MINE

# Cloth Simulation for Computer Graphics

Tuur Stuyck

*SYNTHESIS LECTURES ON VISUAL COMPUTING: COMPUTER GRAPHICS, ANIMATION, COMPUTATIONAL PHOTOGRAPHY, AND IMAGING #32*

## ABSTRACT

Physics-based animation is commonplace in animated feature films and even special effects for live-action movies. Think about a recent movie and there will be some sort of special effects such as explosions or virtual worlds. Cloth simulation is no different and is ubiquitous because most virtual characters (hopefully!) wear some sort of clothing.

The focus of this book is physics-based cloth simulation. We start by providing background information and discuss a range of applications. This book provides explanations of multiple cloth simulation techniques. More specifically, we start with the most simple explicitly integrated mass-spring model and gradually work our way up to more complex and commonly used implicitly integrated continuum techniques in state-of-the-art implementations. We give an intuitive explanation of the techniques and give additional information on how to efficiently implement them on a computer.

This book discusses explicit and implicit integration schemes for cloth simulation modeled with mass-spring systems. In addition to this simple model, we explain the more advanced continuum-inspired cloth model introduced in the seminal work of Baraff and Witkin [1998]. This method is commonly used in industry.

We also explain recent work by Liu et al. [2013] that provides a technique to obtain fast simulations. In addition to these simulation approaches, we discuss how cloth simulations can be art directed for stylized animations based on the work of Wojtan et al. [2006]. Controllability is an essential component of a feature animation film production pipeline. We conclude by pointing the reader to more advanced techniques.



## KEYWORDS

physics-based simulation, cloth simulation, computer graphics, explicit integration, implicit integration, adjoint optimization

# Preface

This book has grown from a desire to make cloth simulation more accessible to people new to the field. It is the hope that this book serves as a good practical guide to bring you up to speed to allow you to implement your own cloth simulator and produce visually pleasing results.

The literature on cloth simulation is very vast and new work is published every year. The intention of this book is not to cover all the topics but rather that this tutorial will provide a solid understanding of the basics so that you will more easily understand technical papers that build upon these foundations.

Tuur Stuyck
July 2018

# CHAPTER 1

# Introduction

Making a feature-length computer-animated movie costs millions of U.S. Dollars and takes several years of planning, script writing, visual development, and eventually modeling and animation on a computer. The computer graphics community is pushed forward by solving the challenges artists are faced with during the development of new movies. One of these research areas is physics-based animation where engineers and researchers use physics and math to make beautiful animations of natural phenomena.

## 1.1 PHYSICS-BASED ANIMATION

The use of physically based simulations is ubiquitous in games and special effects for movies. As an animator, you really don't want to be tasked with animating water or cloth by hand since these materials need to follow strict physical laws in order to be plausible and believable to the audience. Every one of us knows how water is supposed to behave so it is very difficult and time consuming to recreate this by hand.

This is where simulation shines. We can model the real world in the computer and compute how the materials would behave under the influence of the environment. A few examples of simulations are fluid simulations that are used for modeling flowing rivers, explosions, and smoke. Other applications are the simulation of rigid body interactions such as the destruction and collapsing of a building. Also, soft body simulations such as flesh and muscle simulations are used for virtual surgery. Of course, there is also cloth simulation that will allow artists to obtain detailed and natural looking geometry that reacts to the movement of the character and wind forces.

Highly believable simulated motions are typically generated by numerical algorithms that evolve discrete mathematical models over time. The model describes how the material should move, taking into account the material properties, boundaries, external forces, and collision objects in the scene. In computer graphics, we are mostly concerned with the look of the final result and physical accuracy is by no means our main goal. This is in stark contrast to the engineering community. For their purposes, physical accuracy is a top priority in order to be able to run simulations that are helpful in modeling and predicting real-world-scenarios. Obviously, physical accuracy helps achieve these visually pleasing and physically plausible goals for applications in computer graphics.

## 1.2 APPLICATIONS OF CLOTH SIMULATION

Before we overload you with mathematical expressions and derivations, let's get you excited by talking about common applications of cloth simulation. The applications can typically be categorized in one of the following two categories.

- **Offline simulations** are computed, tweaked, and post processed before being rendered on screen. The artist has time to run multiple simulations with different settings in order to find the desired results. These methods typically target high believability and controllability.
- **Real-time simulations** involve computing the simulation dynamics at runtime. This will allow the simulation to interactively react to user input and changes in the virtual environment. This type of simulations have very limited computation time available to them and are commonly implemented on GPU hardware. Real-time simulation algorithms are required to be fast and stable.

Specific examples of both categories are given in the following subsections.

### 1.2.1 OFFLINE SIMULATIONS

The most obvious applications are the use in special effects, digital doubles, computer animation, and virtual prototyping. The special effects industry has advanced so much that, instead of hiring a stunt double, it is sometimes easier to just digitally recreate the actor. This requires that we can also accurately model their clothing and the way the cloth behaves. That way, a smooth transition can be made from the real actor to the digital double, leaving the viewer none the wiser on how they performed the actual stunt. Spoiler: it's all computers and the amazing craftsmanship and dedication of animators and technical directors.

Computer animation is very similar to special effects, although the focus is often a little different. Special effects want to stay close to reality in order to truthfully recreate actors. Computer animation often involves virtual characters created by the director and their highly talented development team. Their focus is artistic expression. Directors are very concerned with being able to convey a very stylized style in order to tell the story the best way they can. The focus in computer animation is thus mostly controllability and art directability.

As a last example, fashion designers can use virtual cloth models to find the right 2D patterns that make up garments. A computer implementation of the cloth dynamics allows them to quickly iterate on designs and visualize how the garment will drape and where folds and wrinkles will be created naturally due to the material and sewing patterns. Virtual prototyping allows them to save on material and fabrication costs and accelerates the design process.

# CHAPTER 2

# Cloth Representation

**"All beginnings are difficult."**

While this popular saying might be true, we'll try to present you with an understandable explanation.

---

So you want to learn more about physically based simulation of cloth? That's great! Let's start with the basics. We are assuming you know a little bit about computer graphics already so this chapter will be a very quick overview.

## 2.1 TRIANGLES

One way to represent geometry on a computer is by using triangles. A single triangle is pretty boring but by combining many triangles into a triangle mesh we have the capability to create astonishingly complex geometrical shapes. Just think about all the special effects you see in movies and video games these days. Almost indistinguishable from real life, except, in real life, things don't blow up as easily as in the movies.

An example of a virtual garment is shown in Figure 2.1. The dress is made up of numerous small triangles. The right figure shows the wireframe of these triangles. The garment has a natural drape over the body of the virtual character thanks to physically based cloth simulation. The garment reacts to external forces such as gravity or wind and moves with the character due to collisions with the body.

A triangle is made up of three *vertices* or *particles*, connected by edges. These terms can be used interchangeably in most settings. Have a look at Figure 2.2 to see what this looks like, particles are shown in red and the triangle is shown in grey. Having a gorgeous 3D model is pretty neat, but you know what's even neater. Having it move! That's what physics-based simulation is all about.

Cloth is a continuous material but in what follows we will work with a discrete particle representation, this will become more clear later in the tutorial. For now, just blindly trust us.

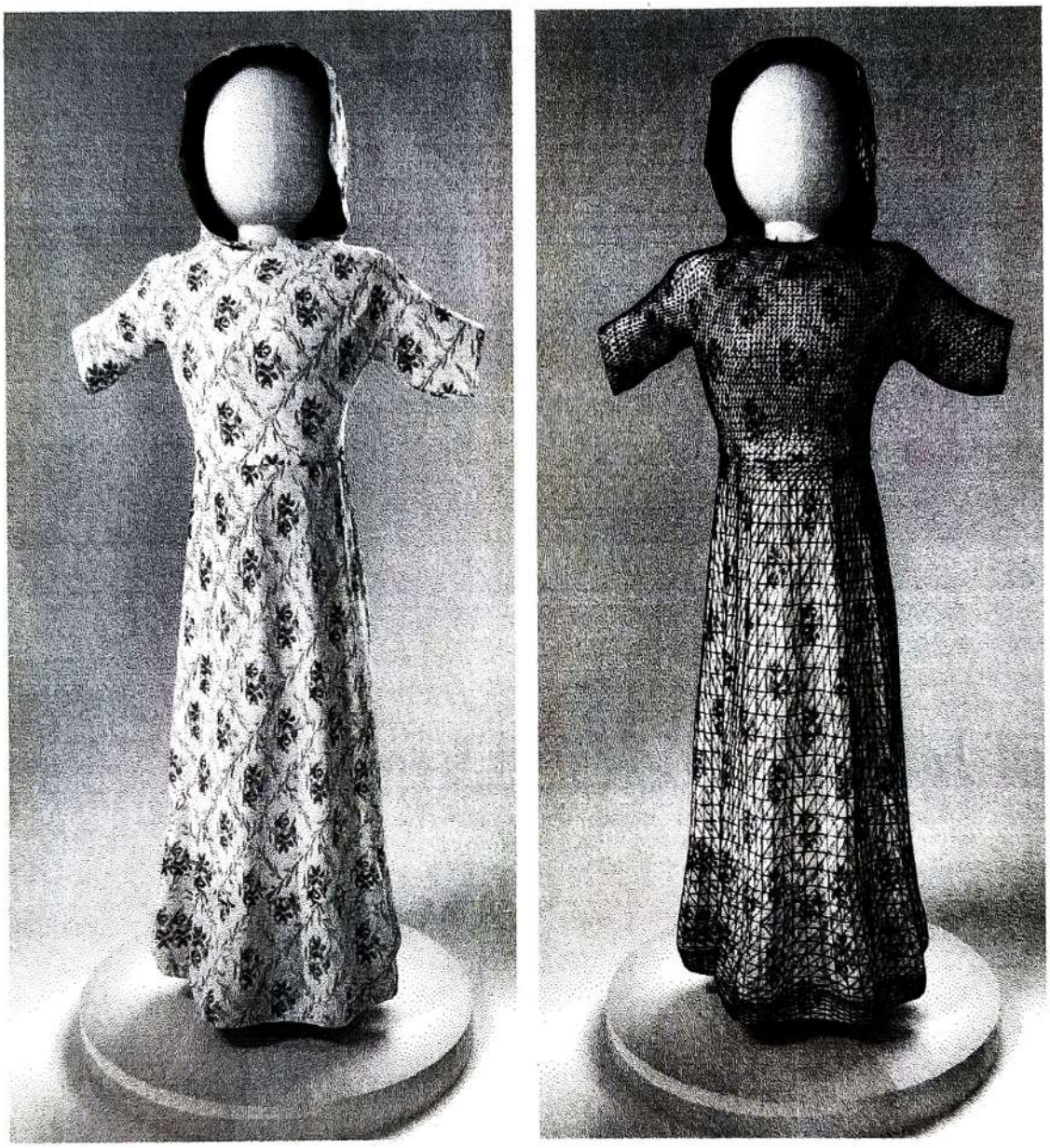

Figure 2.1: The dress has a natural drape on the body of the character thanks to physically based cloth simulation. The garment is made up of multiple triangles which are shown using a black wireframe overlay in the right image. The cloth model and textures were obtained from user *mnphmnmn* on turbosquid: `https://www.turbosquid.com/Search/Artists/mnphmnmn`.

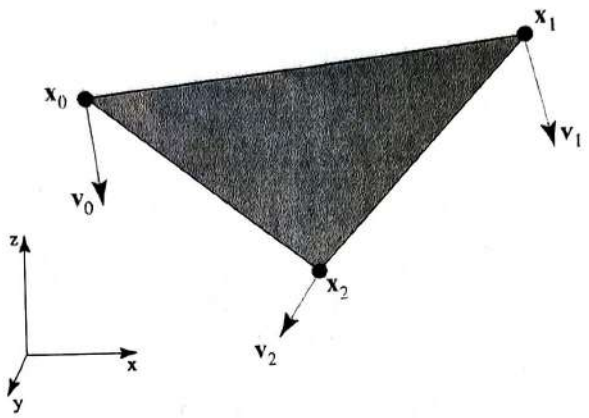


Figure 2.2: A triangle is made up of three particles or vertices with positions $\mathbf{x}_i$ shown in red and velocities $\mathbf{v}_i$ shown in blue.

## 2.2 PARTICLES

A particle $i$ is defined by a 3D position $\mathbf{x}_i \in \mathbb{R}^3$ and velocity $\mathbf{v}_i \in \mathbb{R}^3$. The combination of particle position and velocity is also referred to as the *particle state* $\mathbf{q}_i = (\mathbf{x}_i, \mathbf{v}_i)$. The positions and velocities of the particles will change over time because they have to obey the physical laws that describe the material properties. For cloth, this means that it won't stretch too much but it might shear and bend, creating folds pretty easily. As you know, a wool sweater behaves differently from a linen shirt. This is described by the material model.

We can group all the positions and velocities of the entire particle system with $N$ particles with positions and velocities $\mathbf{x}_i, \mathbf{v}_i \in \mathbb{R}^3$ in a single long vector $\mathbf{x} \in \mathbb{R}^{3N}$ and $\mathbf{v} \in \mathbb{R}^{3N}$

$$\mathbf{x} = \begin{bmatrix} x_{0_x} \\ x_{0_y} \\ x_{0_z} \\ \vdots \\ x_{N-1_x} \\ x_{N-1_y} \\ x_{N-1_z} \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \\ \vdots \\ v_{N-1_x} \\ v_{N-1_y} \\ v_{N-1_z} \end{bmatrix}. \tag{2.1}$$

We can store the triangle meshes using a few different data structures. The most straightforward way is to store the particle positions and velocities in separate arrays that can be indexed using the unique and unchanging particle index $i$. Triangles are then represented using a list

of three integer particle indices per triangle. This is a very simple approach and more complex techniques that encode triangle connectivity can be found in Botsch et al. [2010].

## 2.3 FORCES

The cloth is affected by external forces such as gravity, wind, or collisions with a body. But, what really makes the cloth behave like it should are the internal forces. These are the stretch, shear, and bend forces that act on the particles to make it behave like textile. Throughout this document we will look at different ways to formulate these internal forces acting on the vertices of the triangles. Once we have these forces we can use numerical integration to advance the simulation over time. That brings us to another important point. We'll be working on a computer and that means we don't have infinite resources available. A typical way to compute these simulations is to only compute the particle states at discrete time steps. Starting from some time $t_0$, the simulation will advance with small steps of duration $h$ to continue to time $t_0 + h$, $t_0 + 2h$, and onward!

### 2.3.1 FRAMES AND STEPS

In computer graphics and video, in general the illusion of continuous motion is created by showing the viewer many images per second. The number of images per second is called *frames per second* or *frame rate*. Commonly used frame rates are 24, 30, or even 60 frames per second. This is the number of images we have to create to obtain 1 second of video. However, as we'll see later in this book. We might have to take multiple simulation steps per frame to obtain stable results. The number of simulation steps can thus be much higher than the frame rate. We have to compute the motion of the cloth for every step but we only have to create an image to show the viewer for steps that coincide with frames.

To summarize, we have two types of discretizations in our computer model for cloth:

- **Discretization in space.** The continuous cloth is represented by a finite number of triangles that are made up by particles with positions and velocities.
- **Discretization in time.** Continuous time will be divided into discrete time steps of duration $h$.

# CHAPTER 9

# Collision Detection and Response

"OH YEAH!!!!!"

*Kool-Aid Man*

He can often be found answering the call of children by running through walls and furnishings.

## 9.1 INTRODUCTION

You might have noticed but we barely mentioned collision detection and response at all throughout this text. It is an important but tricky aspect of cloth simulation that it is often the bottleneck in modern visual effects. Collision handling is split into two phases: the **collision detection phase** and the **collision response phase**. First, we try to find colliding triangles or particles. Once we know the culprits we will try to resolve the collisions using an appropriate collision response. Two different types of collisions can be discerned.

- **Cloth-cloth** collisions happen when the garment collides with itself or another garment. For example, when wearing a shirt and sweater, you wouldn't want the shirt to pass through the sweater.
- **Object-cloth** collisions happen when the cloth collides with other objects. Think about how a shirt is constantly colliding with the body.

Collision detection is probably the easier task of the two to get working correctly. Many geometric tests exist to figure out whether triangles or particles are colliding and it is a matter of implementing these correctly. However, a lot of computation time will be spend on detecting collisions.

Collision response can be hard to get right. Applying changes to fix the collision could easily create odd-looking artifacts in the simulation. Additionally, resolving one collision can create new collisions which in turn need to get resolved as well. There's no guarantee that the

algorithm ever converges! Obviously, this can be problematic and there exist techniques to deal with this situation which will be discussed later in this chapter. Another difficult situation is when cloth slides over itself or another garment. The cloth is in contact with itself so it is colliding but we do not want to restrict it too much since this would create snagging artifacts.

## 9.2 COLLISION DETECTION

The collision detection phase can be split into a broad phase, a mid phase, and a narrow phase. We recommend the excellent book by Ericson [2004] and the freely available chapter on collision detection in Akenine-Möller et al. [2018]. The broad phase is first performed and is meant to quickly discard object pairs that are definitely not colliding. As a second step, the mid phase will look at the overlapping primitives between object pairs. As a final step, the narrow phase then takes a closer look at primitive pairs that could potentially be close to each other. We give a quick overview.

- The **broad phase** works by looking at objects that overlap. The workhorse for the broad phase is the sweep and prune algorithm [Akenine-Möller et al., 2018]. However, for cloth simulations we typically assume a limited number of meshes in the scene and we can safely skip the broad phase and start with the mid phase.

- The **mid phase** works on pairs of objects to find primitives in the object that overlap. The phase makes use of spatial acceleration structures that can quickly prune particle pairs that definitely won't collide. This significantly reduces the number of expensive collision tests that need to be performed in the narrow phase. Possible acceleration data structures are bounding volume hierarchies, acceleration grids, or k-d trees.

- In the **narrow phase**, the remaining potential cloth-cloth intersections can then be computed using particle-triangle and edge-edge collision tests. The cloth-solid intersections are computed by checking the cloth particles with respect to the faces of the solid object.

When using axis-aligned bounding boxes for the triangles in the acceleration structures, we typically enlarge the bounding box by the thickness of the cloth, e.g., $10^{-3}$m. Of course, as particles and triangles move around in the simulation the acceleration structure has to be updated in every iteration.

Collision detection algorithms can be classified by when they look for collisions. To be more precise, we have the following.

- **Discrete time** collision detection will look for all particles that are in close proximity at the beginning of the time step and will add constraints or penalty forces to particles that appear to be colliding. This will hopefully prevent the collision but there are no guarantees and this is why algorithms need failure modes to recover from collision. These methods are also known as *a posteriori*. To reliably resolve the collision continuous time algorithms will be needed.

closer look – пристальный взгляд

e.g. — например, ⇒

hopefully – надеюсь (тут, вполне)

- In contrast, **continuous time** algorithms look at the particle trajectories and will look for the instant in time that the first collision happens. The simulation will then be advanced until this time of first collision at which the collision can be resolved. This is also known as *a priori*. This is performed in an iterative fashion until all collisions are resolved.

### 9.2.1 BOUNDING VOLUME HIERARCHIES

Detecting a collision between two complex geometries can be very expensive, especially with ever increasing complexity of geometry. Geometry meshes with tens of thousand of triangles are pretty common nowadays. It would be extremely inefficient to simply compare every particle on the mesh with every other particle in the scene. The Bounding Volume Hierarchy (BVH) is one particular acceleration structure that can be used in the broad phase to get better efficiency and time complexity. In this text, we will focus on the BVH. For completeness, other algorithms that are frequently used for efficient pruning are uniform grids, hierarchical grids, binary space partitioning, and k-d trees.

To accelerate collision detection, complex geometries are often contained in a surrounding bounding volume. This volume can be a box, sphere, cylinder, or any other primitive that's cheap to test for intersections. Only when there's an intersection with the bounding volume do we need to intersect with the primitives inside. Another way of saying this is that only when the bounding volumes overlap could there potentially be an intersection and further investigation in the narrow phase is needed. This will save a lot of computation since large pieces of geometry can quickly be pruned since there won't be any intersections with the surrounding volume.

At the lowest level, triangles are embedded in a bounding volume. If we perform an intersection test of a single point with all the triangles then the complexity would scale linearly with the number of triangles $n$ in the mesh. This gives the following time complexity $\mathcal{O}(n)$ using big O notation, where $n$ is the number of triangles. We can do better than this. By grouping bounding volumes in a hierarchy; simply put, by constructing bigger bounding volumes containing many smaller volumes. The hierarchical bounding volume will have a way better time complexity of $\mathcal{O}(\log n)$.

The most commonly used bounding volume is the Axis-Aligned Bounding Box, often abbreviated as AABB. The box is aligned with the $xyz$-axis of the world coordinate system and it is just big enough to fully contain the geometry inside. It has the following desired properties that we are always looking for in a bounding volume.

1. It is cheap to test for intersection with the bounding volume.
2. It tightly fits around the geometry inside.
3. It is inexpensive to compute.
4. It is easy to transform.
5. It makes efficient use of computer memory.