hold specific functions, like being part of the cell membrane or represent an organelle. It is suitable to simulate single cells, a few interacting cells or multi-cellular systems.

The use of Delaunay triangulations allows a faster update of the neighborhood relations between the interacting particles in relation to traditional off-lattice methods. Some methods from molecular dynamics (like the one used in [Newman 2005, Sandersius 2008]) also provide efficient procedures to update neighborhood relations, as long as the motility of the cells is limited.

Delaunay triangulations have often been used in Biology, usually in the simulation of systems where many cells interact with each other (for example tissues, primary lymphoid follicles formation) [Meineke 2001, Beyer 2008]. The model described here is an extension on a framework developed in the last years [Schaller 2004, Beyer 2005]. The program generates kinetic (moving vertices) and dynamic (changing number of vertices) Delaunay triangulations for a set of agents, being particularly useful for the simulation of evolving biological systems, where the number of agents varies and at every time step the neighborhood relations must be quickly adjusted.

This framework was extended by adding the possibility that particles interact only in a subset of the defined interaction relations, more specifically in a subset of the Delaunay triangulation restricted to a two-dimensional surface embedded in three-dimensional space. The model will be applied to define the interaction relations between membrane sub-particles of a three dimensional cell. The details of the mathematical implementation will be discussed in chapter 3.

Chapter 3

The model

Contents

The goal of this work was to develop a framework for the simulation of cells in an agent-based fashion. The intended framework was designed so that it would include the following desired properties:

1. It should take the internal structure of the cells into consideration. Practically, this means that some resolution inside the cell is needed. Therefore, each cell cannot be represented by a single particle, but should be simulated as a set of sub-particles. The sub-particles, naturally, need to interact to each other. This leads to the next desired property of the framework:

2. The interaction relations between the sub-particles should be defined via a Delaunay triangulation. Many reasons were involved in this decision. First, Delaunay triangulation is an efficient method to define neighborhood relations between particles, and it is widely used in the simulation of multi-cellular systems [Meineke 2001, Beyer 2008, Bock 2009, Galle 2008], usually with each cell in the system simulated as a single point of the triangulation. Second, former members of the Systems Immunology group at FIAS developed during the last years a framework to calculate kinetic (i.e., with the ability to handle moving vertices) and dynamic (allowing the insertion and deletion of vertices) Delaunay triangulations [Schaller 2004, Beyer 2005, Meyer-Hermann 2008]. Therefore, one of the goals of the project was to expand the already existing framework to facilitate the simulation of sub-cellular structures.

3. The framework should be flexible. It should be easy to change the number of sub-particles in a cell, as well as to increase the complexity of the model (for example, with the inclusion of cellular organelles) according to the complexity needed for the system in consideration.

4. The minimum setup of the framework should already include two basic types of sub-particles, distinguishing the particles belonging to the membrane from the particles that make up the internal structure of the cell.

In this chapter the basic mechanisms of the framework will be explained, as well as the necessary mathematical background. In section 3.1 an introduction to the Delaunay triangulation (as well as to its dual, the Voronoi tessellation) will be given. In section 3.2 the interaction potentials used during dynamics will be introduced and justified, and in section 3.3 the equations of motion used throughout this work will be presented. Finally, in section 3.4 the technique used to solve the dynamics of the system will be explained.

3.1 The Delaunay triangulation

The Delaunay triangulation of a given set of points P is a special triangulation that fulfills a set of mathematical criteria. In two-dimensions, it triangulates the set of points in such a way that no point in the set is inside the circumcircle of any triangle in the triangulation. Both the Delaunay triangulation and its dual, the Voronoi tessellation, are well covered topics in the literature (specially in two-dimensional space). A good review, as well as the fundamental definitions and properties of Delaunay triangulations can be found in [Okabe 2000]. Additionally, detailed explanations about the numerical implementation of a kinetic and dynamic triangulation supporting insertion, removal and movement of spheres was also throughly discussed in [Schaller 2005, Beyer 2007]. In this thesis, therefore, only a simple introduction to the subject and the information relevant for understanding the model will be given.

3.1.1 The Voronoi tessellation

Given a sample of points P, a Voronoi cell can be defined for every point $\mathbf{p}$ in the sample as all points in $\mathbb{R}^3$ nearer to this point than to any other point $\mathbf{p}'$:

$$V_{\mathbf{p}} = \{x \in \mathbb{R}^3 : |x - \mathbf{p}| < |x - \mathbf{p}'|, \forall \mathbf{p}' \in P, \mathbf{p}' \neq \mathbf{p}\}. \tag{3.1}$$

A Voronoi tessellation (see figure 3.1) is defined as the set of Voronoi cells for all points of a sample P, and decomposes the $\mathbb{R}^3$ in convex polyhedrons.

3.1.2 Defining the Delaunay triangulation

The Delaunay triangulation (see figure 3.1) of P is the dual graph of the Voronoi tessellation. Every point in P is a Delaunay vertex, and (in three dimensions)

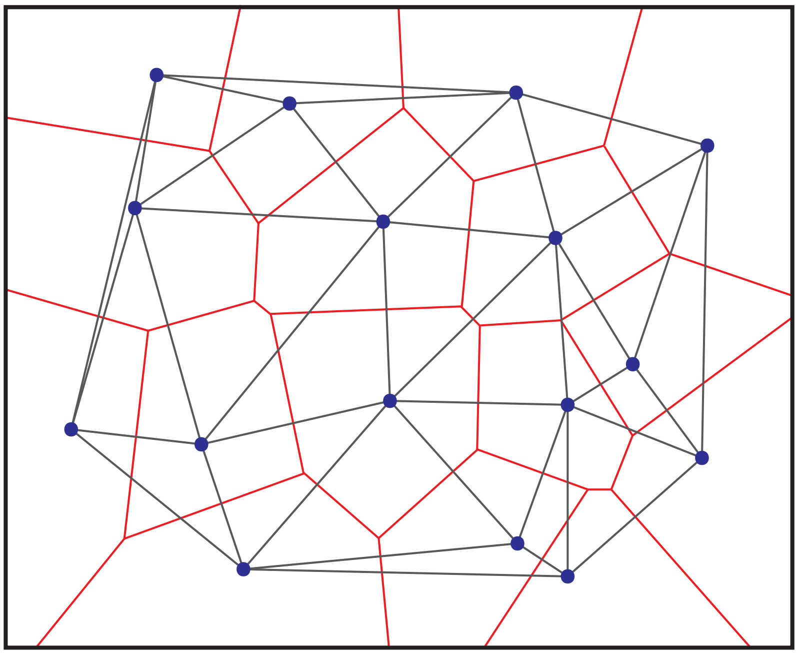


Figure 3.1: Voronoi tessellation and Delaunay triangulation of a sample P of points in two-dimensions. The round dots are the points in the sample P, with the connections between them forming their Delaunay triangulation. All other drawn connections belong to the Voronoi tessellation of the sample.

a simplex is defined as the convex hull of four points in P whose corresponding Voronoi cells intersection is non-empty. Starting with the Voronoi diagram for a sample P, the Delaunay triangulation of the sample can be intuitively constructed by connecting every pair of points in P whose Voronoi cells have a common facet.

Throughout this work the terms vertex and point will refer to Delaunay vertices, and edges in the Delaunay triangulation will also be referred to as connections.

3.1.2.1 The Delaunay criterion

When the Voronoi tessellation is not known a priori, the Delaunay criterion is used in order to construct a Delaunay triangulation.

Every simplex of vertices $\mathbf{p}_i$ in $\mathbb{R}^3$ has a circumsphere. The radius and the center of the circumsphere can be derived from the four sphere equations

$$(\mathbf{m} - \mathbf{p}_i)^2 = R^2, \qquad i = 1, ..., 4, \tag{3.2}$$

where $\mathbf{m}$ are the coordinates of the center of the circumsphere and R is its radius.

For a triangulation to be considered a Delaunay triangulation, all of its simplices must satisfy the empty-circumsphere-criterion (or Delaunay criterion), i.e., no vertex of the triangulation may lie inside the circumsphere of the triangulation simplices. Therefore, a Delaunay triangulation is uniquely defined if the points in P are in extended general position, i.e., no two points are identical, no three points lie on a common line, no four points lie on the same plane and no five points lie on a common sphere (for the three-dimensional case discussed here) [Schaller 2004, Okabe 2000, Mücke 1998, Cazals 2006].

A more general concept for the Delaunay criterion can be obtained by extending the Euclidean distance measure to vertices with weights. In this case, a weighted vertex can be defined as $\hat{\mathbf{p}} = (\mathbf{p}, w_p)$, and the orthogonal distance between two points $\hat{\mathbf{p}}_1 = (\mathbf{p}_1, w_1)$ and $\hat{\mathbf{p}}_2 = (\mathbf{p}_2, w_2)$ as $\pi(\mathbf{p_1}, \mathbf{p_2}) = (\mathbf{p}_1 - \mathbf{p}_2)^2 - w_1 - w_2$. A vertex with positive weight can be understood as a sphere with radius $R = \sqrt{w}$ situated at position $\mathbf{p}$. The problem presented in this work does not include weights on the vertices of the triangulation, but it could be extended to include weights if necessary.

The easiest way to verify whether or not a vertex lies inside the circumsphere of a simplex is to solve the four sphere equations 3.2. However, a more efficient method (called lifting transformation) to solve the problem exists, and it involves the addition of one more dimension to the problem [Ferrez 2001, Shewchuk 1997]. For that, the coordinates in $\mathbb{R}^3$ are projected onto a paraboloid in $\mathbb{R}^4$ via

$$\mathbf{p} = (p_x, p_y, p_z) \rightarrow \mathbf{p}^\dagger = (p_x, p_y, p_z, p_x^2 + p_y^2 + p_z^2). \tag{3.3}$$

Therefore, if a simplex is composed by four points $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, the four points $\mathbf{p}_1^\dagger$, $\mathbf{p}_2^\dagger$, $\mathbf{p}_3^\dagger$, $\mathbf{p}_4^\dagger$ will define a hyperplane in $\mathbb{R}^4$. If $\mathbf{q}$ is a point that lies within the circumsphere of $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, then $\mathbf{q}^\dagger$ will be below this hyperplane in $\mathbb{R}^4$ and above otherwise. Consequently, the Delaunay criterion in $\mathbb{R}^3$ reduces to a simple orientation computation in $\mathbb{R}^4$.

This lifting transformation will give the following:

$$\begin{aligned}
&\text{in\_circumsphere}[(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4), \mathbf{q}] = \text{orientation}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{q}) \\
&= \text{sign} \begin{vmatrix} p_{1x} & p_{1y} & p_{1z} & p_{1x}^2 + p_{1y}^2 + p_{1z}^2 & 1 \\ p_{2x} & p_{2y} & p_{2z} & p_{2x}^2 + p_{2y}^2 + p_{2z}^2 & 1 \\ p_{3x} & p_{3y} & p_{3z} & p_{3x}^2 + p_{3y}^2 + p_{3z}^2 & 1 \\ p_{4x} & p_{4y} & p_{4z} & p_{4x}^2 + p_{4y}^2 + p_{4z}^2 & 1 \\ q_x & q_y & q_z & q_x^2 + q_y^2 + q_z^2 & 1 \end{vmatrix} \\
&= \text{sign} \begin{vmatrix} p_{1x} - q_x & p_{1y} - q_y & p_{1z} - q_z & (p_{1x}^2 + p_{1y}^2 + p_{1z}^2) - (q_x^2 + q_y^2 + q_z^2) \\ p_{2x} - q_x & p_{2y} - q_y & p_{2z} - q_z & (p_{2x}^2 + p_{2y}^2 + p_{2z}^2) - (q_x^2 + q_y^2 + q_z^2) \\ p_{3x} - q_x & p_{3y} - q_y & p_{3z} - q_z & (p_{3x}^2 + p_{3y}^2 + p_{3z}^2) - (q_x^2 + q_y^2 + q_z^2) \\ p_{4x} - q_x & p_{4y} - q_y & p_{4z} - q_z & (p_{4x}^2 + p_{4y}^2 + p_{4z}^2) - (q_x^2 + q_y^2 + q_z^2) \end{vmatrix},
\end{aligned} \tag{3.4}$$

where sign gives the sign of the calculated determinant. A positive sign indicates that the point q lies inside the circumsphere of $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$, if the simplex is positively oriented in three dimensions. The lifting transformation gives rise to a different viewpoint of Delaunay triangulations: a n-dimensional Delaunay triangulation is determined by the boundary of the convex hull of the lifted vertices in n + 1 dimensions.

3.1.2.2 Computational implementation of the three-dimensional Delaunay triangulation

The underlying algorithms to calculate Delaunay triangulations used in this work have been published before [Schaller 2004, Beyer 2005, Meyer-Hermann 2008] and are available upon request.

The software used in this work calculates a fully kinetic and dynamic three-dimensional Delaunay triangulation of a given set of particles, supporting both the movement of vertices as well as their dynamic insertion and deletion [Schaller 2004, Beyer 2005, Meyer-Hermann 2008]. It is therefore suitable for systems where the number of particles may vary.

The implementation is done in C++ and is parallelized [Beyer 2005, Beyer 2006]. The software package offers many functions to deal with vertices and simplices which will be considered as given during the description of the algorithms used to develop this work.

- Vertex list: list of all vertices belonging to the triangulation. A vertex is composed of a position in space and the associated object in this position.
- Neighbor list: function of a vertex. Given a determined vertex, this function returns all neighbors of this vertex in the triangulation.
- Simplex list: function of a vertex. Given a vertex, this function returns all simplices this vertex belongs to.

3.2 Interaction potentials

Any kind of internal structure belonging to a cell can, in principle, be included in the model, by controlling the properties of the sub-particles. This allows, for example, the definition of sub-particles as organelles inside the cell, as belonging to the actin network, to the cytoskeleton or to any other sub-cellular structure one can think of. However, the more detailed the internal structure of the cell, the more time consuming the computation is. Therefore, the refinement of the model has to be done with care, taking into consideration the growth in complexity as well.

In the present setup of the model only two different groups of particles are considered: the ones belonging to the membrane (called membrane particles) and the ones belonging to the internal structure of the cell (called cytosolic elements). Membrane particles and cytosolic elements have different properties, and therefore should interact under different conditions. In this simple setup, two interaction potentials were set, one to mediate the interactions between membrane particles, and one to mediate the interactions between cytosolic elements and interactions between both types of particles.

3.2.1 Cytosolic sub-particle interactions

In principle, all sub-particles interact with each other via the three-dimensional Delaunay triangulation. It will be shown later on, however, that the three-dimensional Delaunay triangulation is not the best solution to define the interaction relations between membrane particles. With exception to the neighborhood relations between membrane particles, all other neighborhood relations on the system are given by the three-dimensional Delaunay triangulation.

The potential chosen for the interaction between cytosolic elements is the Lennard-Jones potential [Jones 1924]:

$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{3.5}$$

where ε is the depth of the potential and σ is the equilibrium distance where the potential is zero. Here $\varepsilon = \varepsilon^*\varepsilon_0$, where ε_0 is the universal energy scale, $\sigma = \sigma^*\sigma_0$ and $r = r^*\sigma_0$, where σ_0 is the universal length scale. From now on reduced units in terms of ε_0 and σ_0 will be used. In this case, equation (3.5) becomes

$$U(r^*) = 4\varepsilon^*\left[\left(\frac{\sigma^*}{r^*}\right)^{12} - \left(\frac{\sigma^*}{r^*}\right)^{6}\right]. \tag{3.6}$$

For convenience we drop the stars in the remainder of this thesis.

The Lennard-Jones potential was proposed by John Lennard-Jones in 1924 [Jones 1924], with the goal to describe the interactions between a pair of neutral atoms or molecules. The r^{12} term describes repulsion at short ranges and the r^6 term describes attraction at long ranges. Originally, the repulsion term represented Pauli repulsion at short ranges due to overlapping electron orbitals and the attraction term represented forces like van der Waals force or dispersion forces.

Although the Lennard-Jones potential was originally developed to describe interactions between molecules, it is essentially a phenomenological potential. In this case, it can be used to simulate the interaction between any kind of particles where repulsion is expected at short ranges and attraction at long ranges. This potential was chosen because it matched the desired behavior while being also convenient, due to the ease and efficiency of computing r^{12} as the square of r^6. Evidently, the parameters of the potential (ε_0, σ_0 and r_0) should, in principle, be connected to relevant quantities from experiments related to the system. A longer discussion about the connection between the simulations and physical units will be given during the calculation of initial configurations (chapter 5).

3.2.2 Cross-interaction between membrane and cytosolic sub-particles

The cross-interactions between membrane particles and cytosolic elements are governed by the same rules as the interactions between cytosolic elements only. The neighborhood relations are defined via the original three-dimensional Delaunay triangulations, and the particles interact via a Lennard-Jones potential as well. However, albeit the potential is the same, the parameters used in the simulations for the interactions may differ depending on the interacting particles.

For the initial configurations used during this work, the depth of the potential ε was the same for both interactions between cytosolic elements and cross-interactions between cytosolic elements and membrane particles. The equilibrium distance (σ), however, was modified. More details about the chosen parameters are given in chapter 5.

3.2.3 Membrane sub-particle interactions

The properties expected from particles simulating a cell membrane are quite different from the properties expected from particles simulating cytosolic elements. The interactions between cytosolic elements should be flexible and, to a certain extent, reproduce the characteristics of a liquid. The cell membrane, however, should be elastic and behave as a liquid confined to two-dimensions. Therefore, the interactions between membrane particles should be able to simulate these properties.

Additionally, as explained in section 5.1, the cell membrane is mainly composed by a lipid bilayer. Little to no long range interactions happen in the lipid bilayer and, therefore, it is reasonable to approximate the membrane by a two-dimensional bended surface. As a result of this assumption, the interactions between particles used to simulate the membrane should be local.

The initial intention was to let the three-dimensional Delaunay triangulation handle all the neighborhood relations of the model. Early tests, however, showed that, although the neighborhood relations coming from the Delaunay triangulation are local, they are not sufficiently restrictive to be used to describe the interactions between membrane particles. This happens because the connections generated by the three-dimensional Delaunay triangulation still allow interactions to happen outside the two-dimensional surface that defines the cell membrane.

With the original Delaunay triangulation not being suitable to define the membrane neighborhood relations, a new solution had to be proposed. However, the three-dimensional Delaunay triangulation is still the method to get all other neighborhood relations in the model. Therefore, a solution that could still make use of the Delaunay triangulation (that is in any case calculated) would be welcome. With that in mind, the following solution was proposed: the neighborhood between membrane particles should be represented by a subset of the original three-dimensional Delaunay triangulation. Additionally, the connections belonging to this subset should be the ones restricted to the membrane two-dimensional surface.

Obtaining this restricted Delaunay triangulation from the original Delaunay triangulation is a rather complicated problem. Although some methods from surface reconstruction are available [Amenta 1998, Amenta 1999], they have specific properties that are not desired when applied to biological systems. In order to solve this problem, an algorithm to define which connections belong to this restricted triangulation was developed. This algorithm is an important part of this thesis, and it will be explained in details in chapter 4.

Due to the cell membrane elastic properties, the potential chosen to mediate the interactions between membrane particles was the elastic potential:

$$U(r) = \frac{1}{2}\kappa(r - r_0)^2, \tag{3.7}$$

where κ is the elastic constant and r_0 is the equilibrium distance of the potential. As before, κ and r_0 are given in reduced units, and need to be connected to experimental parameters in order for the results to be given in physical units.

3.3 Equations of motion

Considering a system of a single cell that does not interact with the environment, the equations of motion for membrane particles and cytosolic elements can be written down separately. For each cytosolic element, the equation of motion will be

$$m_c \frac{d^2}{dt^2}x(t) = \sum_{N_c}(F_{LJ})_c + \sum_{N_m}(F_{LJ})_{cm}, \tag{3.8}$$

while for each membrane particle it will be

$$m_m \frac{d^2}{dt^2}x(t) = \sum_{N_c}(F_{LJ})_{cm} + \sum_{N_m}(F_{el})_m, \tag{3.9}$$

where m_c and m_m are, respectively, the cytosolic and membrane particles' masses. Additionally, N_c and N_m represent the cytosolic and membrane neighbors of a particle, $(F_{LJ})_c$ is the force due to the Lennard-Jones potential between cytosolic elements, $(F_{LJ})_{cm}$ is the cross force between cytosolic elements and membrane particles (also due to the Lennard-Jones potential), and $(F_{el})_m$ is the elastic force between membrane particles.

Using the elastic and the Lennard-Jones potential from equations (3.7) and (3.5), the forces in equations (3.8) and (3.9) can be written as:

$$(F_{LJ})_c = 24\varepsilon_c \left(2\frac{\sigma_c^{12}}{r^{13}} - \frac{\sigma_c^6}{r^7}\right), \tag{3.10}$$

$$(F_{LJ})_{cm} = 24\varepsilon_{cm} \left(2\frac{\sigma_{cm}^{12}}{r^{13}} - \frac{\sigma_{cm}^6}{r^7}\right), \text{ and} \tag{3.11}$$

$$(F_{el})_m = -\kappa(r - r_0). \tag{3.12}$$

3.4 Dynamics

The evolution of the system in time is calculated using the Verlet algorithm [Verlet 1967, Verlet 1968], a numerical method commonly used to integrate Newton's equations of motion. The algorithm is often used in molecular dynamics, and it is more stable than the usual Euler method. The Verlet algorithm calculates the position at the next time step from the positions at the previous and current time steps, and it does not depend on the velocity of the particle. This reduces the level of errors introduced in the integration.

In order to derive the Verlet algorithm, Taylor expansions of $x(t)$ both forward and backward in time are needed:

$$\begin{aligned} x(t_0+\Delta t) &= x(t)+v(t)\Delta t+\frac{1}{2}a(t)(\Delta t)^2+\frac{1}{6}b(t)(\Delta t)^3+\mathcal{O}(\Delta t^4), \quad \text{and} \\ x(t_0-\Delta t) &= x(t)-v(t)\Delta t+\frac{1}{2}a(t)(\Delta t)^2-\frac{1}{6}b(t)(\Delta t)^3+\mathcal{O}(\Delta t^4), \end{aligned} \tag{3.13}$$

where x is the position, v is the velocity, a in acceleration and b is the third derivative of the position with respect to time. Summing the two expansions gives the following equation for the position of a particle in the system in an instant of time $t_0+\Delta t$:

$$x(t_0+\Delta t) = 2x(t_0) - x(t_0-\Delta t) + a\Delta t^2. \tag{3.14}$$

As it can be seen in the equation above, summing the two Taylor expansions cancels both the first and the third-order terms out, making the Verlet integrator an order more accurate than integration by simple Taylor expansion alone.

3.4.1 Overdamped approach

Biological systems are usually under a lot of drag and friction forces. A simple way to implement this on the Verlet algorithm is to consider that part of the particle's speed is lost to the environment. This can be easily implemented in the form of a factor in equation (3.14).

$$x(t_0+\Delta t) - x(t_0) = (1-f)(x(t_0)-x(t_0-\Delta t)) + a\Delta t^2, \tag{3.15}$$

where f varies from 0 to 1 and is the percentage of speed lost due to friction. This equation can also be written as:

$$x(t_0+\Delta t) = (2-f)x(t_0) - (1-f)x(t_0-\Delta t) + a\Delta t^2. \tag{3.16}$$

As stated in section 2.2.2.1, the cytosol is basically a gel, with a network of fibers dispersed through water. Each cytosolic element can therefore be considered to be moving through a gel environment, and an overdamped approach can be implemented. In this case, the friction in the system is considered to be maximum, setting $f=1$. As a result, the displacement of a particle after the next time step is purely proportional to its acceleration:

$$x(t_0+\Delta t) - x(t_0) = a\Delta t^2. \tag{3.17}$$

3.4.2 Adaptive time step

In order to avoid artificial forces when particles happen to start the simulation too near to each other, a routine to dynamically adapt the size of the time step used by the system was implemented. In order to do that, a fixed value for the maximum displacement of any particle in the system had to be chosen. This fixed value D_{max} is usually expressed as a percentage of the particle diameter.

Because of the enforced maximum displacement, not every size of time step is allowed to be executed anymore. Instead, each time step will be chosen so that no particle in any time step can move further than the chosen D_{max}. This ensures that no artificial displacement happens on the program. The following procedure is used to implement the adaptive time step procedure:

- The displacement of every particle using the current time step size (Δt_{cur}) is calculated.
- The biggest displacement (D_{big}) is stored. This displacement is then compared to D_{max}. Using equation (3.17) for D_{max} and D_{big} one obtains:

$$\frac{D_{max}}{D_{big}} = \frac{a(\Delta t_{max})^2}{a(\Delta t_{cur})^2} \tag{3.18}$$

- The maximum time step allowed can be derived from equation (3.18):

$$\Delta t_{max} = \sqrt{\frac{D_{max}}{D_{big}}}\Delta t_{cur}. \tag{3.19}$$

- The original Δt_{cur} is then substituted by the Δt_{max} calculated above. With this substitution, the particle that originally had the biggest displacement D_{big} will only be displaced by the maximum displacement D_{max}, and all the other particles will have their displacements changed by a factor D_{max}/D_{big}.

Even though this procedure is important to the correct physical behavior of the system, it slow down the simulations. This happens because, even if only one particle would be moved by a displacement over D_{max}, all the particles in the system have their displacements lowered accordingly. A possible way to speed up the program without allowing non-physical displacements is to modify this algorithm so that the movement of particles is asynchronous. In this case, fast particles would have their displacement broken down in many time steps while slow particles would be moved in bigger time steps. However, the implementation of this kind of algorithm is trick and requires the speed of the few fast particles to be really higher than the average speed on the system. Since no extended simulations were used in this work, the implementation of such algorithm was deemed unnecessary.

Chapter 4

Surface reconstruction

Contents

As explained in chapter 3, the biological properties of the cell membrane restrict the interactions between membrane particles to the surface they form. This makes the original three-dimensional Delaunay triangulation inadequate to describe the membrane particles' interactions, and raises the necessity of the construction of a restricted Delaunay triangulation to handle these interactions. In this chapter the algorithm developed to generate this restricted Delaunay triangulation is described.

The method's input is a given set P of points (membrane particles) defined to belong to a surface embedded in $\mathbb{R}^3$, plus a set Q of points defined to belong to the internal structure of the cell (cytosolic elements). The three-dimensional Delaunay triangulation of $P \cup Q$ is calculated, as it is anyway needed for the interaction relations between cytosolic elements. This three-dimensional Delaunay triangulation will be used as the starting point for the definition of the restricted Delaunay triangulation.

The method works as follows: from the three-dimensional Delaunay triangulation of $P \cup Q$ a structure called *quasi-surface* is obtained. The quasi-surface is a subset of connections from the Delaunay triangulation of $P \cup Q$, and it includes all connections belonging to the desired surface plus some undesired ones (see figure 4.1). Therefore, the quasi-surface still has undesired three-dimensional structures (simplices or groups of simplices). The formal definition of the quasi-surface, as well as an alternative method to define a similar structure when no cytosolic elements are present, will be presented in the section 4.1.

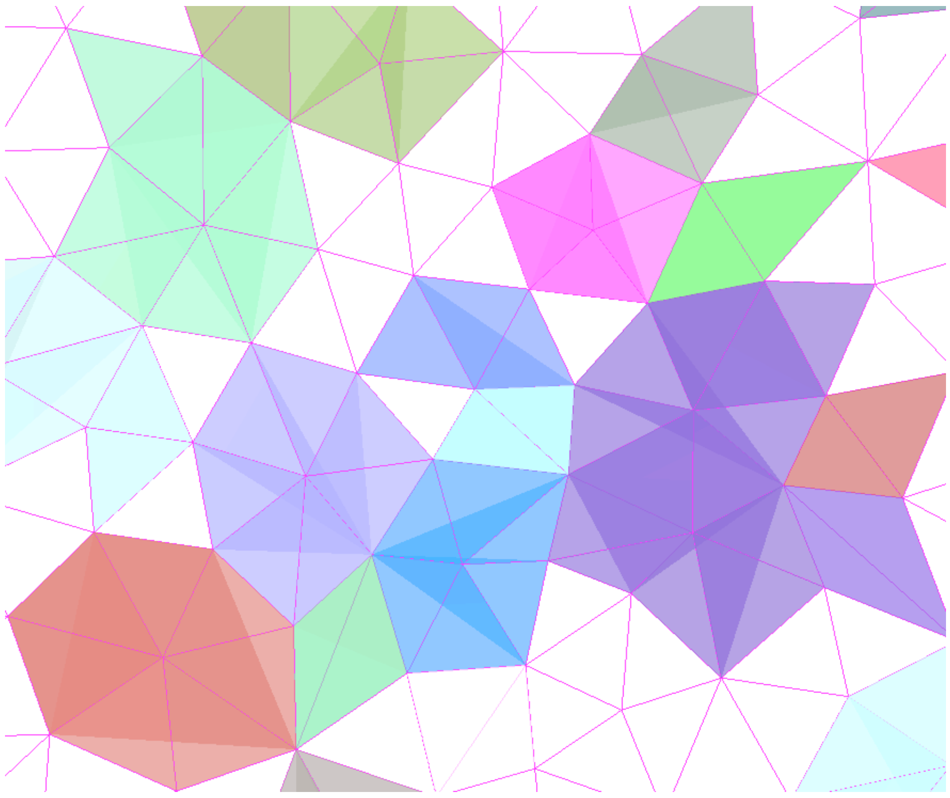


Figure 4.1: The quasi-surface is obtained from the three-dimensional triangulation involving membrane particles and cytosolic elements. Notice that it has areas already triangulated on a two-dimensional surface and areas (highlighted) still including three-dimensional structures.

The left-over three-dimensional structures belonging to the quasi-surface have then to be eliminated. For that, the following sequence of steps is applied: all sets of connected simplices (these structures will be called *clusters*) in the quasi-surface are selected. The boundaries of the selected clusters are then identified and their vertices ordered. At last, an algorithm is applied in order to select a subset of all connections in the cluster which re-triangulates it on a proper surface. Notice that the resulting triangulation of a cluster on a surface is a subset of the cluster's connections and, therefore, a subset of the three-dimensional Delaunay triangulation.

In the remaining of this chapter the steps briefly introduced above will be thoroughly explained. The chapter starts with the definition of the quasi-surface, followed by the introduction of the concept of clusters. The algorithm used to define a cluster's boundary is then introduced, followed by the explanation of how to put the vertices belonging to a cluster's boundary in order.

Once the cluster's boundary is identified and ordered, it is possible to apply to it a re-triangulation algorithm. Two different re-triangulation algorithms are presented: one to deal with clusters that have internal points and one to deal with clusters that do not have internal points. These two algorithms are explained in the last section of this chapter.

4.1 Obtaining the quasi-surface

The method to obtain the quasi-surface takes advantage of specific features of the model developed for a cell. In this model, presented in chapter 3, a cell consists of membrane particles (in this case, the set P of input points) and cytosolic elements. In order to define the neighborhood relations between the cytosolic elements or between them and the membrane particles, a Delaunay triangulation of the union of the two sets of points is always calculated. One can then make use of this triangulation in order to define the quasi-surface.

> **Definition:** The quasi-surface is the subset of the three-dimensional Delaunay triangulation formed by all connections between two membrane points.

The quasi-surface will always be a rough surface (including all connections belonging to the desired surface plus some undesired ones) as long as the targeted surface is convex and closed. This is true because, in this case, the surface's *medial axis* is always inside the surface.

> **Definition:** The medial axis of an n-dimensional surface embedded in $\mathbb{R}^{n+1}$ is defined as the closure of the set of points with more than one closest point on the surface.

If there were enough points (cytosolic elements or any kind of artificial points that could have been included in the triangulation) inside the surface that fall into the medial axis, there would be no long range connections between membrane points.

In the setup presented here, no artificial points are included inside the surface. However, in order for the simulation to be meaningful, sufficient resolution is needed inside the cell. This implies that there must always be sufficient cytosolic elements so that no connection between two membrane particles cross the medial axis of the surface.

Moreover, problems could arise if these cytosolic elements were not far enough from the cell membrane, in which case holes could be generated. However, this would not only cause problems in the algorithm but also in the model, as holes in the membrane are seldom desired and should never happen randomly. Therefore, the interaction potential between cytosolic elements and membrane particles must be carefully chosen, in order to guarantee that cytosolic elements are far enough from the membrane, being unable to cause holes on the surface.

In case the model to be simulated does not obey these assumptions, the following algorithm to obtain a structure similar to the quasi-surface (the *crust*) may be used instead.

4.1.1 The crust (in two-dimensions)

Similarly to the quasi-surface, the crust is a set of connections that contains the required surface but still contains some three-dimensional structures. It can be obtained as an alternative to the quasi-surface when applying the model to a system with no equivalent to the cytosolic particles. The concept of the crust was developed by Nina Amenta et al [Amenta 1998, Amenta 1999], and is used in some of their algorithms for surface reconstruction.

Consider the problem of obtaining a curve in two-dimensional space from a set of sample points C. This is the two-dimensional problem analogous to the one of obtaining a surface in three-dimensional space form a set of points P.

The algorithm to solve this problem is the following:

- Calculate the Delaunay triangulation of all points in C (see figure 4.2 - a).
- Calculate the Voronoi tessellation of the points in C (figure 4.2 - b). The set of all Voronoi vertices will be called V.
- Now consider the set of points in $\mathbb{R}^2$ given by $C \cup V$. Calculate the Delaunay triangulation of this set (figure 4.2 - c).
- To finally obtain a polygonal approximation of the curve, only the connections (in the last calculated Delaunay triangulation) between points belonging to C are selected (figure 4.2 - d).

Amenta proved that, given certain restrictions to the set of points C, this method always gives as output the desired polygonal approximation of the curve [Amenta 1999].

This method works because, in two-dimensions, Voronoi vertices have the property of, for sufficiently dense samples, always falling into the medial axis. However, this method cannot be immediately generalized to solve the analogous problem in three-dimensions. Unlike in the two-dimensional case, not all Voronoi vertices in the three-dimensional problem fall into the medial axis. Some of them fall arbitrarily near to the surface. Nevertheless, some of the Voronoi vertices do have this property, lying in the medial axis. These Voronoi vertices are called poles. If, instead of using all Voronoi vertices to calculate the second Delaunay triangulation (like in figure 3-c), only the poles are included, the method can be generalized to three-dimensions. For a detailed explanation of the method in three-dimensions, see [Amenta 1999].

When using the three-dimensional version of the method discussed above, the final result will not be a polyhedral approximation of the surface. The method to calculate the crust always works for well behaved samples (see [Amenta 1999] for the method's mathematical requirements) but, because it requires the calculation of two separated Delaunay triangulations, it is not a sufficiently efficient algorithm for the present purpose. Therefore, the crust should only be used when the calculation of the quasi-surface is not possible.

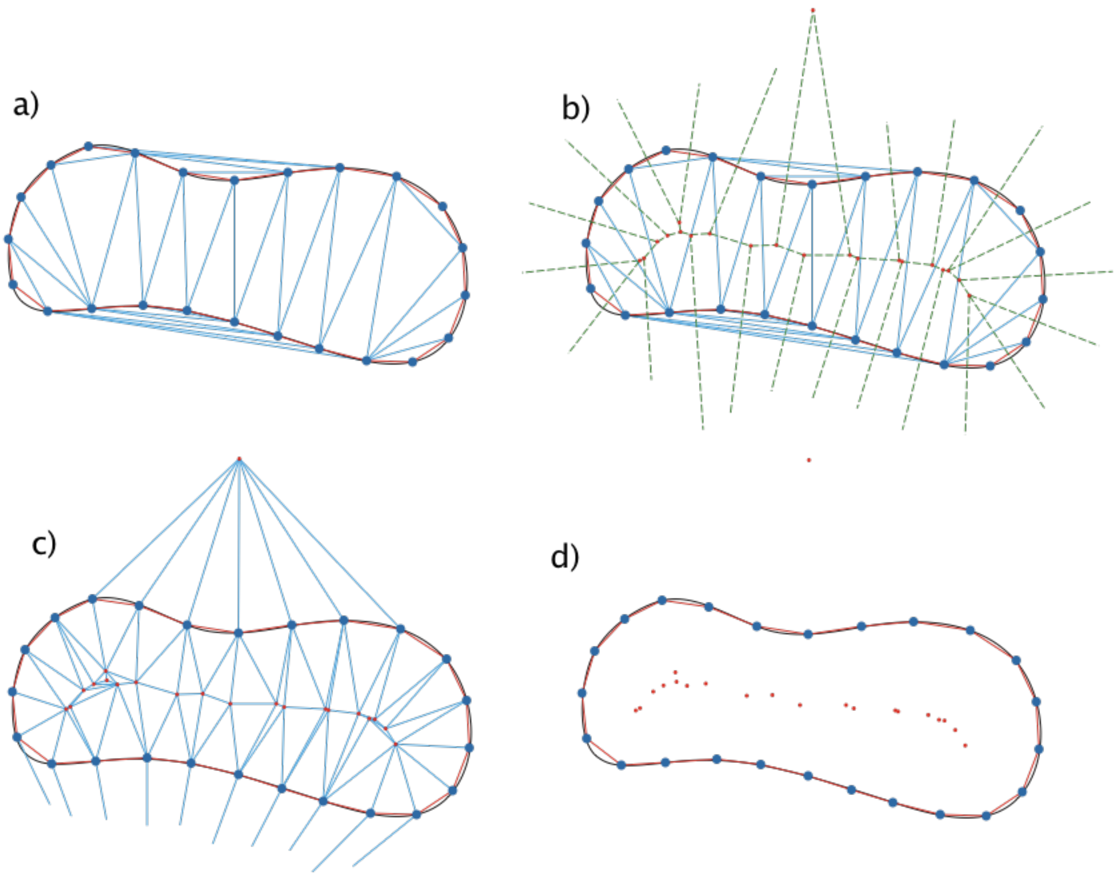


Figure 4.2: This figure illustrates the method to find the crust. a) Delaunay triangulation of the points belonging to the curve C. b) Voronoi tessellation of the points in C. c) Delaunay triangulation of $C \cup V$. d) The polygonal approximation of the curve is composed of the connections on the Delaunay triangulation of $C \cup V$ between points in C.

4.2 Clusters

Once obtained, the quasi-surface can be then subdivided in smaller structures called clusters. Clusters are formed by the three-dimensional structures that are left in the quasi-surface (see figure 4.3). Once the clusters are identified, each of them can be triangulated separately, breaking an initially complex problem in many simpler independent ones.

In order to identify each cluster on the quasi-surface, a list of all simplices in the quasi-surface is needed. For a quasi-surface, this list is a subset of the list of simplices of the original triangulation (involving membrane points and cytosolic elements). If a simplex of the original list is composed of four points also belonging to the quasi-surface, then it also belongs to the list of simplices of the quasi-surface.

Two simplices are neighbors if they have a common face. A cluster is defined

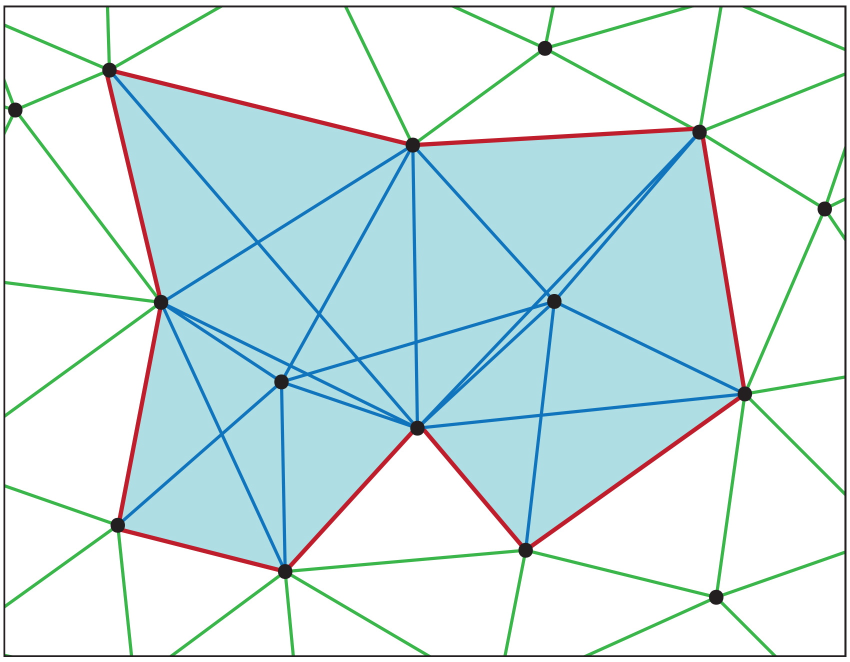


Figure 4.3: The cluster (filled area in the figure) is a not yet re-triangulated area in the quasi-surface that is surrounded by a surface properly restricted to two-dimensions. The connections belonging to the cluster and its surrounding (in red in the figure) compose the cluster boundary. Only end vertices of boundary connections have a common neighbor outside of the cluster.

as a set of simplices where each simplex has a common face with at least one other simplex on the set. Each cluster is therefore stored as a list of simplices. A list is also be used to store all the clusters in the quasi-surface. The algorithm used to identify the clusters is the following:

1. A random simplex from the list of all simplices in the quasi-surface that does not belong to any cluster yet is chosen.
2. A new cluster is then created, and the simplex included in the list of the new cluster.
3. Every neighbor of the simplex that also belongs to the list of simplices of the quasi-surface necessarily belongs to the same cluster. These simplices are also included in the list of the cluster.
4. The procedure in 3 is recursively repeated to all the neighbors of the selected neighbors and so on, until no neighbor satisfying the original condition is found.

The quasi-surface can only be separated into clusters if some parts of it are already triangulated on the desired two-dimensional surface, thus disconnecting the three-dimensional structures from each other. The validity of this assumption is discussed in Section *Testing*.

4.3 Cluster boundaries

Every cluster has a two-dimensional boundary where it touches the surrounding properly triangulated surface (see figure 4.3). Identifying this boundary is a necessary step in order to obtain a valid two-dimensional triangulation for it.

A boundary can be uniquely identified by a set containing all its connections. In order to obtain this set, a list of all vertices belonging to each cluster is needed. This list is easily obtained by including all vertices belonging to the simplices of a cluster into a hash set.

4.3.1 Selecting connections

All simplices originally in the quasi-surface were included in one of the clusters. It is therefore assumed that two vertices that are endpoints of a connection belong to the boundary if and only if they have at least a common neighbor outside the cluster (see figure 4.3). This defines the algorithm to determine the boundary of a cluster:

- For each pair of neighbor vertices (p_1, p_2) in the cluster, all other neighbors of p_1 are selected.
- For every neighbor of p_1, it is checked whether it belongs to the cluster.
- If a neighbor of p_1 does not belong to the cluster, it is checked whether it is also a neighbor of p_2.
- If yes, the connection (p_1, p_2) belongs to the boundary of the cluster, and the procedure is stopped.
- If not, the procedure has to be repeated until a common neighbor of p_1 and p_2 outside of the cluster is found, or until all neighbors are tested and no common neighbor is found.
- If no common neighbor outside the cluster is found, the connection (p_1, p_2) does not belong to the boundary of the cluster.

4.3.2 Wrongly selected connections

Even though every simplex in the quasi-surface is included in a cluster, three-dimensional simplex-like structures are still spread through the quasi-surface. The figure 4.4 illustrates such a structures. The configuration shown has four points belonging to the surface sample and one external point (for example, a cytosolic element). When all connections between the surface sample points and the external point are cut (in order to obtain the quasi-surface), the four points belonging to the surface will still form a three-dimensional simplex-like structure. However, since they were not forming a simplex in the original triangulation, they will also not appear as a simplex in the quasi-surface, will not be in the simplex list and will not be part of any of the clusters.

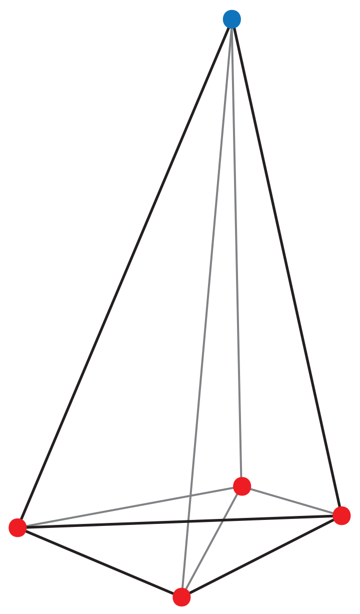


Figure 4.4: Example of a three-dimensional structure left on the quasi-surface. The four red points belong to the surface sample, while the blue point is an external point (for example, a cytosolic element). When all connections between the surface sample points and the external point are cut (in order to obtain a quasi-surface), the four points belonging to the surface will still be all connected to each other, forming a simplex-like structure in the surface that does not belong to any cluster, since they do not form a simplex.

This, in some rare cases, renders the method to select the boundary connections wrong, allowing connections that do not belong to the boundary to have common neighbors outside the cluster, as illustrated in figure 4.5. These connections will be wrongly selected as belonging to the boundary. Further checks can be applied to make sure that all selected connections do belong to the boundary, and to remove wrongly selected connections.

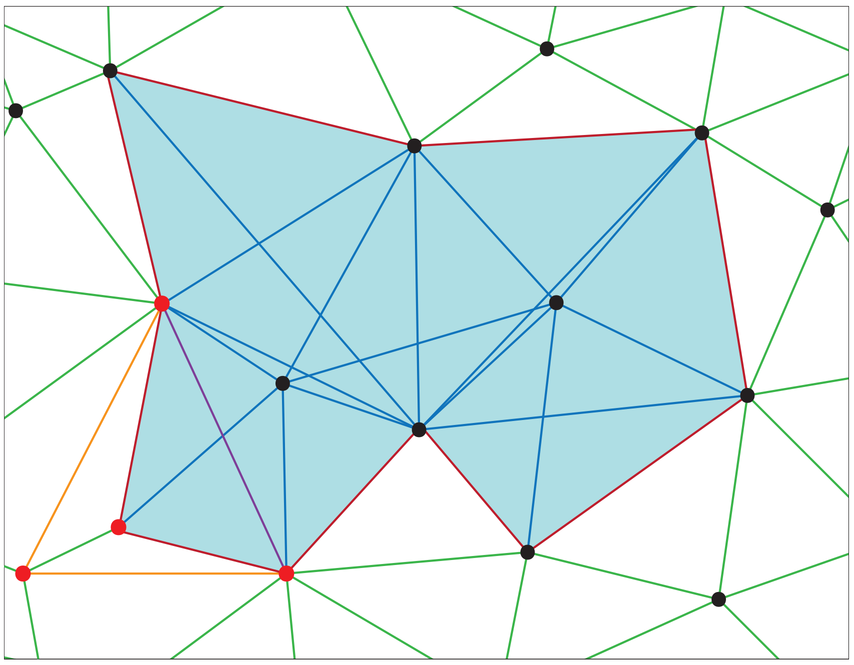


Figure 4.5: Three-dimensional structure next to a cluster. The four points in red form a three-dimensional structure as the one shown in figure 4.4. The purple connection will be wrongly identified as belonging to the boundary of the cluster, since its vertices have a common neighbor outside of the cluster (connections to the common neighbor are shown in orange).

4.3.3 Detecting wrongly selected edges

The number of connections in the boundary of a cluster shall be equal to the number of vertices in the boundary. Moreover, every vertex in the cluster that belongs to its boundary shall be part of exactly two connections also belonging to the boundary. This can be used to define whether the boundary of a determined cluster was or was not properly selected and, in the case it was not, count how many wrongly selected connections are present.

For this purpose a list of pairs is created, where the first element of the pair is a vertex and the second is the number of times this vertex is found in a connection at the boundary. If any of the vertices is found in more than two boundary connections, some connections were mistakenly selected in this cluster. These cases are restored during ordering of the boundary.

4.4 Boundary ordering

The next necessary procedure is the ordering of the boundary connections. Wrongly selected connections will also be thrown away in this step.

Any vertex on the list above that belongs to exactly two connections in the boundary can be taken to be the initial vertex of the ordered boundary. This vertex is then included in the container where the ordered boundary will be kept. Having the initial point, the procedure to obtain the boundary if no connections were mis-selected is:

- Search for a connection in the list of boundary connections that contains the initial vertex.
- Include the second vertex of this connection to the ordered boundary, and delete the connection from the list of boundary connections.
- Repeat the procedure, but now looking for a connection containing the last vertex that was included in the ordered boundary. When the second vertex in a connection is the initial vertex, the boundary has been ordered.

This algorithm does not always work if connections that do not belong to the boundary were mis-selected as belonging to it. Additionally, it does not take into consideration that some clusters might have more than a boundary (there is nothing that forbids an area inside a cluster to be already properly triangulated in a surface). In order to take these more delicate cases into considerations, a more complex variation of the algorithm above has to be implemented.

The algorithm to identify and order a boundary is similar to the one above. However, the latter supposes that only one path can be constructed using the connections in the boundary list. When this is not the case, all possible paths have to be identified. The path that represents the ordered boundary of the cluster is the one that contains all points in the boundary, each one a single time.

Dealing with multiple boundaries in a cluster is also not complicated. After a boundary for a cluster is identified, it must be checked whether all connections belonging to the original list of boundary connections were used. If some of these connections were not used, they belong to a cluster boundary (or boundaries) that has yet to be identified and ordered. The algorithm to identify a boundary has then to be once more executed, but only with the connections left.

Clusters with multiple independent boundaries need an additional step before they can be re-triangulated, because the algorithms used for the re-triangulation suppose that each cluster only has a single boundary. However, solving that is not difficult. First, the biggest boundary of the cluster has to be identified. This will be the final boundary of the cluster. Then, all vertices that belong to the smaller boundaries, as well as all vertices that belong to the areas bounded by these boundaries (outside the cluster) must be included as internal particles of the cluster. The normal algorithms to re-triangulate the clusters can then be used normally in these clusters as well.

4.4.1 Exceptions

In the extremely majority of the cases, the algorithm described above is able to retrieve the boundary of a cluster from a list with mis-selected connections. However, there are two cases where problems might still occur.

First, the procedure has to start with a vertex that belongs to exactly two connections on the cluster. In the unlikely case where every vertex in the boundary also belonged to a mis-selected connection, such a vertex would not be found and the procedure would not start.

Second, the mis-selected connections in the boundary can be positioned in a way that generates a boundary obeying the rules above, but that it is still not the real boundary of the cluster (see figure 4.6). In this case, the procedure to find the boundary would finish with two equally possible boundaries.

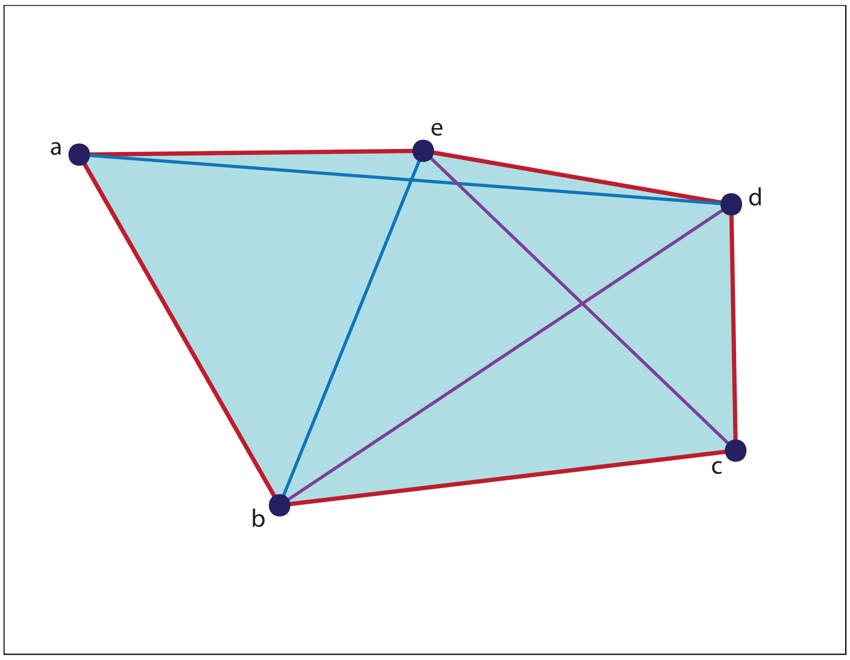


Figure 4.6: Configuration with two possible boundaries. If the connections in purple are wrongly selected as belonging to the cluster boundary, then the following connections are in the boundary list: (a,b), (b,c), (c,d), (d,e), (e,a), (b,d) and (c,e). These connections allow two possible orders for the boundary: (a,b,c,d,e) or (a,b,d,c,e). At this point the algorithm is unable to identify which one is the real boundary.

4.4.2 Troubleshooting

In the cases above, the re-triangulation of the cluster is not possible. When a cluster without a properly identified boundary is found, the program simply skips the cluster and goes on to the next one. After a run through all other clusters on the quasi-surface, it is checked whether any cluster was left untriangulated. When the answer is positive, the program will once more try to order the boundaries and re-triangulate the problematic clusters. This might be possible because, sometimes, neighboring clusters are the cause of problems with some clusters boundaries.

This procedure is repeated until either all clusters are re-triangulated or the number of not-triangulated clusters stops changing from a run to the other. In the first case the configuration was successfully triangulated, and in the latter the re-triangulation algorithm failed. A discussion about what can be done when the algorithm fails to properly re-triangulate a surface is given in the first sections of chapter 7.

4.5 Clusters re-triangulation

With the boundary of a cluster ordered, all the information about the cluster necessary to its re-triangulation is available. It is important to remark that the triangulation of each cluster is independent, i.e., when the boundary of a cluster is identified, this cluster can be re-triangulated independent of the state of any other cluster. This is important because, in case the boundary of a cluster is not identified, all the other clusters can still be re-triangulated. Additionally, due to the deletion of joined connections, the re-triangulation of clusters surrounding a cluster with a non-ordered boundary may help its ordering.

In this section the procedures used to the re-triangulation of clusters is explained. The main goal of the re-triangulation is to delete connections belonging to the interior of a cluster until all vertices in this cluster are re-triangulated in a surface, with no holes and no crossed connections (see figure 4.7).

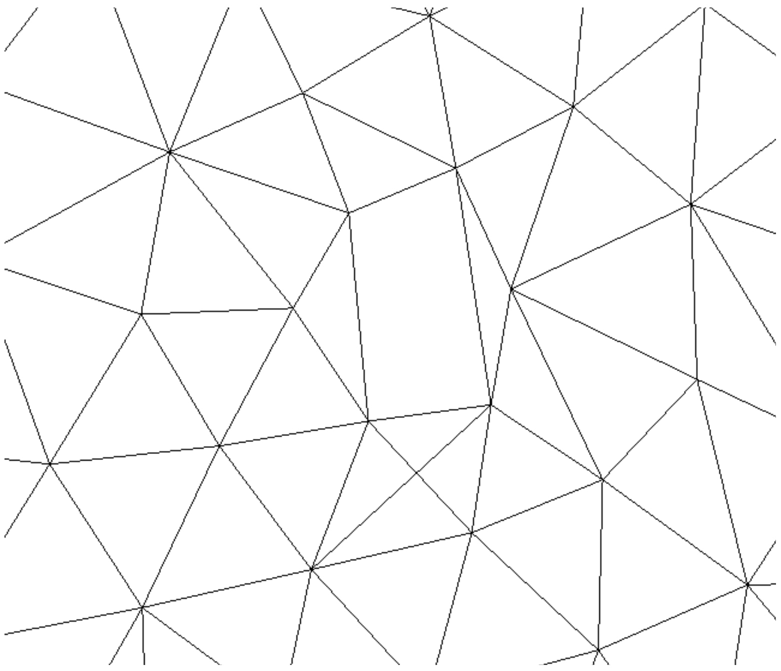


Figure 4.7: Example of a surface with a hole and crossed connections.

Two distinguished procedures are used, with either being chosen according to the number of points in the selected cluster that do not belong to its boundary.

Points in a cluster that do not belong to its boundary will be called *internal points*. Clusters, therefore, are divided in two groups: cluster without internal points (all points in the cluster belong to its boundary) and clusters with at least one internal point.

For both cases a vector is created to store all connections selected to belong to the final triangulation of the cluster. The vector is initialized with all connections belonging to the cluster's boundary, as all of them will be part of the final surface configuration. After the re-triangulation of a cluster, the connection list contains all the connections kept in the cluster.

4.5.1 No internal points

In a schematic way, the boundary of a cluster that has no internal points can be understood as a polygon whose vertices have to be triangulated. This problem can be stated as follows: given a convex polygon with n sides, divide it into triangles by drawing non-intersecting diagonals connecting some of the polygon's vertices. The problem of re-triangulating polygons is known in computational geometry. A polygon can be re-triangulated in many different configurations, and the first person to offer a good counting argument for the number of possible configurations was the mathematician Gabriel Lamé [Lamé 1838] (see figure 4.8).

For the specific problem of re-triangulating a cluster, however, not any possible triangulation of the polygon is accepted, but only one of those in which all chosen connections belong to the original triangulation (involving membrane particles and cytosolic elements). Any selection of connections fulfilling this criterion that properly re-triangulates the cluster is considered a good one (see figure 4.9).

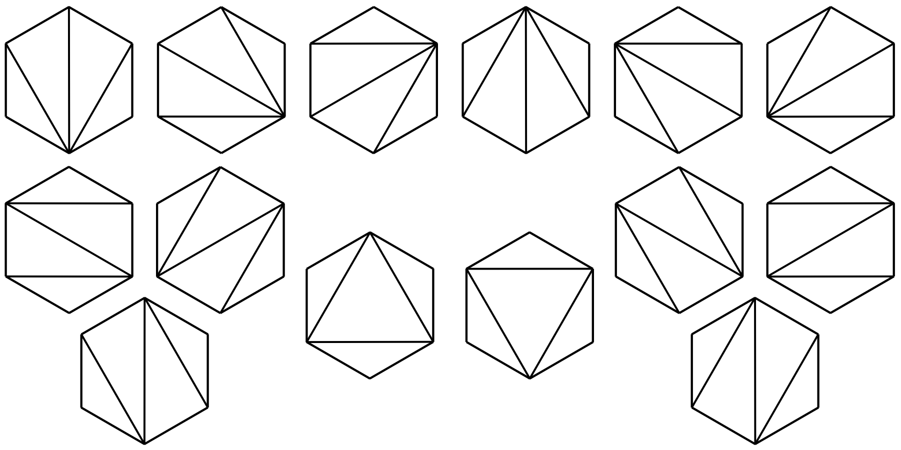


Figure 4.8: All possible triangulations of a hexagon.

The following algorithm is used to re-triangulate a cluster where all vertices belong to its boundary. This algorithm was based on mapping all possible triangulations of a polygon, and it was implemented as a recursive function:

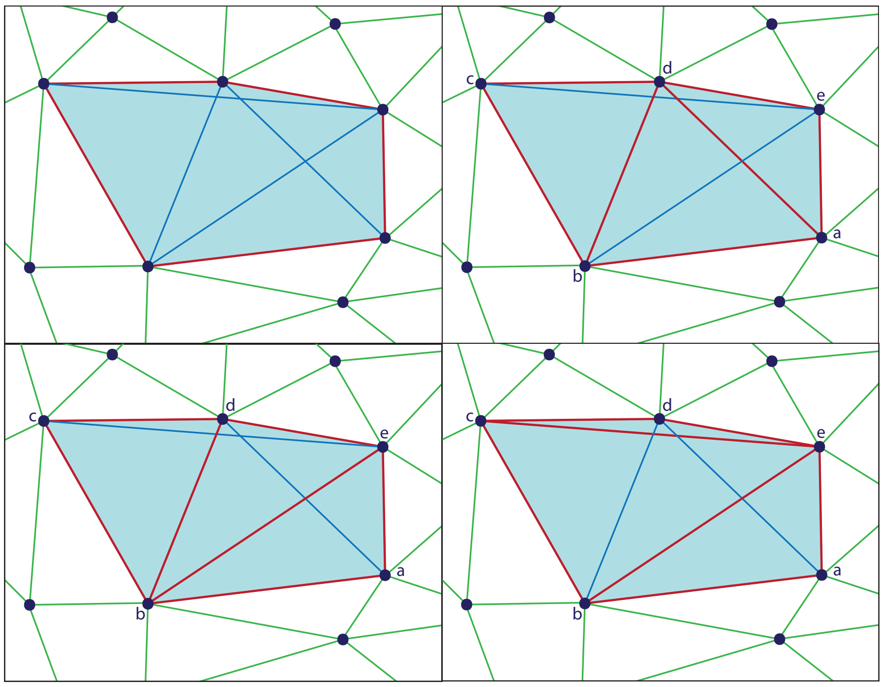


Figure 4.9: Three possible final triangulations for the same cluster. Any re-triangulation obtained for a cluster is acceptable.

- For all vertices i in the ordered boundary, it is checked whether the vertex $i+2$ is a neighbor. Periodic boundary conditions are necessary, since the boundary is closed.
- If i and $i+2$ are not neighbors, the procedure is restarted using the next vertex on the list. If they are neighbors, the connection $i, i+2$ is included on the vector of connections created beforehand.
- The boundary of the cluster is updated. This is done by deleting the vertex $i+1$ from the ordered boundary. The updated boundary will be the boundary of the part of the cluster still left to be re-triangulated.
- The routine is called again with the updated boundary as input.
- The routine finishes when the number of vertices in the cluster is equal to four.

Once the requirement to leave the recursive function is reached, only two possible connections are left. Either one that exists in the original triangulation can be chosen and is included in the connection list.

This procedure is based on two assumptions: first, that every connection selected must belong to at least one possible triangulation of the cluster; second, that all clusters are convex. While the first assumption is easily proven to be true, as every cluster is composed by a collection of simplices, it is pretty obvious that the second assumption is false, since there is no property on the triangulation that could force clusters to be convex.

Even though it is not true that every cluster is convex, the method used for the re-triangulation still works. The reason for that lies on the second restriction in the choice of a re-triangulation: that all connections must belong to the original three-dimensional triangulation involving membrane points and cytosolic elements. When a cluster is not convex, the connections between two vertices in the cluster that would not lie inside the cluster simply do not exist in the original Delaunay triangulation and, therefore, cannot be selected (see figure 4.10).

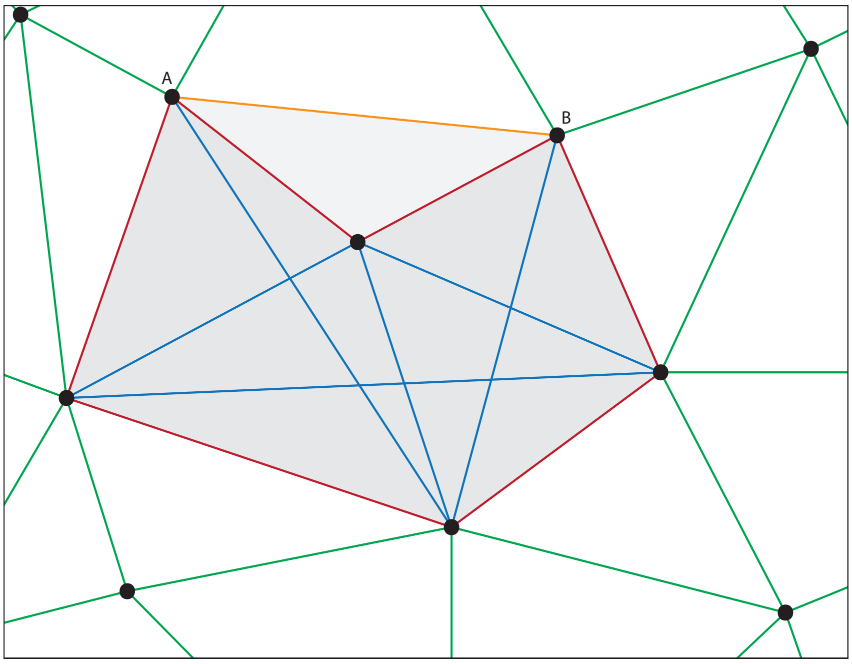


Figure 4.10: If the connection between the vertices A and B is inexistent, the cluster is concave. If the connection exists, it will necessarily belong to the cluster. In this case, the cluster is convex and contains one internal point.

In the case where a connection between two vertices in the cluster that lies outside the cluster exists, this connection will be included in the cluster, since every connection between two vertices belonging to a cluster is also considered to be in the cluster. But, if this connection belongs to the cluster, than the cluster is convex, has an internal point, and will therefore not be solved by this method, but by the method explained below. This can be visualized with the help of figure 4.10.

4.5.2 Internal points

Clusters with many internal points can be rather challenging to be re-triangulated. Their properties might vary a lot depending on the number of internal points and in the spatial organization of these points. The success of an algorithm to re-triangulate such clusters involves a delicate balance of restricting sufficiently which connections can be selected so that the final configuration re-triangulates the surface, while still leaving enough flexibility so that a configuration can at all be found.

Every restriction included in the algorithm discussed here was included with the goal of selecting the smoothest re-triangulation possible. First, shorter connections are preferred (longer connections are rare, and the selection of a long connection is likely to lead to a situation where the triangulation of the cluster cannot be finished – see figure 4.11). Additionally, it is required the all triangles formed by two accepted connections have the same orientation (the definition of orientation will be given below). The problem with wrongly oriented triangles is that, in order to finish the re-triangulation, a long connection will likely be required (see figure 4.11. At last, it is required that, when projected to the plane defined by a triangle, other points in the cluster do not fall inside this triangle (this would likely end up in a wrongly oriented triangle – see figure 4.11).

The algorithm presented below works for the majority of cases but it is not fail safe. A discussion of the limitations of this procedure and possible solutions is given in chapter 7.

In order to re-triangulate a cluster with internal points, the following procedure is used:

- One of the two possible ordering for the boundary connections is arbitrarily defined as positive.
- Each triangle composed by at least two boundary points has its orientation calculated.
- A global orientation of each cluster is calculated as the average orientation of all the triangles.
- Distances from all internal points to the boundary are calculated.
- The number of consecutive connections between an internal point and the boundary is calculated.
- For each triangle formed by an internal point and two of its consecutive connections to the boundary, it is checked whether there is either another internal point or a boundary point whose projection onto the plane defined by the triangle falls inside the triangle.
- For each triangle formed by an internal point and two of its consecutive connections to the boundary, it is checked whether its orientation is the same as the cluster orientation.
- The internal point that has at least two consecutive connections to the boundary with no projection inside and the same orientation as the cluster with the

smallest distance to the boundary is then selected. The consecutive connection between this point and the boundary that obey the required restrictions belong to the final configuration.

- The internal point used and its selected connections to the boundary are included in the cluster boundary, modifying it.
- The whole process is repeated, with one less internal points and the updated boundary.
- When all internal points are incorporated to the boundary, the hole left is triangulated using a method similar to the case of clusters with no internal points, but with the additional restrictions about orientation and projection of points. It is important to notice that, due to these stronger restrictions, it cannot be proven anymore that the two assumptions made for the method to re-triangulate clusters without internal points are still valid here.

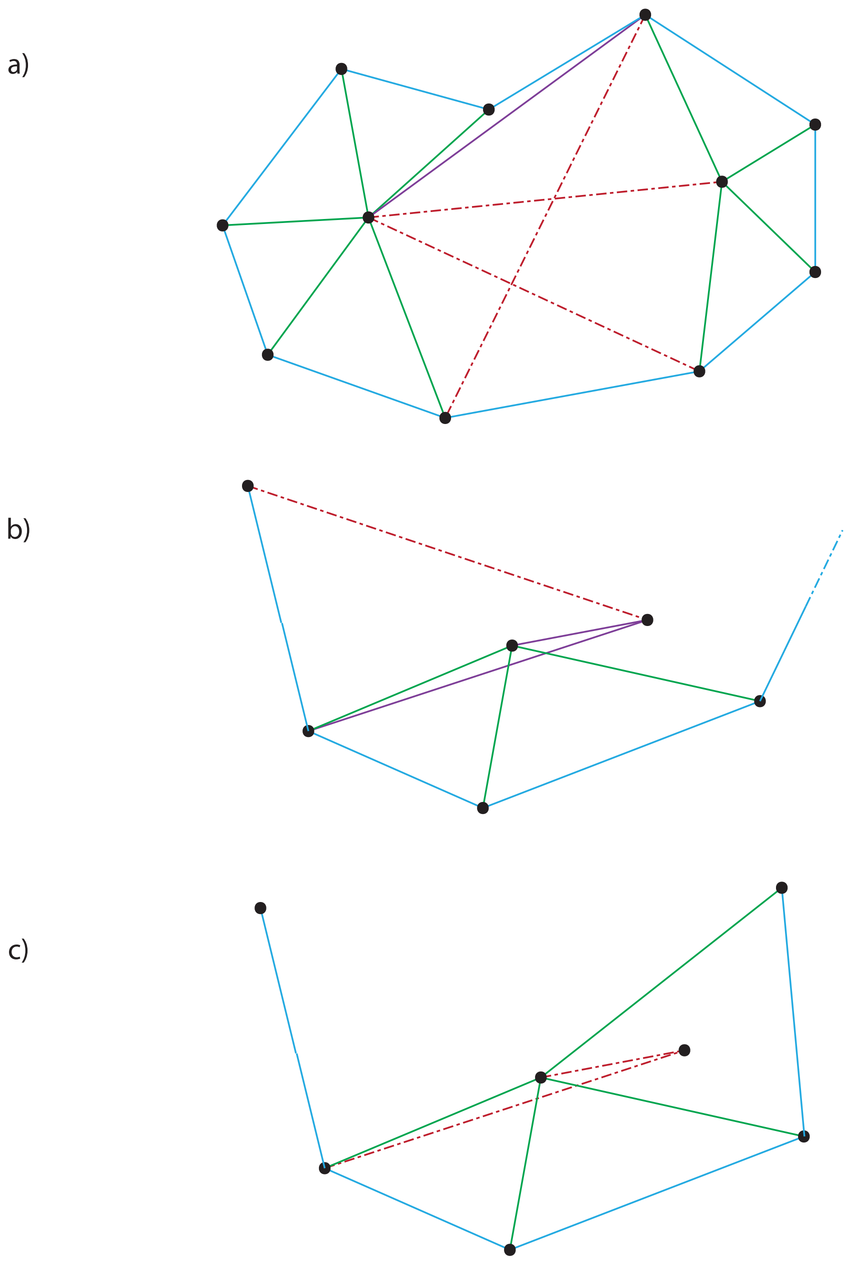


Figure 4.11: Illustration of the most common problems on the re-triangulation of a cluster. a) Long selected connection (in purple) leads to unfinished re-triangulation (if connections in red do not exist in the original triangulation); b) Wrongly oriented triangle leads to long connection. In order to the triangle in purple to belong to the final triangulation, the connection in red must exist; c) Allowing the projection of a point to fall inside a triangle leads to wrongly oriented triangles. If the green triangle with the point inside is selected, the only way to re-triangulate this point will be via the triangle made by the red connections, that have the wrong orientation.

CHAPTER 5

Generating initial configurations

Contents

In this chapter the thermalization of cell configurations prior to a simulation will be explained. As detailed in chapter 3, a cell is composed of two different types of sub-particles: membrane particles and cytosolic elements. The interaction potentials and dynamics are also used as explained in chapter 3.

The generation of a initial configuration starts only with membrane particles. These particles are generated randomly on an artificial spherical shell. In order for these particles to stay on this spherical shell, a hard sphere where the membrane particles cannot penetrate is included. Additionally, a single artificial particle is generated and put to the center of the sphere. This particle interacts with the membrane particles attractively, like a central potential.

Within the spherical shell, the membrane particles interact elastically. No cytosolic elements are yet included, only the artificial particle responsible for the central force. The membrane particles will interact in this setup until they are uniformly distributed on the spherical shell. The configuration is accepted as thermalized when a defined accuracy of the distance between the particles is achieved.

The Delaunay triangulation of all particles involved in the interactions is calculated and used in order to obtain the neighborhood relations of the system. The triangulation is updated at every time step accordingly to the changes in the particles' positions.

Once the membrane particles are thermalized on the spherical shell, the artificial particle is removed and the cytosolic elements are randomly placed inside the shell. They interact with each other and with the membrane particles using a Lennard-Jones potential (equation (3.5)). However, the membrane particles, at this stage, are fixed and therefore don't feel the interaction with the cytosolic elements. The cytosolic elements then interact until they are thermalized.

5.1 Thermalizing membrane particles

Supposing a cell with $N_{memb} = 1000$ membrane particles and a rest distance between the particles equal to 1, the radius R of the spherical shell can be calculated as

$$R = \sqrt{\frac{\sqrt{3}}{2}\frac{N_{memb}}{4\pi}}. \tag{5.1}$$

The elastic constant used for the elastic interaction between membrane points (equation (3.7)) is $\kappa = 30$. Since the variance for the average distance between two particles tends to an asymptotic value between 10% and 15%, thermalization is defined to be achieved when the variance on the average distance is smaller than 14.5%. In figure 5.1 the membrane particles are shown before and after thermalization.

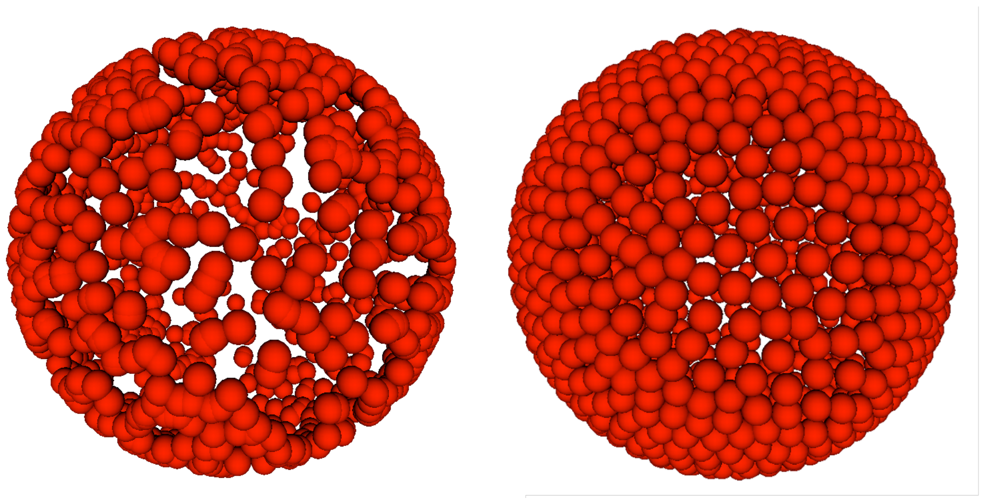


Figure 5.1: Configuration of membrane particles before (left) and after (right) thermalization.

5.2 Thermalizing cytosolic elements

In this example, cytosolic elements are set to have a radius (and, therefore, an interaction rest distance) twice as big as membrane particles. The number of cytosolic elements is calculated by

$$N_c = 0.74\frac{(R - r_m)^3}{r_c^3}, \tag{5.2}$$

rounded up to the next integer. Here r_m is the radius of a membrane particle and r_c is the radius of an internal particle. The factor 0.74 comes from the supposition that, after thermalization, the spheres will be approximately in the densest possible packing (known as cubic close packing). It has been proven [Hsiang 1993] that, in this case, the density of spheres will be given by

$$\frac{\pi}{3\sqrt{2}} \approx 0.74. \tag{5.3}$$

The set of parameters used for the Lennard-Jones interaction between cytosolic elements (equation (3.5)) was $\varepsilon = 1$ and $\sigma = 2r_c 2^{-1/6}$, where $2r_c$ equals the rest distance between the elements. For the Lennard-Jones interaction between membrane particles and cytosolic elements (equation (3.5)) the parameters were $\varepsilon = 1$ and $\sigma = (r_c + r_m)2^{-1/6}$.

The criteria to define whether thermalization was achieved is the same as for membrane particles. In figure 5.2, a final thermalized configuration is shown. This configuration can then be used as the initial configuration for a cell in a simulation. Additionally, figure 5.3 visualizes all connections between cytosolic elements (i.e., of the Delaunay triangulation) after thermalization.

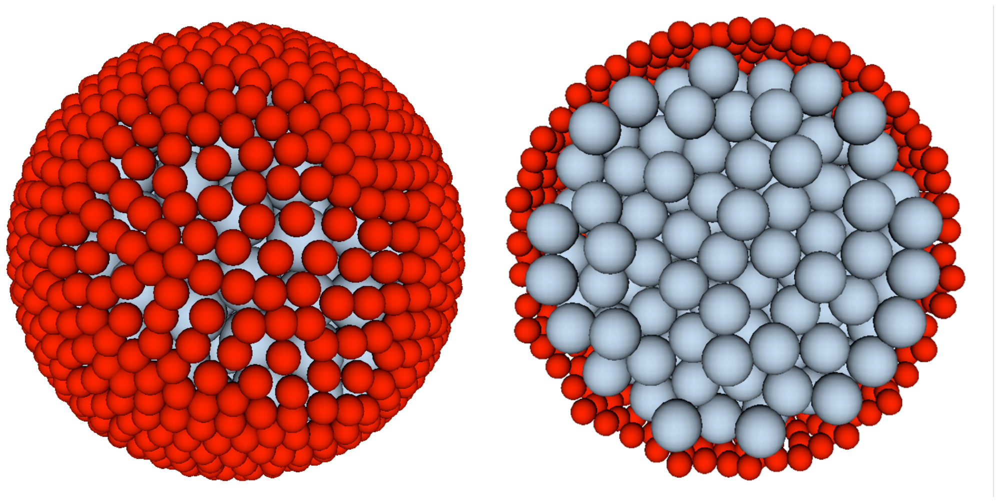


Figure 5.2: Thermalized configuration with membrane particles (dark colored) and cytosolic elements (light colored). In the right side, the cell is shown sliced in half. Notice that in this example the radius of the cytosolic elements is twice as big as the radius of the membrane particles. The size of the particles is related to the strength of the potential as well as to the resolution used to describe the cell.

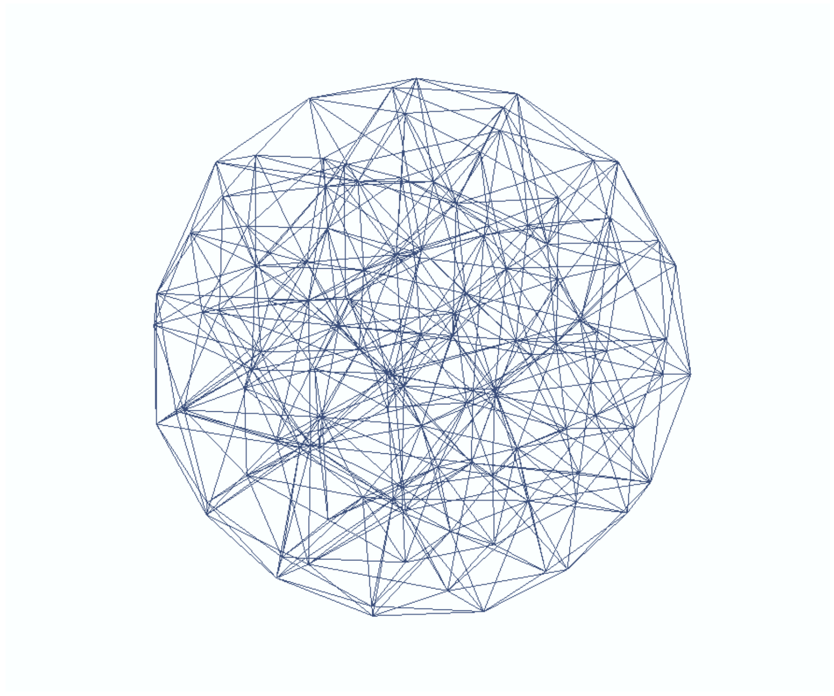


Figure 5.3: The three-dimensional Delaunay triangulation of all cell's cytosolic elements after thermalization, for a simulated system with 500 membrane particles (not shown) and 115 cytosolic elements.

Chapter 6

Testing the surface reconstruction

Contents

In this chapter results from tests performed with the surface reconstruction method discussed in chapter 4 are presented. In section 6.1 the parameters used in the generated initial configurations are introduced, and the computational setup used in the simulations presented. Section 6.2 shows images of the generated configuration, while section 6.3 presents a discussion of the convergence and simulation times obtained with the surface reconstruction method.

6.1 Setup

For the simulation tests, many different sizes of systems were studied. Small systems (with less then 500 membrane particles) function well, but the small number of cytosolic elements is likely to limit the eventual production of blobs on the cell due to the lack of internal resolution. On the other side, big systems (with much more than 1000 membrane elements) were much more prone to lead to errors on the re-triangulation and significantly slower. Therefore, for the results shown here, systems with 500 and 1000 membrane points were chosen.

A system with 500 membrane points has 115 cytosolic elements, while a system with 1000 points has 352, as calculated by equation (5.2). In order to have a variety of systems to test, five initial configurations (with different random numbers)were generated for each system size.

Additionally, the initial systems were set further apart from each other with the use of a parameter called in this section *variation*. The variation represents a displacement applied in a random direction for each membrane particle in the initial configuration used. Variations were also useful for the initial systems to get away from the original spherical configurations imposed during the construction of initial conditions.

The values used for the variations of the position of the membrane particles were $0.01, 0.02, 0.03, 0.04$ and 0.05, in the reduced units used in the program. For comparison, the rest distance between membrane particles is, in the same units, equals to 1. Variations also give a measure of how resilient the algorithm is to changes in the membrane, i.e., how far a configuration can be from the spherical initial system and still be properly re-triangulated. The bigger the variation, the farer from the original initial configuration the new system is.

For each of the five initial conditions for each system size, 100 random applications of each variation value were made. Therefore, 500 different systems were generated for each cell size and value of variation. Each pair of random application of variation and initial condition was additionally run 500 times, solely for statistical reasons. In total, 25000 runs were made for a single system composed of a number of particles with a determined variation.

All simulations presented in this chapter were done in a Linux machine with an Intel dual core of 2.4 GHz and 2 Gb of RAM memory.

6.2 Visualization

In order to better visualize what is happening when the re-triangulation method is applied to a cell, snapshots were done before and after running the algorithm. These snapshots show the connections between membrane points. A visualization of all connections between cytosolic elements in a cell is shown in chapter 5 (figure 5.3).

On figure 6.1 the initial configuration of a cell with 500 membrane points and variation 0.05 is shown. In the top panel the whole cell with all its clusters highlighted can be seen. The lower panels show a zoom to a set of clusters in the membrane. The clusters in the left panel are highlighted with different colors, while in the right panel the connections belonging to the clusters are shown.

Figure 6.2 shows the same cell as figure 6.1, but with the final configuration for the membrane. The left panel shows an overview of the whole cell, while the right panel shows a zoom to a small piece of the membrane.

Another interesting possibility is to compare how the number of clusters in a cell changes with the variation. In figure 6.3 the triangulation of a cell membrane with 1000 points is shown. The variation used to generate the configurations were 0.01, 0.03 and 0.05. Both the quantity of clusters and their size grow withe the variance.

6.3 Testing results

In this section an analysis of the performance of the re-triangulation method presented in chapter 4 is given. First, the time needed to re-triangulate a single cluster

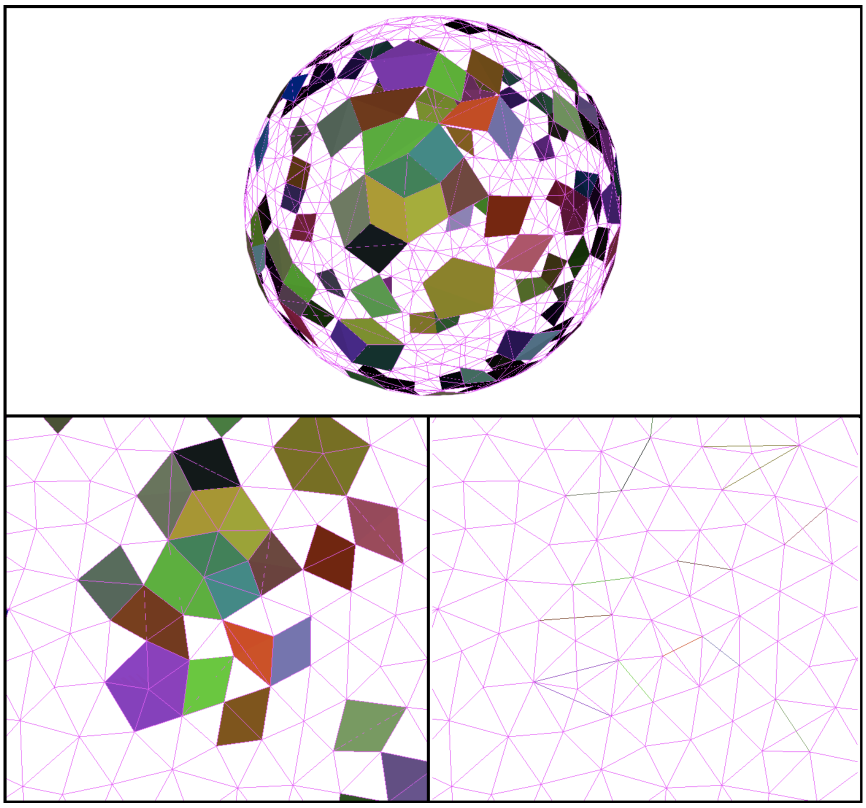


Figure 6.1: Visualization of the initial configuration of a cell with 500 membrane points and variation of 0.05. In the upper panel the membrane connections and the clusters in the cell are shown. In the lower left panel a close up in a set of clusters is shown, while in the right panel the same close up is shown, but now with the connections in the cluster highlighted.

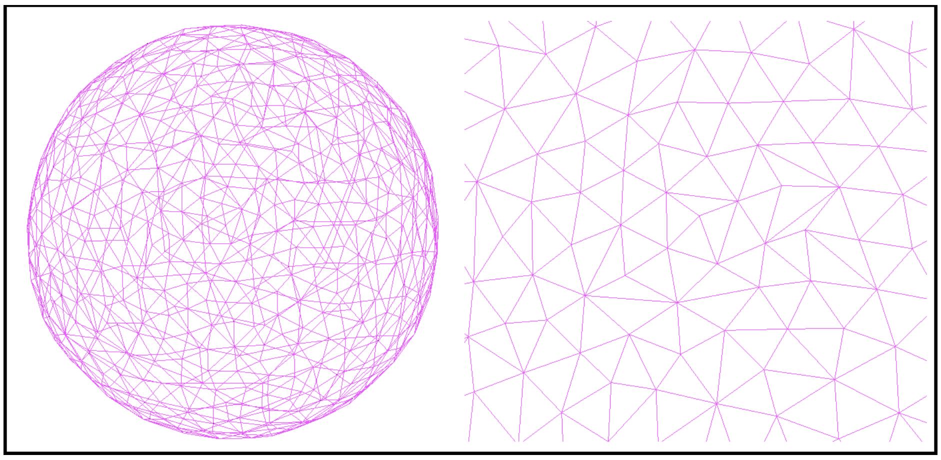


Figure 6.2: Visualization of the final configuration of a cell with 500 membrane points and variation of 0.05. Left panel: overview of the whole membrane. Right panel: zoom of a small part of the membrane.

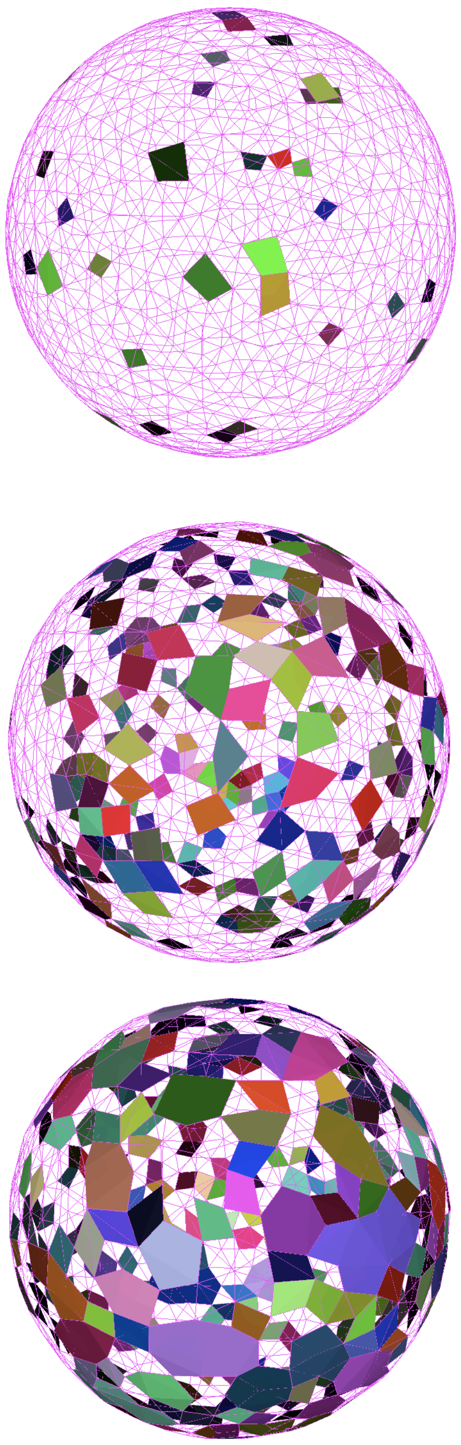


Figure 6.3: Visualization of the initial cluster configuration for cluster with 1000 membrane points and variances of 0.01, 0.03 and 0.05. Each cluster is identified by a random color.

is studied, followed by an analysis on how the time needed to re-triangulate each point in a cluster varies with the cluster size. The frequency of appearance for each cluster size are also compared, as this information is necessary to guarantee that the total time needed by to re-triangulate a cell is convergent. At last, the total time spent re-triangulating a surface is plotted, as well as a scaling for the total time with the variation.

6.3.1 Solving time per cluster

The first property analyzed in the system was the scaling of the time needed to re-triangulate a cluster versus the cluster size. In figure 6.4, the scaling behavior for a system with 1000 membrane particles and variation of 0.05 is shown. Since small clusters are much simpler to be re-triangulated than big clusters, the expected behavior in that the time necessary to solve a cluster increases with the size of the cluster.

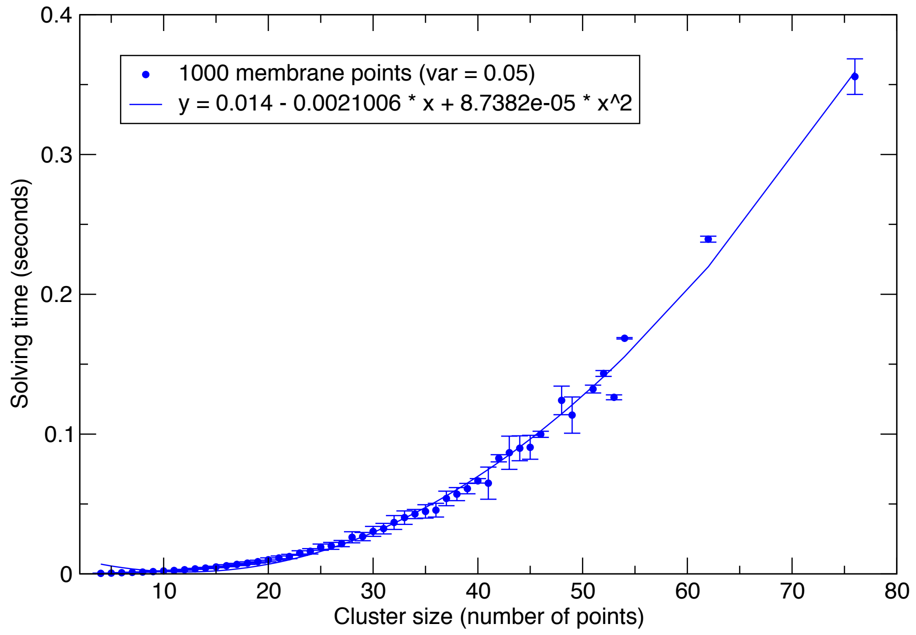


Figure 6.4: Average time needed to re-triangulate a cluster versus the cluster size, for a system with 1000 membrane points and variation equals 0.05. Errors were calculated using 25000 runs. The full line shows a quadratic fit for the experimental data. The fit and the experimental data deviate for small cluster sizes.

Indeed, this expected behavior is observed on the simulation results. Additionally, the observed behavior is approximately quadratic, as can be seen in the quadratic fit plotted also in figure 6.4. However, for small cluster sizes, the observed behavior deviates from the quadratic curve. This deviation occurs because the re-triangulation routine is not quadratically dependent on the number of total points in the cluster, but on the number of internal points only.

While the majority of the points in a big cluster is indeed internal, small clusters tend to have few or none internal points, explaining both the quadratic behavior for big clusters and the deviation for small clusters. An additional factor to explain the deviation is the use of different algorithms to solve clusters with and without internal points.

In the upper panel of figure 6.5 the curves for 500 and 1000 points are compared. The average time needed to solve a cluster in the smaller system is in agreement to the time needed to solve a cluster of the same size in the bigger one. In the lower panel of figure 6.5, the time to solve a cluster is compared for different variations in the surface. As the variation shouldn't affect the time needed to solve a cluster, there is no relevant difference between the results as well.

6.3.2 Solving time per cluster point

A noteworthy behavior easily derived from the one above is how much time in average it takes to re-triangulate each point in a cluster. This result can be generated by the previous one by dividing the total re-triangulation time by the cluster size. Because the solving time per cluster is approximately quadratic with the cluster size, it is expected that the solving time per point in a cluster versus the cluster size approximate a linear behavior (for higher cluster sizes).

In figure 6.6 the re-triangulation time per point is plotted as a function of the cluster size, for a system with 1000 membrane points and variation 0.05. The linear result can be observed for cluster with more then 20 points.

Figure 6.7 plots the solving time per cluster point versus the cluster size, but now comparing the results for, on the upper panel, systems with 500 and 1000 points and, on the lower panel, clusters with 1000 points but distinguished variations. As expected from the results shown in section 6.3.1, the average time per point does not depend on the system size or on the variation value. Since these graphs only show small cluster sizes (the values are only plotted up to the highest cluster size observed in the smallest system), the presented behavior is not linear.

Despite being directly obtained from the results presented in section 6.3.1, knowing how the solving time per point in the cluster behaves with the cluster size is still important. The fact that it behaves approximately linear means that the complexity of the clusters grows with the cluster size. Re-triangulating a point in a bigger cluster is more difficult than in a small cluster. Therefore, an important property to be checked is how frequently big clusters appear on the system. This property can show whether the re-triangulation algorithm implemented converges or not.

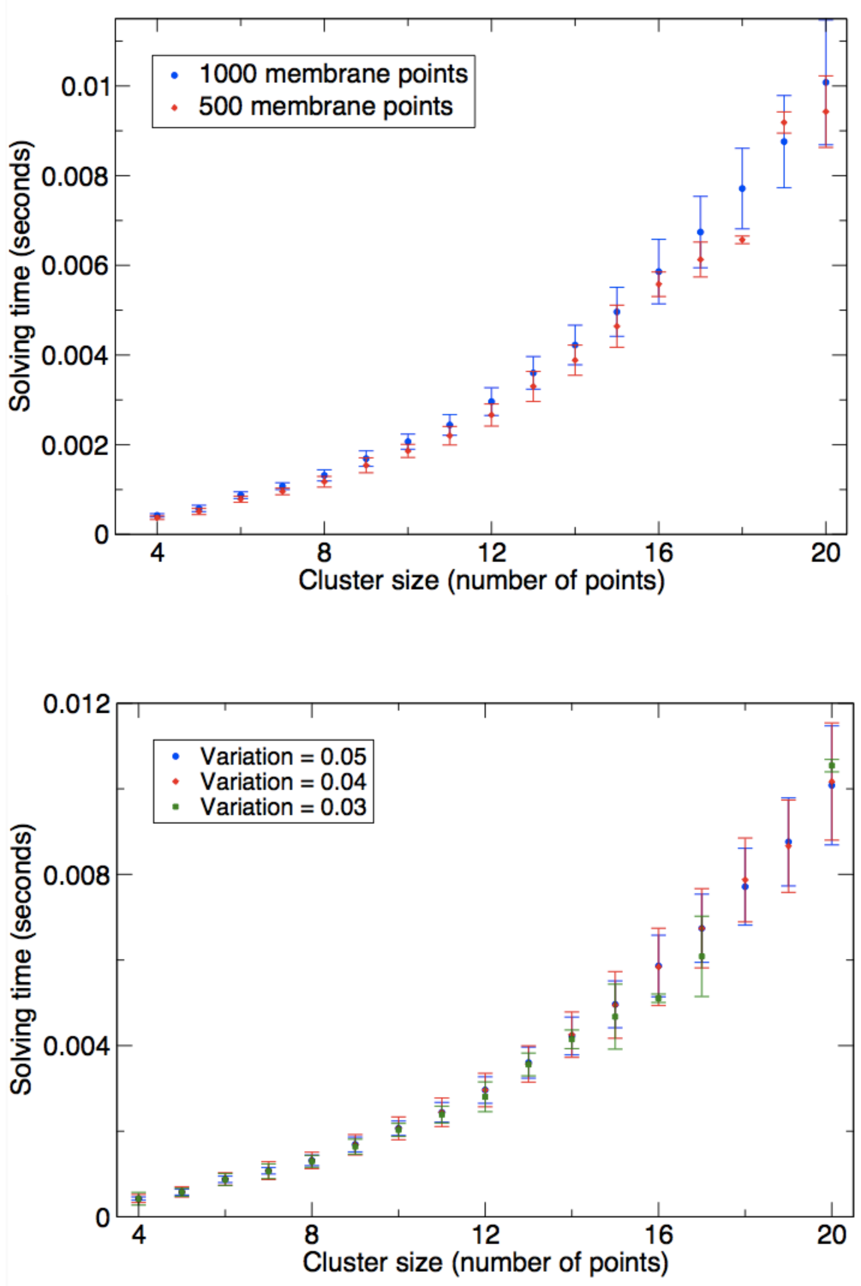


Figure 6.5: Average time needed to re-triangulate a cluster versus the cluster size. On the upper panel, curves for systems with 1000 and 500 points (variation = 0.05) are plotted. On the lower panel, curves for systems with 1000 points but different variations are given. Values are plotted up to the highest cluster size observed in the smallest system (the one with 500 points in the upper graph and the one with variation = 0.03 in the lower one).

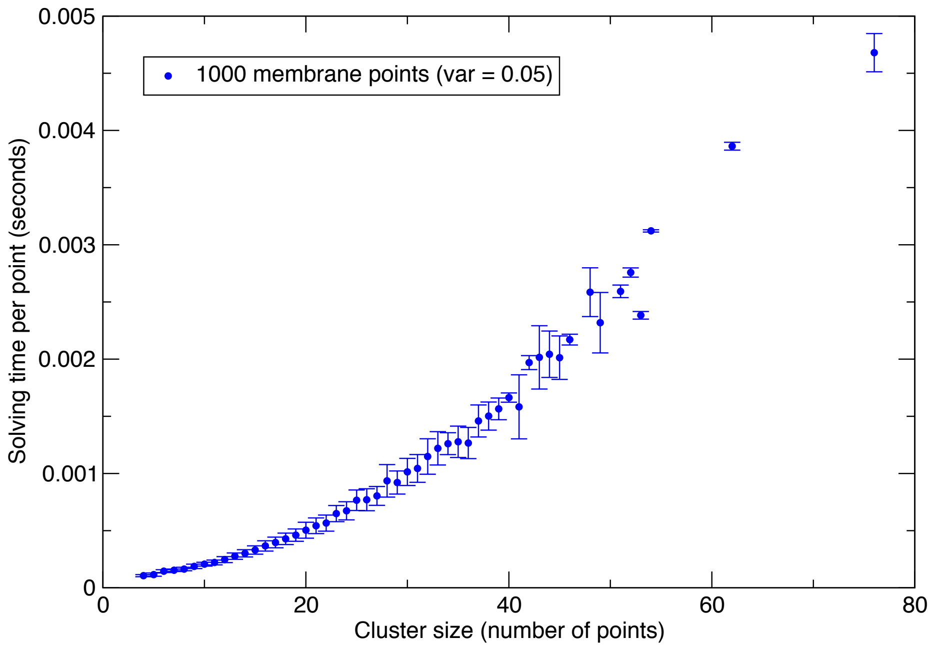


Figure 6.6: Average time needed to re-triangulate a cluster per cluster point versus the cluster size. The simulated system has 1000 membrane points and variation = 0.05. Although not plotted, an approximately linear behavior can be observed for cluster above 20 points.

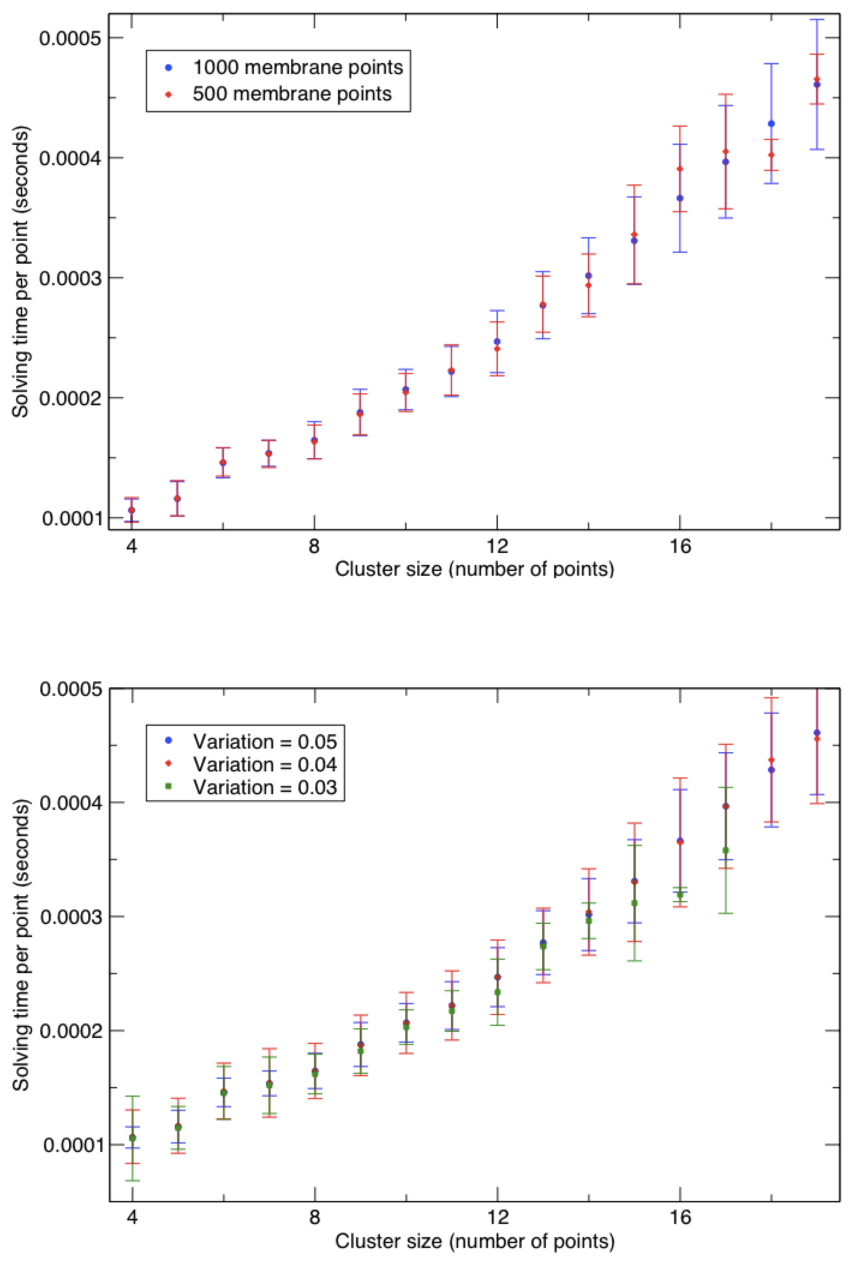


Figure 6.7: Average time needed to re-triangulate a cluster per cluster point versus the cluster size. On the upper panel, the results for systems with 1000 and 500 membrane particles (variation = 0.05) are compared, while on the lower the results for three distinguished variations (in a system with 1000 particles) are seen. Values are plotted up to the highest cluster size observed in the smallest system (the one with 500 points in the upper graph and the one with variation = 0.03 in the lower one).

6.3.3 Cluster frequency

A fundamental property directly related to the convergence of the re-triangulation algorithm is the frequency of the appearance of clusters with a determined size in the simulations. Since, as seen in section 6.3.1, the re-triangulation time for bigger clusters scales approximately with their size squared, simulation time risks to explode if the frequency of appearance of big clusters does not scale down at least with the same behavior.

In figure 6.8 the frequency of appearance of a cluster versus its size is shown. In its upper panel, the frequency for systems with 500 and 1000 particles are compared. For both system sizes the frequency decreases exponentially with the cluster size but, as expected, the frequency decays faster for the smaller system. In the lower panel of the figure the frequency is also plotted against the cluster size, but now for the same system size (1000 membrane particles) and different variation values. As the smaller the variation the less perturbation on the system, it is also expected that the frequency decays faster for smaller variations.

The fact that the frequency scales exponentially with the cluster size is relieving, as it suggests that the total simulation time for bigger systems will not diverge.

6.3.4 Total solving time

To confirm that the total re-triangulation time is not divergent, the total time spent solving all clusters of each size was calculated. This was done by multiplying the frequency of appearance of each cluster size by the average time needed to re-triangulate a cluster of this size.

Since the frequency decays exponentially with the cluster size and the re-triangulation time scales quadratically with it, the behavior for the total time is expected to also exponentially decrease with the size of the clusters. This is confirmed for the different system sizes and variation degrees (see figure 6.9).

6.3.5 Analyzing variation

The scaling behavior of the the total time needed to re-triangulate all clusters versus the variation was plotted (figure 6.10). The figure shows the total re-triangulation time for a system with 1000 membrane particles and 5 variation values.

For a system with 1000 membrane particles and variation 0.05, the total time needed to obtain the re-triangulated surface is around 0.26 seconds. Notice that this time is only spent in the processes specific to the membrane re-triangulation and does not involve the calculation of the original Delaunay triangulation.

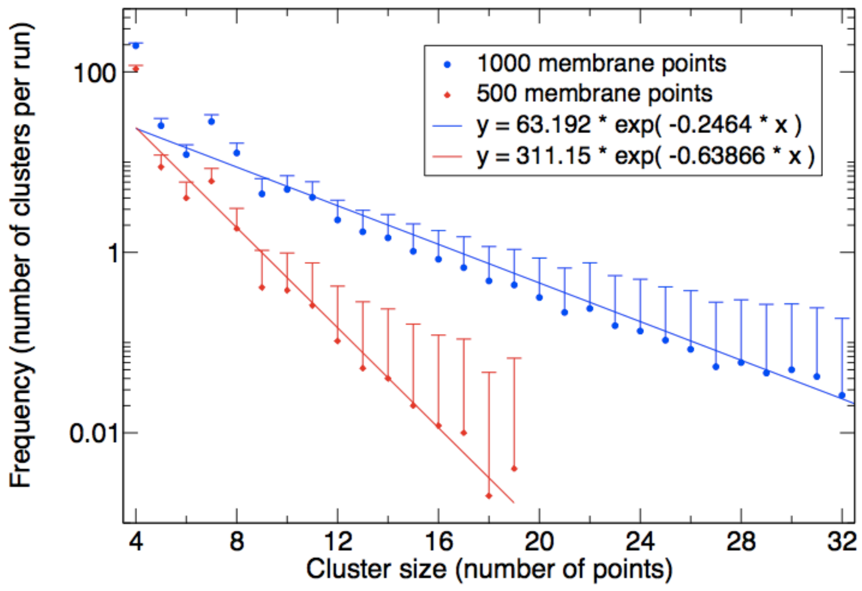
6.3.6 Time efficiency of the algorithm

At last, a comparison between the time spent on the re-triangulation algorithm, on the usual update of the three-dimensional Delaunay triangulation and on the application was considered.

Evidently, the time spent in each of these algorithms depends on the number of particles in the system. Additionally, the algorithm to update the three-dimensional Delaunay triangulation and the algorithm for the application depends on how big the maximum displacement allowed to the particles in the system is, and the re-triangulation algorithm presented in this thesis depends on how big the variation chosen is.

As shown in figure 6.10, the total time needed to obtain the re-triangulated surface was about 0.26 seconds for a variation of 0.05. However, the variation of a cell will certainly not remain constant during the course of a simulation. Actually, the most probable is that the variation of a cell gets bigger with the simulation time, hopefully reaching a stable value after some time.

As, to the present moment, an application of the model was not done, it is difficult do really make calculations about the time spent by the program in each of the algorithms involved. However, preliminary studies made with small applications show that, for a cell with 1000 membrane particles and variation of 0.05, whose particles can move up to 10% of their radius each time step, the update of the three-dimensional Delaunay triangulation takes around 0.7 seconds, the re-triangulation algorithm takes from 0.15 to 1.5 seconds and the time spent in dynamics is virtually negligible compared to the other two, in the order of 0.01 seconds.



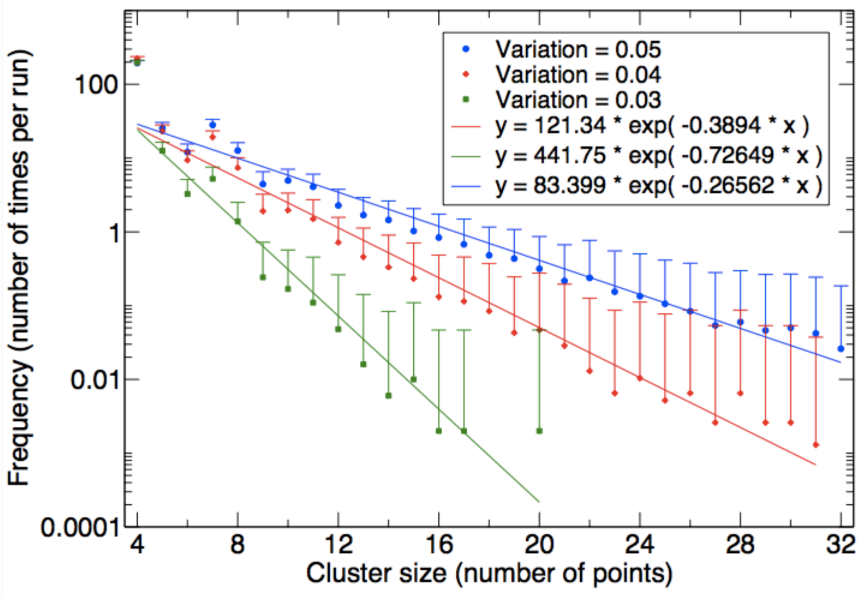
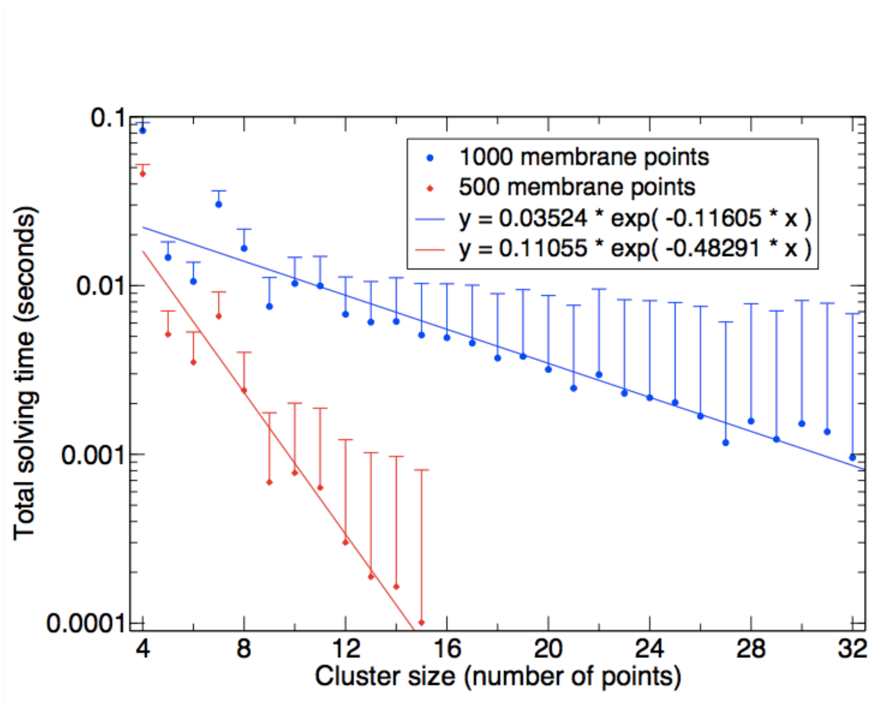


Figure 6.8: Frequency of appearance of a cluster of a determined size versus the cluster size. On the upper panel, two system sizes (500 and 1000 membrane points) are compared. On the lower panel, systems of the same (1000 membrane points) size but with different variation values are compared. The full lines show the best exponential fits for each plot.



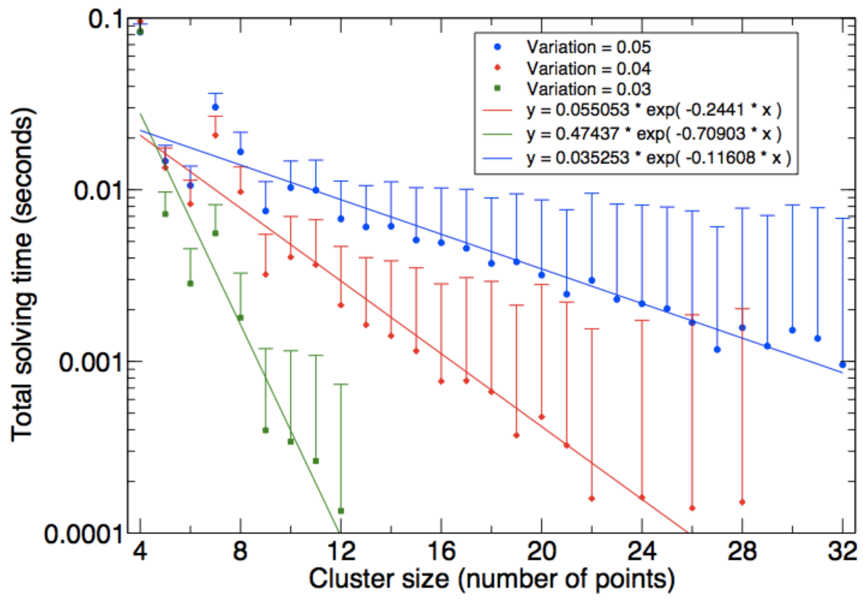


Figure 6.9: Total time spent re-triangulating all cluster of a determined size versus the cluster size. Upper panel: plots for systems with 500 and 1000 membrane points (variation = 0.05). Lower panel: system with 1000 membrane points and variations = 0.03, 0.04 and 0.05. The full lines show the best exponential fits for each plot.

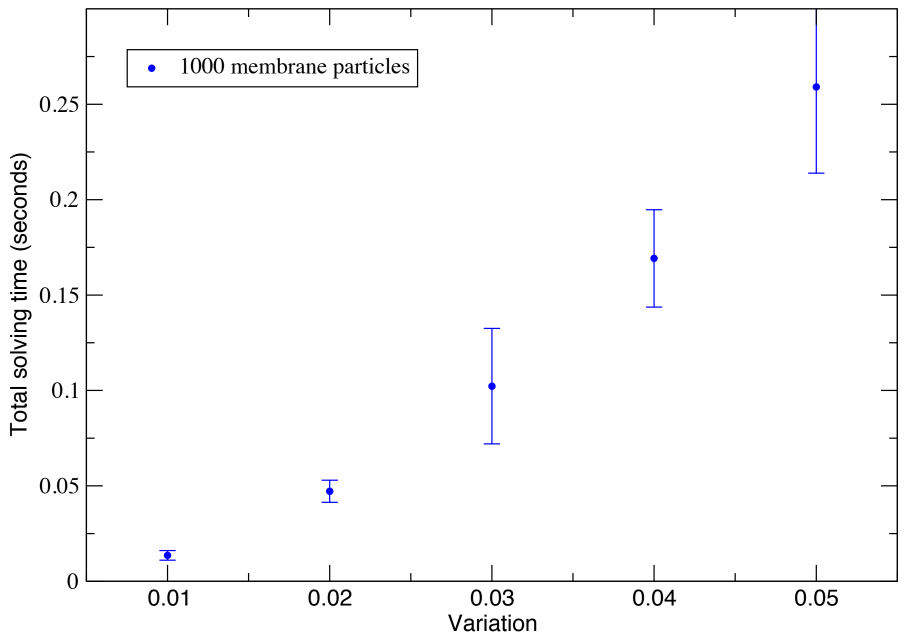


Figure 6.10: Scaling of the total time needed to re-triangulate a surface versus its initial variation. The system used had 1000 membrane points and the variation was raised from 0.01 to 0.05.

Chapter 7

Discussion and conclusions

Contents

7.1 Error handling on the surface reconstruction

The problem of re-triangulating a set of points restricted to a two-dimensional bended surface is very complex and not completely solved. The algorithm presented here is highly tailored to a specific application, but still not fail-safe.

The method explained in chapter 4 has two weak points from where errors can arise: boundary selection and too strong restrictions in the re-triangulation algorithm for clusters with internal points. When no boundary can be selected, the specific cluster cannot be re-triangulated and the whole process fails, as without boundary selection no re-triangulation algorithm can be applied to the cluster. When the restrictions are too strong, none of the connections belonging to the cluster manage to fulfill the algorithm requirements at some point, and a hole is left in the surface.

The number of surfaces not properly triangulated vary with the system characteristics. It increases with both the variation used and the size of the system. The number of runs that failed to generate a properly re-triangulated surface were 2 out of 500 and 15 of 500 for systems with 1000 membrane particles and variations (random displacements on the initial positions of membrane points) of 0.04 and 0.05, respectively. Every configuration generated with a variation smaller then 0.04 successfully generated a properly triangulated surface.

Smaller systems (500 membrane particles) can tolerate higher values of variation without failing. The first variation value for which a system with 500 membrane particles fails is 0.1, with 8 out of 500 runs finishing not properly re-triangulated.

7.1.1 Possible improvements

Although failures on the algorithm used to select the clusters' boundaries happened during the test simulations, no re-triangulation ever failed because of it. The problems were always reversed when the procedure to identify the boundary was rerun after some of the clusters on the surface had been successfully re-triangulated.

Still, improvements on this routine could be achieved by using a boundary orientation in order to decide, when two possible boundary orders are found for a given cluster, which one is the right one. That would be a simple addition to the existing procedure.

The impossibility to find a suitable triangulation for a cluster, leaving it partially untriangulated (with a hole) was the cause for all failed configurations found in the test simulations. Yet, finding a solution for these failures might, in the best case, take a lot of tweaking of parameters in the model and, in the worst case, be impossible.

A possible improvement could involve a re-triangulation algorithm that imposes restrictions in a progressive way according to the number of internal points in the cluster. The more internal points a cluster has, the more complex it is, and the more restrictions needed in order to find a suitable re-triangulation. However, these restrictions are, sometimes, to strict when the cluster being re-triangulated does not involve too many internal particles. By allowing connections in a cluster with medium complexity (less than 10 internal points) to be selected using less restrictive conditions, some of the wrong configurations obtained with the tests could be avoided.

The biggest problem with all restrictions included in the method to solve clusters with internal points is related to the initial assumption that every connection in a cluster belongs to a possible triangulation of this cluster. When, for example, long connections are automatically deleted from the re-triangulation because they have non-physical sizes, part of the cluster may only be able to be triangulated in a single configuration. If then, by any reason, one of the connections left cannot be chosen because of any of the other criteria included, the re-triangulation will fail to be concluded.

In reality, every time all internal points of a cluster are re-triangulated and there is a hole left, if this hole is not a set of simplices there is a probability that it will not be properly triangulated. As stated in chapter 4, the clusters can only certainly be re-triangulated because they are formed by simplices. With that in mind, another possibility of improvement would be to, when a re-triangulation fails, to try to recover a configuration including the not triangulated part that is composed of simplices. The algorithm to re-triangulate a cluster without internal points could then be used on the recovered configuration without modifications.

A more radical approach would be, when a cluster re-triangulation fails, to restart it with different initial conditions until a valid re-triangulation is achieved. This possibility could be implemented relatively fast, but has the drawback that it would slow down the simulations substantially. It is, however, a last resort approach that could be used in sporadic cases.

When using the method to define neighborhood relations in a dynamic system, a last approach for dealing with not re-triangulated clusters would be to simply ignore them for a couple of runs, using instead the last connection this particle had before the triangulation failed. This could lead to defects on the membrane (or whatever surface is being simulated), like small holes or crossing connections. However, since the system is dynamic, the position of the particles in the cluster would change every time-step, and the configuration could evolve to a state where the cluster could then be solved.

7.2 Adaptive membrane re-triangulation

Apart from the procedures discussed in the section above, there are other improvements that could theoretically be applied to the surface reconstruction method. One of the most interesting possibilities is the inclusion of an algorithm that just updates the re-triangulation of a surface when it breaks, instead of re-triangulating the whole surface at every time step. This idea is inspired by the function to update a Delaunay triangulation dynamically. Such function is already implemented in the program used to handle the three-dimensional Delaunay triangulations.

An algorithm that just updates the re-triangulated surface instead of calculating a totally new re-triangulation could be extremely time efficient. The outline of such an algorithm would be the following: before any dynamics is implemented in the program, the re-triangulation of the initial surface is done using the surface reconstruction algorithms presented in chapter 4 of this thesis. Then, at each time step, after the update of the three-dimensional Delaunay triangulation, the existence of the connections that belong to the actual restricted triangulation on the surface are checked. The re-triangulation algorithm is then performed only in the areas where the connections chosen before hand are no longer available.

The tricky part of the implementation of this idea comes from defining the criteria to select the areas to be re-triangulated. Each area then would be handled as a cluster, and the algorithms to re-triangulate clusters discussed in chapter 4 used as normal. In this case, a total re-triangulation of the surface would only be needed in rare cases where the original three-dimensional Delaunay triangulation changed substantially between two time steps. This, however, happens rarely and usually require a re-calculation of the original Delaunay triangulation from scratch as well.

7.3 Re-triangulating non-convex surfaces

The surface reconstruction algorithm discussed in chapter 4 was also tested in non-convex surfaces, with mixed results. The test-surfaces were prepared using two initial configurations for two spheres with different sizes, with half of the smaller sphere being connected to the bigger sphere as a protrusion (see figure 7.1).

Although the algorithm was able to re-triangulate some of these surfaces, it was not nearly as reliable as when used to re-triangulate convex surfaces. Non-convex

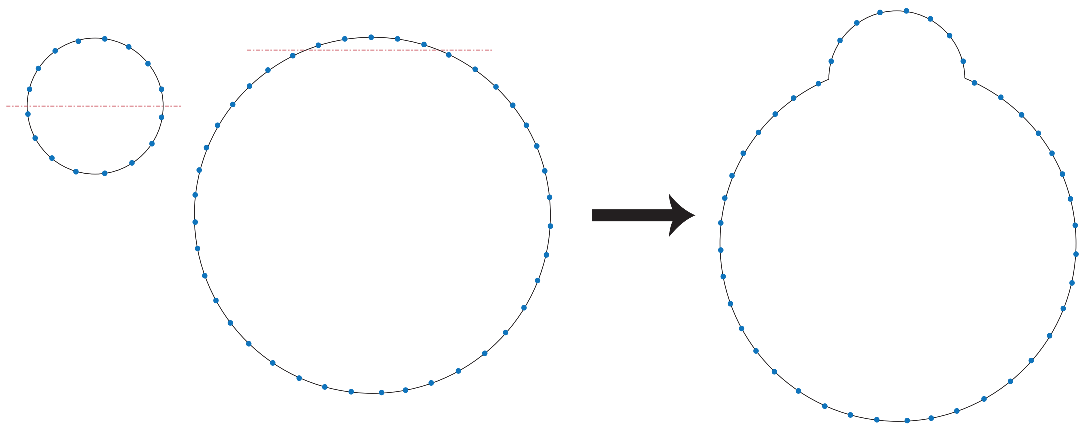


Figure 7.1: Schematics on how to generate a non-convex initial configuration for a cell. A small and a big spherical initial configurations are generated. The lower half of the small sphere and part of the top of the big sphere are deleted, and both spheres are glued together.

surfaces are particularly tricky with the cluster approach developed in this thesis. Because of the non-convexity, long connections are generated connecting parts of the membrane (figure 7.2). These long connections do not happen inside the membrane due to the presence of the cytosolic elements. Even though long connections in the membrane are always dismissed as non-physical, this only happens after the identification of the clusters. And clusters formed as a sum of simplices including these long connections have a tendency to be really big, even when the variation in the surface is small.

Evidently, the bigger the cluster the more complicated it is to find a re-triangulation. Additionally, initial configurations for cells with blobs are not properly thermalized, as they are generated by the composition of two spheres thermalized separately (as shown in figure 7.1). This makes the connections on the edge between the two spheres to be reasonably out of equilibrium.

In order for non-convex clusters to be properly re-triangulated, the first step would be to smooth the initial configuration composed by the two spheres. Another possibility is to include artificial points outside the cell. This would be a hybrid solution between the methods to generate the quasi-surface and the crust (chapter 4). Improvements on the re-triangulation algorithm should make this case more reliable as well.

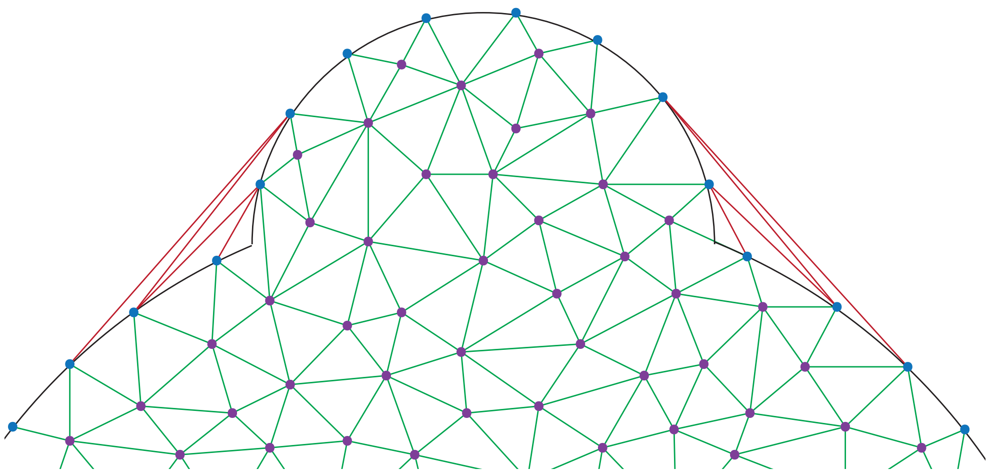


Figure 7.2: Long connections that are generated when the surface is non-convex (shown in the figure in red). This connections do not happen inside the cell because of the presence of cytosolic elements.

7.4 Simulating biological cells

In order for the framework presented in this thesis to be used in simulations for biological cells, the robustness of the surface reconstruction method for the membrane neighborhood relations must be improved. At the moment, the eventual failure on the membrane re-triangulation, specially on non-convex surfaces, can drastically reduce the situations where the method could be useful. The situation is definitely better the smaller the system considered, showing that one possibility of success could be to restrict the work to cells with up to 500 membrane particles.

Additionally, the path chosen for this dissertation was to stick to the originally decided idea of using a subset of connections from the original three-dimensional to define the membrane neighborhood relations. Even though this path led to the development of all the surface reconstruction techniques, the neighborhood relations between membrane particles need not to be related to the original Delaunay triangulation.

Alternatives methods to define the neighborhood relations between membrane points include fixed neighborhood relations (that could become complicated with the inclusion or deletion of membrane points – necessary for changing the cell's size) or neighborhood relations that depend on a distance cutoff. This method, however, has a quadratic behavior with the number of particles, and does not make use of the original available three-dimensional Delaunay triangulation.

Apart from the definition of the neighborhood relations, the next important step would be to define more complex interactions between the sub-cellular particles. The potentials included so far are good for simulating resting cells, but would not be sufficient in case of active cell movement or protrusions were included. The nature of these interactions depends mostly on the specific application of the model.

7.5 Conclusions

In this thesis a framework for the simulation of cells taking into consideration their internal structure was introduced. The framework involved sub-cellular particles, divided in two main groups: membrane particles and cytosolic elements. The neighborhood relations used in the model were defined via a three-dimensional Delaunay triangulation. They were used for all interactions, with exception of the interactions between membrane particles. The interactions and dynamics of the model were discussed, together with rules on how to generate initial configurations for the system.

Because of the nature of the interaction between membrane particles, it was conjectured that the best way to define neighborhood relations between them would be via a restricted Delaunay triangulation at the two-dimensional surface defined by these particles.

A method to define this restricted Delaunay triangulation was not available for the specific conditions required by the problem and, therefore, a method to define, from a set of sample points belonging to a determined surface, a Delaunay triangulation of the points restricted to this surface was developed and presented in this thesis. The method did not involve the deletion of sample points, being therefore specially suitable for applications in physics or life sciences.

The method consisted of many separated steps: obtaining a quasi-surface, subdividing the quasi-surface in independent structures (clusters), obtaining and ordering the boundary of each of the clusters and re-triangulating each of the clusters independently.

The algorithm used to obtain the quasi-surface required a closed convex surface and a minimum number of particles internal to the system (in this case, the cytosolic particles). These restrictions can be circumvented by the use of a crust instead of a quasi-surface.

At last, results for the time efficiency and reliability of the model, in particular of the surface reconstruction method developed, were presented. The possibilities to improve the methods and discussions about its limitations were included in this chapter.

Even though much is still to be done, the author believes that the work presented here is an important step on the way to the main goal of this project: to create an agent-based framework that not only allows the simulation of any sub-cellular structure of interest but also provides meaningful interaction relations to particles belonging to the cell membrane.

This framework has a vast field of potential applications, that go from the study of cell shape and migration to the study of intracellular dynamics, including the ability to follow molecules inside the cell in a space-resolved manner. The framework could, in principle, be applied to different problems in fields other than cell biology as well.

The author hopes to, with this work, have contributed in some meaningful way for both the field of sub-cellular models and the field of surface reconstruction, at least with new ideas and a fresh approach when trying to connect so different areas of knowledge.

Appendix A

Algorithm implementation

As an example of how the algorithms explained in this thesis were implemented, the details of the implementation of the routine used in the process of selecting and ordering the boundary of a cluster will be given in this appendix.

The input for the method is the three-dimensional triangulation generated for the membrane particles and cytosolic elements together. The program used to generate this triangulation gives access to the following, important containers:

1. a list of all simplices on the triangulation;
2. a list of all vertices on the triangulation;
3. for each vertex on the triangulation, a list of all its neighbors.

From the list of all simplices on the triangulation, it is possible to obtain a list of all clusters in the cell membrane. As explained in chapter 4, a cluster is defined as a set of simplices from the original triangulation formed only by membrane points, where every simplex shares a common face to at least another simplex in the set.

Each cluster on the membrane is an instance of a class. This class was called *Cluster* on the simulation. This class provides the following containers and variables (all private):

1. `list_simp` – a vector of all its simplices (the clusters are initialized with this vector);
2. `all_vert_in_cluster` – a hash set with all its vertices;
3. `ord_boundary` – an ordered vector with all vertices in its boundary;
4. `internal_points` – a list of all vertices in the cluster that do not belong to the boundary;
5. `connections` – a final vector of connections that belong to the cluster after re-triangulation (a connection is itself a vector containing two vertices);
6. `bound_properly_selected` – a bool variable that is true when the boundary is properly selected (each vertex in the boundary only belong to two connections);
7. `retriangulated` – a bool variable that is true when the cluster is successfully re-triangulated.

Additionally, a cluster contains (among others) the following private functions:

1. `get_vert_in_cluster()` – generates a vertex list;
2. `define_boundary()` – selects the connections that belong to the its boundary;
3. `find_boundary()` – orders the boundary;
4. `call_retriangulation()` – re-triangulates the cluster ;

and a public function `retriangulate()`, that can be called from the main program. This public function calls all the other functions on the class, in order (see algorithm 1).

Algorithm 1 function `retriangulate( )`

```
call get_vert_in_cluster()
call define_boundary()
call find_boundary()
call call_retriangulation()
```

The first of these functions (`get_vert_in_cluster()`) is a straightforward function that goes through the list of simplices on the cluster (`simp_list`), look for the vertices contained in each simplex and include them in a hash set. The second, `define_boundary`, classifies every connection (`i,j`) as belonging or not to the boundary of the cluster. The procedure is explained in algorithm 2. The auxiliary lists `boundary` and `not_boundary` are use to store the connections.

Algorithm 2 Separating connections as belonging or not to the cluster's boundary

```
for all pairs (i,j) of vertices in the cluster do
  bool in_boundary = false
  if i and j are neighbors then
    for all k that is a neighbor of j do
      if k is also a neighbor of i then
        if k does not belong to the cluster then
          include the connection (i,j) in boundary
          in_boundary = true
        end if
      end if
    end for
    if !in_boundary then
      include the connection (i,j) in not_boundary
    end if
  end if
end for
```

Even though all connections in a cluster are, according to the routine above, separated as belonging or not to the cluster boundary, some connections might have been falsely selected as boundary connections (the reason for that is explained in section 4.3.2 of chapter 4).

The third function called by `retriangulate()`, `find_boundary()`, is significantly more complex than the two described above, since it is responsible to extract and order the cluster boundary, despite the mis-selected connections in `boundary`. Additionally, the routine checks whether the cluster has more than a boundary and, if this is the case, defines which one will be used as the main boundary of the cluster, and includes the vertices forming the other ones (and the vertices that do not belong to the cluster that are bounded by them) as internal points in the cluster.

This routine is responsible to call three other big functions (see algorithm 3). The first is called `order_boundaries(ordered_boundaries, boundary)`, and selects and orders all boundaries belonging to a cluster. These boundaries are then stored in the `ordered_boundaries` vector. Additionally, this routine updates `bound_properly_selected`, a bool variable belonging to the cluster that indicates whether its boundary was properly identified. The second and third routines are only called if this variable is returned `true`.

The second routine, `get_biggest_boundary(ordered_boundaries)`, selects the biggest boundary stored in `ordered_boundaries` and store it in the `ord_boundary` vector. This boundary is then deleted from `ordered_boundaries`. The last routine is called `include_left_bound_to_int_points(ordered_boundaries)`. Its input is `ordered_boundaries` (that now does not contain the biggest boundary of the cluster anymore), and its function is to include all other boundaries of the cluster (that are not the biggest one), as well as all connections and vertices surrounded by these boundaries (that do not belong to the cluster), as internal points and connections of the cluster.

Algorithm 3 Obtaining the ordered boundary of a cluster

```
    vector ordered_boundaries
    call order_boundaries(ordered_boundaries, boundary)
    if bound_properly_selected then
      for every pair (i,j) in boundary do
5:      is_in_final_boundary = false
        for every pair (k,l) of consecutive vertices on ordered_boundaries do
          if (i,j) = (k,l) then
            is_in_final_boundary = true
          end if
10:     end for
        if !is_in_final_boundary then
          delete (i,j) from boundary and include in not_boundary
        end if
      end for
15:   call get_biggest_boundary(ordered_boundaries)
      include all connections left in ordered_boundaries to not_boundary
    end if
```

From the three routines called, it will first be shown how the `order_boundaries(ordered_boundaries, boundary)` was implemented (algorithm 4). The function is recursive and it calls another routine (`call_get_boundary`) that is responsible to select one single boundary of a cluster. If there are some connections left unused in the boundary list, this means the cluster has more than a boundary. The routine to find a boundary will then be once more called, and the procedure repeats until no connection in the boundary list is left unused.

Algorithm 4 `order_boundaries(ordered_boundaries, boundary)`

```
vector obtained_boundary
vector copy
copy = boundary
bound_properly_selected = call_get_boundary(obtained_boundary, copy)
if !bound_properly_selected then
  print ‘‘ERROR! Boundary could not be properly selected’’
else
  include obtained_boundary in ordered_boundaries
  if copy not empty then
    call order_boundaries(ordered_boundaries, boundary_copy
  end if
end if
```

The routine `call_get_boundary` called in algorithm 4 receives a copy of the vector `boundary` and gives as output the vector `obtained_boundary`, that contains one of the boundaries of the cluster. Additionally, it returns a bool variable that is set to `true` if the boundaries are properly selected (see algorithm 5).

The routine works as follows: it initializes a vector called `possible_boundary`, needed by the `get_boundary` routine, the main routine that identifies a single boundary. Besides `possible_boundary`, `get_boundary` needs `copy` as input, and it writes its output in `final_boundary`. If the algorithm worked properly, `final_boundary` only contains one element (a vector of the encountered boundary). This vector is then copied to `obtained_boundary`, the vector that `call_get_boundary` gives as output.

The last routine to be outlined is `get_boundaries`. It is a recursive function that identifies every set of connections in the `boundary` vector that form a loop, and evaluate if this loop might be a boundary of the cluster (in the function `copy` is being used, a vector that is simply a copy of the `boundary` vector) – see algorithm 6. For that, it starts with one vertex that belongs to one of the boundaries in the cluster, sent from `call_get_boundary` in the vector `possible_boundary`.

The `possible_boundary` also assures that this vertex only belongs to two connections, so that any connection chosen to start the `get_boundaries` routine was not selected to belong to the boundary of the cluster erroneously (in this case, the vertex chosen would belong to at least three connections).

One of the two connections to which the initial chosen vertex belongs is then

Algorithm 5 `call_get_boundary(obtained_boundary, copy)`

```
vector possible_boundary
initialize possible_boundary with any vertex from copy that belongs to exactly
two connections
if possible_boundary is empty then
  print ‘‘ERROR, no vertex belongs to only two connections’’
  return false
else
  find a connection in copy that contains the first element of possible_boundary
  include the second vertex of the connection in possible_boundary
  delete the connection from copy
  vector< vector > final_boundary
  get_boundaries(possible_boundary, copy, final_boundary)
  if final_boundary size equals 1 then
    obtained_boundary = final_boundary[0]
    delete all connections used in final_boundary[0] from copy
    return true
  else
    print ‘‘ERROR ORDERING BOUNDARY’’
    return false
  end if
end if
```

chosen, and used to initialize a vector called `tmp_connections`. A loop is over the vertices in `tmp_connections` is then started. First, it is checked whether the current vertex in the loop is already an element of `possible_boundary` (it is not checked whether this vertex is equal to the first vertex in `possible_boundary`). If this is the case, a sequence of connections in the boundary that does not lead to a cluster boundary is found, and this instance of the algorithm does not need to be finished. Even though this is obviously not true at the first time the routine is called, this check makes sense in later recursive calls of the routine.

When the above check comes negative (the chosen vertex is not an element of `possible_boundary` – except maybe the first), a vector to hold a tentative boundary for the cluster is created (`my_possible_boundary`) and initialized with a copy of `possible_boundary`. Additionally, the current vertex in the loop over `tmp_connections` is also included in `my_possible_boundary`. This vector will be used as input to `get_boundaries` (in the place of `possible_boundary`) when the routine calls itself.

The condition for the routine not to call itself is that the first and last vertices in `my_possible_boundary` are equal. If this is the case, a sequence composed by connections from `boundary` that could be a boundary of the cluster was found. The found boundary will be included in the `final_boundary` vector, a vector that stores all sequences of connections in the `boundary` vector that could be a cluster boundary.

Two possibilities exist: first, `final_boundary` might still be empty. In this case, `my_possible_boundary` is included as the first element of `final_boundary`. Second, there is already an element in `final_boundary`. In this case, the sizes of the vector already stored and of the new vector are compared.

If the vector already stored is smaller than the new one, it is deleted from the `final_boundary` container, since if a bigger possible boundary started from the same initial point exists, it will be the right one. If, however, both vectors have the same size, the new found possible boundary is included in the `final_boundary` vector without the deletion of the previous one. Notice that, by construction, it is not possible for a later found possible boundary to be smaller than one already stored in `final_boundary`. When the condition for the routine not to call itself is not met, the routine will simply call itself once more.

If two (or more) possible boundaries are still on `final_boundary` when it is received back by `call_get_boundary`, some of the wrongly selected connections in `boundary` made to sequences of the same set of vertices possible to a boundary of the cluster. In this case, there is a degeneration and the algorithm, up to the moment, cannot define which configuration is the right one. Therefore, if this happens, the algorithm to identify the boundary of the membrane failed.

Algorithm 6 `get_boundaries(possible_boundary, copy, final_boundary)`

```
vector tmp_connections
find a connection in copy that contains the last vertex in possible_boundary
include the second vertex of the connection in tmp_connections
delete the connection from copy
for each vertex i in tmp_connections do
  bool repeated = false
  for each vertex j in possible_boundary except the last one do
    if possible_boundary[j] == tmp_connections[i] then
      repeated = true
    end if
  end for
  if repeated then
    continue
  end if
  vector my_possible_boundary
  my_possible_boundary = possible_boundary
  include tmp_connections[i] in my_possible_boundary
  if first and last elements of my_possible_boundary are equal then
    if final_boundary is empty then
      includes my_possible_boundary in final_boundary
    else if my_possible_boundary > last vector included in final_boundary
    then
      clear final_boundary
      include my_possible_boundary in final_boundary
    else if my_possible_boundary's size == final_boundary's last element size
    then
      include my_possible_boundary in final_boundary
    end if
  else
    call get_boundaries(my_possible_boundary, copy, final_boundary)
  end if
end for
```

Acknowledgments

This work would not have been done without the help and support of several other people. Therefore, I would like to thank:

- My parents and my family for, although living far away, being still so close to me.
- My husband, for being the sweetest person on Earth, and for being there for me every single time I needed. Without him I would never have considered calling Germany home.
- Prof. Dr. Michael Meyer-Hermann for the great supervision, allowing me to be independent but still being present when I needed.
- All integrants of the Systems Biology group in Frankfurt for a nice work environment and helpful discussions, in particular Harald Kempf.
- Prof. Dr. Horst Stöcker and Prof. Dr. Takeshi Kodama, for giving me the opportunity to come to study in Germany.
- The Vereinigung von Freunden und Förderern der Johann Wolfgang Goethe-Universität Frankfurt am Main e.V. for financial support.
- And, at last, many many friends, in Germany or not, for making my life outside of work more enjoyable.

List of Figures

Bibliography

[Alberts 2002] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts and Peter Walter. Molecular biology of the cell. Garland Science, fourth édition, 2002. 9, 10, 12

[Allen 2007] Christopher D. C. Allen, Takaharu Okada, H. Lucy Tang and Jason G. Cyster. *Imaging of Germinal Center Selection Events During Affinity Maturation.* Science, vol. 315, no. 5811, pages 528–531, 2007. 15, 17

[Alt 1995] W. Alt and R. T. Tranquillo. *Basic morphogenetic system modeling shape changes of migrating cells: how to explain fluctuating lamellipodial dynamics.* Journal of Biol. Systems, vol. 3, no. 4, pages 905–916, 1995. 2, 19

[Amenta 1998] Nina Amenta, Marshall Bern and Manolis Kamvysselis. *A new Voronoi-based surface reconstruction algorithm.* In SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pages 415–421, New York, NY, USA, 1998. ACM. 3, 34, 40

[Amenta 1999] N Amenta and M Bern. *Surface reconstruction by Voronoi filtering.* Discrete & Computational Geometry, vol. 22, no. 4, pages 481–504, December 1999. 3, 34, 40

[Beyer 2005] T Beyer, G Schaller, A Deutsch and M Meyer-Hermann. *Parallel dynamic and kinetic regular triangulation in three dimensions.* Computer Physics Communications, vol. 172, no. 2, pages 86–108, November 2005. 3, 25, 27, 31

[Beyer 2006] Tilo Beyer and Michael Meyer-Hermann. *The treatment of non-flippable configurations in three dimensional regular triangulations.* WSEAS Trans. Syst., vol. 5, no. 5, pages 1100–1107, 2006. 31

[Beyer 2007] Tilo Beyer. *Spatio-Temporal Dynamics of Primary Lymphoid Follicles During Organogenesis and Lymphneogenesis.* PhD thesis, Frankfurt Institute for Advanced Studies – Goethe University, 2007. 28

[Beyer 2008] Tilo Beyer and Michael Meyer-Hermann. *Mechanisms of organogenesis of primary lymphoid follicles.* Int. Immunol., vol. 20, no. 4, pages 615–623, 2008. 23, 25, 27

[Bock 2009] Martin Bock, Amit Kumar Tyagi, Jan-Ulrich Kreft and Wolfgang Alt. *Generalized Voronoi Tessellation as a Model of Two-dimensional Cell Tissue Dynamics*, 2009. 23, 27

[Canadas 2002] Patrick Canadas, Laurent Valerie M, Christian Oddou, Daniel Isabey and Sylvie Wendling. *A Cellular Tensegrity Model to Analyse the*

Structural Viscoelasticity of the Cytoskeleton. Journal of Theoretical Biology, vol. 218, no. 2, pages 155–173, 2002. 24, 25

[Cazals 2006] Frédéric Cazals and Joachim Giesen. *Delaunay Triangulation Based Surface Reconstruction.* In Jean-Daniel Boissonnat and Monique Teillaud, editeurs, Effective Computational Geometry for Curves and Surfaces, pages 231–276. Springer-Verlag, Mathematics and Visualization, 2006. 3, 30

[Crick 2003] Francis Crick and Christof Koch. *A framework for consciousness.* Nature Neuroscience, vol. 6, no. 2, pages 119–126, February 2003. 2

[Denk 1990] W Denk, JH Strickler and WW Webb. *Two-photon laser scanning fluorescence microscopy.* Science, vol. 248, no. 4951, pages 73–76, 1990. 2, 14, 15

[Ferrez 2001] J.-A. Ferrez. *Dynamic triangulations for efficient 3d simulations of granular materials.* PhD thesis, EPFL, 2001. 30

[Galle 2008] Jörg Galle, Martin Hoffmann and Gabriela Aust. *From single cells to tissue architecture – a bottom-up approach to modeling the spatio-temporal organisation of complex multi-cellular systems.* Journal of Mathematical Biology, vol. 58, no. 1, pages 261–283, 2008. 23, 27

[Glazier 1993] James A. Glazier and François Graner. *Simulation of the differential adhesion driven rearrangement of biological cells.* Phys. Rev. E, vol. 47, no. 3, pages 2128–2154, Mar 1993. 19, 21

[Graner 1992] François Graner and James A. Glazier. *Simulation of biological cell sorting using a two-dimensional extended Potts model.* Phys. Rev. Lett., vol. 69, no. 13, pages 2013–2016, Sep 1992. 19

[Hauser 2007] A.E. Hauser, T. Junt, T.R. Mempel, M.W. Sneddon, S.H. Kleinstein, S.E. Henrickson, U.H. von Andrian, M.J. Shlomchik and A.M. Haberman. *Definition of Germinal-Center B Cell Migration In Vivo Reveals Predominant Intrazonal Circulation Patterns.* Immunity, 2007. 17

[Helmchen 2005] Fritjof Helmchen and Winfried Denk. *Deep tissue two-photon microscopy.* Nature methods, vol. 2, no. 12, pages 932–940, 2005. 14, 15, 17

[Hsiang 1993] Wu-Yi Hsiang. *On the sphere packing problem and the proof of Kepler's conjecture.* Internat. J. Math., vol. 5, no. 4, page 739, 1993. 56

[Ingber 2003a] Donald E. Ingber. *Tensegrity I. Cell structure and hierarchical systems biology.* J Cell Sci, vol. 116, no. 7, pages 1157–1173, 2003. 24

[Ingber 2003b] Donald E. Ingber. *Tensegrity II. How structural networks influence cellular information processing networks.* J Cell Sci, vol. 116, no. 8, pages 1397–1408, 2003. 24

[Jones 1924] J. E. Jones. *On the Determination of Molecular Fields. II. From the Equation of State of a Gas.* Proceedings of the Royal Society of London., vol. 106, no. 738, pages 463–477, 1924. 32

[Lamé 1838] Gabriel Lamé. *Un polygone convexe étant donné, de combien de manières peut-on le partager en triangles au moyen de diagonales?* Journal de Mathématiques Pures et Appliquées, vol. 3, pages 505–507, 1838. 49

[Meineke 2001] F. A. Meineke, C. S. Potten and M. Loeffler. *Cell migration and organization in the intestinal crypt using a lattice-free model.* Cell proliferation, vol. 34, no. 4, pages 253–266, August 2001. 23, 25, 27

[Meyer-Hermann 2005] Michael E Meyer-Hermann and Philip K Maini. *Interpreting two-photon imaging data of lymphocyte motility.* Phys Rev E Stat Nonlin Soft Matter Phys, vol. 71, no. 6 Pt 1, page 061912, Jun 2005. 20, 22

[Meyer-Hermann 2008] Michael Meyer-Hermann. *Delaunay-Object-Dynamics: cell mechanics with a 3D kinetic and dynamic weighted Delaunay-triangulation.* Curr Top Dev Biol, vol. 81, pages 373–99, 2008. 3, 27, 31

[Miller 2002] M. J. Miller, S. H. Wei, I. Parker and M. D. Cahalan. *Two photon imaging of lymphocyte motility and antigen response in intact lymph node.* Science, vol. 296, pages 1869–1873, 2002. 2, 17

[Mücke 1998] Ernst P. Mücke. *A Robust Implementation for Three-Dimensional Delaunay Triangulations.* Int. J. Comput. Geometry Appl., vol. 8, no. 2, pages 255–276, 1998. 30

[Murray 2002] J. D. Murray. Mathematical biology i: An introduction, volume 17 of *Interdisciplinary Applied Mathematics.* Springer, third édition, 2002. 8

[Murray 2003] J. D. Murray. Mathematical biology ii: Spatial models and biomedical applications, volume 18 of *Interdisciplinary Applied Mathematics.* Springer New York, third édition, 2003. 8

[Newman 2005] Timothy J Newman. *Modeling multicellular systems using subcellular elements.* Mathematical Biosciences and Engineering, vol. 2, no. 3, pages 611–622, 2005. 23, 25

[Okabe 2000] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara and Sung Nok Chiu. Spatial tessellations: Concepts and applications of Voronoi diagrams. Probability and Statistics. Wiley, NYC, 2nd édition, 2000. 671 pages. 28, 30

[Sandersius 2008] Sebastian A Sandersius and Timothy J Newman. *Modeling cell rheology with the Subcellular Element Model.* Physical Biology, vol. 5, no. 1, page 015002 (13pp), 2008. 23, 25

[Schaller 2004] G Schaller and M Meyer-Hermann. *Kinetic and dynamic Delaunay tetrahedralizations in three dimensions.* Computer Physics Communications, vol. 162, no. 1, pages 9–23, September 2004. 3, 25, 27, 30, 31

[Schaller 2005] Gernot Schaller. *On selected numerical approaches to Cellular Tissue.* PhD thesis, Frankfurt Institute for Advanced Studies – Goethe University, 2005. 28

[Schubert 2003] Walter Schubert. *Topological proteomics, toponomics, MELK technology.* Adv. Biochem. Engin. Biotechnol., vol. 83, pages 189–209, 2003. 2, 15, 17

[Schubert 2006] W. Schubert, Bernd Bonnekoh, Ansgar J. Pommer, Lars Philipsen, Raik Böckelmann, Yanina Malykh, Harald Gollnick, Manuela Friedenberger, Marcus Bode and Andreas W. M. Dress. *Analyzing proteome topology and function by automated multidimensional fluorescence microscopy.* Nat. Biotechnol., vol. 24, pages 1270–1278, 2006. 2, 15, 17, 18

[Schwickert 2007] Tanja A. Schwickert, Randall L. Lindquist, Guy Shakhar, Geulah Livshits, Dimitris Skokos, Marie H. Kosco-Vilbois, Michael L. Dustin and Michel C. Nussenzweig. *In vivo imaging of germinal centres reveals a dynamic open structure.* Nature, vol. 446, no. 7131, pages 83–87, March 2007. 17

[Shewchuk 1997] J. R. Shewchuk. *Adaptive precision floating-point arithmetic and fast robust geometric predicates.* Discrete and Computational Geometry, vol. 18, no. 305–363, 1997. 30

[So 2001] Peter TC So. *Two-photon Fluorescence Light Microscopy.* Encyclopedia of life sciences, May 2001. 2, 14, 15

[Stoll 2002] S. Stoll, J. Delon, T. M. Brotz and R. N. Germain. *Dynamic Imaging of T Cell-Dendritic Cell Interactions in Lymph Nodes.* Science, vol. 296, pages 1873–1876, 2002. 17

[Verlet 1967] L. Verlet. *Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules.* Phys. Rev., vol. 159, no. 1, pages 98–103, 1967. 35

[Verlet 1968] L. Verlet. *Computer experiments on classical fluids. II. Equilibrium correlation functions.* Phys. Rev., vol. 165, no. 1, pages 201–214, 1968. 35

[von Andrian 2002] U. H. von Andrian. *T cell activation in six dimensions.* 296, pages 1815–1817, 2002. 17

[von Neumann 1966] J. von Neumann. Theory of self-reproducing automata. University of Illinois Press, 1966. 8

[Wolfram 1994] S. Wolfram. Cellular automata and complexity: Collected papers. Wolfram Research, 1994. 9