

学位論文

Doctoral Dissertation

**A Study on Simulations of Granular Materials and
Strands Taking Wetting Effects into Account**

(吸湿現象を考慮した粒状および
線状物体のシミュレーションに関する研究)

ウィッタワット ルンチラタナーノン

Abstract

There is considerable recent progress in physically-based simulations, driven by the high demands in computer animated movies. Realistic animations of rigid bodies, fluid, deformable objects and couplings among different objects became possible. However, simulating wetting effects between solid objects and water is still relatively in its infancy despite the fact that it is a common phenomenon in the real world. The main cause is the complexity of physical mechanism of the wetting effects. The wetting effects are interesting effects in our daily life and could greatly enrich the realism of games and animations.

An object that has water absorbency is called a porous medium. An internal structure of the porous medium consists of a solid part of the medium mixing with a vast number of pores. These pores result in the capillary action where water is absorbed into and flows through the medium. Pores inside porous media such as cloth and sponge have their locations fixed in the media, while pores in porous media such as granular material and hair are dynamically and temporally arise from complex contact regions among the media. This dissertation categorizes the former porous structure as a rigid porous structure and the latter as a deformable porous structure. Recently, novel approaches for wetting effects in the rigid porous structure have been proposed in the computer graphics field. However, only a few studies have been introduced for the deformable porous structures. Accordingly, this dissertation targets wetting effects in the deformable porous structures, especially, wetting effects in sands and hair strands which are the objects in our daily life and important in CG.

When granular material comes into contact with water, water is absorbed into pores among granular particles and diffused from particle to particle throughout the material by the influence of capillary action. Then, wet granular particles stick together due to liquid bridges that are formed between particles. The liquid bridges subsequently disappear when there is too much water penetrates into the material. The similar physical mechanism occurs in the case of hair as well. However, the wetting effects of hair are more complicated. There are three main differences from the granular material. First, a single hair strand is also a kind of porous medium. Water is absorbed not only into small contact regions among hair strands, but also into the inner layer of each hair strand. When water percolates into the inner layer of hair strand, a chemical reaction inside the strand causes the shape of the hair strand to temporally change. Second, the contact regions among hair strands are costly to examine, since a hair strand is a long cylindrical deformable object. Third, hair is an anisotropic porous medium, as water diffuses more in the hair strand direction.

To develop models for the wetting effects, the underlying simulation models for granular material and hair strands are also important. While well-studied particle systems can be used for simulating fine-scale of granular material, simulation of hair is still a challenging problem in computer graphics. Traditional simulation techniques handle hair as clumps or continuum for efficiency; however, the visual quality is limited because they cannot represent the fine-scale motion of individual hair strands. Although a recent mass-spring approach tackled the problem of simulating the dynamics of every strand of hair, it suffered from high computational cost and required a complicated setting of springs. The morphological shape transformation of wet hair requires a model that can easily modify the shape of hair, thus the recent models are not suitable. In this dissertation, the hair simulation model on such a fine-scale is built up from a novel single strand simulation model. Strand-like objects in our daily lives have a wide variety of materials, e.g., extensible and inextensible strands, rubber, threads, plastic. Such strand-like objects also exhibit interesting behaviors such as bending, twisting, tearing (by stretching or twisting), and bouncing back when pulled and released.

This dissertation presents a strand simulation model that enables these behaviors, handles wide variety of materials, and is suitable for building a hair simulation model for wetting effects. Specifically, this dissertation offers the following four contributions. First, this dissertation introduces a strand simulation model that based on Lattice Shape Matching (LSM), which has been successfully used for simulating deformable objects. Each strand is represented as a chain of particles, and its deformation are handled by geometrically-derived forces of the chain based on shape matching. The shape matching can simulate a stiff strand in a numerically stable way. This benefits in handling stiff hairstyles such as curly hair and afro. Second, this dissertation introduces a method for handling twisting effects with both uniform and non-uniform torsional rigidities. Third, this dissertation presents a method for estimating the tension acting on inextensible strands in order to reproduce tearing and flicking (bouncing back), whereas the tension for an extensible object can be computed via stretched length. The length of an inextensible object is maintained constant in general, and thus, a novel approach is needed. Fourth, this dissertation introduces an optimized grid-based collision detection for accelerating the computation. In the simulation of a large number of hair strands, this dissertation develops a GPU-based simulator which achieves visually-plausible animations consisting of several tens of thousands of hair strands at interactive rates.

For the wetting effects in granular material, this dissertation introduces a wetness value for each granular particle and integrates wet behaviors dependent on

the wetness value into a simple particle-based framework. Using this method, a GPU-based simulator can achieve dynamic animations of granular material including wetting effects at interactive rates. For the wetting effects of hair, this dissertation introduces a simulation model that reproduces interactions between water and hair as a dynamic anisotropic porous medium. An Eulerian approach is utilized for capturing the complex deformable porous structure of hair and the wetting effects are efficiently handled using a Cartesian bounding grid. The proposed model and simulation generate many interesting effects of interactions between fine-detailed dynamic hair and water, i.e., water absorption and diffusion, cohesion of wet hair strands, water flow within the hair volume, water dripping from the wet hair strands and morphological shape transformations of wet hair.

要旨

映画などの映像制作分野において、より高品質なコンピュータアニメーションが求められていることを背景に、コンピュータグラフィクス(CG)分野において物理ベースシミュレーションの研究は近年著しく進歩している。その結果、剛体、流体、柔軟物体などの写実的なアニメーションが可能となってきた。しかしながら、実世界では一般的な現象であるにもかかわらず、流体と物体の連成シミュレーションにおいて物体の吸湿現象はその複雑さからこれまであまり扱われてこなかった。吸湿現象は日常的に見受けられるため、吸湿現象の再現は現在のゲームや映画における表現を高めるために重要な研究テーマである。

吸湿性のある物体は多孔質媒体と呼ばれる。多孔質媒体は、スポンジなどのように、内部に無数の小さな空孔を持つ。この空孔により、水分が媒体中で保持され、その中を伝播するといった毛細管現象が起こる。多孔質媒体はそれらの多孔質構造により、固定多孔質媒体と変形多孔質媒体に分類できる。布やスポンジなどは固定多孔質媒体に分類され、空孔の位置は媒体中で固定されている。一方、本研究で対象とする粒状および線状物体は変形多孔質媒体に分類され、水分が保持されるのは物体の接触領域であり、その位置や保持できる水分量が動的に変化する。近年、固定多孔質媒体と流体のシミュレーションモデルが提案されたが、変形多孔質媒体を扱うモデルはほとんどなかった。本研究では、変形多孔質媒体として粒状物体および線状物体における、吸湿現象のシミュレーションを対象とする。なお、吸湿現象を考慮した粒状物体と線状物体の具体例として、日常的に見受けられ、かつCG分野で重要な例である砂と髪の毛を扱う。

粒状物体と線状物体における吸湿現象の物理的機構について簡単に説明する。まず、粒状物体では、水が粒状物体と接触することにより、水が粒状物体の空孔に吸収され、粒状物体は水気を帯びる。水気を帯びた粒状物体は、流体の架橋 (*liquid bridge*) によって互いに吸着するようになる。この流体の架橋は水の表面張力によるもので、水分量が過度になると失われる。こういった吸湿現象は髪の毛の場合において、同様に起る。しかし、髪の毛の吸湿現象はより複雑であり、粒状物体と比べて三つの主な違いがある。(1) 一本の髪の毛はさらに一種の多孔質媒体である。水は、髪の毛の間の小さな接触領域だけでなく、各髪の毛の内層にも吸収される。水が髪の毛の内層にしみ込むと、髪の毛の内層の化学反応により、髪の毛の形が一時的に変形する。(2) 髪の毛

毛は長い円筒状の変形物体であり、髪の中の毛の間の接触領域を調べるコストが高い。(3) 水は髪の中の毛の接線方向に沿ってより多く伝播するため、髪は異方性の多孔質媒体である。

吸湿現象を考慮した粒状物体と髪の中の毛のシミュレーションを実現するために、吸湿現象をモデル化するだけでなく、粒状物体と髪の中の毛の適切なシミュレーションモデルも必要である。粒状物体については、よく研究された従来の粒子ベースのシミュレーションモデルが用いられる。一方、髪の中の毛については、人の頭部にある髪の中の毛(通常10万本以上)の動きをシミュレーションすることは、コンピュータグラフィクスの分野において長く挑戦的な課題であった。近年の手法の多くは、計算の効率化のため、髪の中の毛を粗い束か連続体として扱っており、髪の中の毛の動きの自由度が制限されてしまう。最近のパネモデルの研究は髪の中の毛の一本ずつの動きをシミュレートする手法を提案した。しかし、計算コストが高く、パネの構造が複雑である。濡れた髪の中の毛の形状変化のために、変形を効率的かつ安定的に扱える髪の中の毛モデルを必要とし、従来法は適切ではない。本研究では、詳細な髪の中の毛のシミュレーションモデルは単一次元変形物体のシミュレーションモデルから組み立てる。我々の日常生活での1次元変形物体の例はケーブル、プラスチック糸、ゴム糸などの様々な物体がある。本研究はこれらの1次元変形物体をストランドと呼ぶ。ストランドは、伸び縮み、ねじれ、断裂、および伸長時間から急激に収縮するといった性質も持っている。

本研究は、ストランドのそれらの効果を実現し、様々なストランドを扱い、ストランドのシミュレーション手法を提案する。具体的には、本研究は次の4つを提案する。一番目に、**Lattice Shape Matching (LSM)**に基づき、ストランドのシミュレーション手法を提案する。**LSM** は変形物体のシミュレーションにおいて成功を収めてきた。提案法は**LSM**を単純化し、1本のストランドを粒子の鎖(チェーン)で表現し、そのチェーンの変形を形状一致法に基づいた幾何的な力を実現する。二番目に、ストランドの均一・不均一なねじり剛性を考慮したねじれる効果を扱う手法を提案する。三番目に、ストランドの断裂と急激な収縮を実現する。断裂と急激な収縮はチェーンのひずみの値(伸びた長さ)から計算できるが、一般の伸びないストランドは物体が伸びすぎないように長さを制約してしまう。つまり、伸びた長さに基づいてチェーンの断裂や急激な収縮を再現することが難しくなる。提案法では、伸びないストランドの張力を推定し、断裂と急激な収縮を実現する。四番目に、衝突処理を高速化するために、最適化したグリッドベース衝突検出を提案する。また、本研究ではGPUを使用したシミュレータを開発し、数万本の髪の中の毛のアニメーションを対話的な速度で生成することがで

きる。

提案法では、粒状物体や線状物体の吸湿現象を扱うため、下記の手法を導入する。粒状物体の吸湿現象を実現するために、提案法では粒状物体の各粒子に対して、水分量の値を導入し、その値に応じた粒状物体の挙動を、粒子ベースシミュレーションのフレームワークに統合する。また、提案手法はGPUを用い、吸湿現象を考慮したダイナミックなアニメーションを対話的な速度で生成することができる。髪の毛の吸湿現象について、本研究は、髪を異方性の変形多孔質媒体として、水との相互作用を再現するシミュレーション手法を提案する。髪を格子化することにより、髪の複雑な変形多孔質構造を扱う。直交座標系のバウンディング・グリッドを使用し、効率的に吸湿現象を実現する。提案法は、詳細な髪の毛と水の相互作用の多様な興味深い効果を再現する。特に本研究では、髪の毛の吸水、水分の伝播、髪の毛が束になる様子、水がしたたり落ちる様子、濡れた髪の毛の形状変化を扱う。

Acknowledgements

First of all, I would like to express my gratitude to my advisor, Professor Tomoyuki Nishita, for his invaluable support, continuous encouragement and guidance. I am also grateful to Assistant Professor Yoshihiro Kanamori (University of Tsukuba). His helpful discussions and tutoring greatly helped me to complete my work successfully. I also would like to thank all the members of Nishita laboratory for their valuable advice and opinions, especially Zoltan Szego and Dr. Yonghao Yue for his great support.

As ever, I would like to thank for love and support from my family and friends. During the past years as an international student, my life would be tough without them. I would like to thank Panasonic Scholarship for giving me the opportunity to attend the University of Tokyo and exposing me to many new experiences. Finally, I would like to thank Japan Society for the Promotion of Science (JSPS) for a PhD research grant. The work presented in this dissertation was partly supported by Grant-in-Aid for JSPS Fellows (22-4748).

Contents

1	Introduction	1
1.1	Strands Simulation	3
1.2	Wetting Effects in Granular Materials	5
1.3	Wetting Effects in Strands	8
1.4	Organization	9
2	Related Work	10
2.1	Fluid Simulation	10
2.2	Granular Materials Simulation	11
2.3	Strands Simulation	11
2.4	Wetting Effects in Granular Materials	14
2.5	Wetting Effects in Strands	15
3	Fundamental Methods	17
3.1	Smoothed Particle Hydrodynamics (SPH)	17
3.2	Discrete Element Method (DEM)	18
3.3	Lattice Shape Matching (LSM)	20
4	Strands Simulation	22
4.1	Chain Shape Matching (CSM)	23
4.2	Strain Limiting	25
4.3	Twisting Effects	27
4.4	Tearing and Flicking Effects	30
4.4.1	Stress and Strain	30
4.4.2	Tension Estimation	32
4.4.3	Tearing and Flicking Effects	33
4.5	Collision Handling	34

4.6	GPU Implementation	36
4.7	Results	39
4.7.1	Simulation Results	39
4.7.2	Parameters Setting	41
4.7.3	Limitations	43
4.8	Summary	44
5	Wetting Effects in Granular Materials	56
5.1	Overview	56
5.2	Wetting Model for a Granular Particle	59
5.3	Interactions between Fluid and Granular Particles	61
5.4	Interactions among Granular Particles	61
5.5	Control of Propagation Speed	62
5.6	GPU Implementation	63
5.7	Results	66
5.7.1	Simulation Results	66
5.7.2	Parameters Setting	68
5.7.3	Limitations	70
5.8	Summary	71
6	Wetting Effects in Strands	85
6.1	Overview	85
6.2	Grid Construction	88
6.3	Water Propagation	91
6.3.1	Voxel Absorption	91
6.3.2	Macroscopic Propagation	92
6.3.3	Microscopic Propagation	93
6.3.4	Water Dripping	94
6.4	Cohesion of Wet Hair	95
6.5	Morphological Shape Transformation	96
6.6	Results	97
6.6.1	Simulation Results	97
6.6.2	Discussions and Limitations	100
6.7	Summary	101
7	Conclusion and Future Work	104

Chapter 1

Introduction

Recent advances in physically-based simulation have made it possible to generate realistic animations of rigid bodies, deformable objects and fluid. However, in the case of solid-fluid coupling, wetting effects have rarely been noticed despite their visual importance and contribution to realism. Most objects in the real world are permeable; when the objects come into contact with water, the water absorption and diffusion happen. There is also changes in physical properties of the wet objects, e.g., stickiness of wet sands, self-assemble of wet hair, weakening of wet paper and sponges.

The mechanism behind the wetting effect is mainly caused by the capillary action. The capillary action is the result of intermolecular attraction within the fluid and between solid and fluid which are surface tension and adhesion, respectively. To demonstrate this phenomenon, a capillary tube, a very thin glass tube with an internal bore, is commonly used. If we place one end of the tube into water, the water will permeate into the bore. When withdrawing the tube from the water, some amount of the water is trapped in the tube. This phenomenon can be observed on such other small open spaces among solid material, such as small gaps among a pile of sand and small open spaces inside a sponge. The object with such small open spaces is called a *porous medium*, and the small open space is called a *pore*.

The porous medium can be categorized into four fundamental classes accord-

ing to its dimensions in space, i.e., zero, one, two and three dimensional porous media (Figure 1.1). Although, the zero dimensional object is generally a point with no volume, this dissertation defines the zero dimensional object as a particle with a small volume such as a granule of sand. The pore structures in these four classes of porous media are different. The pores in zero dimensional porous medium reside in a vast amount of small gaps between individual granules. Likewise, the pores in one dimensional porous medium, such as hair strands, reside in complex contact regions among the medium. Since the pores are formed up from moving objects, the pore structures in zero and one dimensional porous media are highly deformable. Only a small deposition of a granule or a hair strand can drastically change the pore structures. In contrast, the pores structures in two and three dimensional porous media, such as cloth and sponge, are more rigid. The media themselves may deform, but the locations of pores are static in these media.

Several researches in computer graphics field have proposed methods that targeted only the two and three dimensional porous media, e.g., wet cloth [37, 60] and wet sponge [46]. The models for two and three dimensional porous media are not suitable for zero and one dimensional porous media whose porous structures are highly deformable as mentioned above. Therefore, this dissertation focuses on the zero and one dimensional porous media. The general examples of zero and one dimensional porous media are granular material (sand) and human hair, respectively. Therefore, for concise expressions, the zero-dimensional porous medium is shortly called as *granular material* and the one-dimensional porous medium is shortly called as *strands* in this dissertation.

The ultimate goal of this dissertation is to create simulation models for granular material and strands taking wetting effects into account. In order to achieve this goal, two main studies are required; (1) Simulation models for both granular material and strands and (2) Wetting effects handling models of those two kinds of porous media. As granular material consists of many small particles, it is suitably to be modeled as a particles system. Particle-based simulation has a long history



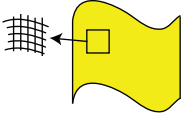
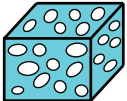
Porous medium	Zero-dimensional (Granular material)	One-dimensional (Hair, String)	Two-dimensional (Cloth, Paper)	Three-dimensional (Sponge, Brick)
Pores	 Gaps among granules	 Contact region among material	 Gaps among fiber network	 Air pockets among material
Porous structure	Highly deformable porous structure		Rigid porous structure	

Figure 1.1: Permeable materials and their porous structures.

in computer graphics, and is well-studied. The particle-based simulation system benefits the small-scale interactions between water and sand, thus highly dynamics animations with wetting effects could be simulated. For a small rigid object such as a grain of sand, the method called *Discrete Element Method* (DEM) has been developed and used in various works. The DEM is the choice for basis simulation model of granular material in this dissertation. For fluid, the well-known *Smoothed Particle Hydrodynamics* (SPH) is used. On the other hand, the simulation of strands is a challenging problem, especially simulating hair in fine-scale as individual strands. The problem further challenges in wet hair, since wet hair temporally change shape when wet. A model that can dynamically transform shape of hair is also necessary.

In this chapter, the background of simulation models for strands, wet granular material and wet strands are described following with an organization of this dissertation.

1.1 Strands Simulation

Simulating the dynamics of a full head of hair (i.e., typically one hundred thousand hair strands) has, for a long time, been a challenging task in computer graphics. The difficulty mainly stems from the computational cost and numerical sta-

bility, especially in the interactions between individual strands of hair, which are essential for animation. Other important concerns for hair simulation include how to model each strand to represent various hairstyles; many recent techniques handle hair as clumps or a continuum to avoid the computational issues and focus on specific hair styles such as straight only [3, 70, 91] or straight plus curly hairstyles [9, 93]. Recently, an extended mass-spring model [80] undertook to simulate the dynamics of up to ten thousand individual hair strands. However, this required a complicated configuration for the spring structure and suffered from high computational cost.

This dissertation develops a fine-detailed hair simulation model carving from a single strand level. There are many strand-like objects in our daily lives, e.g., shoelaces, threads, rubber cords, plastic fiber and spaghetti. The strand-like objects have a wide variety of materials, and a simulation of its dynamics is a challenging problem. Such strand-like objects exhibit interesting behaviors such as twisting, tearing (by stretching or twisting) and bouncing back when pulled and released. However, all behaviors of a strand-like object are not introduced together in a single framework in previous methods.

Handling of inextensible strands, such as hair strands and threads, poses another technical challenge. To prevent inextensible strands from excessive elongation, many length-constraint schemes called *strain limiting* have been developed [4, 21, 26, 63, 75]. With strain limiting, however, the tearing simulation becomes difficult; whereas an *extensible* strand will break when its length or strain reaches a certain breaking point, it is difficult to see when an *inextensible* strand will tear based on the constrained length. Moreover, beside the fact that an inextensible strand is not elongated by their own weight, under a large applied force such as a large pulling force, the strand should be elongated according to its material property. However, the strain limiting causes the material property unrelated to the applied force.

Contributions: This dissertation presents a model that handles the deformation

of an individual strand based on *Lattice Shape Matching* (LSM) [76]. LSM has been successfully used for simulating deformable objects because it is simple, fast and numerically stable. While LSM assumes that a deformable object can be approximated by a set of particles aligned in a lattice, we represent a hair strand as a chain of particles; therefore we call our method *Chain Shape Matching* (CSM). CSM is much simpler than LSM because it only considers deformations along a single chain. We can immediately use the particles' positions in the original shape to create a chain structure, which greatly benefits the shape transformation process. Then, the model is further developed to enable interesting behaviors of strand-like objects, i.e., twisting, tearing (by stretching or twisting) and bouncing back when pulled and released. Specifically, the following three contributions are offered. First, a method for handling twisting effects with both uniform and non-uniform torsional rigidities is introduced. Second, this dissertation presents a method for estimating the tension acting in inextensible strands in order to reproduce tearing and flicking (bouncing back); whereas the tension for an *extensible* object can be computed via stretched length, the length of an *inextensible* object is maintained constant in general, and thus we need a novel approach. Third, an optimized grid-based neighbors search for accelerating the collision detection is introduced. For a simulation of large number of hair strands, the proposed method can be implemented entirely on the GPU and achieve interactive performance for up to several tens of thousands of hair strands such as that in Figure 1.3(a).

1.2 Wetting Effects in Granular Materials

The physics of granular materials is far more complex than it looks. To the best of our knowledge, a lot of researches are currently going on trying to understand the physics of what happens to the physical properties of a granular material when wet, but there are no reasonable theoretical models available to describe their properties. This is mainly due to the difficulty of quantitative experiments that

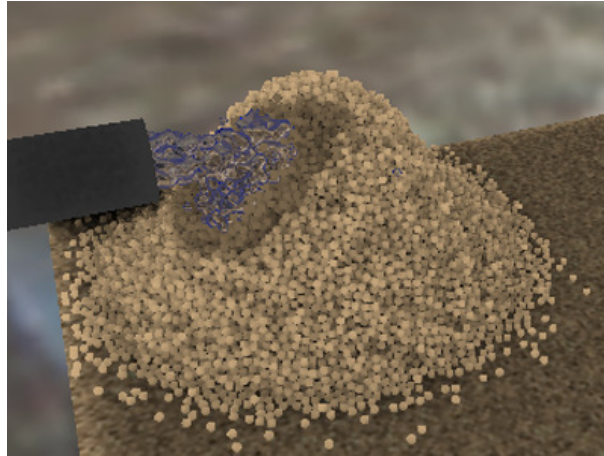


Figure 1.2: A sand pile consisting of 32,000 particles with a water stream emitted from a black faucet. The frame rate is about 49 fps.

can be used to develop models.

Phenomena of both dry and wet granular material taken into consideration in this dissertation are splashes of sand particles from interactions between sand and water, traceable *wetting fronts* (the moving boundary between wet and dry region), the travel of water through granular materials and the changes in physical properties of granular material due to the wetness.

A pile of granular materials includes a vast number of small gaps between the individual particles. When such a pile comes into contact with a water, water is absorbed into the spaces and propagated through the material, mainly induced by capillary actions. The capillary actions occur by the small gaps that are small enough to have the capillary phenomena. Water is trapped into these small gaps, so the gaps determine how much fluid can be held. To model this, the proposed method introduces a capacity of wetness (a wetness value) for each particle, and describe the physical phenomena based on the wetness value.

Contributions: This dissertation presents a simple particle-based method to model the physical mechanism of wetness propagating through granular materials; fluid particles are absorbed in the open spaces between the granular particles

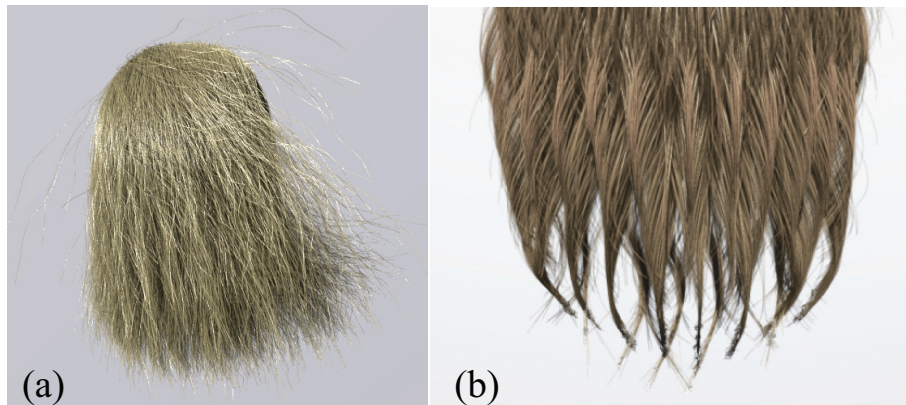


Figure 1.3: (a) Animating 10,000 straight strands of hair (16 particles per strand, about 12 fps on the GPU), with the dynamics of each strand of hair. (b) A simulation result of 5,000 wavy hair strands with water showered onto them until completely wet. The water is absorbed and diffused, making the hair wet and saturated, then the excess water flows along the hair strands until dripping out at the tips. The wet hair strands stick to surrounding strands forming several clumps, while the detail of some wet stray hair linking between clumps can be seen as well.

and then these wet granular particles stick together due to liquid bridges that are caused by surface tension and subsequently disappear when there is too much fluid penetrates into the spaces. The proposed method can handle these phenomena by introducing a wetness value for each granular particle and by integrating those aspects of behavior dependent on wetness into the simulation framework. For the use in interactive applications, this dissertation seeks a simple yet practical model that simulates the physical phenomena of granular materials taking wetting effects into account. Therefore, the proposed method is a simplified physical model. The simulator, running entirely on a GPU, can produce animated scenes such as that in Figure 1.2 in real time.

1.3 Wetting Effects in Strands

Wetting of hair is an interesting phenomenon in our everyday life. When water contacts with hair, the water is absorbed and diffuses into the hair, making the hair wet. Then, the wet hair strands stick to surrounding strands and form several clumps due to cohesive forces caused by water bridges. The absorbed water also makes the hair strands heavier and temporally alters the protein structure (keratin) of a hair strand, causing its shape to change. For example, one's straight hair may turn into wavy hair or vice versa when wet. With enough water, the hair will be saturated and the excess water will flow down the hair strands to the tips. When the water at the tips grows large enough, it will turn into a water drop dripping out off the hair.

Hair strands have special characteristics differ from granular material. The permeability (the water absorption and diffusion) of hair originates in not only a vast amount of microscopic void spaces between the hair strands, but also an inner part of each hair strand. Each hair strand is also a kind of porous medium which can absorb water up to 30%-45% of its mass [52]. The microscopic void spaces among hair strands form a porous structure similar to granular material. However, open spaces in hair volume are different and difficult to capture. The open spaces in hair usually lie in complex contact regions among the hair strands that can be drastically changed with hair motion. Furthermore, hair is an anisotropic porous medium; water diffuses more in the hair fiber direction than the orthogonal direction. Please see [77] for more detail on chemical and physical behaviors of a hair strand.

Contributions: This dissertation introduces a simulation model that reproduces full interactions between water and hair as a dynamic anisotropic porous medium. The proposed method captures the wetting effects of fine-detailed dynamic hair simulation using a Cartesian bounding grid. Hair can be handled as a dynamics anisotropic porous medium. The complex porous structure of hair can be captured

even under the motion. While the wetting effects of hair are handled by an Eulerian approach, the proposed CSM is used to simulate every single hair strand as an individual entity, yielding the best plausible results of fine-detailed hair dynamics. The CSM can efficiently handle individual hair strands as well as an easy shape controllability of hair strands which advantages morphological shape transformation of wet hair. An interesting interactions between fine-detailed hair and water can be handled by our method as shown in Figure 1.3(b), i.e., water absorption and diffusion, cohesion of wet hair strands, water flow within the hair volume, water dripping from the wet hair strands and morphological shape transformations of wet hair.

1.4 Organization

The rest of this dissertation is organized as follows. Chapter 2 introduces related previous work. Chapters 3 describes fundamental simulation methods for fluid, granular material (DEM) and the basis deformation behaviors for a strand (LSM). Chapter 4 explains the proposed simulation model for strands. Chapters 5 and 6 introduce the proposed models for handling wetting effects in granular material and strands, respectively. Finally, the conclusion and the future work are discussed in Chapter 7.

Chapter 2

Related Work

Physically-based simulation has a long history in computer graphics. This chapter introduces some of the many techniques that have been proposed for simulating fluids, granular materials and strands as well as their interactions with fluid taking wetting effects into account.

2.1 Fluid Simulation

Fluid simulations are divided into the Eulerian and Lagrangian methods. The Eulerian methods observe and analyze fluid at fixed positions by using a data structure such as a grid [23, 24, 59, 88]. In contrast, Lagrangian methods approximate a continuous fluid using a set of particles and model the behavior, making Lagrangian techniques useful for simulations where the topology of the fluid largely changes. Therefore, this dissertation uses particle-based methods in order to generate dynamic animations.

In the particle-based methods, *Smoothed Particle Hydrodynamics* (SPH) [62] and the *Moving Particle Semi-implicit* (MPS) method [44, 74] have already been studied well. The former solves the particle advection equation explicitly, while the latter solves it implicitly. The former seems the preferred method for computer graphics due to its computational simplicity. Following the proposal to apply SPH-based fluid simulations to interactive applications by Müller *et al.* [62], vari-

ous other methods, e.g., viscoelastic fluids [18] and multiple interacting fluids [65] have already been presented. Several hybrid methods of Eulerian/Lagrangian [54, 81, 85] have been proposed as well. Techniques to accelerate SPH make use of hierarchical data structures [42], adaptive sampling density [1] or GPU-based computation [33, 35, 34, 50, 100]. The proposed method also uses SPH for the fluid simulation.

2.2 Granular Materials Simulation

The dynamics of materials such as sand, gravel or grain are usually represented either as a continuum or as a set of individual particles. For the continuum approach, several methods use height fields [15, 48, 69, 90] or handle the material as a fluid [66, 103]. Bell et al. [7] used the *Discrete Element Method* (DEM) [19] to represent a granular material as separate particles, and Harada [32] implemented a DEM simulation on the GPU. The proposed method use DEM in order to be able to handle dynamic animations.

2.3 Strands Simulation

Strand dynamics: The early research on strand simulations handled the dynamics of individual strands of hair using the mass-spring system [78] and projective dynamics [2]. However, these methods ignore hair-hair interactions, curliness and torsion. Hadap and Magnenat [31] use a rigid multibody serial chain to represent a hair strand which can capture the torsion effect, but still ignore curliness. Pai [71] and Bertails et al. [9] introduced physically-based models based on Cosserat's and Kirchhoff's theories, respectively, to represent a strand of hair. The models can capture bending, torsion, non-stretching and the curliness behavior of hair at high expense of computational cost.

For computational efficiency, several techniques regard hair as a continuum [3, 31, 93] or disjoint groups such as wisps [20, 45, 73, 72] and strips [27, 41, 43, 49,

89, 92]. Instead of a continuum and fixed groups of strands, Ward et al. introduced adaptive [96] and Level-of-Detail [98] approaches to increase the detail of hair. However, these models limit the degrees of freedom (DOFs) of the hair motion; although human hair has a collective tendency, a high number of DOFs is required to represent the fine-scale motion of hair blown about by the wind. Please refer to the survey [94] for advances on hair modeling, styling, simulation and rendering.

Recently, mass-spring systems have commonly been used for simulating dynamics of hair and other strand-like objects. Integrations in mass-spring systems are performed using explicit or implicit schemes. Explicit schemes are often preferred due to the low computational cost and ease of implementation. However, for stable simulation, the time step in an explicit scheme should be inversely proportional to the square root of the spring constant (the Courant condition). As a result, highly stiff hair is difficult to simulate at an interactive rate when using an explicit scheme. Implicit schemes can solve the stability problem, with higher computational cost. Selle et al. [80] proposed the use of additional altitude springs to simulate the complex behavior of human hair and semi-implicit springs to solve the stability problem. However, this is not suitable for stylized hair in interactive applications due to the expensive cost and complex configuration of the springs.

As for hair-hair interactions, most of the previous methods only consider collisions occurring on guide hair strands [9, 17] and possibly miss collisions between interpolated hair strands when the guide strands do not collide. There has been little research that takes full hair-hair interactions into consideration, except for the time-consuming bounding box hierarchy used in [80] and the hybrid technique of Eulerian and Lagrangian methods introduced in [56]. Tariq and Bavoil [91] introduced a fast technique for inter-hair collisions using a hair density field. All the hair strands are voxelized into a grid then repulsive forces are applied to hair particles in high density areas. However, their technique only considers volume preservation of hair, not collisions between each individual hair strands. The same limitation can be seen in the continuum methods [3, 31, 93].

Unlike most of the previous methods, the proposed method is easy to implement, can easily handle complex hairstyles, is interactive and numerically stable even if the hair is very stiff. Also, our method handles collisions between individual hair strands.

Twisting effect of a strand: Considerable research on the twisting effects in strand simulation introduced various models for solving the Cosserat and Kirchhoff energy equations. Bertails *et al.* [9] introduced a mechanical model called *super helices* for simulating human hair based on the Kirchhoff theory. However, handling collision responses is not straightforward due to the implicit representation of hair strands. Spillmann and Teschner [86] explicitly represented the centerline of an elastic strand and used the finite element method (FEM) to solve the Cosserat energy equation. Recently, Bergou *et al.* [8] introduced a discrete model for simulating elastic strands based on the Kirchhoff theory. However, the twisting angles are computed with a quasi-static assumption, thus the twisting of non-uniform torsional rigidity along the strand is not addressed. There are also several works on pseudo-physical models that can capture the twisting effect without solving the energy equations. Hadap [30] introduced a model for capturing the torsion effect by integrating a torsion spring into each joint of rigid links. However, strands cannot be stretched and collision handling is not straightforward, because the motion is propagated from top to bottom in one single pass (no backward propagation). Selle *et al.* [80] represented a hair strand by a chain of tetrahedrons of springs and captured the torsion effect by introducing appropriate altitude springs. However, the configuration of springs is complex, and auxiliary particles are required along a strand.

The proposed method is a pseudo-physical model which is easy to implement and can easily handle twisting effects with both uniform and non-uniform torsional rigidities.

Strain limiting for an inextensible strand: In order to handle inextensible objects simulated by deformation models, a variety of methods for stretch resis-

tance have been continuously proposed; from Provot's iterative post-processing edge constraint [75], to a more recent constraint method based on impulse [21]. Some alternative ways of stabilizing stiff simulation were also proposed [4, 26, 63]. These methods, and many of their sequels, have a common goal to limit the maximal strain to a certain threshold. Accordingly, these kinds of methods are problematic in case of excessive stretch or when rupture should occur. Metaaphanon *et al.* [57] proposed a method to deal with cloth tearing using a mass-spring model. However, it tears cloth by checking lengths of springs; when and where yarns of cloth are cut were not directly related to user-applied external forces and cloth material properties, but dependent on how the method constrains the springs.

The proposed method can estimate the tension acting in inextensible strands. The estimated tension is used for reproducing tearing and flicking (bouncing back).

2.4 Wetting Effects in Granular Materials

Several methods handle the interactions of different material types such as fluids and rigid bodies [1, 5, 6, 14] or fluids and soft bodies [16, 28, 84]. However, there are only a few researches taking the wetting effects into account. Despite the fact that employing wetting effects into solid-fluid coupling simulations could greatly enrich the realism.

There are models for wetting effects in soil or pavement based on Darcy's law [22, 46] consider permeable materials as continuum, and thus they could not be used in highly dynamic animations of granular materials or hair where the structures have very large discontinuities.

Liu *et al.* [51] used height fields and the *volume of fluid* method to model the case of fluids on the surface of an object being absorbed and causing erosion. However, their method cannot represent small-scale movements or complex

changes in the object's topological structure. Wojtan *et al.* [99] handled the animation of natural phenomena such as erosion, sedimentation, and acidic corrosion. They provided an example in which a sandcastle is washed away, but while the volume of sand decreases, there are no signs of splashing of the sand or of water absorption. Lenaerts *et al.* [47] presented a unified SPH framework to simulate the interactions between fluids and granular materials. The volume of materials is approximated by particles and the propagation of wetness between particles is computed. They can generate the scene of water absorption and propagation in granular materials. However, the large discontinuities of granular materials such as spattering effects could not be generated, because the granular materials are simulated as a continuous material. Moreover, the simulation has a high computational cost and the results are rendered off-line. It could not be used in the interactive applications which this dissertation targets.

By using a particle-based method, the proposed method can represent even small-scale interactions between water and sand. Furthermore, by taking into account the wetness of the granular material when calculating its behavior, the proposed method can model phenomena such as cohesion, erosion and the absorption of water.

2.5 Wetting Effects in Strands

Most of previous work regarding wet hair did not target simulations but focused on rendering [29] and modeling [12, 82]. Only a few research studies have been introduced for the dynamics of strand-like objects with wetting effects. Bruderlin [12] proposed a method for modeling wet fur used in the movie production, *Stuart Little*. The wet fur is modeled as a cone whose shape is changed depending on the amount of water absorbed. Silva *et al.* [82] introduced a method for handling curling and clumping of animal fur using a 3D texture. Ward *et al.* [95, 97] introduced a method for simulating wet hair whose wetness is supplied by user

interactions. Their method is a clump-based model which sacrifices detail for speed. None of these methods is a full simulation between hair and water. To the best of our knowledge, the method of Ward et al. [95] is the latest approach in wet hair simulation model.

The wetting effects in porous media and water simulation has been studied recently. Lenaerts et al. [46] integrated porous flow into SPH framework for permeable media (rigid and elastic bodies) and water simulation. Later, Lenaerts et al. [47] applied their framework to porous flow in granular material such as sand. Their methods sample solid permeable media in a macroscopic scale by particles and handle wetting effects as an interactions between particles. However, these models do not consider the dynamic anisotropic porous medium such as hair. Huber et al. [37] introduced a simulation model for wet cloth including water absorption, diffusion and stickiness. However, their model is limited to wetting effects on a flat surface.

The proposed method can reproduce full interactions between water and hair as the dynamic anisotropic porous medium, i.e., water absorption and diffusion, cohesion of wet hair strands, water flow within the hair volume, water dripping from the wet hair strands and morphological shape transformations of wet hair.

Chapter 3

Fundamental Methods

In this chapter, the simulation methods for fluid, granular material and the basis method for our strand dynamics are briefly introduced, i.e., *Smoothed Particle Hydrodynamics* (SPH), *Discrete Element Method* (DEM) and *Lattice Shape Matching* (LSM), respectively. From this chapter, the particle in the DEM domain is called a *DEM particle* and that in the SPH domain is called as a *SPH particle*.

3.1 Smoothed Particle Hydrodynamics (SPH)

SPH is a particle-based simulation method, which was originally developed for use in astronomy [25, 55]. It uses a set of particles as a discrete approximation of a continuum, expressing a field quantity $A(\mathbf{x})$ by interpolating between the respective quantities around point \mathbf{x} , as follows:

$$A(\mathbf{x}) = \sum_i m_i \frac{A_i}{\rho_i} W(\mathbf{x} - \mathbf{x}_i, h), \quad (3.1)$$

where m_i is the mass of particle i , A_i is the respective quantity of particle i , ρ_i is its density and \mathbf{x}_i its position. The function $W(\mathbf{x}, h)$ is a smoothing kernel with core radius h (Figure 3.1).

Using Eq.(3.1) the density of fluid can be approximated as

$$\rho_i = \sum_j m_j W(\mathbf{x}_j - \mathbf{x}_i, h), \quad (3.2)$$

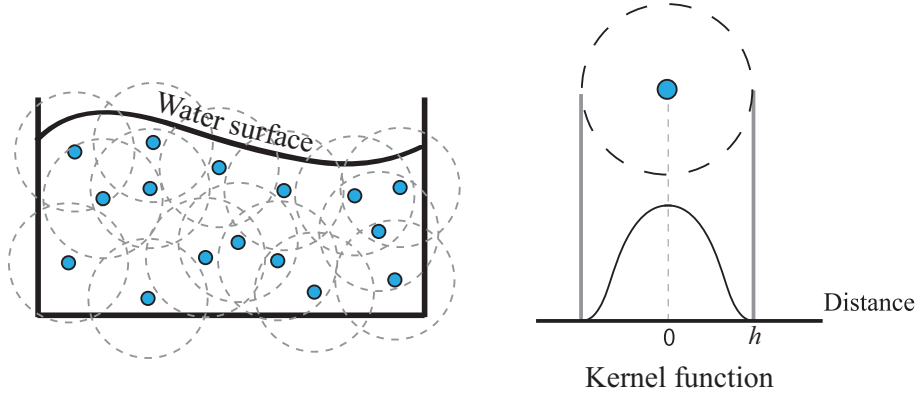


Figure 3.1: SPH model and kernel function. h is a core radius of the kernel function.

then the pressure of particle is calculated.

$$p_i = p_0 + k(\rho_i - \rho_0), \quad (3.3)$$

where p_0 , ρ_0 and k are the rest pressure of the SPH particle, the rest density of the SPH particle and the fluid stiffness, respectively.

When applied to fluids, each of the terms in the governing Navier-Stokes equations are expressed in the above-mentioned form. The formulaization of Müller *et al.* [62], which the proposed method also uses, keeps forces between particles symmetric by calculating the pressure and viscosity terms as follows:

$$\mathbf{F}_i^{pressure} = - \sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (3.4)$$

$$\mathbf{F}_i^{viscosity} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (3.5)$$

where μ , p_i and \mathbf{v}_i represent the viscosity coefficient, the pressure and the velocity of the particles, respectively. For more details, please refer to their paper [62].

3.2 Discrete Element Method (DEM)

DEM is also a particle-based simulation method, which was originally used for rock mechanics problems [19]. In DEM, a sand particle is approximated as a

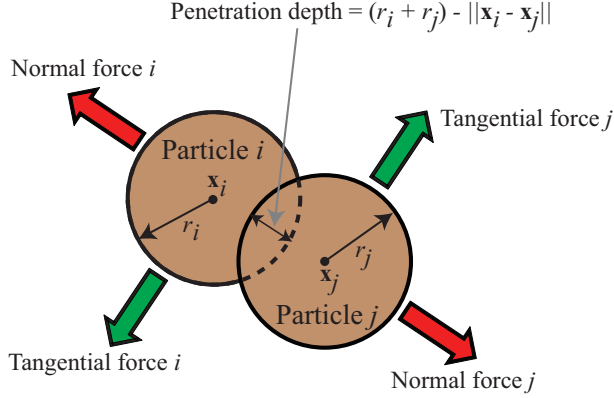


Figure 3.2: DEM model.

sphere, which is also true in the present system. Particles move freely under the external forces such as the gravity force.

For a pair of colliding particles i and j , the proposed method calculates the normal and tangential forces acting on the particles: \mathbf{F}_i^{normal} and $\mathbf{F}_i^{tangential}$ (Figure 3.2). Note that the normal direction is defined by the vector from a center \mathbf{x}_i to \mathbf{x}_i . The normal force is modeled in terms of springs and dampers between the particles, while the tangential force is due to the friction.

$$\mathbf{F}_i^{normal} = \mathbf{F}_i^{spring} + \mathbf{F}_i^{damper}, \quad (3.6)$$

$$\mathbf{F}_i^{spring} = k_s(d_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \quad (3.7)$$

$$\mathbf{F}_i^{damper} = k_d \mathbf{v}_{ij}^{normal}, \quad (3.8)$$

$$\mathbf{F}_i^{tangential} = k_t \frac{\mathbf{v}_{ij}^{tangential}}{\|\mathbf{v}_{ij}^{tangential}\|}, \quad (3.9)$$

where k_s is the spring constant, $d_{ij} = r_i + r_j$ (r_i and r_j are the radii of particles i and j), k_d is the damper coefficient, \mathbf{v}_{ij}^{normal} and $\mathbf{v}_{ij}^{tangential}$ are the particles' relative normal and tangential velocities and k_t is the coefficient of friction.

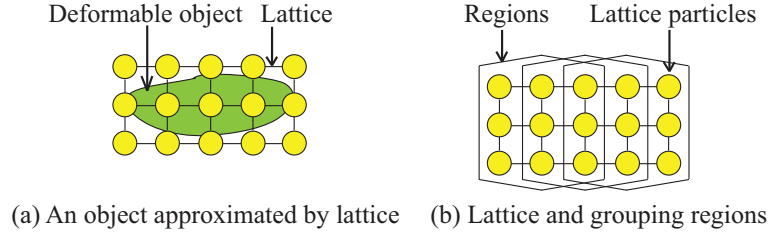


Figure 3.3: LSM model.

3.3 Lattice Shape Matching (LSM)

LSM is an extension of the shape matching method [64]. The main advantages of the shape matching method are unconditional stability and high controllability due to its geometrically-motivated computation. Optimally-transformed positions are computed first, and then particles are moved towards those positions. Since it guarantees that all particles are updated towards the appropriate positions, the overshooting problem that occurs in explicit integration schemes is eliminated. This technique is later generalized as *position based dynamics* [63], which can be applied to general simulation systems.

In LSM, the particles are grouped into multiple-overlapping cubical regions (Figure 3.3). The region half-width value w ($w = 1, 2, 3, \dots$) corresponds to the stiffness of the object. The positions of particles are updated as follows. First, they are moved independently according to the external forces. Next, for each region, LSM computes an optimal rigid transformation (i.e., rotation and translation) based on shape matching [64]. The rigidly-transformed positions of the particles are called *goal positions*. The goal position \mathbf{g}_i of particle i is weighed in the overlapping regions by particle per-region mass $\widetilde{m}_i = \frac{m_i}{N_r}$, where m_i is the mass of particle i and N_r is the total number of regions that the particle i belongs to. The goal position of particle i is computed as follows.

$$\mathbf{g}_i = \frac{1}{N_r} \sum_{r \in \mathcal{R}_i} (\mathbf{R}_r (\mathbf{x}_i^0 - \mathbf{x}_{cm,r}^0) + \mathbf{x}_{cm,r}), \quad (3.10)$$

where \mathcal{R}_i is a set of regions that particle i belongs to, \mathbf{x}_i^0 is the original position, $\mathbf{x}_{cm,r}^0$ is the center of mass of the original shape of the region, $\mathbf{x}_{cm,r}$ and \mathbf{R}_r are the optimal translation and rotation for region r . Finally, for each particle, the velocity is computed toward the goal position.

$$\mathbf{v}_i(t + dt) = \mathbf{v}_i(t) + \frac{\mathbf{g}_i(t) - \mathbf{x}_i(t)}{dt} + dt \frac{\mathbf{f}_{i,ext}(\mathbf{x}_i, t)}{m_i}, \quad (3.11)$$

$$\mathbf{x}_i(t + dt) = \mathbf{x}_i(t) + dt \mathbf{v}_i(t + dt), \quad (3.12)$$

where dt is a time step, \mathbf{v}_i is the velocity and $\mathbf{f}_{i,ext}$ the external force.

Chapter 4

Strands Simulation

This chapter describes how to handle dynamics of a strand using Chain Shape Matching (CSM) based on LSM. Next, a method for handling twisting, tearing and flicking of a strand in CSM is presented. For the twisting effect, this chapter introduces a simple method that adds twisting angles into each segment in a strand which can handle both uniform and non-uniform torsional rigidities. A method for estimating the tension is also explained for tearing and flicking effects in an inextensible strand whose actual tensile stress and strain values are constrained from the strain limiting. In addition, the torsional tension is considered to handle plasticity and tearing from twisting as well. This chapter also introduces a collision searching scheme for efficient collision handling of strands using a grid-based data structure. The proposed searching scheme has a less number of neighbors to be searched compared to typical searching schemes.

In the case of full head of hair strands, this chapter introduces a GPU implementation of CSM without twisting, tearing and flicking effects. For the GPU implementation, the twisting, tearing and flicking effects are neglected for the efficiency. This does not affect the realism. These effects are not significant in common animations of human hair, because the external forces that cause these effects in hair are rare.

Algorithm 1 Pseudocode of CSM algorithm.

```

1: for all particles  $i$  do
2:   initialize original position  $\mathbf{x}_i^0$ ,  $\mathbf{x}_i \leftarrow \mathbf{x}_i^0$ 
3: end for
4: loop
5:   BucketGeneration() // Sections 4.5 and 5.6
6:   for all particles  $i$  do
7:      $\mathbf{f}_{i,ext} \leftarrow \text{ComputeCollisionForce}() + \mathbf{f}_{gravity}$ 
8:   end for
9:   for all particles  $i$  do
10:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + dt \frac{\mathbf{f}_{i,ext}}{m_i}$ 
11:     $\mathbf{x}_i \leftarrow \mathbf{x}_i + dt \mathbf{v}_i$ 
12:  end for
13:  for all chain regions  $R_i$  do
14:     $\mathbf{x}_{cm} \leftarrow \text{ComputeOptimalTranslation}()$ 
15:     $\mathbf{R} \leftarrow \text{ComputeOptimalRotation}()$ 
16:  end for
17:  for all particles  $i$  do
18:     $\mathbf{g}_i \leftarrow \text{ComputeGoalPosition}()$  // Eq.(3.10)
19:  end for
20:  for all particles  $i$  do
21:     $\mathbf{g}_i \leftarrow \text{StrainLimiting}()$  // Section 4.2
22:     $\mathbf{x}_i \leftarrow \mathbf{g}_i$ 
23:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{\mathbf{g}_i - \mathbf{x}_i}{dt}$ 
24:  end for
25: end loop

```

4.1 Chain Shape Matching (CSM)

A strand is represented by a chain of particles grouped into multiple overlapping chain regions (Figure 4.1). Each particle i is associated with a chain region $R_i \in \mathcal{R}_i$ that centers the particle i and contains adjacent particles within the region half-width w . Each chain region uses the same shape matching method as used in LSM (Figure 4.2). Therefore the proposed algorithm is called Chain Shape Matching (CSM). Algorithm 1 describes the pseudocode of the CSM.

In CSM, we can design complex hairstyles at the strand-level as follows. The shape of the strand can be defined by the original particle positions, since these

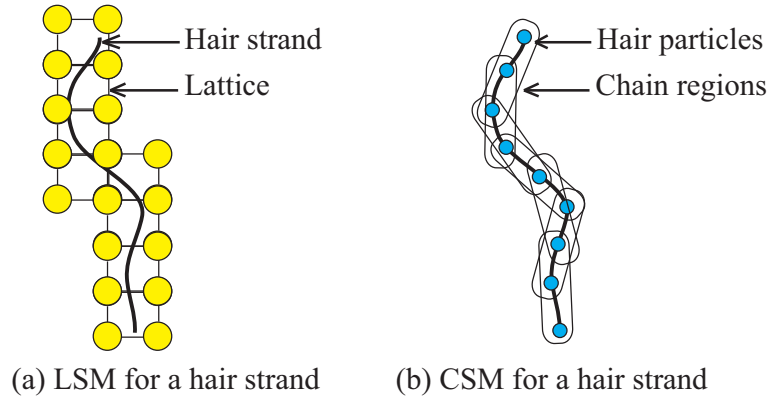


Figure 4.1: Illustrations of LSM and CSM models for a hair strand.

original positions define the shape of the strand at rest (Figure 4.3). The root of each strand is fixed by several particles which are constrained on the head. The number and direction of the constrained particles partially determine the behavior of the strand. The stiffness of the strand is defined by the chain region half-width w (Figure 4.4), which can be partially modified to generate complex hairstyles; e.g., soft straight hair near the root and stiff curly hair near the tip of the hair strand.

Regarding the stiffness control of hair strands, one might consider the use of parameters $\alpha, \beta \in [0, 1]$, as presented in the original shape matching paper [64]; α controls the tendency that goal positions are moved towards rigidly-transformed positions, and β allows goal positions to undergo a linear transformation (see [64] for more details). While α and β can control the stiffness independently of w , we simply fix $\alpha = 1$ and $\beta = 0$ according to the LSM paper [76], taking into account that α and β can make a region softer but not stiffer. Although reducing the number of particles makes a region stiffer, it also reduces the hair strand's DOFs required especially for complex hairstyles. Nevertheless, the use of α and β in conjunction with w might benefit to advanced stiffness control, which is left as future work; a study on the relationship between β and w can be found in [68].

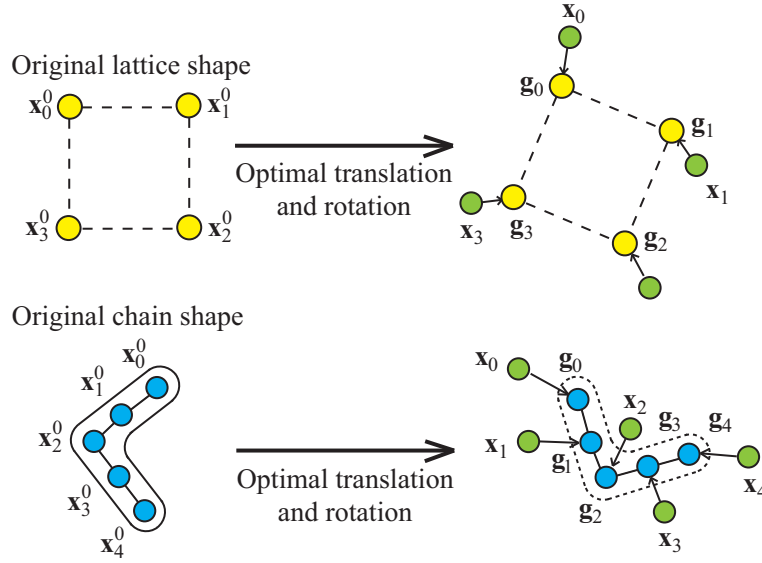


Figure 4.2: \mathbf{x}_i^0 is the original position, \mathbf{x}_i is the position updated by external forces and \mathbf{g}_i is the goal position of particle i .

4.2 Strain Limiting

Techniques for constraining stretching were originally proposed in research on cloth simulation, known as *strain limiting*. Most of the cloth simulations use an elastic system to model cloth such as popular mass-spring systems. As a drawback, elastic objects are usually excessively stretched under its own weight and large applied force. Using large stiff force can ease the excessive elongation problem, however, it leads to numerical instability. Instead, position constraints are often imposed so that the length of each segment i does not exceed a certain threshold L_i^{max} [4, 21, 26, 63, 75]. For our implementation, we used the position constraints method of [63].

Denoting the position vector of particle i by \mathbf{x}_i , we constrain the length $L_i = \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$ of segment i below L_i^{max} by moving \mathbf{x}_i and/or \mathbf{x}_{i+1} according to the state of a strand. There are three possible cases.

1. A strand clamped at one end: Length-constraint is applied from the clamped

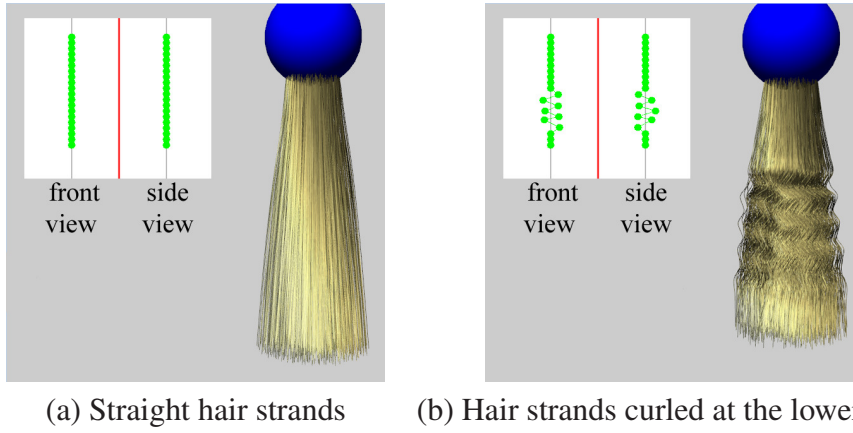


Figure 4.3: The original particle positions define the shapes of the hair strands. Green particles show the original particle positions.

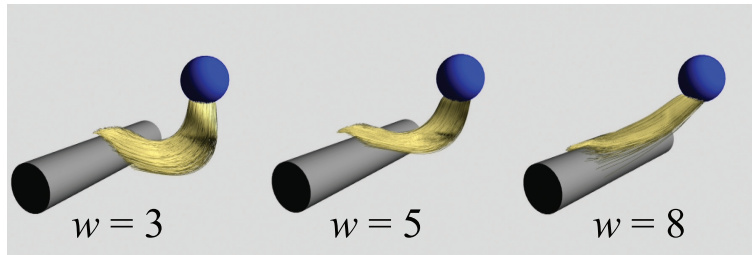


Figure 4.4: A bunch of 1k strands is dragged over a cylinder, with different chain region half-width w . Small w makes the strands softer while large w stiffer.

end sweeping through to another end. E.g., in a strand clamped at $i = 0$, \mathbf{x}_{i+1} is moved sweeping from the clamped end to another end.

2. A strand clamped at both ends: We do multiple adjustments from one end to another end of the strand sweeping back and forth repeatedly, since correcting the length of one segment may change the length of other segments.
3. A strand clamped at multiple points: We separate the strand into strands clamped at both ends and strands clamped at one end, and do the length-constraint accordingly.

However, case 1 is generally applied in case of hair since the root of a hair

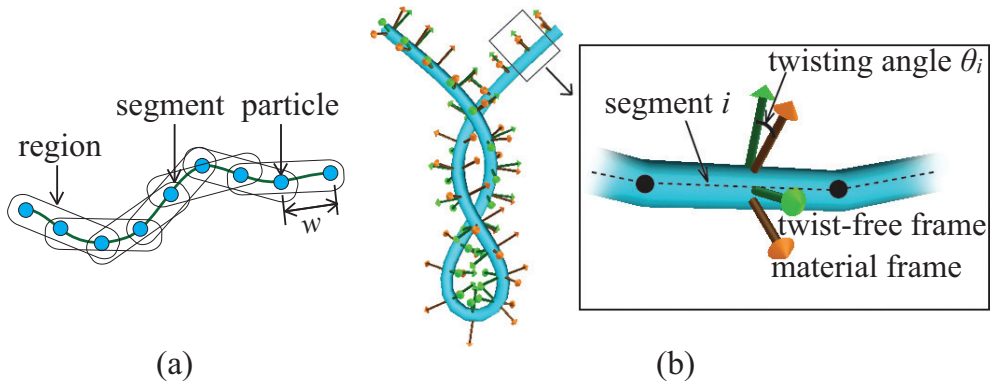


Figure 4.5: A strand model. (a) Multiple overlapping chain regions in CSM. (b) A twisting angle of a segment of an elastic strand is an angle between a twist-free frame and a material frame.

strand is always attached to a head. Note that just moving particles to the non-stretched positions leads to the loss of linear and angular momenta. To conserve the momenta, our method modifies velocity, similarly to [63].

4.3 Twisting Effects

Based on CSM, a strand is represented as a chain of $(n + 1)$ particles connected by n segments (Figure ??(a)). A segment $i \in \{1, 2, \dots, n\}$ has a twisting angle θ_i tracking how much the segment is twisted. The twisting angle can be represented as an angle between a *twist-free frame* (*bishop frame*) and a *material frame* (Figure 4.5(b)). In the initial state, we specify an initial angle θ_i^0 of each segment i according to the shape of the strand. The twisting angle is assigned for each segment, not for each particle, to avoid the ambiguity.

The behavior of twisting can be clearly observed when a strand clamped at both ends is twisted at one end. Therefore, we use this scenario for our explanation (Figure 4.6). When we twist one clamped end with an angle θ_t , the angle θ_i of the segment is increased. The increment of the twisting angle of the segment is propagated to the next segments in order to minimize the elastic energy in the strand. In other words, the strand tries to minimize the twisting angles between

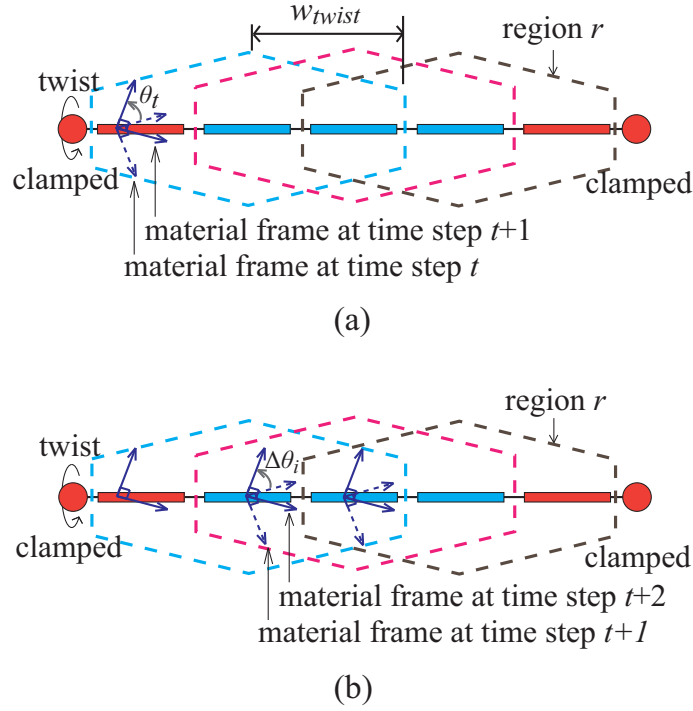


Figure 4.6: (a) An elastic strand clamped at both ends is twisted at one end with a twisting angle θ_t . (b) The increment of the twisting angle is propagated to next segments.

each connected segment. We compute and update a goal twisting angle for each segment, similarly to finding a goal position for each particle in shape matching.

First, we group the segments into multiple overlapping chain regions with the region half-width $w_{twist} \in \{1, 2, 3, \dots\}$ which affects the propagation speed of twisting angles in the strand or the torsional rigidity; the larger the w_{twist} is, the faster the change of twisting angles is propagated. The size of each region in a strand can be varied for handling non-uniform torsional rigidity. The minimized twisting angle increment $\Delta\theta_k^{region}$ of each region k is computed by averaging the twisting angle increment $\Delta\theta_j = \theta_j - \theta_j^0$ of the segments in the region k weighted

by mass m_j :

$$\Delta\theta_k^{region} = \frac{\sum_{j \in S_k} m_j \Delta\theta_j}{\sum_{j \in S_k} m_j}, \quad (4.1)$$

where S_k is a set of segments within region k . Then, θ_i of each segment i is updated with the twisting angle increment $\Delta\theta_i^{segment}$. The goal twisting angle increment $\Delta\theta_i^{segment}$ is calculated by summing the twisting angle increment $\Delta\theta_k^{region}$ of each region k that segment i belongs to:

$$\Delta\theta_i^{segment} = \sum_{k \in \mathfrak{R}} \Delta\theta_k^{region}, \quad (4.2)$$

$$\theta_i = \Delta\theta_i^{segment}, \quad (4.3)$$

where \mathfrak{R} is the set of regions that segment i belongs to.

While a segment is updated to the goal twisting angle, a torque occurs in the cross-section, causing the change of rotational velocity $\omega_i^{segment}$ in the segment's tangential axis. In each time step Δt , a segment is rotated by the rotational velocity first, then it is updated to the computed goal twisting angle in Eq.(4.3). The rotational velocity and the time evolution of twisting angle are computed as follows:

$$\omega_i^{segment} \leftarrow \omega_i + I_i \Delta\theta_i^{segment}, \quad (4.4)$$

$$\theta_i \leftarrow \theta_i + \Delta t \omega_i^{segment}, \quad (4.5)$$

where I_i is an inertia moment of segment i .

The twisting force \mathbf{f}_i^{twist} can be treated as an external force to particle i and derived from the elastic energy equation [8] as follows:

$$\mathbf{f}_i^{twist} = \frac{\beta}{L} (\theta_{i+1} - 2\theta_i + \theta_{i-1}) \left(\frac{-\kappa \mathbf{b}_{i+1} - \kappa \mathbf{b}_{i-1}}{2l} \right), \quad (4.6)$$

$$\kappa \mathbf{b}_i = 2 \frac{\mathbf{e}_{i-1} \times \mathbf{e}_i}{|\mathbf{e}_{i-1}| |\mathbf{e}_i| + \mathbf{e}_{i-1} \cdot \mathbf{e}_i}, \quad (4.7)$$

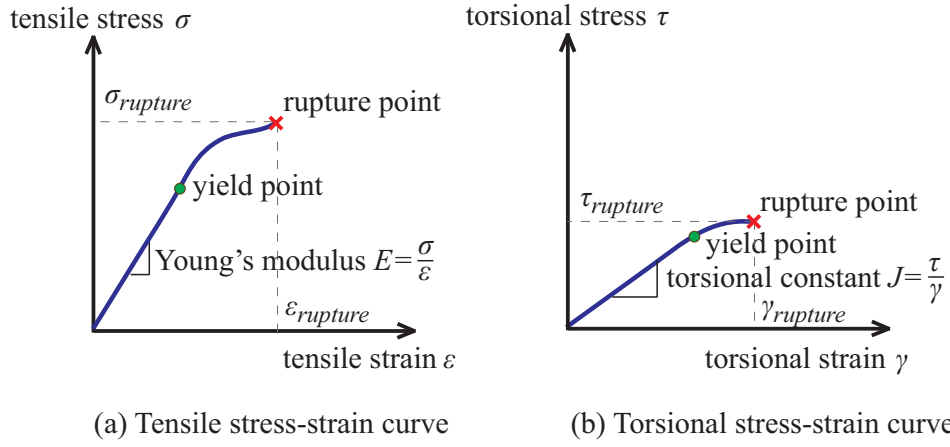


Figure 4.7: Typical tensile and torsional stress-strain curves of a material.

where κ is the curvature, \mathbf{b}_i is the binormal vector, \mathbf{e}_i is the segment vector, l is the length of the segment, β is the twisting stiffness of the strand and L is the total length of the strand.

4.4 Tearing and Flicking Effects

This section briefly reviews the material science of a strand, and then describes the method for handling tearing and flicking effects which was previously difficult due to the strain limiting.

4.4.1 Stress and Strain

In material science, the strength of a strand is associated with its stress-strain curve [10]. There are many kinds of the stress-strain curves depending on the direction of force used in the material strength test, e.g., tensile, compressive, shear and torsional stress-strain curves. Since tearing is typically affected by tensile and torsional stresses, we consider only tensile and torsional stress-strain curves of material in our model. The tensile stress-strain curve shows the relation between an average force per unit area of a cross-section surface and elongation of a strand. The torsional stress-strain curve shows the relation between an average torque on a

cross-section surface and twisting of a strand. Examples of both curves are shown in Figure 4.7.

The tensile stress σ and torsional stress τ of a strand are the average force and the average torque per unit area of a cross-section surface, respectively:

$$\sigma = \frac{\|\mathbf{F}_n\|}{A}, \quad (4.8)$$

$$\tau = \frac{\|\mathbf{T}_n\|}{A}, \quad (4.9)$$

where A is the cross-sectional area, \mathbf{F}_n is the normal force and \mathbf{T}_n is the normal torque. The normal direction is the vector in the cross-section surface's normal direction.

The tensile strain ε and torsional strain γ of a strand are expressed as the ratio of the elongation ΔL to the initial length L_0 and the change of twisting along the axis of the segment, respectively:

$$\varepsilon = \frac{\Delta L}{L_0} = \frac{L - L_0}{L_0}, \quad (4.10)$$

$$\gamma = (\theta_{i-1} - \theta_{i+1})r, \quad (4.11)$$

where L is the current length of the strand and r is a radius of the segment. Note that $\Delta\theta_i = \theta_i - \theta_i^0$ is the change of twisting angle from the rest state, not the tensional strain. The torsional strain comes from the difference of twisting angles between connected segments which are segments $i - 1$ and $i + 1$.

Along the curve, the material exhibits elastic behaviors until the *yield point*; prior to the yield point the material will return to its original shape if the applied force or torque is removed. The slopes of this elastic region are the *Young's modulus* $E = \sigma/\varepsilon$ and *torsional constant* $J = \tau/\gamma$ in the tensile and torsional curves, respectively. Once the yield point is passed, the material becomes plastic; some fractions of the deformation will be permanent and non-reversible. As deformation continues, the material will break when the stress or strain reaches the *rupture point*.

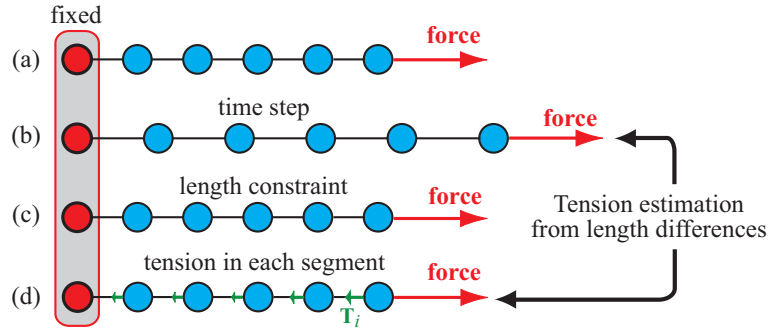


Figure 4.8: A simple example of the tension computation. From (a) to (c), an ordinary length-constrained strand simulation is performed. In (d), tensions are estimated by calculating forces that make the particles move from their unconstrained positions to the constrained positions, yielding an equivalent result with (c).

The stress-strain curve can be derived via a strength testing of a material sample stored as a data set of the experimental result. The stress-strain curve of most materials in the elasticity state is linear, and thus the part of the curve from the origin to the yield point can be stored as a constant value. Still, the data set is required for the curve in the plasticity state. In our implementation, we simply approximate the curve by a line with a constant slope that fits the curve best. As a result, our implementation uses two constant values to represent the stress-strain curve in elasticity and plasticity states together with two constants for the yield point and rupture point.

4.4.2 Tension Estimation

As previously stated, due to the constraint on lengths unrelated to applied forces, actual tensile stress and strain values cannot be directly computed from the simulation result. Here we propose a novel approach to estimate the actual tensile stress and strain values for inextensible strands. The stress and strain are then used for handling elasticity, plasticity, tearing and flicking of strands.

The actual tensile stress and strain values can be computed by estimating the *tensions* in the strand. To derive the tensions, we also consider the particle posi-

tions computed *without* strain limiting. We model the *tension* \mathbf{T}_i of segment i as a stiff force (Figure 4.8d) that makes its particles i and $i + 1$ at both ends move from their unconstrained positions \mathbf{x}'_i and \mathbf{x}'_{i+1} (Figure 4.8b) to the constrained positions \mathbf{x}_i and \mathbf{x}_{i+1} (Figure 4.8c). In our implementation, we compute the tension as follows:

$$\mathbf{T}_i = k_{stiff}(\|\mathbf{x}'_{i+1} - \mathbf{x}'_i\| - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|)\mathbf{t}_i, \quad (4.12)$$

where k_{stiff} is a coefficient and \mathbf{t}_i is a unit vector from particle i to $i + 1$. The tension derived this way is used to reproduce tearing and flicking as well as plastic behaviors of a strand.

4.4.3 Tearing and Flicking Effects

For tearing under a tensile stress, we assign a rupture point or a tensile stress threshold $\sigma_{rupture}$ for each segment. If the segment's tensile stress exceeds its tensile stress threshold, the segment will be broken. The applied tensile stress σ_i can be computed from tension \mathbf{T}_i in each segment using Eq. (4.8) with $\mathbf{F}_n = \mathbf{T}_i$.

Similarly, we can handle the behavior of flicking using the tension. When an inextensible strand is pulled and released or torn apart, the applied stress is vanished but the tensile strain of the segment from the elongated length still remains. The bouncing back force could be computed from an internal tensile stress translated from the tensile strain by referencing the stress-strain curve. However, with our tension estimation technique, we can directly use the tension as the bouncing back force. Note that, without this technique, the strand would just fall down quietly by the gravity force because tensile strain is limited, and thus very small in an inextensible strand.

As can be seen in the tensile stress-strain curve (Figure 4.7(a)), a real strand is lengthened according to the tensile stress in the elasticity and plasticity states prior to the rupture point. Therefore, the maximum length L_i^{max} of each segment used in strain limiting should be updated accordingly (otherwise the strand does not

elongate). For this, we look up tensile strain ε_i corresponding to applied stress σ_i from the tensile stress-strain curve, and use it to compute the appropriate value of L_i^{max} using Eq. (4.10), $L_i^{max} = \varepsilon_i L_0 + L_0$. In the elasticity state, the tensile strain ε_i of a strand becomes zero when applied forces are removed. In other words, the strand returns to its original length. However, when its tensile strain exceeds the yield point (plasticity state), ε_i will remain the same as the last time the forces are applied. Our method also modifies the radius of a segment in order to preserve the volume of the segment when stretched.

The tearing of strand also happens when a sufficient torque is applied to the strand. The tearing caused by twisting regularly arises in several soft strand such as spaghetti and licorice(candy stick). In contrast to tearing by stretching, torsional strain is not limited, and it can be directly calculated from current twisting angles (Eq. (4.11)). Similar to tearing by stretching, when γ_i of segment i reaches a rupture point, we tear the segment. For a plasticity of twisting, when γ_i passes a yield point, we update θ_i^0 with $(\theta_{i-1} - \theta_{i+1})/2$ when it passes a yield point to make the twisted angle of segment i permanent.

4.5 Collision Handling

In this section, we introduce an optimized searching scheme for collision detection of strands. Previous works often use techniques based on bounding volume hierarchy (BVH) [80, 83, 87] and space partitioning using a grid-based data structure [33]. The grid-based data structure is a simple and efficient technique for collision detection of strands which have a large number of self-collisions. Therefore, we based our collision detection on the grid-based structure. Specifically, we treat each segment as a capsule (a cylinder with two spheres at both ends) and search for capsule collision pairs. For neighbor searches, we use a uniform grid of voxels. The number of voxels to be searched is 27 ($= 3 \times 3 \times 3$) in a naïve approach. For better performance, we found that it suffices to search for colliding segments

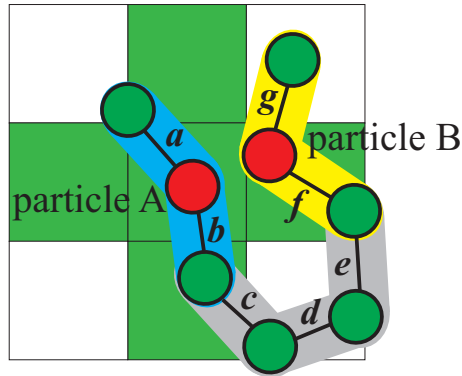


Figure 4.9: A 2D illustration of our optimized searching scheme. When doing a collision detection between particles A and B, segment collision tests between capsules (a, g) , (a, f) , (b, g) and (b, f) are tested.

in only seven neighboring voxels (top, bottom, left, right, front, back and center voxels) under the following three specifications.

- Specifying the voxel size equal to or larger than segment length l
- Storing indices of particles in each voxel
- Searching for capsule collision pairs from two adjacent segments of each particle in the seven neighboring voxels

For a better understanding, we describe using an example in 2D (five neighboring cells). The idea can be generalized to the 3D case in a straightforward manner. In Figure 4.9, particles A and B are neighbors. Our method does the segment collision test between their two adjacent segments, i.e., pairs of segments (a, g) , (a, f) , (b, g) and (b, f) . If two segments have an intersection, there is definitely a pair of their both ends' particles residing in each other seven neighboring cells. This can be easily proved, if one writes all possible cases in 2D with five neighboring cells (center, up, down, left and right).

The closest points of a pair of colliding segments i and j are indicated by fractions $s \in [0, 1]$ and $t \in [0, 1]$, respectively.

$$\mathbf{x}_{i,s} = \mathbf{x}_i + s(\mathbf{x}_{i+1} - \mathbf{x}_i), \quad (4.13)$$

$$\mathbf{x}_{j,t} = \mathbf{x}_j + t(\mathbf{x}_{j+1} - \mathbf{x}_j). \quad (4.14)$$

In order to move the colliding segments to the non-intersection positions, we compute a penalty force between the closest points as follows.

$$\mathbf{F}_{ij} = k_{pn}(d_{ij} - \|\mathbf{x}_{i,s} - \mathbf{x}_{j,t}\|) \frac{\mathbf{x}_{i,s} - \mathbf{x}_{j,t}}{\|\mathbf{x}_{i,s} - \mathbf{x}_{j,t}\|}, \quad (4.15)$$

where k_{pn} and d_{ij} are a penalty force coefficient and a penetration depth between the segments, respectively.

Then, we add the penalty force to the both-end particles of each segment corresponding to the fractions s and t as follows.

$$\mathbf{F}_i \leftarrow \mathbf{F}_i + (1 - s)\mathbf{F}_{ij}, \quad (4.16)$$

$$\mathbf{F}_{i+1} \leftarrow \mathbf{F}_{i+1} + s\mathbf{F}_{ij}, \quad (4.17)$$

$$\mathbf{F}_j \leftarrow \mathbf{F}_j - (1 - t)\mathbf{F}_{ij}, \quad (4.18)$$

$$\mathbf{F}_{j+1} \leftarrow \mathbf{F}_{j+1} - t\mathbf{F}_{ij}. \quad (4.19)$$

For hair-head interactions, the head model is represented by a set of large spheres, and a collision force with each hair particle is calculated as a penalty force between spheres.

4.6 GPU Implementation

For the GPU computation, the physical values of particles are stored in 2D textures in the GPU memory. One texel of each texture has four channels of color; that is red(R), green(G), blue(B) and alpha(A). Instead of the RGBA color data,

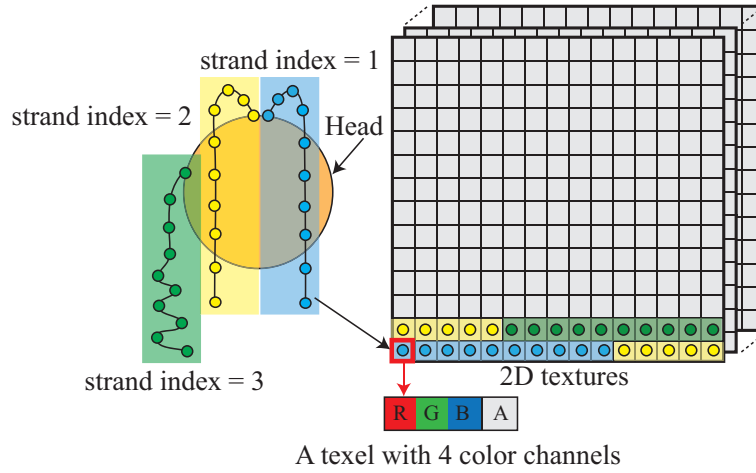


Figure 4.10: Physical values of particles are stored in textures (up to four values per texel). Particles of each hair strand are distinguished by the strand index. These are shown by different colors in this figure.

a floating-point value can be stored in each channel of a texel. Therefore, the proposed method can store up to four physical values of particles into a texel. Our implementation uses ten textures, i.e., two position textures, two velocity textures, one property texture (region half-width w , particle index, strand index and non-stretched length), one original position texture, one texture for optimal translation and three textures for optimal rotation (3×3 matrix, 9 components). The position, velocity, optimal translation and optimal rotation textures require only three channels each, therefore, RGB textures are enough for these textures. Strand indices (see Figure 4.10) are required to distinguish strands because values for all particles are stored in the same textures, and each strand can contain a different number of particles. In addition, our implementation requires one bucket texture, where each texel represents a voxel that stores particle indices for nearest-neighbor searches in the collision detection process.

In each simulation time step, five passes of the GPU computation are assigned (see the black rounded rectangles in Figure 4.11). The detail of each GPU pass is explained together with corresponding line numbers of Algorithm 1 as follows:

Passes 1 and 2 : Position update and collision detection (lines 5-12) First, the bucket texture is generated for nearest-neighbor searches using a GPGPU technique [32]. The velocity of each particle is updated according to external forces such as gravity and penalty forces calculated in the collision detection process (Section 4.5). Then, the position of each particle is updated according to the updated velocity and the time step.

Pass 3 : Chain shape matching (lines 13-16) The optimal rigid transformation of each chain region is computed from the particle positions in the region. Hair particles are stored sequentially in a texture. See Figure 4.12 for an example of the texture layout. Particle a has a half-width size $w = 3$. Particles contained in the region can be found in three texels to the left and right hand sides. In the case of particle b with a half-width size $w = 5$, five adjacent particles can be found on the right hand side, while the last two particles are found in the next row by computing their addresses with the texture width. However, the last three particles on the left hand side have a different strand index. Therefore, only two adjacent particles can be found. The region doesn't have to contain a maximal number of particles (eleven in case of $w = 5$).

Pass 4 : Goal position computation (lines 17-19) After the optimal translation and rotation computation, the goal position of each particle is computed by averaging the goal positions of overlapping regions (Eq.(3.10)). This process can be computed in a similar way to the chain shape matching pass (Pass 3). Instead of particle positions, the computed optimal translation and rotation of the overlapping regions can also be read from the adjacent texels.

Pass 5 : Strain Limiting (lines 20-24) The final pass is the strain limiting process (Section 4.2). The non-stretched position of each particle can be computed from the length starting from the root particle of the strand to the particle (see particle c in Figure 4.12). After updating particle positions, the velocities are also updated in this pass.

4.7 Results

This section shows simulation results of various behaviors of a single strand and full head of hair. Then, this section discusses about parameters setting and some limitations of the proposed model.

4.7.1 Simulation Results

The prototype implementation was written in C++, using OpenGL and GLSL. All experiments were conducted on a PC with an Intel Core i7 3.20GHz, 6GB RAM and an NVIDIA GeForce GTX 480 graphics card. For the hair simulation, the structure of a strand is rendered as connected line segments between particles, and the visual quality is enhanced by Catmull-Rom splines on the GPU using the instanced tessellation technique [11]. As for the shading and self shadowing of hair, we used the Kajiya-Kay shading model [39] and Deep Opacity Maps[102], respectively. All simulation and rendering were entirely conducted on the GPU. The frame rates in this paper include both simulation and rendering. We perform only a single simulation step per frame.

Figure 4.13 shows the result of hair simulation with and without strain limiting described in Section 4.2. The hair strands are stretched due to the external forces without strain limiting. Figure 4.14 shows the result with different numbers of strands on the head. With more strands the visual quality is increased.

Figure 4.15 demonstrates the twisting effects in our model. An application for hanging boxes is presented in Figure 4.15(a), where objects at the tips of strands are rotated by wind forces making the strands twisted. With twisting effects, the strands try to twist back to the initial state, making the rotational velocities increased and the objects rolling back and forth in the wind. The twisting of strands can reproduce phenomena such as an instability of bending and twisting called *buckling* which makes a strand to form a spiral shape (Figures 4.15(b) and 4.15(c)). Figures 4.15(d) and 4.15(e) show the twisting of strands

with uniform and non-uniform torsional rigidities, respectively. In the strand with non-uniform torsional rigidity (thicker at the middle of the strand in this result), the thicker part has a larger torsional rigidity, and therefore has less twisting.

Figure 4.16 shows plasticity handling of strands in our model. When an applied stress from a force or torque passes a yield point, a strand will irreversibly deform; this means the strand will not return to its original length or twisting angle. In the top row, we compare two strands under the applied forces below (left strand) and over (right strand) the yield point. Likewise, two strands in the bottom row show a comparison under the different applied torques. It can be observed both in top and bottom rows that the strand on the left returns to its original shape, while the strand on the right is permanently deformed.

Animation sequences of flicking are shown in Figure 4.17. Without flicking, the strand in Figure 4.17(a) falls naturally when an applied force is removed. In our model, the strand bounces back by the estimated tensions when the applied force is removed as shown in Figure 4.17(b). When the twisted strand in Figure 4.17(c) is pulled and released, the twisting effect also occurs.

To demonstrate the practical uses of our method, Figures 4.18 and 4.19 show applications in an animation and game. Figure 4.18 shows a destruction of a hanging bridge. Wooden boards (rigid bodies) are tied with strands (ropes in this case) to build the bridge. The ropes are gradually torn apart from collisions of the wooden boards and incoming crates that cause high tensions in the ropes. We used a particle-based simulation method [32] for rigid body simulation in our implementation. Figure 4.19 shows a twisting games. A box is hanging to a wooden beam by a strand. The rule of the games is to shoot the box and make the strand twisted until breaking.

The breakdown computational time in each process for strands with a different numbers of particles is shown in Table 6.1. All strands consist of 100 segments, except 150 segments in Figure 4.17, 200 segments in Figure 4.15c and 746 segments in Figure 4.18. The computational time of the results in Fig-

Table 4.1: The computational time in milliseconds of each process in one time step. The time step in our implementation is 10 milliseconds.

No. of segments	Updating particles	CSM	Twisting computation	Tension estimation	Collision handling	Total time
100	0.011	0.086	0.098	0.221	1.31	1.73
150	0.022	0.168	0.184	0.424	1.36	2.16
200	0.029	0.221	0.237	0.564	1.37	2.42
746	0.128	0.501	0.739	1.67	3.13	6.17

ures 4.18 and 4.19 is measured excluding the time for rigid body simulation.

Figure 4.20 shows animation sequences of straight, curly and complex hairstyles flowing in the wind. Each strand of the complex hairstyle is straight around the top and curly around the bottom. Each scene consists of 10,000 strands (160,000 particles), 10,000 strands (580,000 particles) and 17,000 strands (764,000 particles), respectively.

Figure 4.21 shows animation sequences of a head with 120,000 roots of hair strands which is approximately a number of human hair strands in the real world. However, only 80,000 strands that are visible during the animation are simulated. The simulation was run on a CPU, because the current GPU memory is insufficient for such a large number of hair strands. Apart from other results, this result was rendered using an off-line rendering software, POV-Ray 3.7 [67].

The breakdown computational time used in each GPU pass and rendering for each result is shown in Figure 4.22. The simulation and rendering speeds of each sequence are 12, 7 and 4 fps, respectively.

4.7.2 Parameters Setting

To indicate the bending stiffness of hair strand in real world, the elastic modulus such as Young’s modulus could be used. However, the measurement of the elastic modulus of hair is difficult. When a user wants to simulate the particular type of hair in hand, measuring the elastic modulus of a very small hair strand

requires highly precise measurement and experimental settings. There are many measurement methods in textile literature [61] that are applicable for a hair strand such as treating a short hair strand as a cantilever beam center-loaded and loop deformation of a hair strand.

Instead of the elastic modulus, Scott et al. [79] introduced a *stiffness index* to represent the stiffness of hair with a simpler measurement. A hair strand is attached with small plastic tubes (0.1g each) at both ends of the strand and draped over a wire hook. Then, the stiffness index is measured from the distance between two legs of the hair strand in centimeter. Finally the elastic modulus of bending E_B can be calculated as follows.

$$E_B = \frac{\pi T D^2}{2A^2}, \quad (4.20)$$

where T is the applied force, D is the stiffness index and A is linear density of a hair strand in g/cm units.

The stiffness of hair strand simulated by CSM is proportional to the chain region's size which is an integer. Therefore, the modification of the stiffness index of hair in CSM is in a discrete way. As shown in Figure 4.23, the stiffness index is discretely increased according to the chain region size. From our experiments, the stiffness index also depends on the segment length. With the same chain region's size, the stiffness index is linearly increased as shown in Figure 4.24. Therefore, the desired stiffness index of the hair strand can be achieved by the adjustment of chain region size and segment length. For example, to achieve a stiffness index of 8 cm ($D = 8$), several combinations of (*chain region size, segment length*) can be applied by using the experimental data in Figure 4.24. At $D = 8$ on the y -axis, the combinations such as (2, 2.2cm), (3, 1.6cm), (4, 1.25cm) and (5, 1.1cm) can be read from the graph. Each combination achieves the same stiffness index, however, the resolution of strands is different as shown in Figure 4.25. The strand appears smoother with a smaller segment length.

The proposed method also enables simulation of strands with various material

properties, i.e., tensile strength, plasticity and torsional strength, using the stress-strain curve. Figure 4.26 shows the variation tests. The stress-strain curves are shown in the top row, and their corresponding results are shown as animation sequences below each curve. Parameters used in tearing, i.e., rupture points, yield points and Young's modulus, are assigned to all segments in the strand. However, that kind of completely uniform strength is impossible in the real strand, so we randomly altered the parameters in each segment with the range of variation up to 0.01%. To demonstrate the effect of parameters setting, three experiments were conducted as follows:

- Tensile rupture point variation (Figure 4.26(a)): Tensile rupture points of strands are varying, increasing from strand numbers 1 to 4. As expected, the topmost strand (no.1), which has the lowest rupture point, is torn first.
- Young's modulus variation (Figure 4.26(b)): Young's modulus is a measure of the elasticity of material. Since values of Young's modulus of strands in this test are lessened from numbers 1 to 4 while the applied stresses required for breaking the strands are equal, the bottommost strand (no.4) is lengthened most before breaking.
- Torsional rupture point variation (Figure 4.26(c)): Similar to Figure 4.26(a), torsional rupture points are varying, increasing from strand numbers 1 to 4. When the strands are twisted by the applied torques on the right hand side, the topmost strand (no.1), which has the lowest torsional rupture point, is torn first.

4.7.3 Limitations

There are some limitations. As previously mentioned, the proposed method is not a full physically-based model, thus, more advanced physics behaviors such as spring-twisting pendulum and anisotropic bending in [8] are hard to simulate.

The rapid motion of strands could cause the strands to pass through each other or themselves. In case of rapid motion, continuous collision detection should be considered.

4.8 Summary

This chapter has presented a simple model for simulating a strand based on shape matching as well as a model for twisting, tearing and flicking of strands, which is fast and easy to implement. This chapter has demonstrated the the proposed method can handle twisting effects of strands with both uniform and non-uniform torsional rigidities as well as rotational velocity caused by twisting. The tension in an inextensible strand can be estimated for generating tearing and flicking effects. The tearing effects in twisting have also been enabled. A variation in the quality of strands, i.e., elasticity and plasticity, can be achieved. Based on the proposed individual strand model, visually-plausible animations of hair with complex hairstyles can be archived in a numerically stable way, even for highly stiff and curly hair like an afro. This chapter has also demonstrated that a GPU-based simulator can achieve interactive performance up to several ten thousand hair strands.

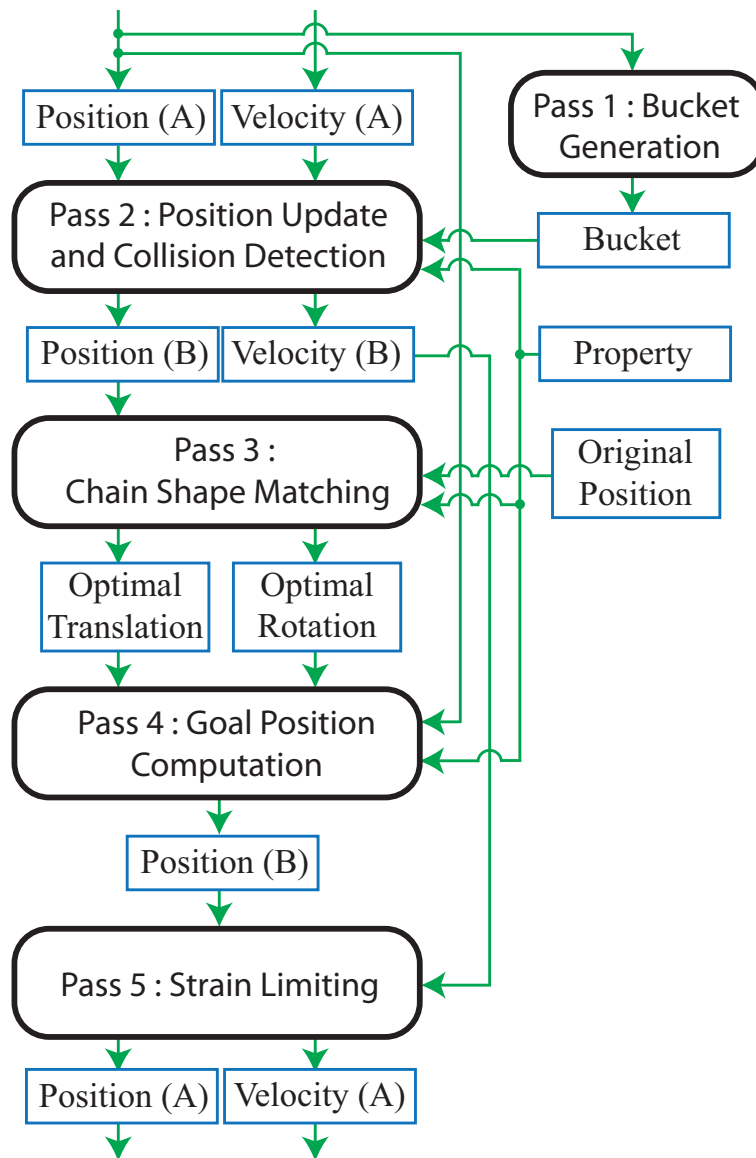


Figure 4.11: Simulation flow of a single step on the GPU. Blue rectangles represent the texture data, black rounded rectangles represent operations and green directed line segments represent the flow of data.

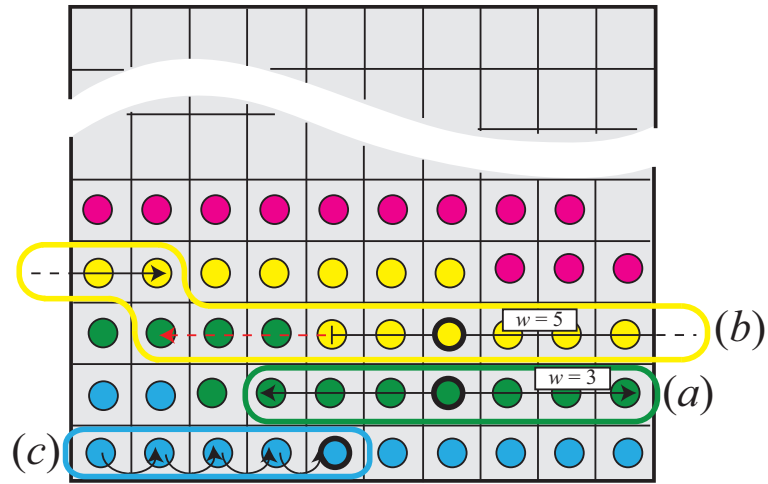


Figure 4.12: Layout of hair particles on a texture memory. (a) The region half-width $w = 3$. The particles in the region can be found in the adjacent texels. (b) The region half-width $w = 5$. All five adjacent particles in the right half can be found. The left half has only two adjacent particles in a strand. (c) Access pattern for strain limiting. The non-stretched position is computed by tracing the length of each segment from the root particle.

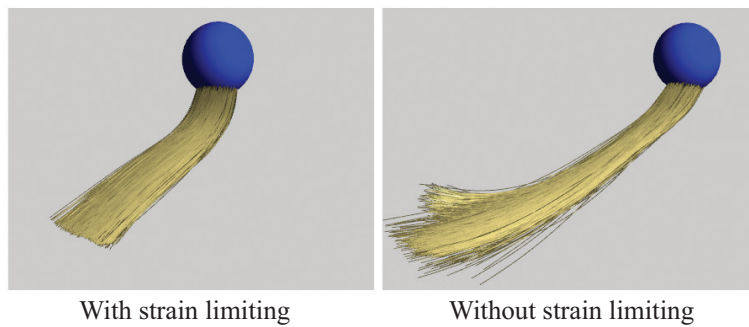


Figure 4.13: Moving bunch of hair strands with (left) and without (right) strain limiting.



Figure 4.14: Increasing hair strands greatly improves the visual quality. From left to right, there are 1,250, 5,000 and 10,000 hair strands on the head.

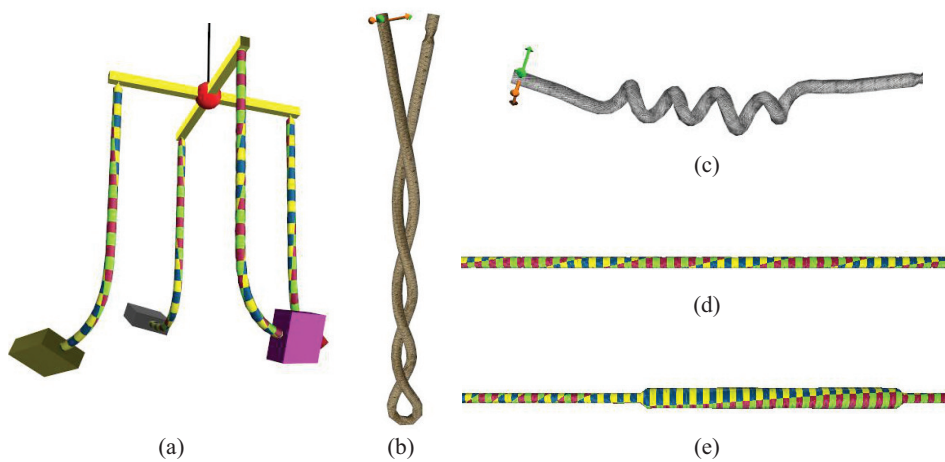


Figure 4.15: Our simulation results of twisting effects. (a) An application for hanging boxes in wind forces. (b) The twisting effect of a strand clamped at both ends. The strand is gradually twisted on the left end and finally twisted to form a loop. (c) A twisted strand that forms a spiral shape like a telephone cord. (d) A strand with uniform torsional rigidity. (e) A strand with non-uniform torsional rigidity.

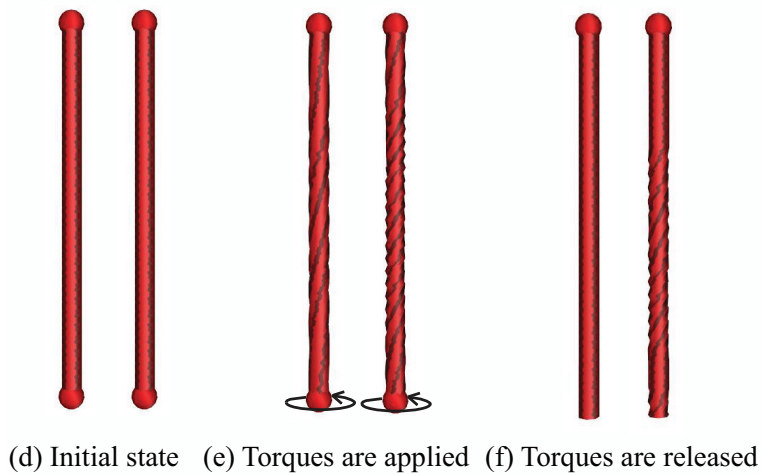
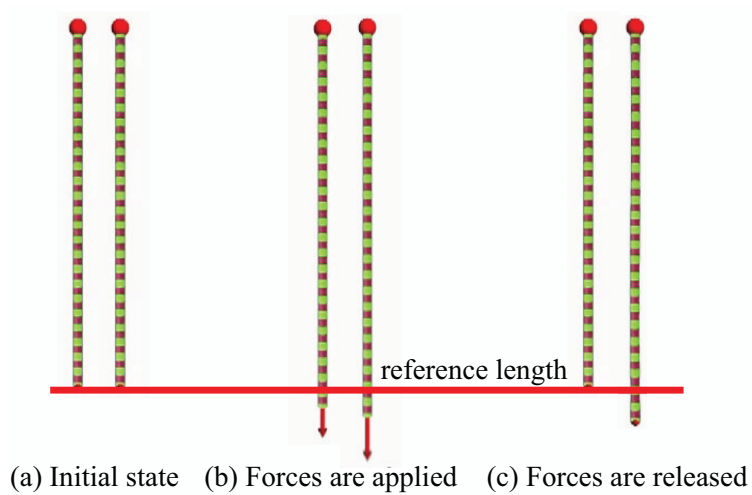


Figure 4.16: Plasticity of the material of extensible strands under tensile and torsional stresses. (a) and (d) are the states before forces and torques are applied in (b) and (d), respectively. In (b) and (d), the strands on the left do not pass the yield points, while the strands on the right do. (c) and (e) are the state after the forces and torques are released.

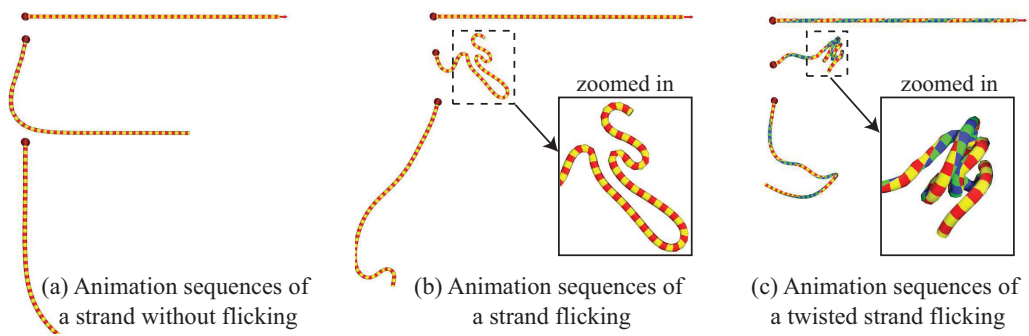


Figure 4.17: Flicking animation sequences of strands from top to bottom.



Figure 4.18: Animation sequences of a hanging bridge colliding with incoming crates.

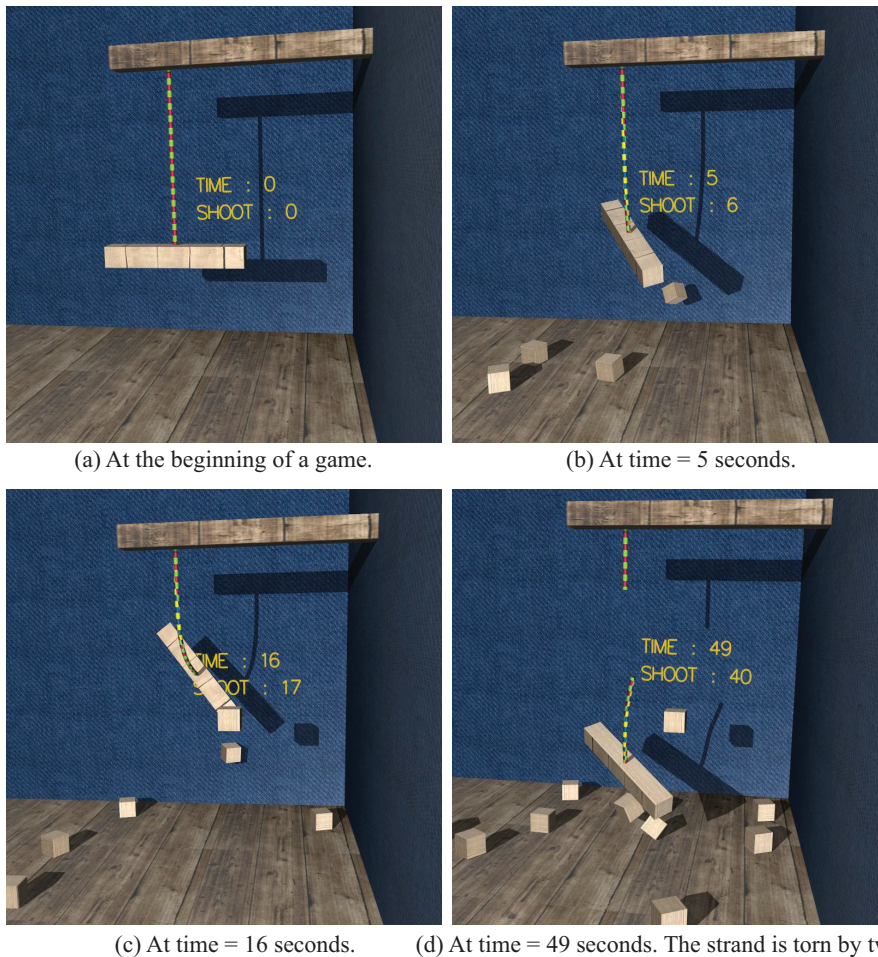


Figure 4.19: A game application using our method. The goal of this game is to shoot a box and make a strand twists until it is torn apart. The more it is twisted the harder it will twist back. Therefore if a player misses, the bar will decrease the twisting angles the player have hit so far. Players can compete the time and the number of boxes they used with others.

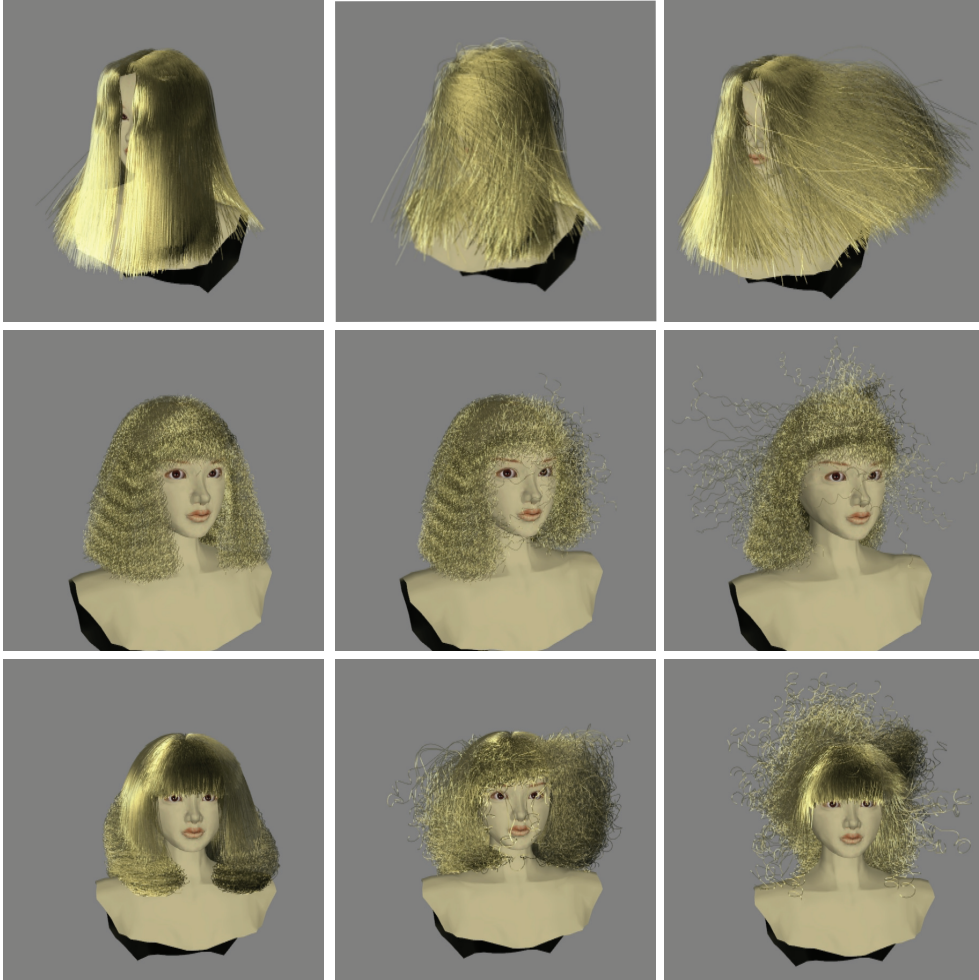


Figure 4.20: Animation sequences of straight (top row, 12 fps), curly (middle row, 7 fps) and complex hairstyles (bottom row, 4 fps) in the wind. There are 10,000 strands (160,000 particles), 10,000 strands (580,000 particles) and 23,000 strands (764,000 particles), respectively. The stiffness configurations of the straight and curly hair are $w = 2$ and $w = 5$. Each strand of the complex hairstyle is straight near the roots and curly near the tips with chain region half-widths of $w = 2$ and $w = 6$, respectively.



Figure 4.21: Animation sequences of 80,000 straight hair strands (1,600,000 particles). The stiffness configuration is $w = 3$.

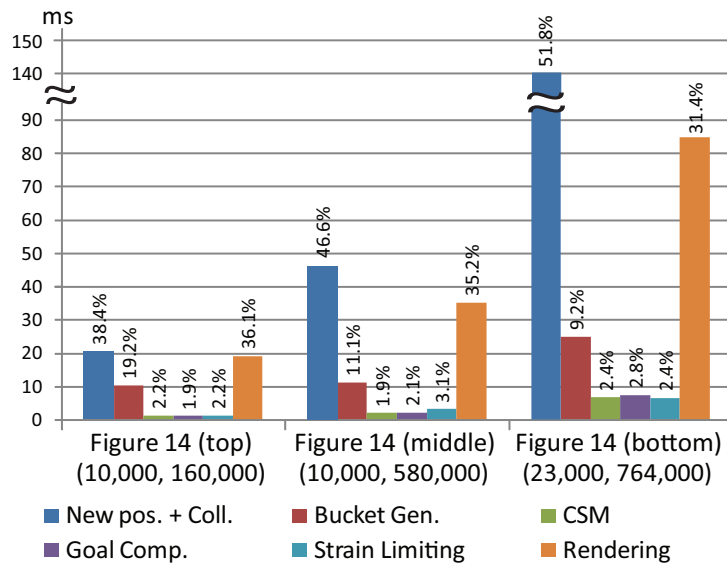


Figure 4.22: The computational time in milliseconds used in each GPU pass and rendering. The detail of each GPU pass is described in Section 5.6. The numbers of hair strands and particles used in each result are shown as (no. of strands, no. of particles).

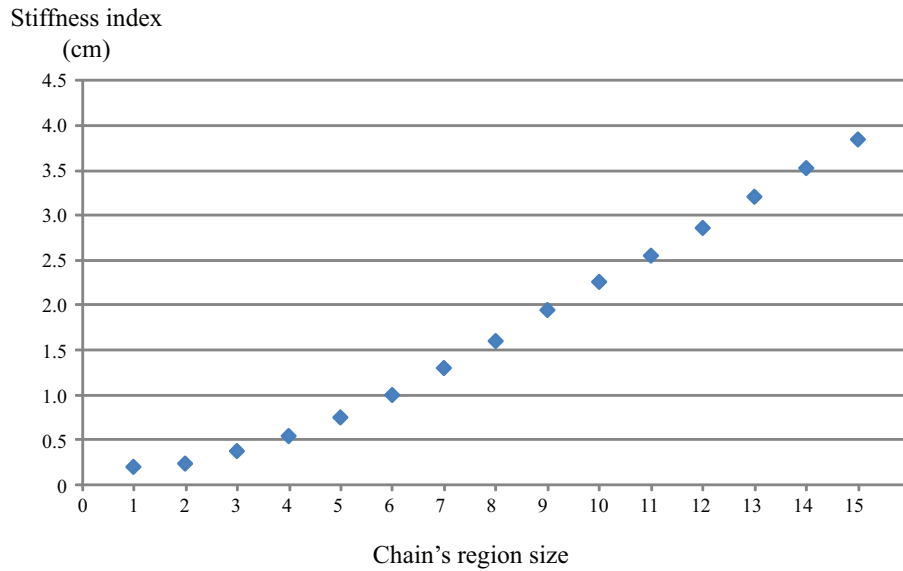


Figure 4.23: A graph shows stiffness indices of hair strands with different chain region sizes. The segment length in this experiment is 1 cm.

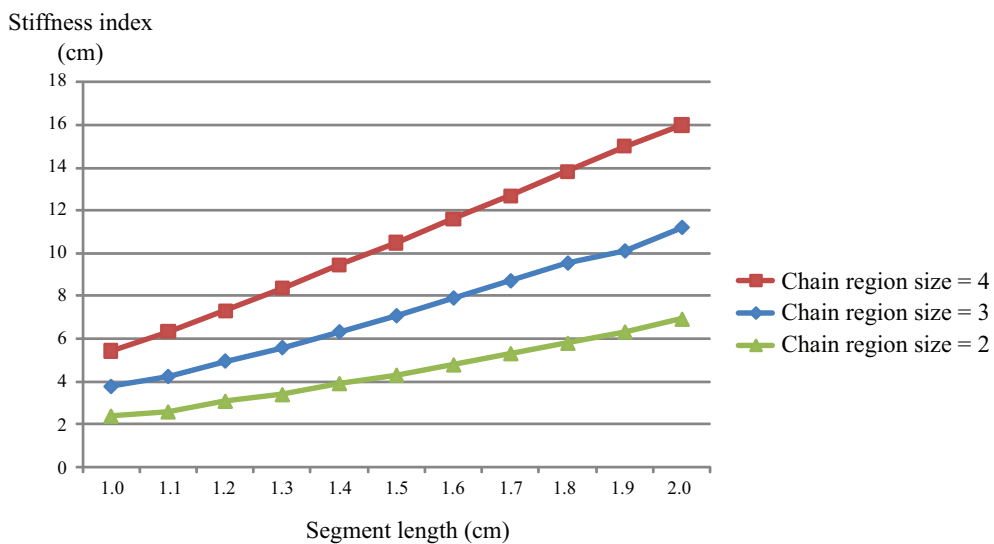


Figure 4.24: A graph shows stiffness indices of hair strands with different segment lengths.

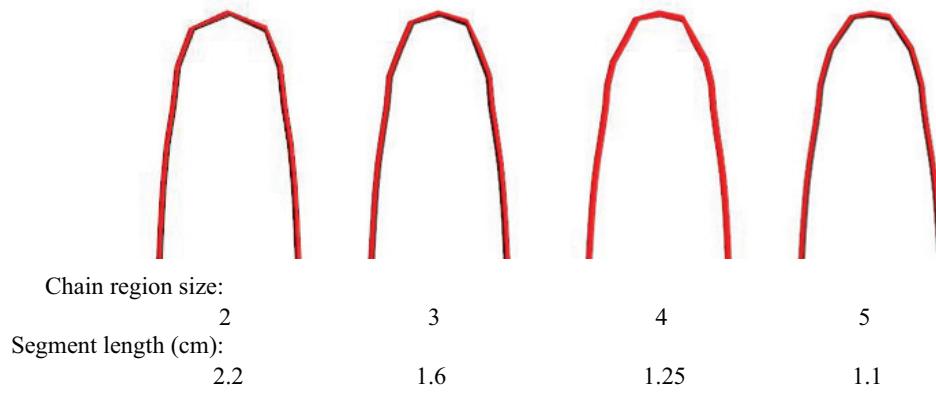


Figure 4.25: Strands have the same stiffness index, but different adjustment of segment length and chain region size.

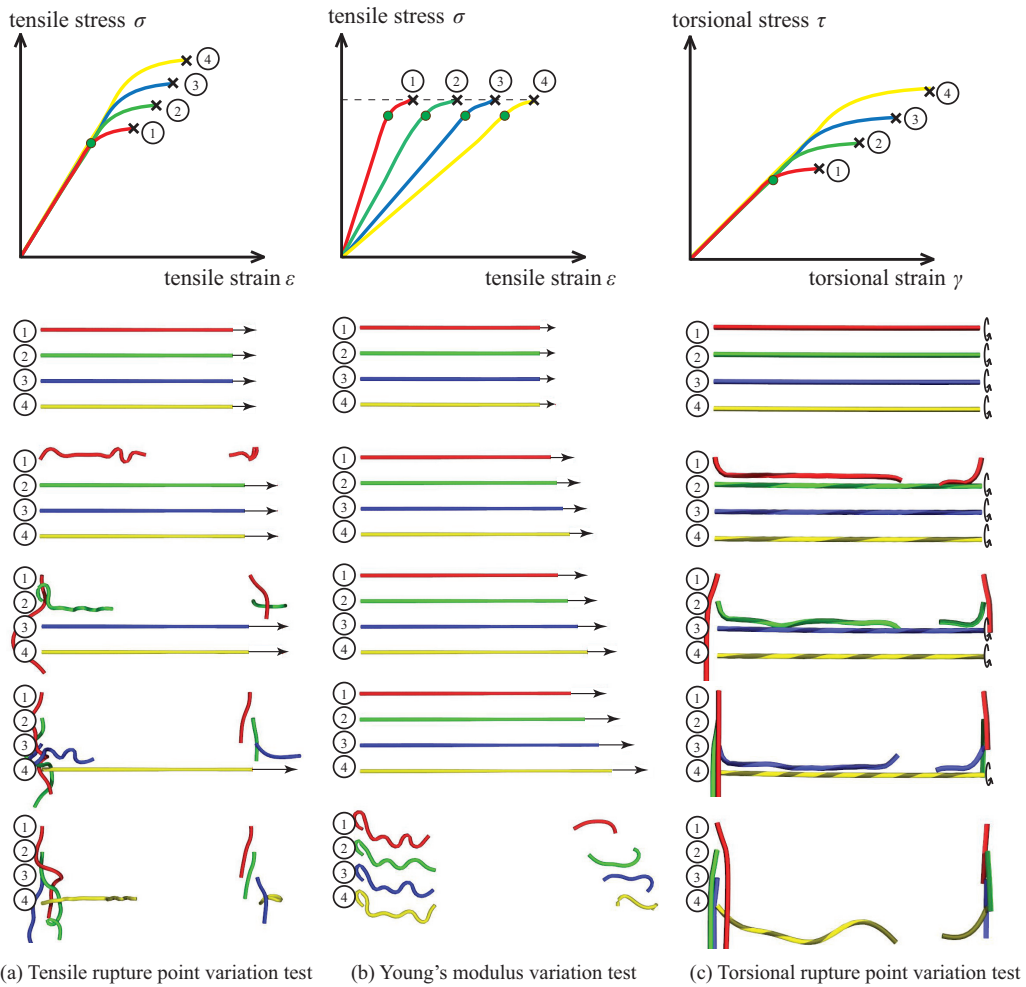


Figure 4.26: Experiments of various stress-strain curves and curves' results. A variety of stress-strain curves are shown in the top row and their results are shown as animation sequences from top to bottom.

Chapter 5

Wetting Effects in Granular Materials

This chapter describes a method for computing the interactions between fluids and granular materials. First, a physical mechanism for the propagation of wetness is introduced, followed by a presentation of the outline of the proposed method, and then the details are finally described.

5.1 Overview

In the real world, most materials are permeable; when the object comes into contact with fluids, the absorption and propagation of the fluids happen. The mechanism behind this wetting phenomenon is mainly caused by the capillary action. The capillary action is the result of surface tension and adhesion which are intermolecular attraction within the fluid and solid materials. To demonstrate this phenomenon, a capillary tube, a very thin glass tube with an internal bore, is commonly used. If one end of the tube is placed into fluid, the fluid will penetrate into the bore (Figure 5.1). When withdrawing the tube from the fluid, some amount of the fluid is trapped in the tube. This phenomenon can be observed in such other small open spaces between two solid materials, such as small gaps between granular particles.

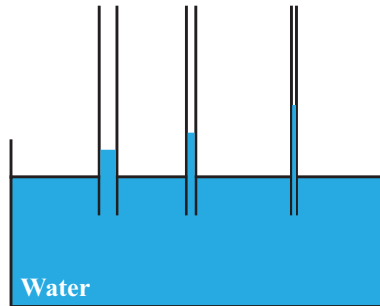
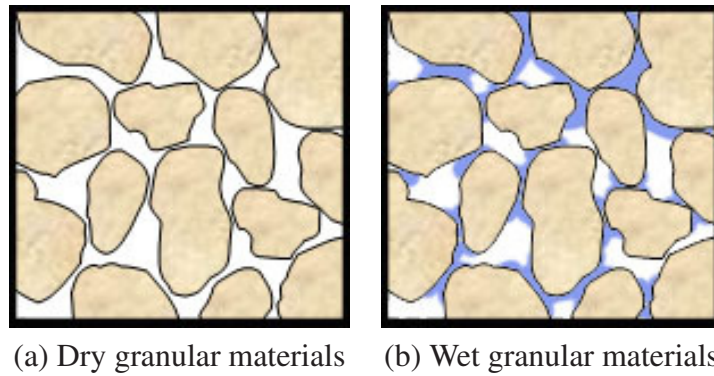


Figure 5.1: The capillary action demonstrated by capillary tubes.



(a) Dry granular materials (b) Wet granular materials

Figure 5.2: The microstructure of granular materials. The small gaps between granular particles that are small enough to have the capillary action or the capability to hold fluid.

A pile consisting of granular materials includes a vast number of small open spaces between the individual particles (Figure 5.2(a)). When such a pile comes into contact with a fluid, wetness is absorbed into the spaces and propagates through the material, mainly induced by capillary forces (Figure 5.2(b)). Similar descriptions can be found in studies of the weathering of stones [22] and on-surface flows [51].

Wetness among granular particles forms structures called *liquid bridges* (Figure 5.3) due to the surface tension of the liquid. These liquid bridges induce grain-to-grain attractive forces and strengthen the cohesion of the material. They are therefore essential for the construction of sand castles. The force yielded by

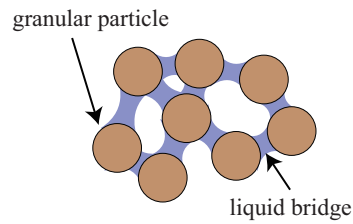


Figure 5.3: Liquid bridges. For illustration purpose, the distances between particles are exaggerated.

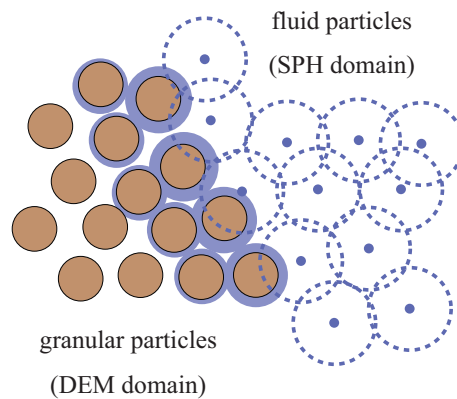


Figure 5.4: Overview of the proposed method. Wetness is provided by fluid particles, and then propagates through granular particles.

a liquid bridge can be computed in an extremely simple case (i.e., two spheres only) using a theoretical formula that agrees well with experimental data. However, in cases where there are many granular particles, the liquid-bridge forces are difficult to consider because their shapes become complicated, and, to the best of our knowledge, there are no reasonable theoretical models available so far. Please refer to the paper [36] for recent advances in the physics of wet granular materials.

This dissertation presents a simple, empirical model for the propagation of wetness and the forces yielded by wetness. In the proposed method, each granular particle is regarded as spherical, and has an individual wetness value. The proposed method assumes that the intensity of the liquid-bridge forces reduce linearly with regard to wetness. The proposed method also assumes that, in the

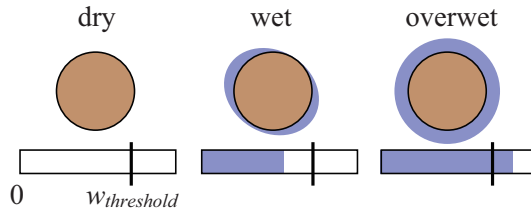


Figure 5.5: Terms according to the wetness value. Each bar indicates the wetness value.

propagation of wetness, gravity has much less influence compared to the capillary forces and thus can be ignored. The proposed method computes the interactions of a granular particle with fluid particles and other granular particles, according to its wetness (Figure 5.4):

Interactions with fluid particles: Inter-particle forces are computed based on SPH. Then, if the wetness value of the granular particle **does not reach** a maximum wetness value, the granular particle receives wetness from the fluid particles, and the fluid particles disappear (Section 5.3).

Interactions with other granular particles: Attractive forces yielded by liquid bridges are computed in addition to the forces used in DEM. Then, if the wetness value of the granular particle **exceeds** a threshold, the excessive wetness is distributed to those neighboring particles whose wetness values are below the threshold (Section 5.4).

Additionally, we control the propagation speed of wetness among granular particles (Section 5.5).

5.2 Wetting Model for a Granular Particle

In our model, each granular particle i has a wetness value $w_i \in [0, w_{max}]$. We refer to a granular particle by different terms according to w_i (Figure 5.5):

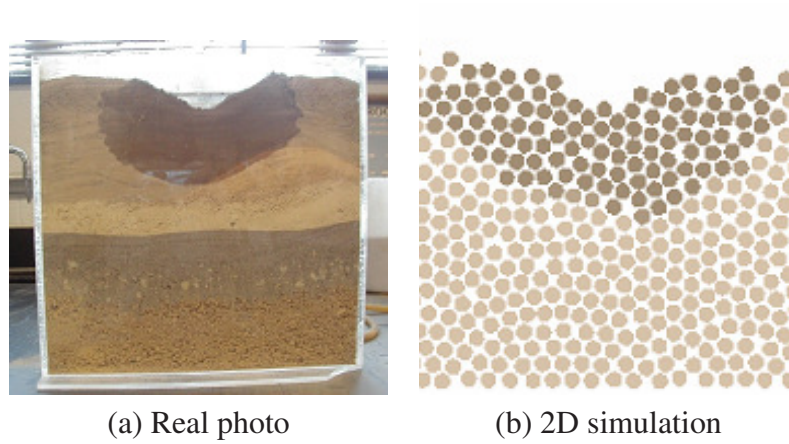


Figure 5.6: 2D comparison in which a flat surface is depressed due to moisture absorption. Photograph courtesy of Daniel D. Fritton and Katharine L. Butler.

dry particle: $w_i = 0$

wet particle: $0 < w_i \leq w_{threshold}$

overwet particle: $w_{threshold} < w_i \leq w_{max}$

where $w_{threshold}$ is a threshold of wetness. Dry or wet particles receive wetness from fluid particles or overwet particles at the time of contact. The wetness value is used to compute the grain-to-grain attractive forces. In addition, in order to represent the aggregation of wet granular materials, the proposed method shrinks the radius r_i of a granular particle i according to the wetness value:

$$r_i = R - k_r w_i, \quad (5.1)$$

where R is the base radius used in DEM, and k_r is a coefficient. This modification allows us to represent the depression of wet surfaces composed of granular materials. Figure 5.6 shows a 2D comparison between a real photo and a simulated result.

5.3 Interactions between Fluid and Granular Particles

When fluid particle i collides with granular particle j , the proposed method first computes the inter-particle forces. The forces for fluid particle i are computed based on SPH (Section 3.1) by regarding granular particle j as a fluid particle that has a constant density and pressure, and by modifying Eqs. (3.4) and (3.5):

$$\mathbf{F}_{ij}^{pressure} = -\frac{V_i V_j}{4}(p_i + p_j)\nabla W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (5.2)$$

$$\mathbf{F}_{ij}^{viscosity} = \mu \frac{V_i V_j}{2} \nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (5.3)$$

where $V_i = m_i/\rho_i$ is the volume of fluid particle i , V_j is the volume of the granular particle, and p_g is a constant pressure. For granular particle j , $-\mathbf{F}_{ij}^{pressure}$ and $-\mathbf{F}_{ij}^{viscosity}$ are added as external forces.

The proposed method assumes that a fluid particle has a wetness value w_{fluid} . After the computation of forces, if there are dry or wet particles in the vicinity of the fluid particle, the fluid particle is absorbed by them. That is, the fluid particle equally distributes its wetness value w_{fluid} to the dry or wet particles, then disappears.

5.4 Interactions among Granular Particles

The proposed method modifies the computation of forces in DEM (Section 3.2), accounting for the amount of wetness. When granular particles i, j collide with each other, the inter-particle forces are computed by modifying Eqs. (3.8) and (3.9):

$$\mathbf{F}_i^{damper} = k_d(1 + w_i + w_j)\mathbf{v}_{ij}^{normal}, \quad (5.4)$$

$$\mathbf{F}_i^{tangential} = k_t(1 + w_i + w_j) \frac{\mathbf{v}_{ij}^{tangential}}{\|\mathbf{v}_{ij}^{tangential}\|}. \quad (5.5)$$

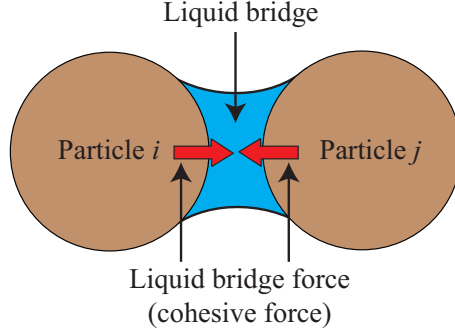


Figure 5.7: The inter-particle force called liquid bridge force between two wet granular particles.

Additionally, the liquid-bridge force \mathbf{F}_i^{bridge} is computed. \mathbf{F}_i^{bridge} is designed to work between wet particles only ($w_i + w_j > 0$), and to reduce as the wetness increases:

$$\mathbf{F}_i^{bridge} = k_{bridge} \max\left\{0, w_f - \frac{w_i + w_j}{2}\right\} (\mathbf{v}_j - \mathbf{v}_i), \quad (5.6)$$

where k_{bridge} is a coefficient and w_f is a threshold for fluidization. The proposed method uses \mathbf{F}_i^{bridge} only when the particles are moving away from each other, that is, $(\mathbf{v}_j - \mathbf{v}_i) \cdot (\mathbf{x}_j - \mathbf{x}_i) > 0$.

After computing the forces, if a granular particle i is overwet, i.e. $w_i > w_{threshold}$, and if there are dry or wet particles in its vicinity, the excessive wetness value $\Delta w = w_i - w_{threshold}$ is distributed equally among them.

5.5 Control of Propagation Speed

To control the speed of wetness propagating among granular particles, the proposed method introduces a coefficient k_p for the propagation rate. Let w_i^t be the wetness value of particle i at time t and Δw_i^t be the excessive wetness of particle i at time t (i.e., $\Delta w_i^t = w_i^t - w_{threshold}$). The proposed method assumes that the propagation speed of wetness from particle i to neighboring particles j ($j = 1, 2, \dots, N_i$) exponentially decreases according to the excessive wetness

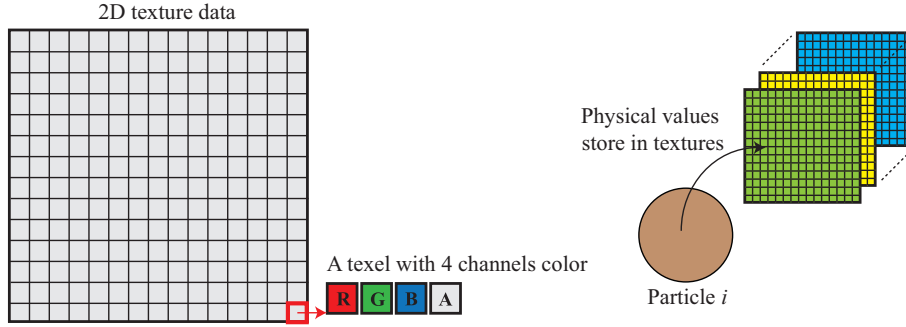


Figure 5.8: 2D texture data and each texel's 4 channels color. The physical values of particles are stored in the textures.

Δw_i^t :

$$w_j^{t+1} = w_j^t + k_p \frac{\Delta w_i^t}{N_i} \Delta t, \quad (5.7)$$

where Δt is the timestep. Larger k_p yields faster propagation of wetness while smaller k_p results in slower propagation.

5.6 GPU Implementation

With modern graphics hardware, many computational processes could be performed on the GPU (Graphics Processing Unit), especially simulation processes. Harada *et al.* [33] have demonstrated that the particle-based simulation system could be entirely implemented on the GPU. The speed-up of computation time is the main advantage of the GPU implementation. Since the proposed method is also based on the particle system, the GPU implementation technique can be used.

For the GPU computation, the physical values of particles, both in DEM and SPH domains, are stored as 2D textures in the GPU memory. The proposed method stores both DEM and SPH particles in the same texture and distinguishes by particle domain flags. In the implementation of this dissertation, the particle domains is set to '1' for DEM particles and '2' for SPH particles. Each particle occupies one texel of the 32bit floating-point RGBA textures (Figure 5.8).

Several computational processes need to read the values and write back into the texture. However, GPUs cannot read and write a single texture at once without data hazard, and thus two textures (read and write textures) are required. In the proposed method, read and write textures are required in the computation of positions, velocities and particle properties. On the other hand, only a single texture is required for each computation of forces and fluid properties because they are not updated in the same computational pass. Consequently, the prototype implementation uses nine textures as follows (Table 5.1):

- two textures (read and write textures) for positions; the RGB channels store position data in xyz coordinates and the alpha channel stores a particle's lifetime that indicates the particle's life in a scene.
- two textures (read and write textures) for velocities; the RGB channels store velocity data in xyz coordinates.
- two textures (read and write textures) for particle's properties; radius, wetness value, SPH particle's deleted flag and a flag for the particle domain.
- one texture for the computed force of each particle; the force in each coordinate is stored in the RGB channels.
- one texture for fluid particle properties; the RGB channels store density, volume and pressure of a SPH particle. The alpha channel is used for storing the number of propagate neighboring particles.
- one texture for a bucket data structure required for nearest-neighbor searches.

In each simulation time step, four passes of the GPU computation is assigned. See blue rectangles in Figure 5.9. Firstly, the proposed method updates the velocity of each particle influenced by the external forces such as the gravity and collision forces. Then the position of each particle is updated according to the updated

Table 5.1: Physical parameters stored in the GPU memory as textures. There are five sets of physical parameters of particles stored in eight 2D textures. Note that another texture is required for nearest neighbor searches. $x, y, z, v_x, v_y, v_z, F_x, F_y$ and F_z are the position, velocity and force data in xyz coordinates, respectively. Positions, velocities and properties textures require two textures each. Properties and fluid properties textures require only one texture each.

Parameter \ Channel	Red	Green	Blue	Alpha
Positions	x	y	z	life time
Velocities	v_x	v_y	v_z	
Properties	r_i	w_i	deleted flag	particle domain
Fluid properties	density	volume	pressure	N_i
Forces	F_x	F_y	F_z	mass

velocity and the time step. From the updated positions, the proposed method generates a bucket texture for nearest-neighbor searches. The 3D simulation space in the proposed method is divided into a uniform grid. The grid is then sliced and represented with a set of 2D bucket textures for the use of nearest-neighbor searches. Each texel of the bucket texture stores four particle indices in its four color channels. Suppose that i_k ($k = 0, 1, 2, 3$) are the indices stored in a texel and $i_0 < i_1 < i_2 < i_3$. These indices are stored in this order using the technique proposed by Harada *et al.* [33]. The bucket texture then can be used in density computation, collision detection, wetness propagation and force computation.

The sizes of these textures, except the bucket texture, depend on the number of particles used in the scene. The size of the bucket texture is linked to the 3D simulation world space's size and the resolution of uniform grids.

For each pair of colliding particles, the forces regarding to their domains are computed (Sections 3.1, 3.2 and 5.3). Then, the wetness propagation among DEM particles is handled (Section 5.5). In the wetness propagation, the number of propagating neighbors N_i has to be known (Eq.(5.7)). Before the wetness value computation pass, the proposed method can employ this process into the SPH computation pass to avoid one more pass for costly nearest-neighbor searches on

the GPU. While SPH particles compute the density, pressure and volume from SPH neighbors, the proposed method computes the number of propagating DEM particles N_i by checking whether the wetness should be distributed or not (Section 5.4).

5.7 Results

This section shows several simulation results of the proposed method. Then, this section makes discussions on computational time and limitations of the proposed model.

5.7.1 Simulation Results

The prototype implementation was written in C++, using OpenGL, GLSL and Cg. All experiments in this dissertation were conducted on a PC with an Intel Core 2 Quad 3.0GHz processor, 2GB memory and an NVIDIA GeForce GTX280 graphics card.

Figure 5.10 shows the 2D simulation result with and without the wetting effects. The simulation result without wetting effects has only the force interactions between fluids and granular materials. With the wetting effects taken into account, the simulation result becomes more realistic.

Figure 5.11 shows an animation sequence of the wetting front spreading through a sand bed from the interface between fluid and granular materials. The boundary between wet and dry part of granular materials (wetting front) can be observed.

Figures 5.12 and 5.13 show animation sequences consisting of a pile of sand with a water stream emitted from a black faucet. These scenes contain 32,000 and 160,000 granular particles, respectively. The emitted fluid in each scene contains 8,000 and 16,000 fluid particles, respectively, although only a fraction of the fluid particles are actually visible. The frame rate is about 49 fps and 13 fps, respectively.

Figure 5.14 is a comparison of the properties of dry, wet and overwet particles. The modified-DEM forces introduced in Section 5.4 result in sand piles with different heights as the sand particles stack to them. Each scene is simulated at 82 fps, with 32,000 granular particles.

Figure 5.15 shows the interactions between granular particles with a rigid shovel (with 32,000 particles, around 70 fps). The liquid-bridge forces and modified-DEM forces proposed in Sections 5.4 result in different behaviors when the wet and dry granular particles interact with a rigid body.

In Figure 5.16, massive fluid interacts with a sand castle. The result shows a sand castle containing 25,000 particles being destroyed by a massive wave with 70,000 particles. Sand particles of the sand castle are washed away while the wetness is propagated into the structure, simultaneously. The proposed method achieves 13fps speed.

Rigid bodies can be integrated into the simulation framework. A rigid bunny is approximated by a set of particles and rendered as a mesh. Figure 5.17 shows the interaction between granular materials containing 35,000 granular particles, fluids containing 64,000 fluid particles and a rigid bunny. The frame rate is about 12 fps.

For rendering, granular materials are rendered as solid spheres using point sprites, and the visual quality is enhanced by a screen-space ambient occlusion technique similar to Mittring's work [58]. And fluid is rendered as metaballs using the GPU-based ray-casting technique proposed by Kanamori *et, al* [40]. The frame rates of all results includes both simulation and rendering.

The simulation times of the results in Figures 5.12 (a sand pile), 5.15 (a large sand pile), 5.16 (a sand castle) and 5.17 (a bunny with fountains) are shown in Table 5.2. The simulation time is divided into the time used in each pass of the GPU computation and rendering. As shown in the table, the most computational cost pays to density and force computation due to costly nearest-neighbor searches. The rendering time directly depends on the number of fluid particles in

the simulating scene. The rendering is more slower with more fluid particles in the simulating scene.

5.7.2 Parameters Setting

This section describes the effects of parameters related to the behavior of granular material and wetness propagation together with simulation results of different parameters setting.

One of the common properties that characterize behavior of granular materials is the *angle of repose*, the angle between a stable pile of granular material and the ground [13]. Different kinds of granular materials have different internal friction, particles' size and shape which result in different angles of repose. In the proposed model, varying angles of repose can be achieved by adjusting the friction coefficient k_t in Eq.(3.9). As shown in Figure 5.18, the larger the friction coefficient k_t between particles is, the steeper the angle of repose is.

The wetness propagation is controlled with parameters as follows. The threshold of wetness value $w_{threshold}$ determines when the granular particles begin propagating wetness, while the maximum wetness value w_{max} determines when the granular particles stop absorbing fluids. If the wetness value of the granular particle exceeds $w_{threshold}$, the excessive wetness is distributed to neighboring particles with propagation rate k_p . The proposed method uses different propagation rate k_p varied in the surrounding region of the particle. The region surrounding the particle can be divided into upward, downward and lateral regions (Figure 5.19).

In the case of absorbing fluids from the surface, the wetness of the particles at the surface exceed $w_{threshold}$ first, then the exceeded wetness is propagated to downward particles. Therefore, the upward propagations rarely occur. Figure 5.20 shows the simulation results with different portion of the lateral and downward propagation rates, the upward propagation is neglected in this result. As shown in the result, the larger the downward propagation rate is, the deeper the wetting front is. Likewise, the larger the lateral propagation rate is, the wider the wetting

Table 5.2: The computation time in milliseconds of each GPU pass and rendering. The numbers in parentheses are the workload percentage of each pass.

Scene	# of granular particles	# of fluid particles	Simulation time (ms)				Rendering time (ms)
			update	bucket	density	force	
A sand pile	32,000	8,000	0.198 (2.11%)	1.752 (18.74%)	2.672 (28.58%)	4.728 (50.57%)	9.147
A sand castle	25,000	70,000	0.36 (1.35%)	6.494 (24.34%)	7.253 (27.19%)	12.572 (47.12%)	41.39
A bunny with fountains	35,000	64,000	0.373 (1.12%)	6.487 (19.46%)	4.742 (14.22%)	21.741 (65.20%)	42.482
A large sand pile	160,000	16,000	0.374 (0.77%)	6.487 (13.36%)	14.205 (29.26%)	27.483 (56.61%)	20.429

front is.

In Figure 5.20(d) where the downward propagation rate is zero, the particles near the surface absorb fluids and exceed the threshold. Without downward propagation, their wetness values gradually reach the maximum value, hence fluid particles are not absorbed.

Several granular materials are aggregated when wet. The proposed method shrinks the radius r_i of a granular particle i according to the wetness value and controls the degree of shrinkage by the coefficient of radius shrinkage k_r . Figure 5.21 shows depressions of the surfaces with different k_r .

The coefficient of liquid-bridge force k_{bridge} defines how strong the liquid bridge force is. In order to generate a structure such as a sand castle, the liquid bridge forces and modified DEM forces in Section 5.4 are required to strengthen the cohesion of the structure.

5.7.3 Limitations

The major limitations in the proposed method are scalability, physical plausibility and stability.

Scalability : In order to achieve highly dynamic animations, the proposed method handles granular materials as individual particles, instead of as a continuum. Therefore, the model suffers from the memory usage and performance in a very large scene. The memory usage and performance are directly influenced by the number of particles used in the scene. Employing an adaptive technique [1] or hybrid methods with height fields [69] could solve these problems.

Physical plausibility : Because the proposed model is a simplified physical model targeting the interactive applications, the parameters related to the wetting effects are not exactly physically-based. The parameters related to the wetting effects are particle's radius, threshold of wetness value $w_{threshold}$, maximum wetness value w_{max} , propagation rate k_p (Eq.(5.7)), coefficient of radius shrinkage k_r (Eq.(5.1)) and coefficient of liquid-bridge force k_{bridge} (Eq.(5.6)). These para-

maters are all user-defined values. Therefore, to simulate a scene with the specific physical behaviors, users have to set parameters with trial and error. In addition, the force interaction between a fluid particle and a granular particle is also computed as the fluid - fluid interaction which is physically not true.

Stability : The proposed method uses the explicit integration scheme for the simplicity. Therefore, the proposed method is conditionally stable. The integrations in the simulation systems can be performed using explicit or implicit schemes. Explicit schemes are often preferred due to its low computational cost and ease of implementation. However, for stable simulations, the time step in explicit schemes should be carefully considered. In other words, the time step is limited by stability. Implicit schemes can solve the stability problem, at the cost of the high computational cost and complexity in the implementation.

5.8 Summary

This chapter has introduced simulations of interactions between fluids and granular materials based on SPH and DEM. Specifically, this chapter has presented the following contributions in order to handle the propagation of wetness provided by a fluid passing through granular materials and the transition of the properties of the granular materials:

1. An empirical model for the propagation of wetness by means of introducing a wetness value for each granular particle,
2. Shrinkage of the radii of granular particles for the aggregation, and
3. Integration of the attractive force due to the amount of wetness introduced into the DEM framework.

This chapter has also demonstrated that a GPU-based simulator can achieve real-time performance.

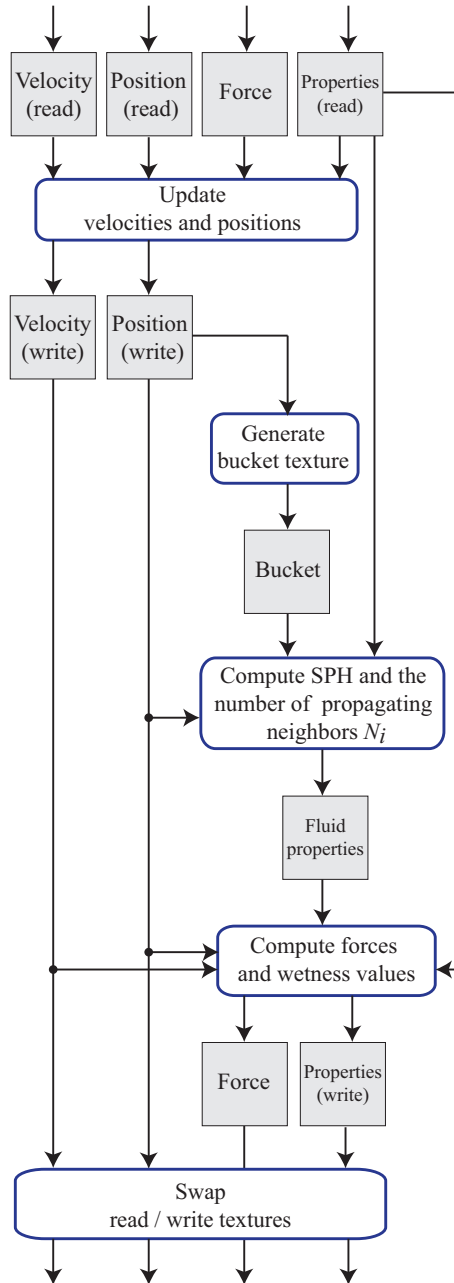
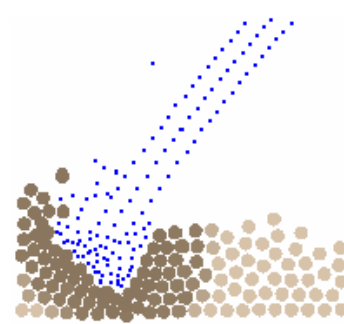
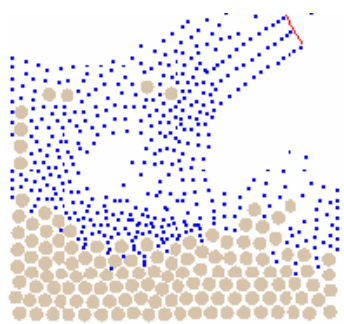


Figure 5.9: Flow chart of a single simulation time step on the GPU. Grey rectangles represent the texture data, blue rounded rectangles represent operations and black directed line segments represent the flow of data.



(a) simulation without wetting effects (b) Simulation with the wetting effects

Figure 5.10: 2D simulation results with and without the wetting effects.

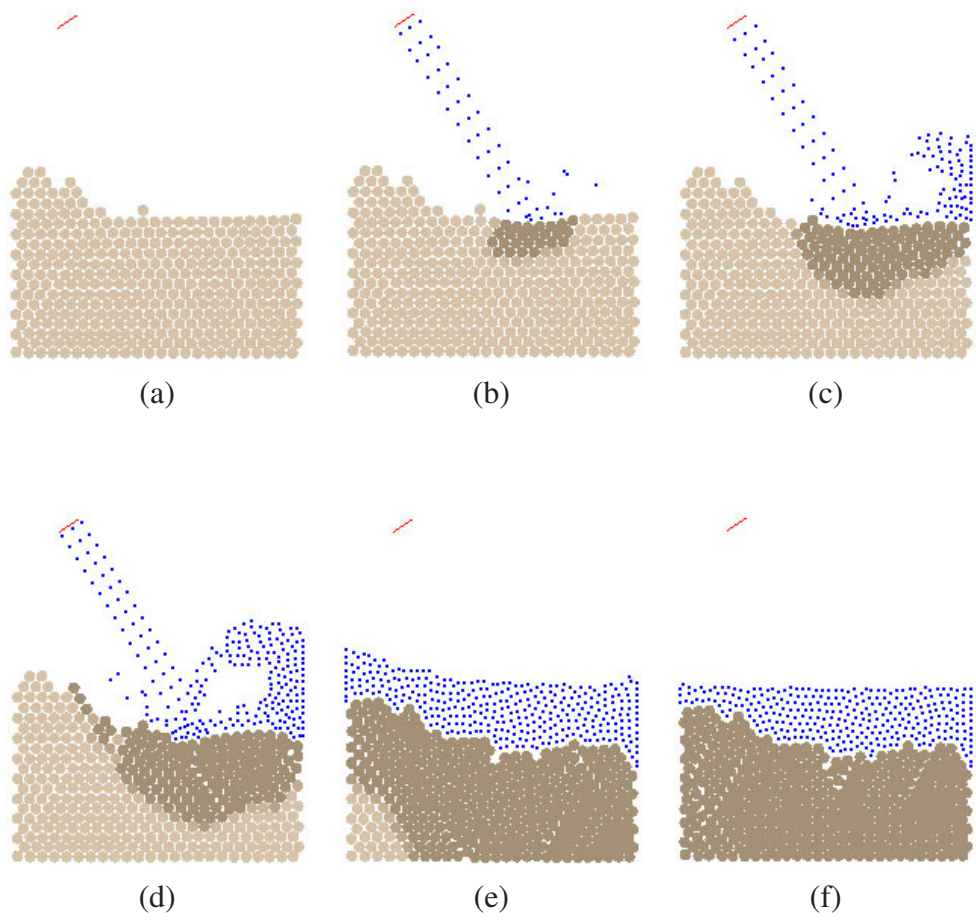
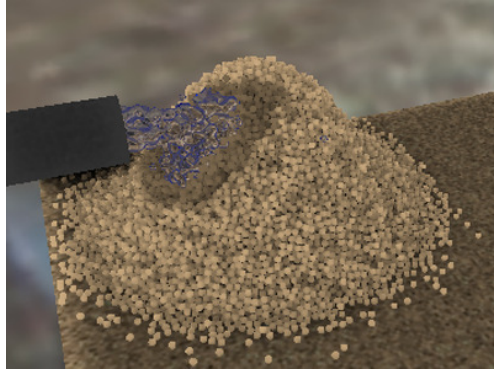
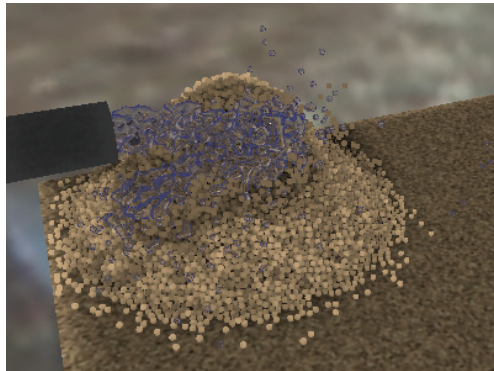


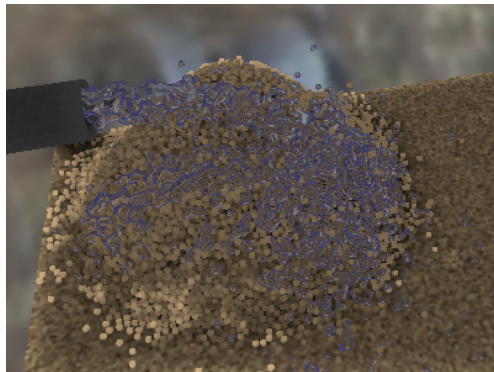
Figure 5.11: Propagation of wetness through a sand bed. A water stream is emitted from a red line in the scene. The fluid particles are rendered as blue dots.



(a)



(b)



(c)

Figure 5.12: Animation sequence of a sand pile and water stream emitted from a black faucet. This scene contains 8,000 fluid particles and 32,000 granular particles. The proposed method can simulate this scene with around 49 fps.

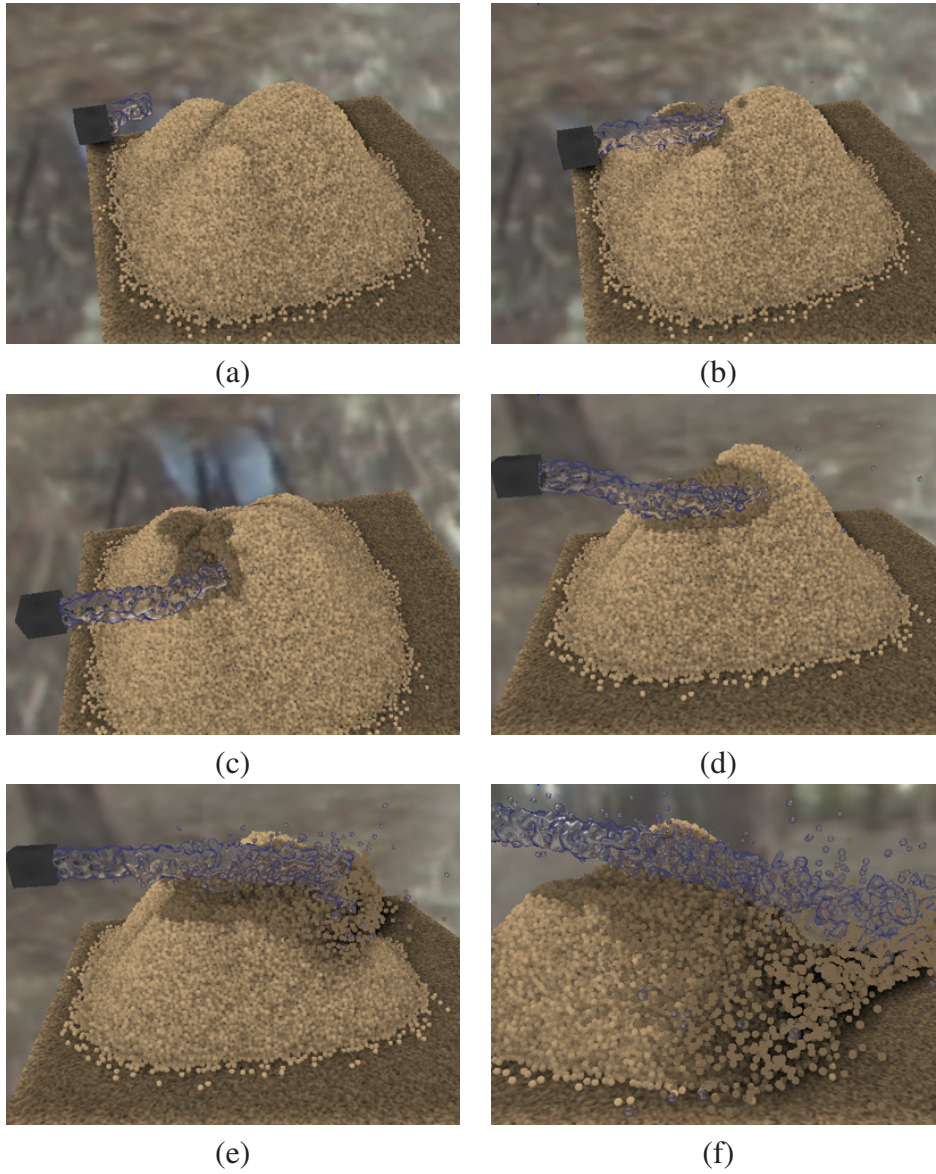
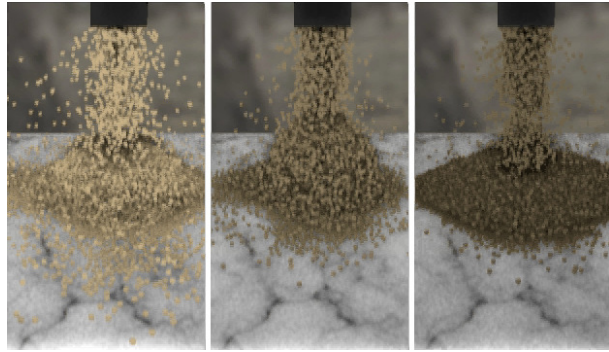
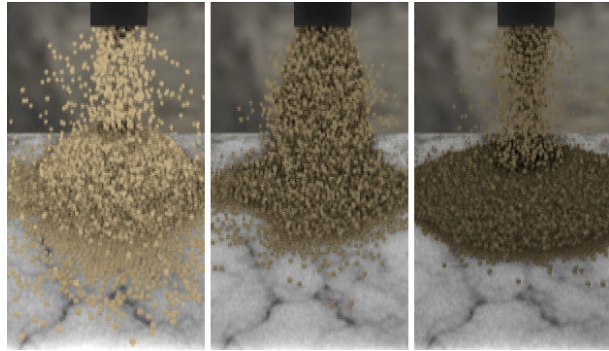


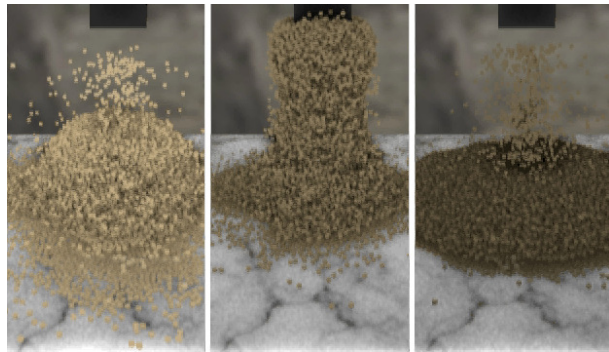
Figure 5.13: Animation sequence of a large sand pile with a water stream emitted from a black faucet. The result shows the destruction of a large sand pile containing 160,000 sand particles by a high velocity flow containing 16,000 water particles with 13 fps simulation and rendering speed.



(a)

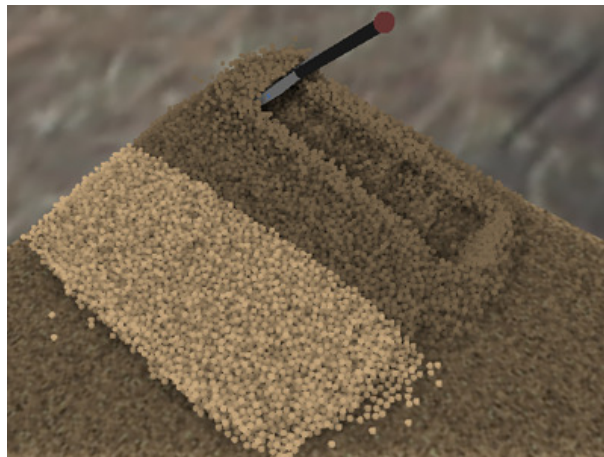


(b)



(c)

Figure 5.14: Comparison of the behaviors of dry (light brown), wet (brown) and overwet (dark brown) particles.

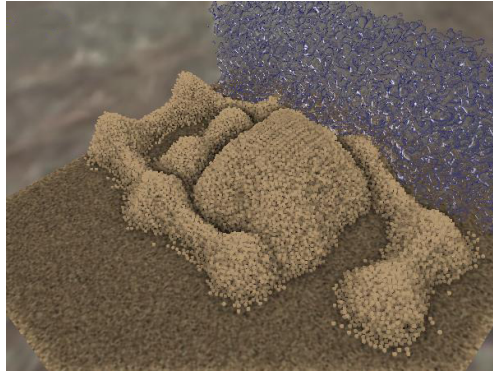


(a)

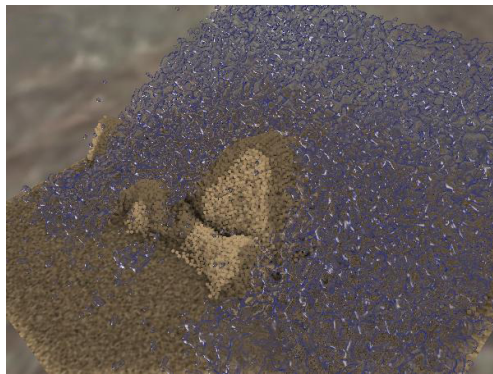


(b)

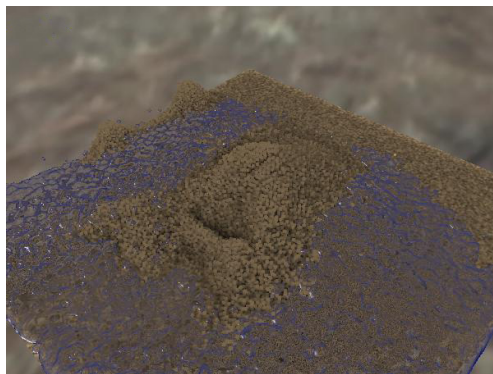
Figure 5.15: Interaction between granular particles with a rigid shovel. The liquid-bridge forces and modified-DEM forces result in different behaviors when the particles interact with a rigid body.



(a)



(b)



(c)

Figure 5.16: Interactions between massive fluid and a sand castle. The interactive speed (13 fps) can be achieved.

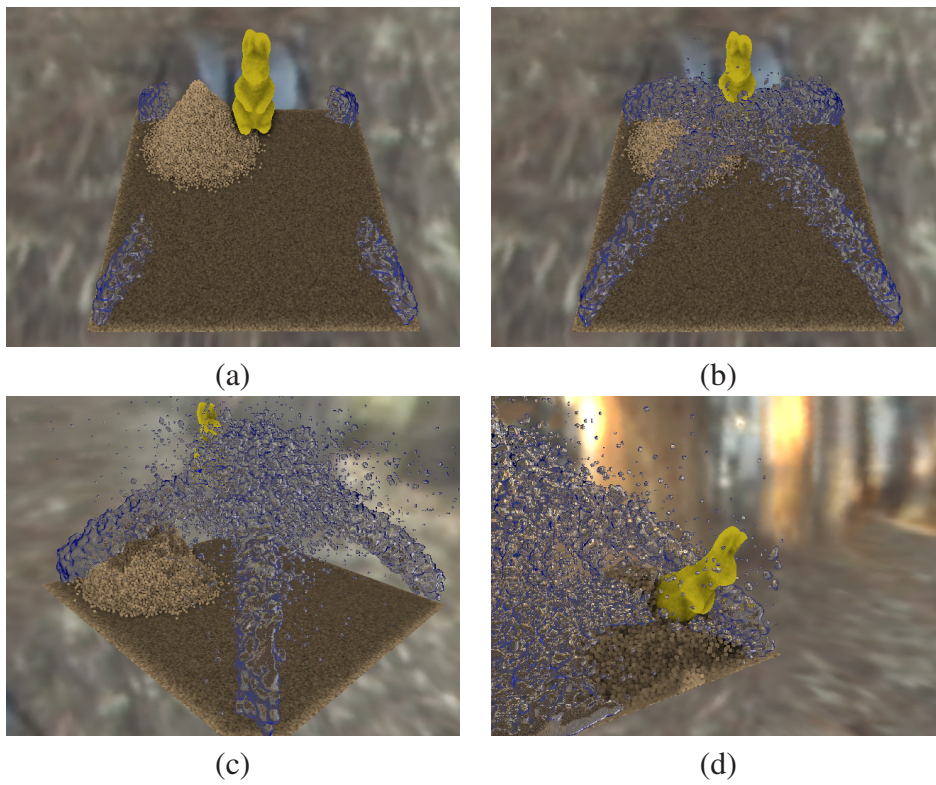


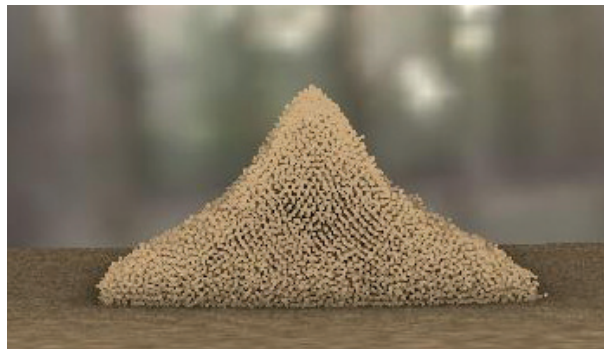
Figure 5.17: Interactions between sands, fountains and a rigid bunny. The proposed method can simulate this scene with around 12 fps.



(a) $k_t=10$



(b) $k_t=50$



(c) $k_t=100$

Figure 5.18: Interactions between massive fluid and a sand castle. The interactive speed (13 fps) can be achieved.

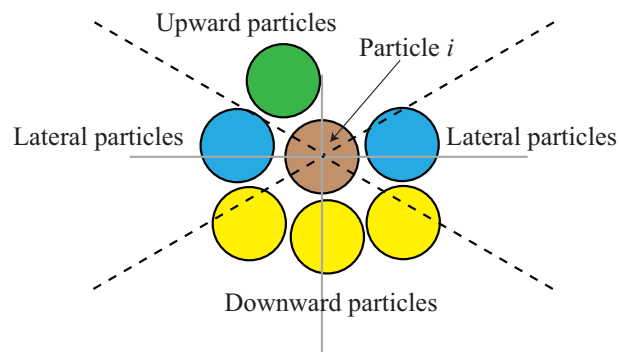


Figure 5.19: The neighboring particles in the vicinity of the particle i . The upward particle is shown as the green particle. Lateral particles are shown as blue particles. Downward particles are shown as yellow particles.

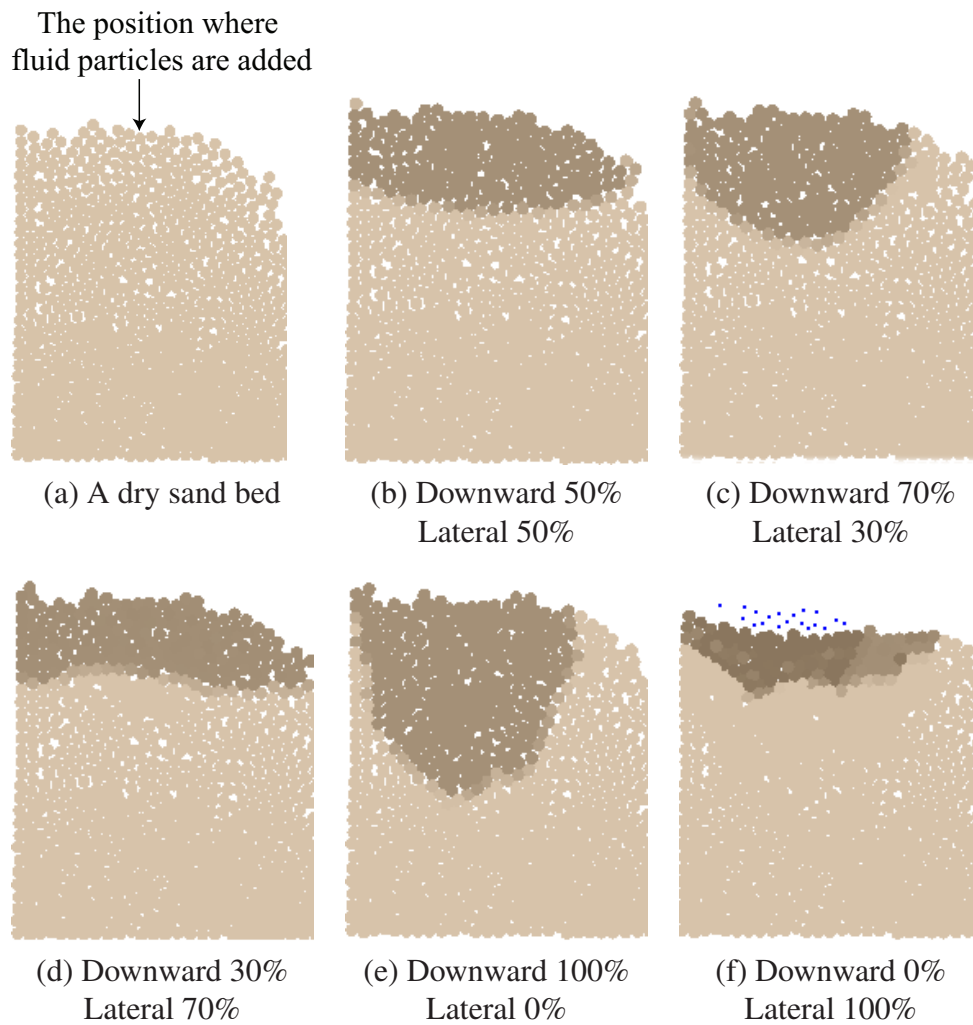


Figure 5.20: The simulation results with different downward and lateral propagation rates k_p . The shapes of the wetting fronts are various according to the propagation rates.

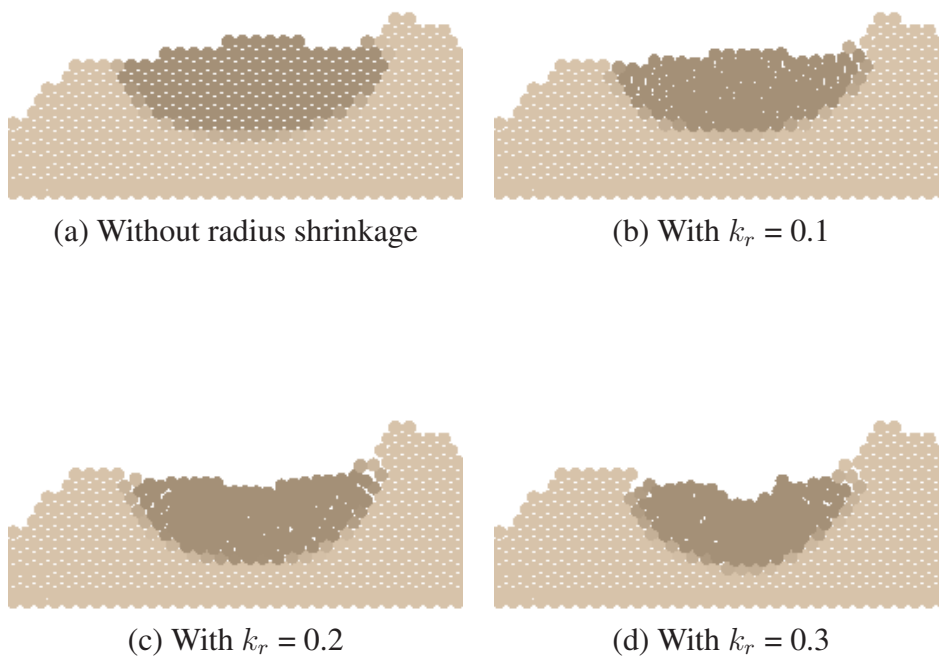


Figure 5.21: The simulation results with different coefficients of radius shrinkage k_r . The larger k_r results in a more depressed surface.

Chapter 6

Wetting Effects in Strands

This chapter gives an overview of the proposed method and how to rasterize a hair volume using a Cartesian bounding grid. Then, the methods for wetting effects of hair utilizing the grid are described.

6.1 Overview

To model wetting effects in coupling simulations of hair and water, our method introduces a Cartesian grid to implicitly represent the dynamic capillary system among hair strands (Section 6.2). Each voxel in this grid defines the unit for water absorption and diffusion. Water travels in the hair volume both in microscopic (hair strands) and macroscopic scales (voxels). In the microscopic scale, each hair segment of each hair strand holds some water inside and around the segment. The water inside the segment diffuses to adjacent segments in a hair strand, and the water around the segment flows along the hair strand. In the macroscopic scale, each voxel containing hair absorbs water and the water diffuses among voxels. The water in each voxel is then distributed to each segment within the voxel. The water propagation processes are summarized in the following four stages:

Voxel absorption: When voxels contacts with water, the water is absorbed to the voxel (Section 6.3.1).

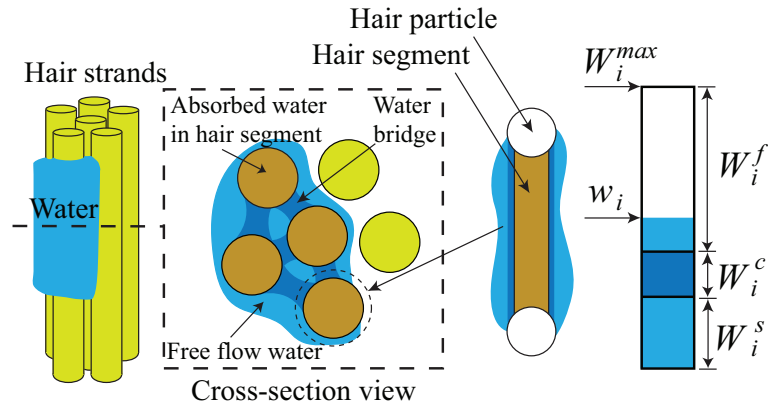


Figure 6.1: A brief microscopic illustration of wet hair and our simulation model of a hair segment. Water permeated into hair is represented as a water mass of a hair segment. The total possible amount of water mass is a summation of absorbed water, water bridge and free-flow water capacities.

Macroscopic propagation (Inter-voxel diffusion): The water propagates to the neighboring voxels according to the difference of capillaries of the voxels (Section 6.3.2). The change of water in each voxel is then distributed to segments in the voxel.

Microscopic propagation: Water inside each segment diffuses to its connected segments and the excessive water around the segment flows along the hair strand, which is followed by intra-voxel diffusion that uniformizes segments' water amount within the voxel (Section 6.3.3).

Water dripping: Water drips out as a droplet from a voxel if the water mass exceeds its capacity in certain duration. (Section 6.3.4).

After these processes, the stored water in each segment causes cohesion forces (Section 6.4) and morphological shape transformation (Section 6.5) as well as increase of the weight of hair strands. Algorithm 2 shows the pseudocode of our algorithm.

Algorithm 2 Pseudocode of our algorithm.

```
1: Given  $V$ : a set of non-empty voxels.
2: Given  $M$ : hair surface meshes (Section 6.3.4).
3: Given  $w_i$ : current water mass in segment  $i$ 
4: Given  $w_v$ : current water mass in voxel  $v$ 
5: Given  $W_v$ : a set of water capacities in voxel  $v$ 
6: loop
7:   SIMULATECSMANDSPH()
8:    $V, w_v, W_v \leftarrow$  CONSTRUCTGRID() // Section 6.2
9:   for all voxel  $v \in V$  do
10:     VOXELABSORPTION( $w_v, W_v$ ) // Section 6.3.1
11:     INTERVOXELDIFFUSION( $w_v, W_v$ ) // Section 6.3.2
12:     for all segment  $i$  in voxel  $v$  do
13:        $w_i \leftarrow w_i +$  INTERPOLATECHANGE()
14:       INSIDEHAIRSTRANDDIFFUSION( $w_i$ )
15:     end for
16:   end for
17:   for all segment  $i$  do
18:     FREEWATERFLOW( $w_i$ ) // Section 6.3.3
19:   end for
20:   for all voxel  $v \in V$  do
21:     INTRAVOXELDIFFUSION() // Section 6.3.3
22:   end for
23:    $M \leftarrow$  CONSTRUCTHAIRSURFACE()
24:   DRIPWATER( $M$ ) // Section 6.3.4
25:   for all segment  $i$  do
26:     COMPUTE COHESIONFORCES() // Section 6.4
27:     SHAPE TRANSFORM() // Section 6.5
28:   end for
29: end loop
```

6.2 Grid Construction

As introduced in Section 1, when hair contacts with water, the water absorption and diffusion happens due to the permeability of hair strands and capillaries. However, the capillaries of hair is difficult to figure out due to the complexity of hair structure. We can do nearest neighbors search for each hair segment and examine contact regions among neighboring hair segments to accurately determine the capillaries, but it is not an efficient way. Therefore, we introduce a grid of uniform voxels to approximate the capillaries of hair.

Hair volume is divided into voxels. Each voxel contains portions of hair segments and acts as a *porous voxel* that absorbs and diffuses water. In a porous voxel v , the maximum water capacity W_v^{max} that a porous voxel v can hold is considered as a summation of three capacities of water as follows (see Figure 6.1).

- **Water capacity in hair segments** W_v^s : The water capacity can be held inside of the hair segments.
- **Water capacity in capillaries** W_v^c : The water capacity held by capillaries (the extremely small gaps) between hair segments. We call the water in the capillaries *water bridges*.
- **Water capacity of free-flow** W_v^f : The capacity of water around the hair segments that is not absorbed into hair segments or held as water bridges, but covers the hair segments due to a cohesion. This capacity is a user-defined value controlling the amount of water considered as an Eulerian free-flow water.

The maximum capacity of the voxel is then: $W_v^{max} = W_v^s + W_v^c + W_v^f$. The W_v^s and W_v^f can be directly computed from a function of the total mass of hair in the voxel, while the capacity of capillaries W_v^c is approximated.

To determine the total hair mass in the voxel, we rasterize each hair segment in the grid using *3D Digital Differential Analyzer (3DDDA)*. 3DDDA yields a set

of voxels that segment i passes through together with its length in each voxel. Let l_{iv} be a length of segment i in a voxel v (Figure 6.2), then the total length of hair segments L_v and total mass of hair m_v^{hair} in the voxel can be computed as follows.

$$L_v = \sum_{i \in \mathcal{R}(v)} l_{iv}, \quad (6.1)$$

$$m_v^{hair} = \rho_{hair} \pi r^2 L_v, \quad (6.2)$$

where $\mathcal{R}(v)$ is a set of segments in the voxel v , ρ_{hair} is the density of hair segment and r is a radius of cylindrical hair segment. Then, we can compute water capacity in portions of hair segments W_v^s and free-flow water capacity W_v^f as follows.

$$W_v^s = K_{hair} m_v^{hair}, \quad (6.3)$$

$$W_v^f = k_{free} m_v^{hair}, \quad (6.4)$$

where K_{hair} is a permeability of a hair segment (usually $0.3 \leq K_{hair} \leq 0.45$, as described in Chapter 1) and k_{free} is a constant controlling the free-flow water capacity.

To approximate W_v^c , we assume that the contact regions of hair strands are proportional to the total mass of hair inside the voxel.

$$W_v^c = \begin{cases} 0, & \text{if } \|\mathcal{R}(v)\| = 1 \\ k_{capillaries} m_v^{hair}, & \text{otherwise} \end{cases} \quad (6.5)$$

where $k_{capillaries}$ is a constant and $\|\mathcal{R}(v)\|$ is the number of segments in the voxel v . Our simple assumption is based on the fact that wet hair segments tend to stick to each other such that the water bridges totally lie in the clump of hair strands. The higher the number of hair strands in the clump is, the larger the water bridges are. However, there is a possibility of large error due to the complexity of dry hair. The more accurate method for approximating topology of contact region inside the voxel is one of our future work.

In our model, a hair segment i stores water mass w_i . The summation of w_i of all hair segments is the total water mass in the hair volume. The current amount of

water mass w_v in a voxel can be computed from a summation of water mass of the portions of segments in the voxel. We consider that the water mass of a segment is uniformly distributed in the segment (Figure 6.2), therefore a water mass of a portion of the segment is linear proportional to its length.

$$w_v = \sum_{i \in \mathcal{R}(v)} w_i \frac{l_{iv}}{l_i}, \quad (6.6)$$

where l_i is the total length of segment. Note that we use a small w for the current amount of water mass and a capitalized W for the water capacity, while subscripted i and v indicate a segment and a voxel, respectively.

Later in this dissertation, the current amount of water mass in each capacity is used to handle the wetting effects. We define the current amount of each kind of water mass as follows.

$$w_v^s = \min\{w_v, W_v^s\}, \quad (6.7)$$

$$w_v^c = \min\{\max\{0, w_v - W_v^s\}, W_v^c\}, \quad (6.8)$$

$$w_v^f = \max\{0, w_v - W_v^s - W_v^c\}. \quad (6.9)$$

Eqs.(6.7), (6.8) and (6.9) represent the current water mass of each kind in the voxel. We use the same equations for the current water amount of a segment as well, i.e., w_i^s , w_i^c and w_i^f .

To handle an anisotropic water diffusion in hair, a representative tangent vector of hair in each voxel has to be known. We use an averaged tangent vector of each segment in the voxel \mathbf{d}_v as the representative tangent vector using the equation belows.

$$\mathbf{d}_v = \frac{1}{L_v} \sum_{i \in \mathcal{R}(v)} \mathbf{u}_i \frac{l_{iv}}{l_i}, \quad (6.10)$$

where \mathbf{u}_i is the tangent vector of segment i .

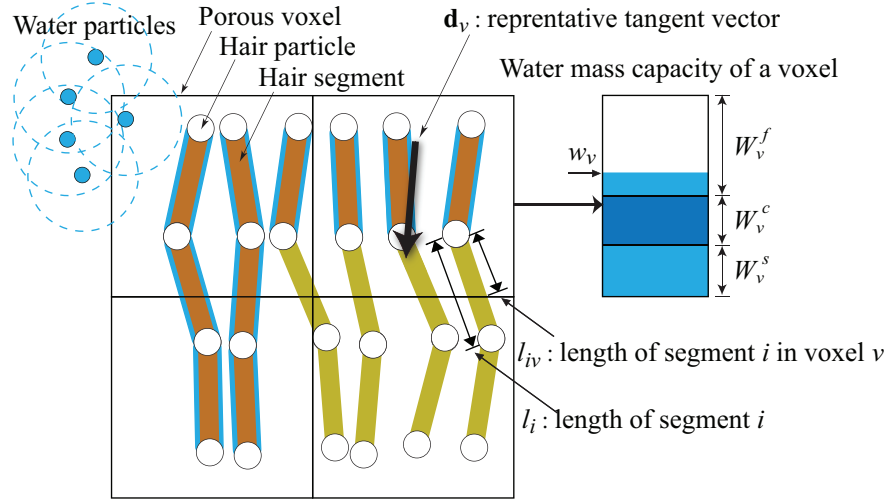


Figure 6.2: An Eulerian grid of uniform voxels for capturing hair porosity. Each voxel contains portions of hair segments.

6.3 Water Propagation

Water flows into and drips out of the hair volume influenced by voxel absorption, macroscopic and microscopic water propagations. In this section, we describe the detail of each stage.

6.3.1 Voxel Absorption

When a water particle p contacts with a porous voxel v , we consider the following conditions to check whether the water particle gets absorbed or not.

- The porous voxel has a capacity left, i.e., $w_v < W_v^{max}$.
- The water particle's velocity \mathbf{v}_p is toward the porous voxel, i.e., $\mathbf{v}_p \cdot \mathbf{n}_{pv} > 0$, where \mathbf{n}_{pv} is the vector from the water particle to the center of the porous voxel.

When the water particle is absorbed into the porous voxel, the water mass of the porous voxel is increased by the water particle's mass $w_v \leftarrow w_v + w_{SPH}$, where w_{SPH} is the water mass of the water particle (Figure 6.2).

6.3.2 Macroscopic Propagation

The difference of water in capillaries between the porous voxels is the main cause of the inter-voxel diffusion. We employ the Fick's second law for the change of the water in capillaries over time.

$$\frac{\partial w_v^c}{\partial t} = \nabla \cdot (D_v \nabla w_v^c), \quad (6.11)$$

where D_v is the diffusivity of porous voxel. The diffusivity of an isotropic porous voxel can be some constant. However, hair is an anisotropic permeable medium where the water diffuses more in the hair tangent direction. Given D_{\parallel} and D_{\perp} are diffusivity constants in the hair tangent and orthogonal directions, respectively, we modify the diffusivity for an anisotropic porous voxel as follows.

$$D_v = D_{\parallel} \lambda + D_{\perp} (1 - \lambda), \quad (6.12)$$

$$\lambda = \|\mathbf{d}_v \cdot \mathbf{u}\|, \quad (6.13)$$

where \mathbf{u} is the unit vector from the voxel v to a neighboring voxel. The evolution of diffused water mass, Δw_v , is then computed on the grid as follows.

$$\frac{\Delta w_v}{\Delta t} = \sum_{u \in \mathcal{G}(v)} D_v (S_u - S_v) \frac{(w_u^c - w_v^c)}{\Delta d}, \quad (6.14)$$

where $\mathcal{G}(v)$ is a set of six connected neighboring voxels of voxel v , Δt is a time step, Δd is a size of voxel, and S_v is a saturation of the voxel v . The saturation is a mass fraction of water in the capillaries of the porous voxel: $S_v = \frac{w_v^c}{W_v^c}$.

After the absorption and diffusion processes, we update the change of water mass in each porous voxel Δw_v to segments in the voxel. The increase is distributed to each segment according to the fraction of its portion length in the voxel (Eq. (6.15)). The longer the portion is, the more change is distributed to that segment.

$$\Delta w_i^{increase} = \sum_{v \in \mathcal{G}(i)} \Delta w_v \frac{l_{iv}}{L_v}, \quad (6.15)$$

where $\mathcal{G}(i)$ is a set of voxels that segment i passes through. However, there's possibly a problem in case that the change of water mass is decreasing and some segments in the voxel has no water mass. The water mass of those segments will be updated to a negative value, which is impossible. Therefore, we use a fraction of water mass that the segment contributes to the voxel w_{iv} for the change (Eq. (6.16)), instead of the fraction of length.

$$\Delta w_i^{decrease} = \sum_{v \in \mathcal{G}(i)} \Delta w_v \frac{w_{iv}}{w_v}. \quad (6.16)$$

6.3.3 Microscopic Propagation

In microscopic level, we consider the diffusion of water inside a segment w_i^s and the flow of free-flow water w_i^f . For each segment, there are also three kinds of water mass capacity. W_i^s and W_i^f are directly computed from the mass of the hair segment, while W_i^c is derived from the approximated W_v^c in the voxels (Eq. (6.5)) that the segment resides in.

$$W_i^c = \sum_{v \in \mathcal{G}(i)} W_v^c \frac{l_{iv}}{l_i}. \quad (6.17)$$

As a result, the current amount of water inside the segment w_i^s and free-flow water mass of the segment w_i^f can be calculated using Eq. (6.7) and (6.9), respectively.

The current water mass inside the segment diffuses to the adjacent segments in its strand. We use a simple diffusion process as follows.

$$\frac{\Delta w_i}{\Delta t} = D_s (w_{i+1}^s + w_{i-1}^s - 2w_i^s), \quad (6.18)$$

where Δw_i is the variation of w_i and D_s is a diffusivity between hair segments.

The water that flows within the hair volume is the free-flow water of the segment, w_i^f . We move this free-flow water along a hair strand from segment to segment controlled by a flow rate k_{flow} and the direction of the segment. We determine the direction of the flow by checking the dot product of the unit tangent vector of hair segment \mathbf{t}_i and the unit gravity direction \mathbf{g} , i.e., $k_{dir} = \mathbf{t}_i \cdot \mathbf{g}$.

1. There will be no flow, if $\|k_{dir}\|$ is less than a threshold ε .
2. If $k_{dir} \geq \varepsilon$, the water will flow to the next segment, $w_{i+1} \leftarrow w_{i+1} + \Delta t k_{flow} \|k_{dir}\| w_i^f$.
3. If $k_{dir} \leq -\varepsilon$, the water will flow to the previous segment, $w_{i-1} \leftarrow w_{i-1} + \Delta t k_{flow} \|k_{dir}\| w_i^f$.

Each segment transfers an amount of free-flow water from voxels to voxels. The update of the water mass w_v in each voxel can be computed using Eq. (6.6). The free-flow water inside the voxel diffuses among the segments as well, e.g., there might be free-flow water inside the voxel that has dry segments. We update w_i of each segment as follows.

$$\Delta w_i = \Delta t k_f \left(w_v^f \frac{l_{iv}}{L_v} - w_i^f \frac{l_{iv}}{l_i} \right), \quad (6.19)$$

where k_f is a constant. The first term on the right hand side in Eq. (6.19) is the probable free-flow water mass of the segment when all segments in the voxel are equally wet. The second term is the current free-flow water mass that the segment contributes to the voxel. The equation can be interpreted that the free-flow water mass of each segment gradually becomes equals to each other.

6.3.4 Water Dripping

Water droplets drips out when the water mass in a voxel exceeds W_v^{max} . Water droplets should be created around hair strands, but voxels are too coarse to specify where hair strands exist. For this, we create a surface mesh from the hair volume using distance transform of hair segments and the marching cubes algorithm [53]. We refer to this mesh as the *hair surface mesh*. If $w_v > W_v^{max}$ in voxel v and a hair surface mesh exists within the voxel, we generate a water particle and decrease the water mass of the voxel by the water particle's mass, $w_v \leftarrow w_v - w_{SPH}$. The decrease of the water mass is interpolated back to the segments in the voxel using

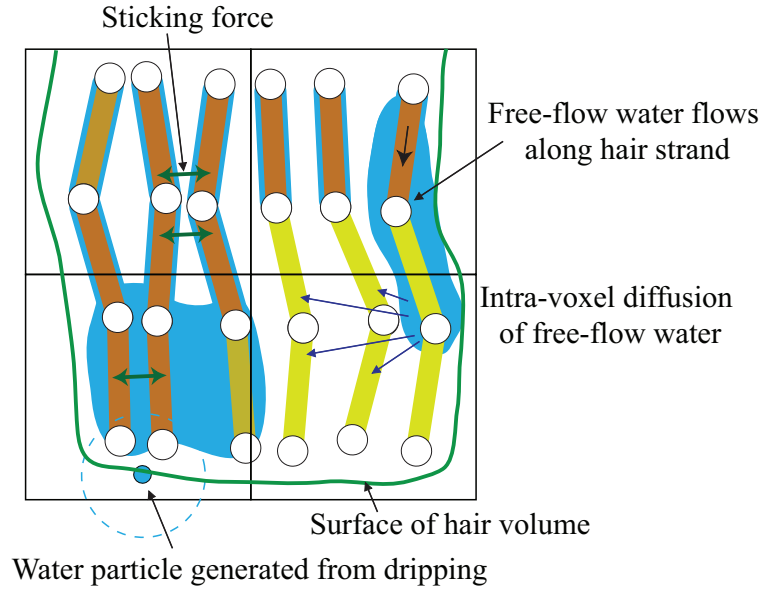


Figure 6.3: Handling free flow water and cohesion of wet hair.

Eq. (6.16). The position of the generated water particle is an averaged barycenter of the surface meshes (Figure 6.3). If there is a water particle within a water particle's radius of the generated position, we add radius and mass to the water particle instead of creating a new one.

To guarantee that absorbed water does not turn into water particles repeatedly or vice versa at the interface between porous voxels and water particles, we add a time delay γ . The dripping occurs, if the voxel has the water mass exceeded W_v^{max} for γ time steps. We also add a small velocity to the generated water particle in the direction of an averaged normal vector of the surface meshes to avoid the absorption of generated water particles.

6.4 Cohesion of Wet Hair

Cohesion of wet hair is influenced by the water bridges and the free-flow water. We add a sticking force between a pair of colliding segments according to those water masses of the segments (Figure 6.3). The result of the collision detection

gives the closest points \mathbf{x}_i and \mathbf{x}_j on the segments i and j , respectively. The pair of segments is a colliding pair when the length of the vector $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ is less than d_{ij} , where d_{ij} is a distance of r_i and r_j including a radius from the water masses.

$$d_{ij} = r_i + r_j + \sigma(w_i + w_j), \quad (6.20)$$

where σ is a ratio of increasing water radius. The sticking force between segment i and j is computed when $d_{ij} \geq |\mathbf{x}_{ij}| > (r_i + r_j)$:

$$\mathbf{F}_{ij}^{stick} = \frac{1}{2} k_{stick} (w_i + w_j) (d_{ij} - |\mathbf{x}_{ij}|) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|}, \quad (6.21)$$

where k_{stick} is a coefficient of the sticking force. If $|\mathbf{x}_{ij}| < (r_i + r_j)$, the penalty force is computed:

$$\mathbf{F}_{ij}^{penalty} = k_p (r_i + r_j - |\mathbf{x}_{ij}|) \frac{\mathbf{x}_{ij}}{|\mathbf{x}_{ij}|}, \quad (6.22)$$

where k_p is a coefficient of the penalty force.

6.5 Morphological Shape Transformation

The absorbed water inside a hair strand can temporally alter the shape of wet hair due to chemical reactions. In the CSM model, a user assigns a predefined shape of a hair strand by giving positions of hair particles or tangent vectors of hair segments (see Figure 6.4). When the hair particles are moved by external forces, CSM tries to maintain the predefined shape. Accordingly, we assign two predefined shapes for a dry hair strand and a wet hair strand. When the hair segment i is wet, we first calculate a tangent vector that segment i should transform into, called *goal tangent* \mathbf{t}_i^{goal} . The goal tangent is interpolated between the tangent of dry segment \mathbf{t}_i^{dry} and wet segment \mathbf{t}_i^{wet} using the absorbed water mass inside the segment w_i^s . The current tangent vector \mathbf{t}_i is then updated towards \mathbf{t}_i^{goal} as follows.

$$\mathbf{t}_i^{goal} = \left(1 - \frac{w_i^s}{W_i^s}\right) \mathbf{t}_i^{dry} + \frac{w_i^s}{W_i^s} \mathbf{t}_i^{wet}, \quad (6.23)$$

$$\mathbf{t}_i \leftarrow k_t \Delta t (\mathbf{t}_i^{goal} - \mathbf{t}_i), \quad (6.24)$$

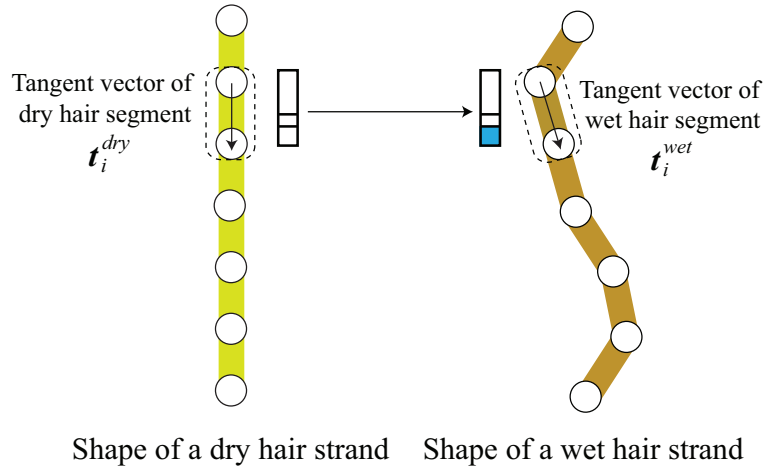


Figure 6.4: Morphological shape transformation of wet hair. Predefined shapes of dry hair and wet hair are assigned by the user.

where k_t is a constant controlling speed of the chemical reactions of hair strand.

6.6 Results

This section shows hair simulation results with wetting effects generated by the proposed method. Then, this section gives discussions and limitations of the proposed model.

6.6.1 Simulation Results

Our implementation was written in C++ with OpenGL. All experiments were conducted on a PC with an Intel Core i7 3.20GHz, 6GB RAM and an NVIDIA GeForce GTX 480 GPU. Our final results are rendered using an off-line rendering software, POVRay 3.7 [67]. Each hair segment is rendered as a semi-transparent cylinder. The color of a hair segment gets darker according to the amount of absorbed water mass. When the amount of free-flow water around a hair segment is large enough, the free-flow water can be seen as a thin water film covering the segment. In our rendering, we add some virtual water particles (not used for simulation, only for rendering) along the hair segment with the size according to

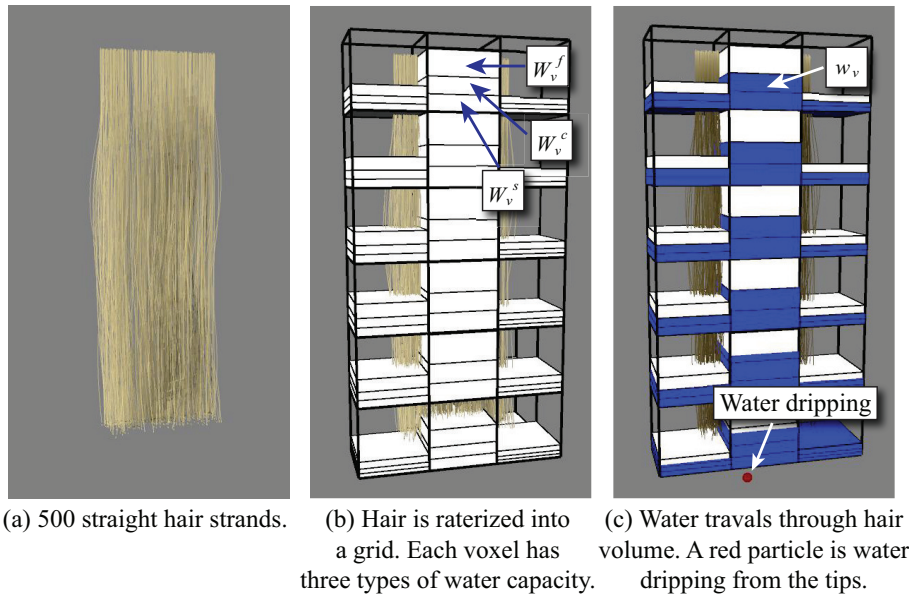


Figure 6.5: A simulation result demonstrating an overview of our model.

the free-flow water amount. Then, we employ the method proposed in [101] for constructing the surfaces of water particles in SPH domain and virtual particles around hair segments.

An overview of water traversal in our model is shown in Figure 6.5. The water capacities in each voxel are shown as a stack of three white boxes. From bottom to top, the boxes indicate the water capacities in hair segments, capillaries and free-flow. As shown in the figure, the higher the number of hair segments in a voxel is, the larger the water capacities are. When a voxel contacts with water, the water gets absorb in hair segments first, then capillaries and free-flow capacities. Afterwards, the absorbed water in hair segments diffuses to their adjacent segments in the same hair strand, while the water in capillaries of each voxel diffuses to its neighboring voxels. If the free-flow water exists, the water flows through hair volume influenced by tangent direction of hair segment until dripping out of hair (a red particle in the figure).

Our simulation result compared with the real world is shown in Figure 6.6. The real hair and our results are shown in the left and right columns, respectively.



Figure 6.6: Comparison of our simulation results and real world experiments.

The top row shows dry hair and the bottom row shows wet hair.

Figure 6.7 shows results of morphological shape transformations according to the absorbed water. Straight hair temporarily turns into wavy hair in Figure 6.7(left), while the wet curly hair in Figure 6.7(right) becomes more straight. In the curly hair, hair strands become straight due to the weight of the absorbed water and the morphological shape transformations, but largely induced by the shape transformations. Figure 6.9 shows results of 5,000 strands of wet mob being pull out of a water tank. The strands are completely wet under the water. The water amount in the strands is maximum. Therefore, a great amount of excess water (free-flow water) flows along the strands until pouring out of the strands when the mob is rising into the air. Figure 6.10 shows animation sequences of 11,000 wavy hair strands interacting with water from a shower. The water is absorbed and diffuses into the hair. The wet hair strands form several clumps. The excess water gradually drips at the tips. The breakdown computational time in each process is

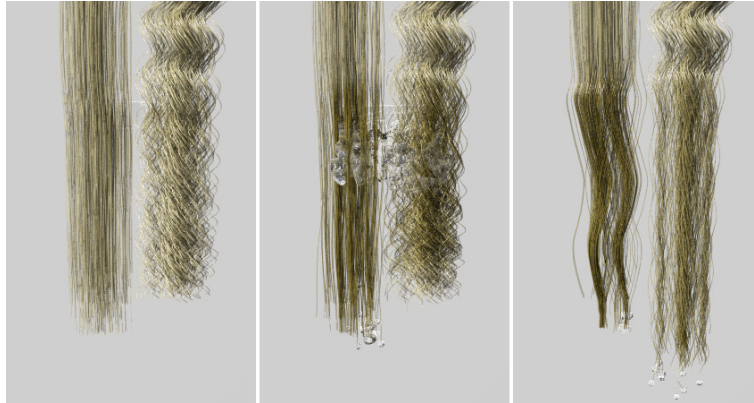


Figure 6.7: Results of morphological shape transformations in our model. The wet parts of straight hair (left) and curly hair (right) turn into wavy and straight hair, respectively.

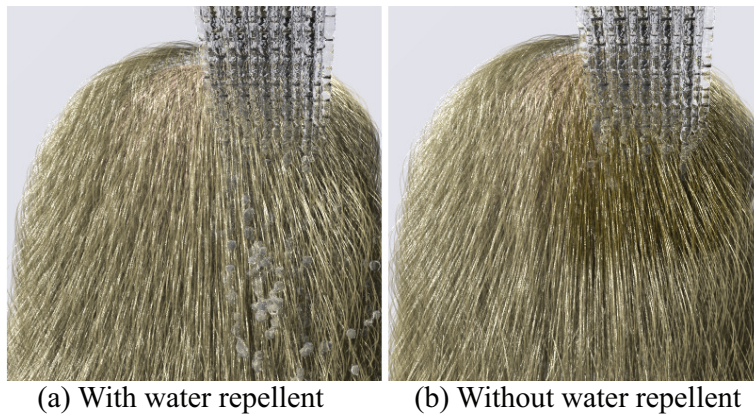


Figure 6.8: Simulation results of hair with and without water repellent.

shown in Table 6.1.

6.6.2 Discussions and Limitations

Compared to the real world, there are much more underlying physics of water and hair interactions, e.g., water repellent, surface tension between fluid and hair. In the real world, sometimes hair has a water repellent where the water is repelled by the oil component or static electric in hair. We demonstrate a simple way to handle this in Figure 6.8. We give each hair segment an α value that indicates the

strength of oil component (or static electric) of the segment. The voxel absorption occurs only when the summation of α values in the voxel is less than a threshold. Unless, the α values of segments in the voxel keep decreasing when the voxel contacts with water. However, this should be handled in a more physics-inspired way.

Regarding the choice of voxel size, we use the size equal to or larger than a diameter of water particle, since a water particle can be totally absorbed into a voxel. Although the overall water mass in the hair volume remains the same with a larger voxel size, the water absorption and diffusion processes affect more hair segments in a time step. An order of computation cost of hair rasterization is $O(n)$, where n is a number of voxels. Therefore, the performance gain is linear to the voxel resolution.

The rendering method we used is a simple hair rendering with a changing of hair color according to absorbed water mass. More realistic rendering method would greatly enhance our results, especially the rendering of water inside the hair volume. The free-flow water should be rendered as a very thin water film, while the water in the capillaries has a complex shape linking between hair strands (water bridges).

In our implementation, we use a simple Euler explicit integrator which has a high possibility of missing collisions when hair has a large motion. However, there are no significant problem in the scenarios of our shown results.

6.7 Summary

We have introduced a model for handling the wetting effects in coupling simulations of hair and water. Hair is modeled as an anisotropic dynamics permeable media. We have proposed a method utilizing Eulerian approach for capturing hair porosity and handling wetting effects on the fine detailed hair simulation using Lagrangian method. Our method have enabled many interesting effects of wet-

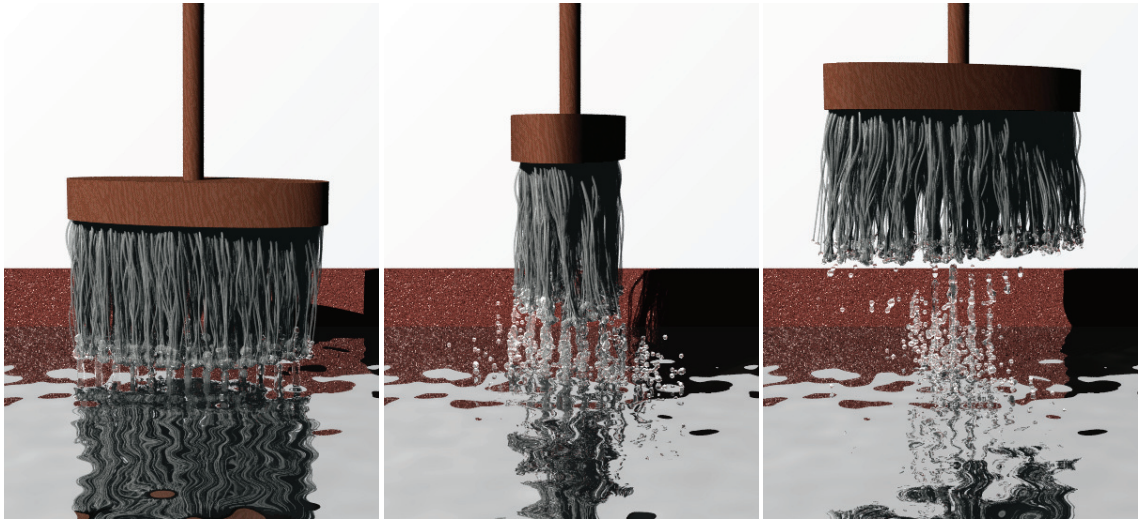


Figure 6.9: Animation sequences of 5,000 strands of wet mop (87,000 segments) being pull out of a water tank.



Figure 6.10: Animation sequences of 11,000 straight hair strands (220,000 segments) interacting with water from a shower.

ting hair. To the best of our knowledge, our system is the first to fully simulate the interactions between hair and water both in macroscopic and microscopic levels.

Table 6.1: The computational time in seconds taken in each process.

No. of hair segments / Max no. of water particles	CSM	SPH	Voxelization	Water absorption and diffusion	Water flow	Collisions and cohesions	Total time
15,000 segments / 1,250 particles (Figure 6.7)	0.034	0.052	0.331	0.003	0.005	0.470	0.895
87,000 segments / 2,250 particles (Figure 6.9)	0.159	0.112	2.982	0.009	0.031	3.078	6.371
220,000 segments / 4,800 particles (Figure 6.10)	0.364	0.522	5.87	0.023	0.0728	8.192	15.044

Chapter 7

Conclusion and Future Work

This dissertation has proposed the methods for handling wetting effects in deformable porous structures of granular materials and strands.

For the wetting of granular materials, this dissertation has introduced water absorption, water propagation between granular particles and changes of physical properties of wet granular particles into a simple particle-based framework of DEM and SPH (Chapter 5). Dynamic animations of granular material including wetting effects were achieved at interactive rates using a GPU-based simulator.

To handle the wetting effects of hair, first, this dissertation has proposed a strand simulation method (Chapter 4). The proposed strand simulation method can handle stiff strand in a numerically stable way, and easily modify the shape of a hair strand. This dissertation has demonstrated a simulation result with a large number of hair strands comparable to the real world, which has never been tackled in the previous works. The twisting, tearing by stretching, tearing by twisting and flicking effects of a strand have also been introduced together with an optimized grid-based collision detection. Then, on top of the proposed strand simulation model, this dissertation has proposed a simulation model that reproduces interactions between water and hair as a dynamic anisotropic porous medium (Chapter 6). This dissertation has demonstrated that the proposed model can generate many interesting effects of interactions between fine-detailed dynamic hair and water, i.e., water absorption and diffusion, cohesion of wet hair strands, water

flow within the hair volume, water dripping from the wet hair strands and morphological shape transformations of wet hair.

For future work, the collision detection of hair was the most time-consuming process in this dissertation, so we would like to develop an efficient model for hair collision detection. The collision between segments is treated as a collision between rigid segments. We would like to improve the collision detection algorithm to handle the collisions between deformable segments. Although, wet granular material and hair simulations were conducted on GPU, wet hair simulation is still run on a CPU. We would like to implement the wet hair simulation on the GPU as well. We would like to consider a deformation of cross-sections during twisting. In the proposed model, each segment is rendered as a spherical cylinder, however, cross-sections of most material can be deformed when it is twisted. The non-uniform torsional rigidity in our model is considered along the length, not the cross-sections. A non-uniform density distribution within the cross-section should be considered to simulate more interesting results. For rendering, the plausibility of wet surface rendering of granular material and hair strands could be improved by employing, e.g., subsurface scattering [38] of wet materials.

With the initiative of this dissertation, we hopefully look forward to more approaches in physics simulation taking wetting effects into account. We strongly believe that the researches in this topic can help animations and games getting closer to the real world in the near future.

Bibliography

- [1] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007)*, 26(3):48, 2007.
- [2] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *ACM SIGGRAPH Computer Graphics (Proc. of SIGGRAPH 1992)*, volume 26, pages 111–120, 1992.
- [3] Y. Bando, T. Nishita, and B.-Y. Chen. Animating hair with loosely connected particles. *Computer Graphics Forum (Proc. of EUROGRAPHICS 2003)*, 22(3):411–418, 2003.
- [4] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of SIGGRAPH 1998*, pages 43–54, 1998.
- [5] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. In *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007)*, volume 26, page 100, 2007.
- [6] M. Becker, H. Tessenorf, and M. Teschner. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503, 2009.
- [7] N. Bell, Y. Yu, and P. J. Mucha. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86, 2005.

- [8] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete Elastic Rods. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2008)*, 27(3):63:1–63:12, 2008.
- [9] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. Lévêque. Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2006)*, 25(3):1180–1187, 2006.
- [10] C. G. Bhuvanesh, D. A. Rajesh, and M. H. David, editors. *Textile sizing*. CRC Press., 2004.
- [11] T. BOUBEKEUR and C. SCHLICK. Generic mesh refinement on GPU. In *ACM SIGGRAPH/Eurographics Graphics Hardware*, pages 99–104, 2005.
- [12] A. Bruderlin. A method to generate wet and broken-up animal fur. *Journal of Visualization and Computer Animation*, 11(5):249–259, 2000.
- [13] A. V. Burkalow. Angle of repose and angle of sliding friction: an experimental study. *Bulletin of The Geological Society of America*, 56(6), 1945.
- [14] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2004)*, 23(3):377–384, 2004.
- [15] B. Chancelou, A. Luciani, and A. Habibi. Physical models of loose soils dynamically marked by a moving object. In *Proceedings of the Computer Animation*, pages 27–35, 1996.
- [16] N. Chentanez, T. G. Goktekin, B. E. Feldman, and J. F. O’Brien. Simultaneous coupling of fluids and deformable bodies. In *SCA ’06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 83–89, 2006.
- [17] B. Choe, M. G. Choi, and H.-S. Ko. Simulating complex hair with robust collision handling. In *SCA ’05: Proceedings of the 2005 ACM SIG-*

- GRAPH/Eurographics symposium on Computer animation*, pages 153–160, 2005.
- [18] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based viscoelastic fluid simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228, 2005.
- [19] P. A. Cundall and O. D. L. Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29:47–65, 1979.
- [20] A. Daldegan, N. M. Thalmann, T. Kurihara, and D. Thalmann. An integrated system for modeling, animating and rendering hair. *Computer Graphics Forum*, 12(3):211–221, 1993.
- [21] R. Diziol, J. Bender, and D. Bayer. Volume conserving simulation of deformable bodies. In *Eurographics 2009 Short Papers*, pages 37–40, 2009.
- [22] J. Dorsey, A. Edelman, H. W. Jensen, J. Legakis, and H. K. Pedersen. Modeling and rendering of weathered stone. In *Proc. of SIGGRAPH 1999*, pages 225–234, 1999.
- [23] R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In *Proc. of SIGGRAPH 2001*, pages 15–22, 2001.
- [24] N. Foster and D. Metaxas. Realistic animation of liquids. In *Graphical Models and Image Processing*, pages 23–30, 1995.
- [25] R. Gingold and J. Monaghan. Smoothed particle hydrodynamics – theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375, 1977.
- [26] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. *ACM Transactions on Graphics (Proc. of SIGGRAPH2007)*, 26(3):49–55, 2007.

- [27] Y. Guang. A method of human short hair modeling and real time animation. In *Proceedings of Pacific Conference on Computer Graphics and Applications, IEEE Computer Press*, pages 435–438, 2002.
- [28] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (Proc. of SIGGRAPH2005)*, 24(3):973–981, 2005.
- [29] R. Gupta and N. Magnenat-Thalmann. Interactive rendering of optical effects in wet hair. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 133–140, 2007.
- [30] S. Hadap. Oriented strands: dynamics of stiff multi-body system. In *SCA’06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 91–100, 2006.
- [31] S. Hadap and N. Magnenat-Thalmann. Modeling dynamic hair as a continuum. *Computer Graphics Forum*, 20(3):329–338, 2001.
- [32] T. Harada. Real-time rigid body simulation on GPUs. *GPU Gems 3, Chapter 29*, pages 123–148, 2007.
- [33] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on GPUs. In *Proc. of Computer Graphics International*, pages 63–70, 2007.
- [34] M. Harris. Fast fluid dynamics simulation on the gpu. In *ACM SIGGRAPH 2005 Courses*, number 220, 2005.
- [35] M. J. Harris, W. V. Baxter, T. Scheuermann, and A. Lastra. Simulation of cloud dynamics on graphics hardware. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 92–101, 2003.
- [36] S. Herminghaus. Dynamics of wet granular matter. *Advances In Physics*, 54:221–261, 2005.

- [37] M. Huber, S. Pabst, and W. Straßer. Wet cloth simulation. In *ACM SIGGRAPH 2011 Posters*, pages 10:1–10:1, 2011.
- [38] H. W. Jensen, J. Legakis, and J. Dorsey. Rendering of wet materials. In *Proc. of Eurographics Rendering Workshop 1999*, pages 273–282, 1999.
- [39] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Proc. of SIGGRAPH 1989*, pages 271–280, 1989.
- [40] Y. Kanamori, Z. Szego, and T. Nishita. GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum (Proc. of Eurographics 2008)*, 27(2):351–360, 2008.
- [41] T.-Y. Kim and U. Neumann. A thin shell volume for modeling human hair. In *Proc. of the Computer Animation*, pages 121–128, 2000.
- [42] P. Kipfer and R. Westermann. Realistic and interactive simulation of rivers. In *GI '06: Proceedings of Graphics Interface 2006*, pages 41–48, 2006.
- [43] C. K. Koh and Z. Huang. A simple physics model to animate human hair modeled in 2d strips in real time. In *Proc. of the Eurographic workshop on Computer animation and simulation*, pages 127–138, 2001.
- [44] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of incompressible flow. *Nuclear Science and Engineering*, 123:421–434, 1996.
- [45] T. Kurihara, K.-i. Anjyo, and D. Thalmann. Hair animation with collision detection. *Models and Techniques in Computer Animation SpringerVerlag Tokyo*, (January):128–138, 1993.
- [46] T. Lenaerts, B. Adams, and P. Dutré. Porous flow in particle-based fluid simulations. In *ACM Transactions on Graphics (Proc. of SIGGRAPH 2008)*, volume 27, pages 49:1–49:8. ACM, 2008.
- [47] T. Lenaerts and P. Dutré. Mixing fluids and granular materials. *Computer Graphics Forum (Proc. of Eurographics 2009)*, 28(2):213–218, 2009.

- [48] X. Li and J. M. Moshell. Modeling soil: realtime dynamic models for soil slippage and manipulation. In *Proc. of SIGGRAPH 1993*, pages 361–368, 1993.
- [49] W. Liang and Z. Huang. An enhanced framework for real-time hair animation. In *Proc. of the 11th Pacific Conference on Computer Graphics and Applications*, pages 467–471, 2003.
- [50] Y. Liu, X. Liu, and E. Wu. Real-time 3d fluid simulation on gpu with complex obstacles. In *Proc. of the Computer Graphics and Applications, 12th Pacific Conference*, pages 247–256, 2004.
- [51] Y. Q. Liu, H. B. Zhu, X. H. Liu, and E. H. Wu. Real-time simulation of physically based on-surface flow. *The Visual Computer (Proc. of Pacific Graphics 2005)*, 21(8-10):727–734, 2005.
- [52] L’Oreal. Unexpected properties of hair, <http://www.hair-science.com>. 2011.
- [53] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987.
- [54] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.
- [55] L. Lucy. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013, 1977.
- [56] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. Detail preserving continuum simulation of straight hair. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2009)*, 28(3):62:1–62:6, 2009.

- [57] N. Metaaphanon, Y. Bando, B.-Y. Chen, and T. Nishita. Simulation of tearing cloth with frayed edges. *Computer Graphics Forum (Proc. of Pacific Graphics 2009)*, 28(7):1837–1844, 2009.
- [58] M. Mittring. Finding next gen – CryEngine 2. *Advanced Real-Time Rendering in 3D Graphics and Games Course – SIGGRAPH 2007*, 2007.
- [59] R. Miyazaki, S. Yoshida, Y. Dobashi, and T. Nishita. A method for modeling clouds based on atmospheric fluid dynamics. In *Proc. of Pacific Graphics 2001*, pages 363–372, 2001.
- [60] Y. Morimoto, M. Tanaka, R. Tsuruno, and K. Tomimatsu. Visualization of dyeing based on diffusion and adsorption theories. volume 0, pages 57–64, 2007.
- [61] W. E. Morton and J. W. S. Hearle. *Physical Properties of Textile Fibers*. Butterworth and Co., Ltd., 1962.
- [62] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, 2003.
- [63] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.
- [64] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. In *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005)*, volume 24, pages 471–478, 2005.
- [65] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 237–244, 2005.

- [66] R. Narain, A. Golas, and M. C. Lin. Free-flowing granular materials with two-way solid coupling. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2010)*, 29(6):173:1–173:10, 2010.
- [67] P. of Vision Raytracer (Version 3.7.RC5). Retrieved from <http://www.povray.org/download/>. 2012.
- [68] M. Ohta, Y. Kanamori, and T. Nishita. Deformation and fracturing using adaptive shape matching with stiffness adjustment. In *Computer Animation and Virtual Worlds*, volume 20, pages 365–373, 2009.
- [69] K. Onoue and T. Nishita. Virtual sandbox. In *Proc. of Pacific Graphics 2003*, page 252, 2003.
- [70] M. Oshita. Real-time hair simulation on GPU with a dynamic wisp model. In *Computer Animation and Virtual Worlds*, volume 18, pages 583–593, 2007.
- [71] D. K. Pai. Strands: Interactive simulation of thin solids using cosserat models. *Computer Graphics Forum*, 21(3):347–352, 2002.
- [72] E. Plante, M.-P. Cani, and P. Poulin. Capturing the complexity of hair motion. *Graphical Models (GMOD)*, 64(1):40–58, January 2002.
- [73] E. Plante, M. paule Cani, and P. Poulin. A layered wisp model for simulating interactions inside long hair. In *Proc. of Eurographics Workshop on Animation and Simulation*, pages 139–148, 2001.
- [74] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, 2003.
- [75] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 1995 Conference Proceedings*, pages 147–154, 1995.

- [76] A. Rivers and D. James. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007)*, 26(3):82:1–82:6, 2007.
- [77] C. R. Robbins. *Chemical and physical behavior of human hair*. Springer, fourth edition, 2002.
- [78] R. E. Rosenblum, W. E. Carlson, and E. Tripp III. Simulating the structure and dynamics of human hair: Modeling, rendering and animation. *The Journal of Visualization and Computer Animation*, 2(4):141–148, 1991.
- [79] G. V. Scott and C. R. Robbins. Stiffness of human hair fibers. *Journal of the Society of Cosmetic Chemists*, 29(8):469–485, 1978.
- [80] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2008)*, 27(3):64:1–64:11, 2008.
- [81] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005)*, 24(3):910–914, 2005.
- [82] P. Silva, Y. Bando, B.-Y. Chen, and T. Nishita. Curling and clumping fur represented by texture layers. *The Visual Computer*, 26:659–667, 2010.
- [83] G. Sobottka and A. Weber. Efficient bounding volume hierarchies for hair simulation. In *Proceedings of the 2005 Workshop On Virtual Reality Interaction and Physical Simulation*, pages 1–10, 2005.
- [84] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid solid interactions. *Computer Animation and Virtual Worlds*, 18:69–82, 2007.
- [85] O.-Y. Song, D. Kim, and H.-S. Ko. Derivative particles for simulating detailed movements of fluids. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):711–719, 2007.

- [86] J. Spillmann and M. Teschner. Corde: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *SCA'07: Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 63–72, 2007.
- [87] J. Spillmann and M. Teschner. An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum*, 27(2):497–506, 2008.
- [88] J. Stam. Stable fluids. In *Proc. of SIGGRAPH 1999*, pages 121–128, 1999.
- [89] E. Sugisaki and Y. Yu. Simulation-based cartoon hair animation. In *In 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 117–122, 2005.
- [90] R. W. Sumner, J. F. O’Brien, and J. K. Hodgins. Animating sand, mud, and snow. *Computer Graphics Forum*, 18(1):17–26, 1999.
- [91] S. Tariq and L. Bavoil. Real time hair simulation and rendering on the GPU. In *ACM SIGGRAPH 2008 talks*, page 1, 2008.
- [92] H. D. Taskiran and U. Gudukbay. Physically-based simulation of hair strips in real-time. In *WSCG (Short Papers)*, pages 153–156, 2005.
- [93] P. Volino and N. Magnenat-Thalmann. Real-time animation of complex hairstyles. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):131–142, 2006.
- [94] K. Ward, F. Bertails, T.-Y. Kim, S. R. Marschner, M.-P. Cani, and M. Lin. A survey on hair modeling: Styling, simulation, and rendering. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 13(2):213–234, 2007.
- [95] K. Ward, N. Galoppo, and M. Lin. Interactive virtual hair salon. *Presence: Teleoperators and Virtual Environments*, 16(3):237–251, 2007.
- [96] K. Ward and M. C. Lin. Adaptive grouping and subdivision for simulating hair dynamics. volume 0, pages 234–243, 2003.

- [97] K. Ward and M. C. Lin. Modeling hair influenced by water and styling products. In *International Conference on Computer Animation and Social Agents (CASA)*, pages 207–214, 2004.
- [98] K. Ward, M. C. Lin, J. Lee, S. Fisher, and D. Macri. Modeling hair using level-of-detail representations. In *Proc. of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, pages 41–47, 2003.
- [99] C. Wojtan, M. Carlson, P. J. Mucha, and G. Turk. Animating corrosion and erosion. In *Proc. of Eurographics Workshop on Natural Phenomena*, pages 15–22, 2007.
- [100] E. Wu, Y. Liu, and X. Liu. An improved study of real-time fluid simulation on gpu: Research articles. *Computer Animation and Virtual Worlds*, 15(3-4):139–146, 2004.
- [101] J. Yu and G. Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *SCA'10: Proc. of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 217–225, 2010.
- [102] C. Yuksel and J. Keyser. Deep opacity maps. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2008)*, 27(2):675–680, 2008.
- [103] Y. Zhu and R. Bridson. Animating sand as a fluid. In *ACM Transactions on Graphics (Proc. of SIGGRAPH 2005)*, volume 24, pages 965–972, 2005.

Publication list

International journal

1. Witawat Rungjiratananon, Yoshihiro Kanamori, Tomoyuki Nishita: "Wetting Effects in Hair Simulation", Computer Graphics Forum (Proc. of Pacific Graphics 2012), Vol. 31, No. 7, 2012, in press. (Chapter 6)
2. Witawat Rungjiratananon, Yoshihiro Kanamori, Napaporn Metaaphanon, Yosuke Bando, Bing-Yu Chen, Tomoyuki Nishita: "Animating Strings with Twisting, Tearing and Flicking Effects", Computer Animation and Virtual Worlds, pp.113-124, Vol. 23, No. 2, 2012. (Chapter 4)
3. Witawat Rungjiratananon, Yoshihiro Kanamori, Tomoyuki Nishita: "Chain Shape Matching for Simulating Complex Hairstyles", Computer Graphics Forum, 2010, pp. 2438-2446, Vol. 29, No. 8, 2010. (Chapter 4)
4. Witawat Rungjiratananon, Zoltan Szego, Yoshihiro Kanamori, Tomoyuki Nishita: "Real-time Animation of Sand-Water Interaction", Computer Graphics Forum (Proc. of Pacific Graphics 2008), pp.1887-1893, Vol. 27, No. 7, 2008. (Chapter 5)

International conference (Peer review)

1. Witawat Rungjiratananon, Yoshihiro Kanamori, Napaporn Metaaphanon, Yosuke Bando, Bing-Yu Chen, Tomoyuki Nishita: "Twisting, Tearing and Flicking Effects in String Animations", In Proc. of Motion in Games 2011 (LNCS7060), pp.192-203, 2011-9. **(Best paper award)**
2. Witawat Rungjiratananon, Yoshihiro Kanamori, Tomoyuki Nishita: "Elastic Rod Simulation by Chain Shape Matching with Twisting Effect", SIGGRAPH Asia sketch, 2010-12.
3. Witawat Rungjiratananon, Yoshihiro Kanamori, Tomoyuki Nishita: "Simulation of Interactions Between Fluids and Granular Materials with Wetting Effects", Proc. of NICOGRAPH International 2008-5.

Domestic conference (Peer review)

1. Witawat Rungjiratananon, 金森 由博, 西田 友是: "スライス法に基づいた効率的な髪の毛の衝突処理", Visual Computing/グラフィクスと CAD 合同シンポジウム 2012-6.
2. Witawat Rungjiratananon, 金森 由博, 西田 友是: "詳細な濡れた髪の毛のアニメーション", Visual Computing/グラフィクスと CAD 合同シンポジウム 2011-6.
3. Witawat Rungjiratananon, 金森 由博, 西田 友是: "ねじれを考慮した形状一致法による弾性ロッドのシミュレーション", Visual Computing/グラフィクスと CAD 合同シンポジウム 2010-6.

4. Witawat Rungjiratananon, 金森 由博, 西田 友是: "形状一致法に基づく様々なヘアスタイルの対話的シミュレーション", *Visual Computing/グラフィクスと CAD 合同シンポジウム* 2009-6.
5. Witawat Rungjiratananon, 金森 由博, 西田 友是: "吸湿現象を考慮した粒状物体と流体の相互作用シミュレーション", *Visual Computing/グラフィクスと CAD 合同シンポジウム* 2008-6.

Domestic conference (No peer review)

1. Witawat Rungjiratananon, 金森 由博, 西田 友是: "髪の毛同士の相互作用を考慮した詳細な髪の毛のアニメーション", 第 72 回情報処理学会全国大会, 2010-3.