GPU Cloth Simulation Pipeline in Lightchaser Animation Studio

Haowei Han
haowei@lightchaseranimation.com
Lightchaser Animation Studio
China

Meng Sun
Lightchaser Animation Studio
China

Dongying Liu
University of Pennsylvania
USA

Siyu Zhang
UC Berkeley
USA

Tiantian Liu
Tachi Graphics
China

Figure 1: Ancient Chinese female character in our project. Simulation of her clothes with more than 300K triangles took 10 ~ 20 substeps and 4 collision detection passes per substeps. In these shots, single CCD(Continuous Collision Detection) calculation costs less than 15ms with NVIDIA RTX 3090.

ABSTRACT

We present the simulation pipeline of character effects in Lightchaser Animation Studio and how we utilize GPU resources to accelerate clothing simulations. Many characters from ancient Chinese tales in our films are in complex costumes.(Figure 1) Such costumes contain five to six layers of fabrics, where several hundred thousand triangles are used to show the delicate folds of different cloth materials. At the same time, there are more than 600 similar shots bringing more than 1000 cloth simulation tasks in a film project. Therefore, the efficiency and accuracy of our cloth simulator is the key to the CFX production pipeline.

SA '21 Technical Communications, December 14–17, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-9073-6/21/12…$15.00
https://doi.org/10.1145/3478512.3488616

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**.

KEYWORDS

cloth simulation, collision response

ACM Reference Format:
Haowei Han, Meng Sun, Dongying Liu, Siyu Zhang, and Tiantian Liu. 2021. GPU Cloth Simulation Pipeline in Lightchaser Animation Studio . In *SIGGRAPH Asia 2021 Technical Communications (SA '21 Technical Communications), December 14–17, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3478512.3488616

1 PROBLEMS IN PRODUCTION

Previously, we built a cloth simulation pipeline based on Houdini Vellum and found two major problems: squeezing and piling of multi-layered clothes could easily cause local jittering; the performance of Vellum is sensitive to the number of substeps.

Due to the complex design of ancient Chinese costumes, it is common to see layers of cloth piling up together, especially in the armpit and elbow areas. There is no way to reduce the jittering

artifact from Vellum unless we add enough substeps. However, it will cause the decreasing performance of Vellum, which is mainly because of the frequent constraint geometry data exchange between the CPU and the GPU. If we want to increase only the substeps but with fixed iterations and collision passes, it will lead to significant performance degradation, especially for assets with more than 300,000 faces.

2 JITTERING ARTIFACTS

Our solver uses PBD method for collision resolve process, so VF (vertex-face) and EE (edge-edge) collision contacts are regarded as one type of constraints for resolving [Müller et al. 2006]. PBD-based collision resolve is an efficient collision handling method, but it could cause jittering artifacts in some complex scenes. In our project implementation, the following three causes of jittering artifacts are found. Firstly, there is not sufficient gap for resolving new positions for each point (moving points around) between layers of clothes where the clothes are stacked or squeezed. Secondly, EE contacts that are almost parallel can easily cause errors in collision response. Thirdly, PBD-based collision response method has poor convergence and it is hard to reach the condition that all constraints satisfy $C_{collision} \geq 0$.

2.1 Guarantee Sufficient Gap Between Models/Objects

Each layer of cloth has its own attribute of thickness. When the layers of cloth are stacked or squeezed, it is possible to have inadequately small gaps between either the collider and the cloth, or among the layers of cloth, which causes the solver to fail to find a solution that satisfies every collision constraint with the condition $C_{collision} >= 0$. Consequently, it cannot guarantee the continuity of the solution at each time step and cause jittering artifacts. In order to solve this, we pre-process the body collider of each character model by reserving space to ensure enough gaps between cloth-cloth and cloth-collider when cloth squeezing and stacking happens. We called it "Animation solvability check."

2.2 Tricks on resolving Edge-Edge contacts

We define a contact constraint as $C(\mathbf{p}_a, \mathbf{p}_b, \mathbf{p}_c, \mathbf{p}_d) = (\mathbf{p}_\alpha - \mathbf{p}_\beta) \cdot \hat{n} - 2r$, where r is edge thickness, $\mathbf{p}_\alpha$ and $\mathbf{p}_\beta$ are nearest points on the edge **ab** and **cd**. $\hat{n}$ is the normalized cross of the two edges and then its sign is decided according to the history. When resolving EE detected contacts, an EE contact pair could produce errors and joggles because when their positional relationship is closer to parallel (Figure 2), the normal direction might significantly change and even cause the flip of the normal direction. Hence, we add the following two restrictive conditions on EE detection before the contact resolve process. Also, we divide EE resolve into self-collision EE (i.e. EE contact produced by self collision of the cloth) and collider collision EE (i.e. EE contact produced by colliders and the cloth)

- For EE contact produced by cloth self-collision, limitations on the construction of a valid EE contact are added. An EE contact pair is allowed to be added to the entire EE contact list only when the angle between two edges is greater than theta (e.g. 10 degrees) at x0 (initial) time and xt (current) time. This prevents ineffective EE contact (i.e. EE pairs that are close to parallel relationships). We assume that it is more likely to have an EE pair detected as a VE (Vertex Edge) pair when the two edges are closer to parallel. Although there might be missing EE contact pairs in theory, there is no serious issue in the final simulated scenes in real production in terms of visual effects.
- For EE contact produced by a collider and the cloth,we directly use the dihedral normal of the edge on the collider as the normal resolving direction.

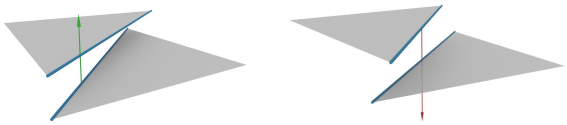

Figure 2: Show an EE contact is unstable when the edge pair is close to a parallel relationship.

2.3 Impact Zone Laplacian Smooth

Due to the nature of PBD collision resolve, the collision constraint and the other types of constraints would affect each other when solving together. It requires a large enough amount of iterations to reach the minimum of all constraint values. However, with this premise, the final linear system would result in a jittering phenomenon in complex collision scenes because of the poor convergence of collision resolve. Therefore, we design a smooth process after finishing all collision detection passes. Specifically, according to the contact connectivity information generated by the last collision detection pass (Figure 3), the Laplacian smooth is applied on the velocity data of the contact constraints block (some connected contact constraints) that is not convergent yet. After this, this block of contact constraints would have a relatively uniform velocity, and then it improves the jittering issue significantly (Figure 4).

Since it is possible for the Laplacian smooth applied for solving the jittering artifacts to cause cloth viscidity, only the contacts that satisfy $C_{collision} < -\epsilon * thickness$ would be processed by Laplacian smooth. In plain English, the Laplacian smooth would only be applied to the velocity of the contacts that still have close enough distances. Generally, the value of ϵ here is 0.1. To be more specific, after the process of collision resolution, the Laplacian smooth is executed when the relative contact depth of a contact pair is still greater than 10% of the pre-defined thickness. When the gap between the contacts is large enough, the collision resolve process is able to move the two contacting layers of cloth away from each other. Hence, the cloth viscidity phenomenon has not been observed in real projects so far.

3 PIPELINE

Data cleaning. Pre-processing can guarantee self-intersections-free of the body during animation, and can also reserve a space for interaction between the clothes and the body.

Firstly, the count of points on the character's body will be calculated by tracing the normal direction of the current point and

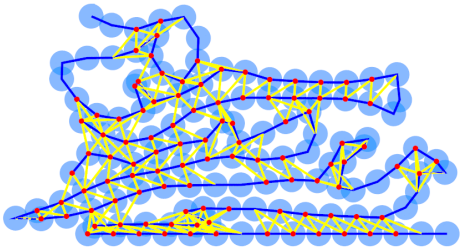

Figure 3: The Range of the Impact Zone: the radius of blue circles: the thickness of cloth at each point; the red dots: the points that are still in the range of reserved thickness (twice of the point thickness) after certain amounts of collision resolve iterations; yellow lines: the sampling connectivity of Laplacian filtering on velocity.

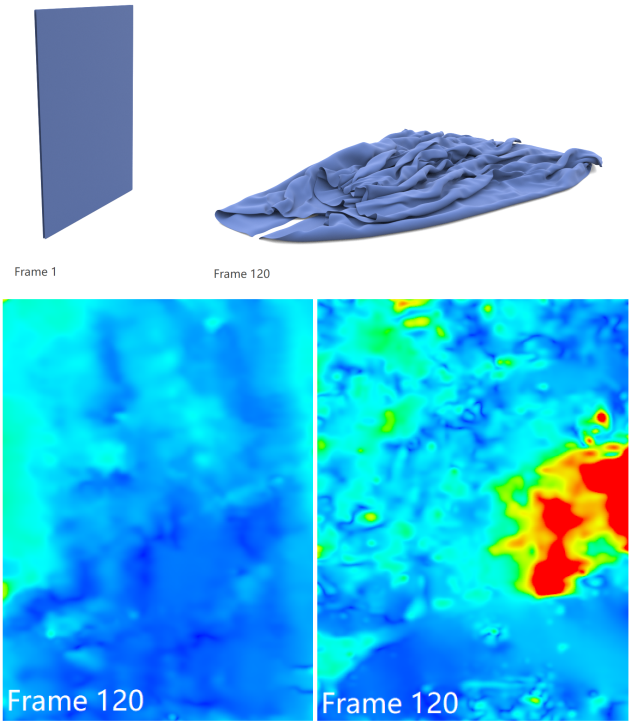

Figure 4: Comparison of turning Laplacian Smooth on (left) and off (right). The two false-color charts above represent the velocity magnitude. At frame 120 the cloth should be in the static state and colored blue.

counting the number of layers of cloth with which the point is intersected. At the same time, the thickness property of each intersected layer would be accumulated, leading to the final reserved thickness for the current point.

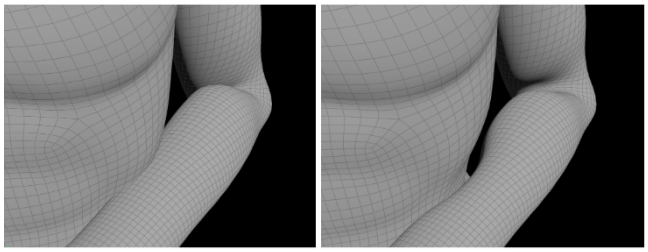

Figure 5: body collider geometry before data cleaning and after data cleaning (A gap of 4 layers of fabric thickness is reserved here)

With the premise that the character's body consists of closed surfaces, the number of the intersections (denoted as n here for later explanation) between every point on the body and the bounding box of the entire body model would be then calculated. This is determined by the parity of n. Specifically, if n is odd, then the point is inside the body model and vice versa. Interior points (points that are intersected with the body model) iteratively move towards the opposite direction of the normal with a certain amount of step size and stop when the entire body is intersection-free in terms of self-penetration. Then, for each point on the body model, a VF contact search would be conducted using the ray-triangle method (find the intersection between a point and a triangle) to find the potential intersections within the range of the twice thickness of the current point. If any VF contacts that do not satisfy the condition of the valid reserved thickness (twice of the point thickness) are found, these contact pairs would be saved and resolved as VF collision later.(Figure 5)

Asset Configuration. To make the CFX artists' lives easier, we develop our in-house software in the manner of having as similar as possible operation designs to the frequently used commercial software. Therefore, during the assets configuration process, we use the Houdini "Vellum Constraint" node directly, for constructing different types of constraints, including distance, triangle stretch, attachment, etc., and the related parameters (e.g. stiffness, attach path, etc.)

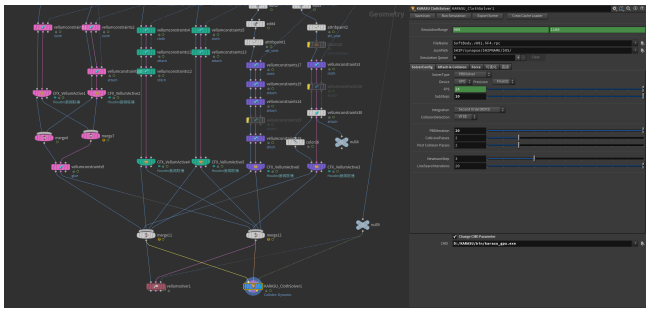

Figure 6: Our solver's node can identify and solve the vellum assets directly. So that the CFX artists who have already mastered the Houdini Vellum can use our in-house cloth simulation solver without any difficulties.

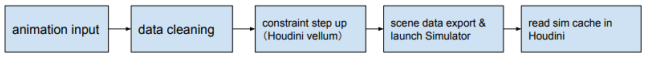

Figure 7: Complete production pipeline.

We also develop an extension of the "Vellum Constraint" node. When the solver switches to the FEM calculation mode, it is required to set additional FEM material properties such as Young's modulus, Poisson's ratio, etc.

Execute the Simulation. After the construction of constraint assets in Houdini, we export the scene as our customized file format. Then, the solver is launched in the manner of command-line statements and the simulation process begins. At the same time, in Houdini, the solver IO node reads the simulation cache and results will be displayed in the Houdini viewport in real-time. In this way, the artists can still use their production experience with the Vellum solver by Houdini and keep the operation habits as well, even though they are using our in-house cloth simulation solver(Figure 7).

GPU implementation. Except for the data write out cache,all of function are implemented on the GPU,so the number of iterations and collision paeese can be spread arbitrarily to multiple substeps, by reducing the time step in exchange for system stability at the same time, the performance is not significantly affected(Figure 8). We first construct R-Triangle information as proposed in [Curtis et al. 2008] for preventing VF and EE contact replication. Secondly, we use Bounding Volume Hierarchy (BVH) as the acceleration structure in the broad phase for culling triangle pairs that are unnecessary to execute CCD calculation [Karras 2012]. During the process of contact traversal in BVH, we use iterative access instead of recursive traversal to improve the optimization efficiency and avoid stack overflow. Finally, we use red-black Gauss-Seidel to execute contact resolve calculation on GPU.

(s/frame)	200 iterations 20 collision passes 1 substeps	20 iterations 2 collision passes 10 substeps	10 iterations 1 collision passes 20 substeps
houdini vellum	2.2s	7.0s	9.6s
our solver	0.34s	0.37s	0.39s

Figure 8: Comparison of Performance between Vellum and our simulator. The data was recorded by 10 frames and then averaged.

4 CONCLUSION

We have presented our cloth simulation pipeline with GPU acceleration in Lightchaser Animation Studio, especially for multilayered traditional Chinese costumes (Figure 10). Our new pipeline not only reduces the time for simulation but also preserves a stable result even in the most extreme cases, such as twisting cloth (Figure 9).

ACKNOWLEDGMENTS

LightChaser Animation Studio Character effects team, KARASU cloth solver early developer Congying Zhang, project manager Xiaoqi Yang, Ye Yuan, CFX leader Yiming Lu, Cen Cui, CFX artist Xinhua Jin, Xianjun Zhu, Pipeline TD Hao Chen.

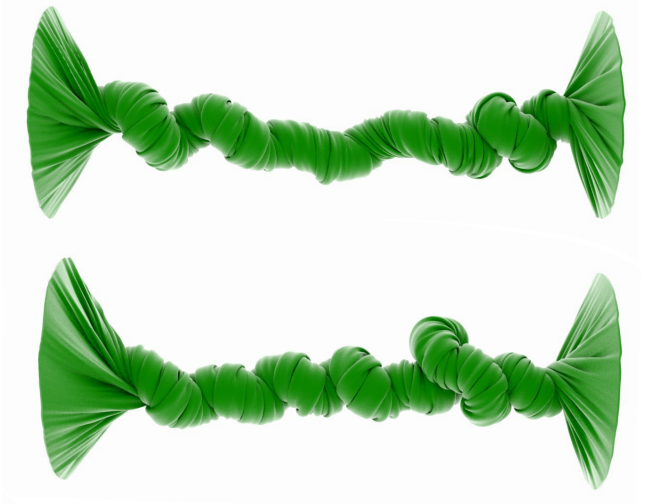

Figure 9: A twist self collision test using our GPU cloth solver

Figure 10: Production frames with various characters in multi-layered cloth.

REFERENCES

Sean Curtis, Rasmus Tamstorf, and Dinesh Manocha. 2008. Fast collision detection for deformable models using representative- triangles. *Proceedings of the Symposium on Interactive 3D Graphics and Games, I3D 2008* (2008), 61–69. Issue Cd. https://doi.org/10.1145/1342250.1342260

Tero Karras. 2012. Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. *High-Performance Graphics 2012, HPG 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings* (2012), 33–37. https://doi.org/10.2312/EGGH/HPG12/033-037

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. Position based dynamics. *3rd Workshop in Virtual Reality Interactions and Physical Simulations, VRIPHYS 2006* (2006), 71–80. https://doi.org/10.1007/978-3-319-08234-9_92-1