한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 31, No. 1, P. 25~33

액체-옷감 상호작용에서 파동 난류, 확산 및 주름을 표현하기 위한 통합 GPU 프레임워크

박은수[1O] 이주용[1] 박인규[1] 김종현[2*]

[1]인하대학교 정보통신공학과

[2]인하대학교 소프트웨어융합대학(디자인테크놀로지학과)

{qkrdmstn1014, juyong9864}@inha.edu, {pik, jonghyunkim}@inha.ac.kr

Unified GPU Framework for Simulating Wave Turbulence, Diffusion, and Wrinkling in Fluid-Cloth Interaction

Eun Su Park[1O] Juyong Lee[1] In Kyu Park[1] Jong-Hyun Kim[2*]

[1]Dept. of Information and Communication Engineering, Inha University,

[2]College of Software and Convergence (Dept. of Design Technology), Inha University

요 약

본 논문에서는 GPU 기반 시뮬레이션 프레임워크를 제안하여, 액체-옷감 상호작용에서 발생하는 난류(Turbulence), 확산(Diffusion), 주름(Wrinkle)과 같은 세부적인 물리 현상을 효율적으로 표현한다. 기존의 방법은 운동량 보존 법칙에만 의존하여 상호작용을 표현하기 때문에 세부적인 시각적 특징을 놓치는 경우가 있다. 이를 해결하기 위해, FLIP(Fluid-implicit particle) 방식을 사용하여 기존 입자 기반의 액체 기법보다 더 세밀한 액체 흐름을 구현하고, PBD(Position-based dynamics) 방식을 통해 옷감의 기본 물리 특성을 시뮬레이션한다. 또한, 난류, 확산, 주름과 같은 세부적인 특징을 효율적으로 처리하기 위해 공간 해싱(Spatial hashing), 그래프 컬러링(Graph coloring), 메모리 최적화 등의 GPU 병렬화 기술을 활용한다. 그 결과, CPU 방식보다 수십 배 이상의 성능을 향상했으며, 세부적인 시각적 특징을 빠르게 표현한다.

Abstract

This paper proposes a GPU-based simulation framework to efficiently represent fine-grained physical phenomena, such as turbulence, diffusion, and wrinkling, that occur in fluid-cloth interactions. Unlike previous methods that rely solely on momentum conservation, this framework captures the intricate interactions between fluids and fabrics. The FLIP approach is used to simulate more detailed fluid flows than traditional particle-based fluid techniques, while the PBD method captures the fundamental physical properties of fabrics. To efficiently handle large-scale details like turbulence, diffusion, and wrinkling, the framework utilizes GPU parallelization techniques, including spatial hashing, graph coloring, and memory optimization. The result is a significant performance improvement, with tens of times faster simulation times compared to CPU methods, enabling the rapid representation of fine-grained visual features.

키워드: GPU, 옷감, 액체, 난류, 확산, 주름, 위치 기반 동역학, FLIP(Fluid-implicit particle)

Keywords: GPU, Cloth, Liquid, Turbulence, Diffusion, Wrinkle, Position-based dynamics, FLIP(Fluid-implicit particle)

1. 서론

액체와 옷감이 동시에 등장하는 장면은 컴퓨터 그래픽스 분야에서 시각적 사실성을 높이기 위한 주요 연구 주제이다. 그러나 대부분의 연구에서는 액체-옷감 사이의 운동량만을 다루고 있으며, 난류, 확산, 주름 등의 세부적인 시각적 특징의

*corresponding author: Jong-Hyun Kim / College of Software and Convergence, Inha University (jonghyunkim@inha.ac.kr)

Received : 2025.01.18./ Review completed : 1st 2025.02.05. / Accepted : 2025.02.11.
DOI : 10.15701/kcgs.2025.31.1.25
ISSN : 1975-7883(Print)/2383-529X(Online)

표현은 많은 연산을 요구하기 때문에 CPU 기반 시뮬레이션만으로는 성능적인 한계가 존재한다. 이를 해결하기 위해, 최근에는 GPU 병렬화 기법을 활용하여 액체-옷감 상호작용을 더욱 사실적으로 표현하기 위한 연구가 활발히 진행되고 있다[1]. 본 논문에서는 FLIP 기법[2]과 PBD[3] 방식을 결합하여 액체와 옷감을 시뮬레이션하고, 난류, 확산, 주름 등의 세부 물리 현상을 GPU 병렬화 기법을 활용하여 효율적으로 처리하는 프레임워크를 제안한다.

2. 관련 연구

본 논문에서는 액체 내 난류 표현과 젖은 옷감 표현에 관련된 연구들을 간략히 살펴본다. 컴퓨터 그래픽스 분야에서 액체 시뮬레이션을 통해 시각적 사실성을 높이는 연구가 지속적으로 이루어지고 있다. Zhu 와 Birdson 이 제안한 FLIP 기법은 격자 기반 방법과 입자 기반 방법의 장단점을 결합하여 비압축성 유체의 압력 계산은 격자 기반으로, 이류 및 외력 처리는 입자 기반으로 수행한다[2]. 이 기법은 높은 안정성과 효율성을 제공하지만, 사실적인 표면 세부 표현, 특히 난류와 같은 파동 표현에는 여전히 한계가 있다. Mercier 등은 입자 기반 액체 시뮬레이션 표면에 파동 시뮬레이션을 결합하여 난류를 표현하는 방법을 제안했지만, 이 방법은 고해상도 시뮬레이션에서 높은 계산 복잡도로 인해 성능적인 한계를 보였다[4]. 최근에는 GPU 병렬화를 활용하여 고해상도 FLIP 기반 시뮬레이션의 성능을 개선하려는 시도와 물리 기반 난류 모델을 입자 기반 유체에 통합하려는 연구가 진행되고 있다[5]. 이에 반해, 본 논문은 GPU 병렬화를 활용하여 고해상도에서도 효율적으로 난류를 표현하는 방법을 제안하여, 기존 난류 표현 기법의 성능적 한계를 극복하고 실시간 응용 가능성을 확장하고자 한다.

Patkar 등이 제안한 젖은 다공성 고체에 대한 기법은 포화도를 기반으로 젖은 옷감을 세 가지 상태로 나누었다[6]. 이 기법은 옷감이 액체와 충돌할 때 흡수, 흡수된 액체의 확산, 최대 포화도를 초과하면 배출로 구성된다. 그러나 이 기법에서는 옷감의 젖음에 따른 기하학적 특징인 주름에 대한 설명이 충분하지 않으며, 옷감에 수분이 흡수되는 과정에서의 질량 변화에 따른 특징과 다공성 고체에 흡수되는 과정에서의 부피 확장만을 표현했다.

Chen 등은 가상의 인간이 젖은 옷을 입은 시뮬레이션을 제안했다[7]. 이 시뮬레이션은 실제 실험에서 측정한 결과를 기반으로 최대한 실제와 유사한 주름을 표현하지만, 실제 액체와의 상호작용을 고려하지 않았다.

Rumman 등은 액체와 변형체의 상호작용을 제안했다[6]. 이 방식은 액체 시뮬레이션에 ISPH(Incompressible Smoothed-Particle Hydrodynamics)기법을 사용하여 상세한 표현을 수행하지만, 액체와 옷감의 충돌 시 상호작용만을 설명하고 옷감이 젖어서 변하는 과정이나 액체 시뮬레이션의 세부적인 시각적 특징을 반영하지 않았다.

최근에 인공지능을 통해 유체를 표현한 연구들이 제안되었지만[14,15], 대부분 비압축성적인 특징만을 학습하려고 했으며, 액체-옷감 상호작용에 대한 디테일을 표현하기에는 어렵다.

3. 사전 연구

3.1 FLIP 기반 액체 시뮬레이션

본 논문에서 액체를 표현하기 위한 시뮬레이션은 FLIP 기법을 활용하며 파동 난류를 표현하기 위해 Mercier 등이 제안한 난류를 활용한다[4] (Figure 1 참조).

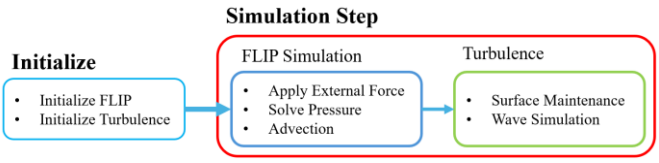


Figure 1. Liquid simulation overview. It consists of an initialization and a simulation step, with turbulence performed as a post-processing following the FLIP simulation.

액체 시뮬레이션은 유체의 움직임을 설명하는 나비에-스톡스 방정식을 통해 수행되며, 이는 수식 1 로 표현된다. 이 방정식은 입자 기반 방법과 격자 기반 방법으로 나뉜다.

FLIP 기법은 액체 시뮬레이션을 크게 이류 과정과 압력 계산 과정으로 나누어 수행한다. 이류 과정은 입자 기반 방법으로 계산되며, 비압축성 유체에 대한 압력은 격자 기반으로 계산된다. 이 기법은 각 기법의 단점을 상호 보완하여 보다 정확한 시뮬레이션을 가능하게 한다. 본 논문에서는 FLIP 기법을 기반으로 3 차원 입자 시뮬레이션을 수행한다.

$$u_t + (u \cdot \nabla)u + \frac{\nabla p}{\rho} = f \qquad (1)$$

여기서 u는 속도장을 의미하며 p는 압력, ρ는 밀도를 의미하고 f는 전체 유체의 힘을 의미한다.

3.2 파동 난류

입자 기반 액체 시뮬레이션에서 파동 난류는 후처리 과정으로, 표면 유지와 파동 시뮬레이션으로 구성된다.

표면 유지 과정에서는 난류 입자의 이류, 표면의 법선 벡터 계산, 표면의 정규화를 수행한다. 난류 입자는 주변 FLIP 입자의 움직임을 따라가며, 메타볼(Metaball) 레벨셋을 사용하여 표면의 법선 벡터를 계산하고, 제약조건 투영을 통해 액체 시뮬레이션 표면에 난류 입자의 위치가 유지되도록 한다 (Figure 2 참조).

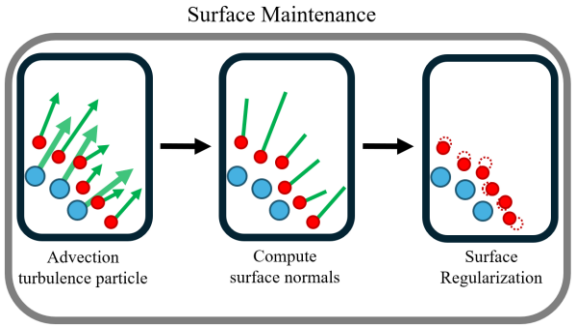


Figure 2. Surface maintenance process of turbulence particles (blue : FLIP particle, red : turbulence particle).

파동 시뮬레이션 과정에서는 곡률 계산, 파동 생성, 표면 파동의 진행을 거친다. 파동은 곡률이 높은 입자로부터 코사인 파동 생성을 시작하며, 1 차원 파동 방정식을 사용하여 표면에 파동을 생성한다 (Figure 3 참조).

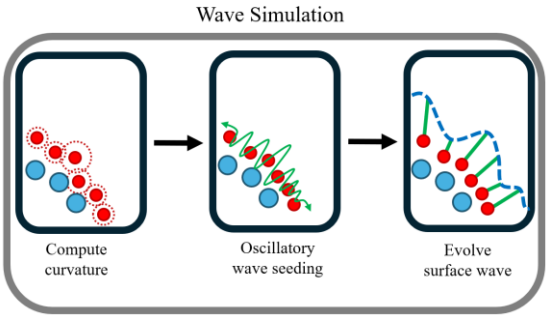


Figure 3. Wave simulation process of turbulence particles (blue : FLIP particle, red : turbulence particle).

이때, 1 차원 파동 방정식은 $u(x,t)$에 대한 편미분 방정식으로 c 는 파동의 속도를 나타내며 $u(x,t)$는 시각 t, 위치 x에서의 파동의 진폭을 나타내는 함수이다 (수식 2 참조). 본 논문에서는 이를 기반으로 액체 시뮬레이션에 파동 난류를 표현한다.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \qquad (2)$$

3.3 젖은 옷감

본 논문에서는 젖은 옷감을 표현하기 위하여 PBD 기법과 S. Patkar 등이 제안한 기법을 활용한다[6] (Figure 4 참조).

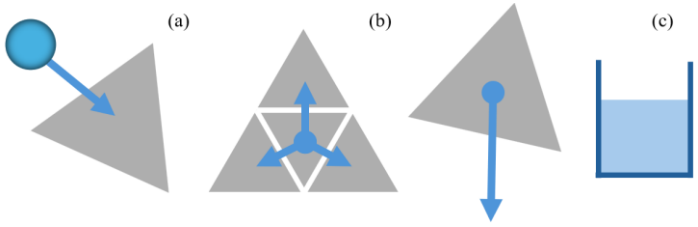


Figure 4. Wet-cloth Simulation steps((a) Absorption, (b)Diffusion, (c) Dripping)

가장 먼저 흡수되는 과정은 면(Face)의 포화도 S에 기반한 수식 3 과 4 를 통해 옷감에 흡수되는 액체 질량 $m_i^{absorbed}$과 포화도를 계산한다. 이때, $\tilde{S}$ 는 최대 포화도이고, 잔여 질량 $m_i^{residual}$가 0 이거나 음수가 되면 액체 입자를 삭제한다 (Figure 5 참조).

$$S = S + \left(\frac{m_i^{absorbed}}{A}\right) \qquad (3)$$

$$m_i^{absorbed} = (\tilde{S} - S)A \qquad (4)$$

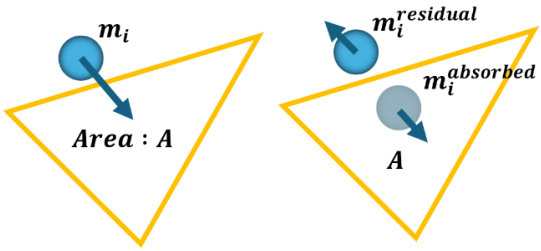


Figure 5. (Left) Fluid particles before colliding with the cloth (Right) Fluid particles after collision with the cloth

충돌처리는 Rumman 등이 제안한 충돌 제약조건 반복 투영 방식[8]과 Lewin 등이 제안한 정점(Vertex)와 면의 충돌 제약조건[9]을 수식 5 를 만족하도록 반복 투영한다 (Figure 6 참조). 제약조건은 면의 세 정점 p_0, p_1, p_2 과 액체 입자 p에 대한 함수로 면의 법선 벡터 $\hat{n}$과 액체 입자의 반지름 r을 고려한다.

$$C(p, p_0, p_1, p_2) = \hat{n} \cdot (p - p_0) - 2r \geq 0 \qquad (5)$$

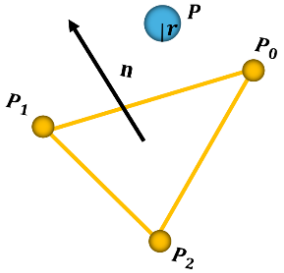


Figure 6. The collision between fluid particles and the cloth is resolved through constraints imposed by vertex-to-face collisions.

충돌로 인해 흡수된 수분은 인접한 면으로 확산된다 (Figure 7 참조). 이때, 포화도 차이에 의한 확산과 중력에 의한 확산도 고려하기 위해서 중력 방향으로의 가중치 $cos\theta_{ln}$ 를 더해서 포화도 변화량 ΔS_{ln}을 수식 6 에 의해서 계산한다. 이때, 포화도 차이에 의한 확산계수 k_d와 중력에 의한 확산계수 k'_d를 통해 영향을 조절할 수 있다.

$$\Delta S_{ln} = \min(0, k_d(S_l - S_n) + k'_d S_l cos\theta_{ln}) \quad (6)$$

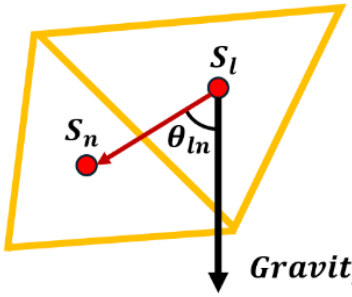


Figure 7. The diffusion of saturation from face $\boldsymbol{l}$ to face $\boldsymbol{n}$.

포화도가 최대 포화도를 초과하면 수식 7 을 통해 초과한 질량 m_{excess}을 계산하고 수식 8 에 의해서 버퍼 $dripbuf_l$에 이를 저장한다. 버퍼가 특정 임계점 $m_{drip-threshold}$을 넘으면 액체 입자를 생성하여 물방울 떨어짐 현상을 표현할 수 있다 (Figure 8 참조). 이때, 포화도의 초과 정도를 중력에 의해 확산되도록 하면 옷감을 타고 흐르면서 떨어지는 물방울을 표현할 수 있다.

$$m_{excess} = (S_l - \tilde{S})A_l \quad (7)$$

$$dripbuf_l = dripbuf_l + m_{excess} \quad (8)$$

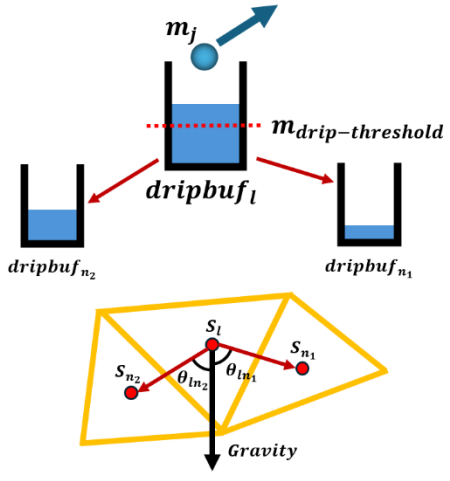


Figure 8. (Top) The diffusion of the drip buffer from surface $\boldsymbol{l}$ to surfaces $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ (Bottom) the generation of fluid particles upon exceeding the drip threshold.

이러한 과정들을 거치면 흡수된 액체의 질량이 옷감의 질량으로 포화도의 형태로 전환되고 최대 포화도를 초과하면 다시 배출되면서 질량을 보존할 수 있게 되어 안정적인 시뮬레이션이 가능해진다.

3.4 GPU기반 공간 해싱 알고리즘

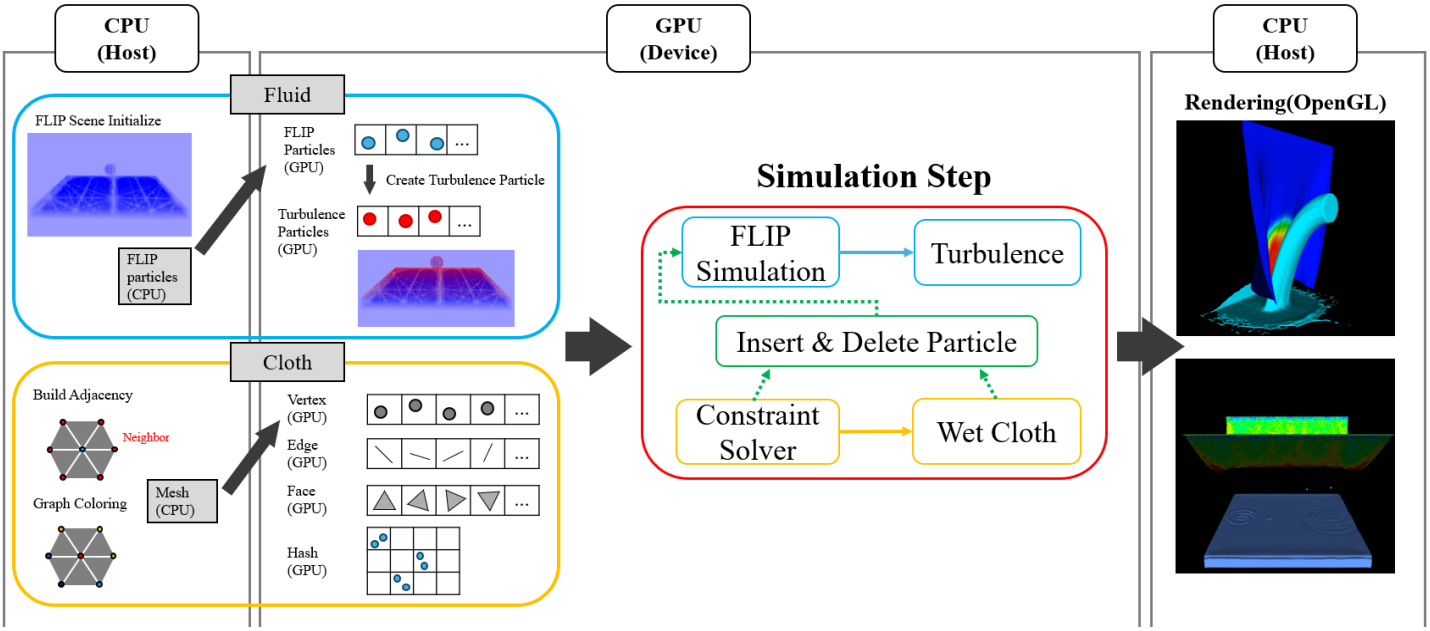
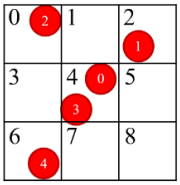


Figure 10. Algorithm overview.

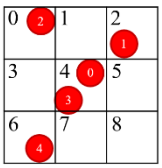
본 논문에서는 옷감과 액체의 이웃 입자를 탐색하기 위해 공간 해싱 알고리즘을 사용하며, CPU 기반에서의 공간 해싱 알고리즘 과정은 다음과 같다 (Figure 9 참조).



(cell index, particle index)

Cell Index	Count	Particle Index
0	1	2
1	0	
2	1	1
3	0	
4	2	0, 3
5	0	
6	1	4
7	0	
8	0	

Figure 9. CPU-based spatial hashing algorithm.



(cell index, particle index)

Index	Unsorted Index	Sorted Index	Cell Start
0	(4, 0)	(0, 2)	0
1	(2, 1)	(2, 1)	
2	(0, 2)	(4, 0)	1
3	(4, 3)	(4, 3)	
4	(6, 4)	(6, 4)	2
5			
6			4
7			
8			

Figure 11. Parallelization of the spatial hashing algorithm without race conditions through sorting and prefix sum of particle information within each cell.

그러나 각 칸(Cell)에 포함된 입자의 개수를 세는 과정에서 경합 조건(Race condition)이 발생하며, 이를 방지하기 위해 Green 이 제안한 정렬을 이용한 공간 해싱 병렬화 기법을 사용한다[10] (Figure 11 참조). 이는 입자가 속한 칸에 대한 정보를 저장한 뒤, cell index 를 이용한 정렬 과정을 거쳐 각 칸에 속한 입자 개수 정보를 얻을 수 있다.

4. 제안하는 프레임워크

본 논문에서 제안하는 프레임워크는 크게 초기화, 시뮬레이션 단계, 렌더링 과정을 거친다.

초기화 과정에서 액체 시뮬레이션은 FLIP 입자를 생성하고, 옷감 시뮬레이션은 인접 정보 생성 및 그래프 컬러링의 초기 설정을 호스트 메모리에서 진행한다. 이후, 디바이스 메모리로 복사하는 과정을 거쳐 액체 시뮬레이션의 난류 입자 초기화 과정을 수행한다.

시뮬레이션 단계에서는 상호작용에 나타나는 난류, 확산, 주름 등 세부적인 시각적 특징 표현을 디바이스 메모리에서 진행한다. 이때, 젖은 옷감에 대한 시뮬레이션을 수행하고, 물의 흡수와 물방울 떨어짐 현상에 따라 액체 입자의 삽입/삭제 과정을 거쳐 액체 시뮬레이션을 수행한다. 최종적으로, 각 시뮬레이션의 입자 정보를 호스트 메모리로 복사하는 과정을 거쳐 OpenGL 을 활용해 렌더링을 진행한다 (Figure 10 참조).

4.1 FLIP 시뮬레이션의 GPU 병렬화

액체 시뮬레이션의 GPU 병렬화 과정은 초기화, 시뮬레이션 단계의 두 과정으로 나뉜다. 초기화 과정의 경우 FLIP 입자의 초기화를 호스트 메모리에서 진행한 뒤, 이를 디바이스 메모리로 복사하여 난류 입자의 초기화를 진행한다.

시뮬레이션 단계의 경우 입자 기반으로 계산되는 외력과 이류 과정은 각 FLIP 입자 단위로 스레드를 할당하며, 그리드 기반으로 계산되는 비압축성 액체에 대한 압력 계산 과정은 그리드 단위로 스레드를 할당한다 (Figure 12 참조).

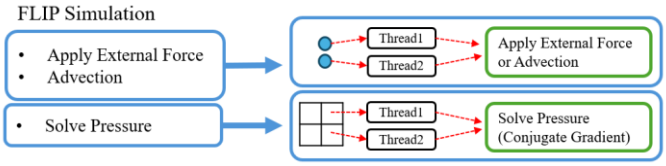


Figure 12. Fluid GPU framework.

4.2 파동 난류의 GPU 병렬화

본 논문에서 초점을 두고 있는 파동 난류의 병렬 처리 부분은 난류 입자의 삽입/삭제 과정과 파동 시뮬레이션 두 가지로 나뉜다. 난류 입자의 초기화 및 표면 유지 과정은 매 프레임 난류 입자의 개수가 달라지는 삽입/삭제 과정이 발생한다. 이를 효율적으로 설계하기 위해 GPU 기반의 정렬과 배타적 스캔(Exclusive scan) 등의 연산을 거쳐 메모리 최적화를 진행한다 (Figure 13 참조).

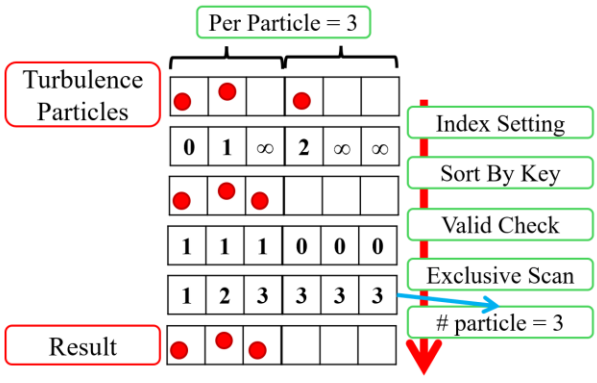


Figure 13. Parallelization of turbulence particle insert/delete process. Setting the maximum number of turbulence particles generated per FLIP particle, followed by efficiently counting and sorting them in device memory.

이후에 진행되는 난류 입자에 의한 파동 시뮬레이션은 각 입자마다 파동의 높이가 계산되는 과정으로, 각 입자마다 스레드를 할당하여 연산을 수행하고, 최종적인 파동 난류 표현을 진행한다 (Figure 14 참조).

Figure 14. Parallelization of wave simulation (blue : FLIP particle, red : turbulence particle, green : turbulence particle with applied waves).

4.3 젖은 옷감의 기하학적 특징

젖은 옷감은 마른 옷감과는 다른 기하학적 특징을 가진다. 본 논문에서는 젖은 옷감이 마른 옷감과 비교하여 더 과장되도록 주름을 표현하였다. Chen 등이 제안한 기법에서는 젖은 옷감에서의 주름이 곡률에 기반한다고 설명하고 있다[7]. 따라서 곡률에 비례하여 정점이 법선의 반대방향으로 과장되도록 하였다. 이때, 곡률을 고려하기 위해서 이면각(Dihedral angle)에 비례하는 가중치를 설정하여 주름을 과장해서 표현한다 (Figure 15 참조).

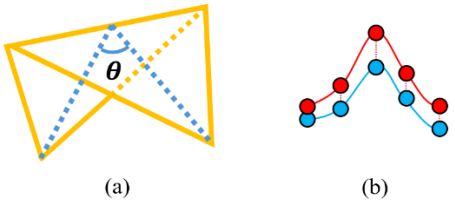


Figure 15. The weight used for representing wrinkles in wet cloth (a) and the result (b) ((a) Dihedral angle, (b) Exaggerated representation of wrinkles.)

4.4 젖은 옷감의 GPU 병렬화

젖은 옷감 시뮬레이션의 GPU 병렬화 과정은 각 과정들마다 연산의 단위가 다르기 때문에 다른 방식의 병렬화를 진행한다 (Figure 16 참고).

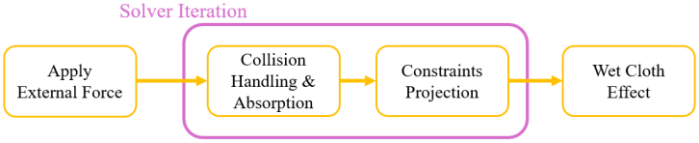


Figure 16. Processes in wet cloth simulation that require parallelization.

가장 먼저 외력의 적용과 적분은 정점(Vertex)의 위치와 속도를 갱신하기 때문에 정점마다 스레드를 할당하여 병렬화 한다 (Figure 17 참조).

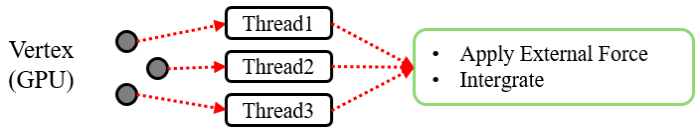


Figure 17. Parallelization of external force application and integration at the vertex level.

PBD 옷감 시뮬레이션의 가장 핵심은 제약조건(Constraint)의 투영을 통해서 위치를 보정하는 것이다. 이때, 제약조건 연산의 단위는 각 스프링마다 스레드를 할당하여 스프링에 연결된 정점의 위치를 갱신하는 것이다. 하지만 하나의 정점에 여러 스프링이 연결되어 있기 때문에 이 경우에는 경합조건이 발생한다. 따라서 Fratarcangelin 등이 제안하는 그래프 컬러링을 활용하여 인접한 정점에 다른 색을 칠하고 같은 색 정점의 스프링마다 스레드를 할당하여 병렬화 한다[11] (Figure 18 참조).

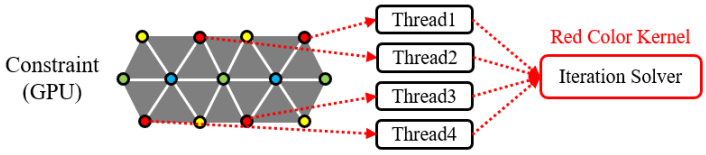


Figure 18. Parallelizing constraints within kernels of the same color using graph coloring.

액체 입자와 옷감사이의 충돌 감지의 경우는 3.4 장의 공간 해싱 알고리즘을 활용한다. 따라서 각 칸마다 스레드를 할당하여 병렬화 한다 (Figure 19 참조).

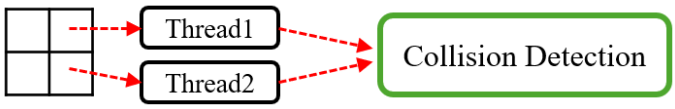


Figure 19. Parallelizing at the spatial hashing grid level to detect collisions only within neighboring grids.

젖은 옷감에서의 흡수, 확산, 물방울 떨어짐 현상은 3.3 장에서의 과정을 따른다. 이때, 면에 포화도를 누적하는 방식을 사용하기 때문에 면마다 스레드를 할당한다 (Figure 20 참조).

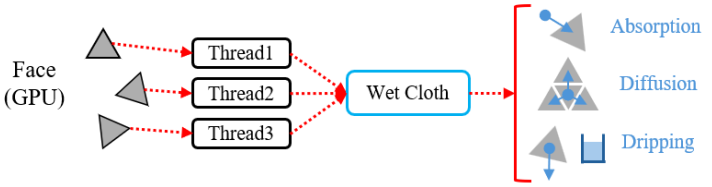


Figure 20. Representing wet cloth by parallelizing at the face level for face saturation calculations.

젖은 옷감의 주름 표현은 4.3 장에서의 과정을 따른다. 이 방식에서는 정점의 곡률을 고려하기 위해서 에지(Edge)마다 스레드를 할당하여 이면각을 계산한다. 그리고 정점의 위치를 갱신하기 위하여 정점마다 스레드를 할당한다 (Figure 21 참조).

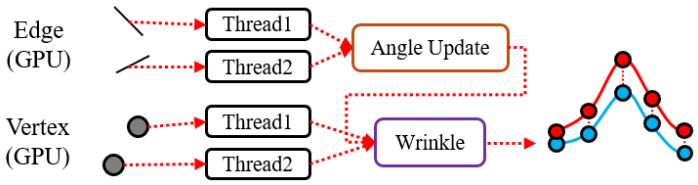


Figure 21. Parallelizing at the edge level for dihedral angle calculations and at the vertex level for position updates to represent wrinkles.

5. 구현

본 논문은 다음과 같은 환경에서 구현되었다: AMD Ryzen Threadripper PRO 7975WX 32-Core CPU, NVIDIA RTX 6000 Ada Generation GPU. FLIP 시뮬레이션의 압력 계산을 위한 수치적 행렬해법은 Li 등이 제안한 GPU 기반 선조건 적용 켤레 기울기법(Preconditioned conjugate gradient)을 사용하였다 [12]. 액체 시뮬레이션의 경우 시뮬레이션 결과가 입자로 나타나기 때문에 입자를 메시(Mesh) 형태로 만드는 표면 복원(Surface reconstruction) 과정을 위해 Lorensen 등이 제안한 마칭 큐브(Marching cube) 기법을 활용한다[13]. 이때, 입자의 밀도를 기반으로 레벨셋을 계산한다.

6. 결과

본 논문에서는 액체-옷감 상호작용 장면의 세부적인 표현 향상을 확인하기 위해 다양한 장면을 생성하여 비교한다.

먼저, 독립적인 장면을 생성하여 각 시뮬레이션의 세부적인 특징이 향상되었는지 실험한다. 물의 움직임이 활발한 장면과 정적인 장면을 통해 액체 시뮬레이션의 세부적인 특징을 확인했으며, 기존의 FLIP 기반 액체 시뮬레이션과 비교했을 때 물 표면에 나타나는 파동 난류 특징이 확연하게 표현됐다 (Figure 22d 와 22e 참조).

옷감 시뮬레이션은 젖은 옷감에서의 중력에 의한 확산과 함께 최대 포화도를 넘어서 물방울 떨어짐 현상을 통해 액체와 옷감의 상호작용 장면을 생성할 수 있다 (Figure 22a~22c 참조). 또한, 액체와 옷감의 충돌을 통해 흡수되는 장면도 생성했으며, 확산이 되었을 때 곡률에 의해 주름이 과장되어 표현되는 장면도 생성했다 (Figure 22f 참조).

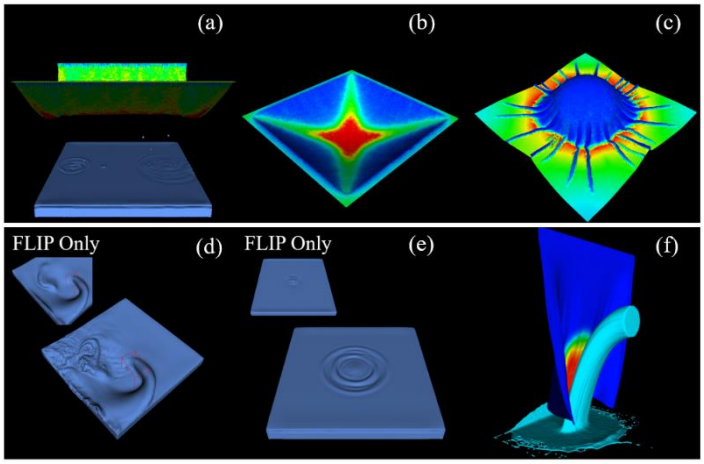


Figure 22. Fluid-cloth interaction results ((a) Dripping, (b) Diffusion, (c) Wrinkle, (d) Box rotation, (e) Single water drop, (f) Water pour to cloth)

Scene		*Water Drop*	*Dam Break*	*Double Water Drop*
GPU	*# FLIP Particle*	20k	24k	135k
	# Turbulence Particle	11k	16k	15k
	frame/sec	2.72	1.76	0.83
CPU	*# FLIP Particle*	20k	24k	47k
	# Turbulence Particle	8k	15k	14k
	frame/sec	0.09	0.035	0.012

Table 1: Liquid simulation performance.

Model	*Dragon*		*Bunny*	
# Vertex/Face	50k	100k	4.8k	2.4k
# Structural/Bend	150k	150k	7.2k	7.2k
CPU/GPU (frame/sec)	0.98	68.3	36.5	92.4

Table 2: Cloth simulation performance.

Scene		*Dripping*	
Fluid	*# FLIP Particle*	135k	
	# Turbulence Particle	197k	
Cloth	*# Vertex/Face*	18k	36k
	# Structural/Bend	54k	54k
Result	*frame/sec*	1.15	

Table 3: Liquid-cloth interaction performance.

추가로, CPU 와 GPU 에서 frame/sec 로 성능 측정을 진행하여 비교한 결과, CPU 대비 액체 시뮬레이션의 경우 약 30~70 배, 옷감 시뮬레이션의 경우 약 2.5~70 배 향상되었다 (Table 1, 2 참조). 두 시뮬레이션 모두 장면의 복잡도가 증가할수록 높은 성능 향상을 보였으며, 액체-옷감 상호작용 장면의 성능 측정 결과는 약 1.15 frame/sec 가 측정되었다 (Table 3 참조). 이 결과, GPU 를 활용한 최적화를 통해 각각의 시뮬레이션 시나리오마다 수십 배에 달하는 성능 향상을 이뤄냈다.

7. 결론 및 향후 연구

본 논문에서는 CPU 기반의 액체-옷감 상호작용 장면에서 성능적인 한계로 인해 표현하지 못했던 세부적인 시각적 특징을 표현하고 이로 인해 증가된 연산을 GPU 기반 병렬화 기법을 통해 효율적으로 처리할 수 있는 프레임워크를 제안했다. 이 프레임워크에서는 빈번하게 발생하는 난류 입자의 삽입/삭제 과정을 메모리 최적화를 통해 효율적으로 설계하였으며, PBD 의 제약 조건 투영 과정에서 발생하는 경합 조건을 그래프 컬러링 알고리즘을 활용하여 해결하는 등의 방식을 사용했다. 결과적으로 이러한 최적화 방식을 통해 CPU 대비 각 시뮬레이션의 성능이 수십 배 향상되었으며, 렌더링 결과에서도 파동 난류, 주름, 확산 등의 세부적인 시각적 특징이 잘 표현되었다. 향후 연구에서는 본 논문의 프레임워크 성능을 개선한 다중 GPU 를 활용한 프레임워크에 대한 연구를 진행할 계획이다. 이 연구를 통하여 액체-옷감 상호작용 장면을 실시간으로 시뮬레이션 할 수 있도록 할 것이다.

감사의 글

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.RS-2022-00155915, 인공지능융합혁신인재양성(인하대학교)). 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아서 수행된 연구임(No. RS-2023-00254695).

References

[1] Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. “A multi-scale model for simulating liquid-fabric interactions,” ACM Trans. Graph. 37, 4, Article 51, 16 pages, August, 2018.

[2] Y. Zhu and R. Bridson, “Animating sand as a fluid,” ACM Trans. Graph, vol. 24, no. 3, pp. 965–972, July, 2005.

[3] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. “Position based dynamics,” J. Vis. Comun. Image Represen, vol. 18, no. 2, pp. 109-118, April, 2007.

[4] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai, “Surface turbulence for particle-based liquid simulations,” ACM Trans. on Graphics, vol. 34, no. 6, pp. 1-10,

November, 2015.

[5] Kui Wu, Nghia Truong, Cem Yuksel, Rama Hoetzlein, “Fast Fluid Simulations with Sparse Volumes on the GPU,” EUROGRAPHICS, vol. 37, no. 2, pp. 157-167, 2018.

[6] S. Patkar, P.huri, “Wetting of porous solids,” IEEE Trans. on Visualization and Computer Graphics, vol. 19, no. 9, pp. 1592-1604, September, 2013.

[7] Yujun Chen, Nadia Magnenat Thalmann, and Brian Foster Allen. “Physical simulation of wet clothing for virtual humans,” Vis. Comput. 28, 6–8 (June 012), 765–774. 2012.

[8] N. Abu Rumman, P. Nair, P. Müller, L. Barthe, D. Vanderhaeghe. “ISPH–PBD: coupled simulation of incompressible fluids and deformable bodies,” The Visual Computer, vol. 36, pp. 893-910, May, 2020.

[9] C. Lewin, “Cloth Self Collision with Predictive Contacts,” Game Developers Conference, 2018.

[10] S. Green, “Particle Simulation using CUDA,” NVIDIA Corporation, pp. 1-12, May, 2010.

[11] M. Fratarcangelin, F. Pellacini, “A GPU-based implementation of position based dynamics for interactive deformable bodies,” Journal of Graphics Tools, vol. 17, no. 3, pp. 59-66, January, 2013.

[12] R. Li and Y. Saad, “Gpu-accelerated preconditioned iterative linear solvers,” The Journal of Supercomputing, vol. 63, no. 2, pp. 443–466, 2013.

[13] William E. Lorensen and Harvey E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” SIGGRAPH Comput. Graph, vol. 21, no. 4, pp. 163–169, July, 1987.

[14] Yu, Hong-Xing, Yang Zheng, Yuan Gao, Yitong Deng, Bo Zhu, and Jiajun Wu. "Inferring hybrid neural fluid fields from videos." Advances in Neural Information Processing Systems, vol. 36, 2024.

[15] Toshev, Artur P., Gianluca Galletti, Johannes Brandstetter, Stefan Adami, and Nikolaus A. Adams. "Learning lagrangian fluid mechanics with e (3)-equivariant graph neural networks." In International Conference on Geometric Science of Information, pp. 332-341. Cham: Springer Nature Switzerland, 2023.

〈 저 자 소 개 〉

박 은 수
- 2025년 인하대학교 정보통신공학과 4학년
- 관심분야 : 물리 기반 시뮬레이션, 게임 그래픽스, 게임 클라이언트
- https://orcid.org/0009-0005-6437-0371

이 주 용
- 2025년 인하대학교 정보통신공학과 4학년
- 관심분야 : 물리 기반 시뮬레이션, 게임 그래픽스, 게임 클라이언트
- https://orcid.org/0009-0003-9878-3292

박 인 규
- 1995년 서울대학교 제어계측공학과 학사
- 1997년 서울대학교 제어계측공학과 석사
- 2001년 서울대학교 전기컴퓨터공학부 박사
- 2001년~2004년 삼성종합기술원 전문연구원
- 2007년~2008년 Mitsubishi Electric Research Laboratories(MERL) 방문연구원
- 2014년~2015년 MIT Media Lab 방문부교수
- 2018년~2019년 UCSD, Center for Visual Computing 방문학자
- 2004년~현재 인하대학교 정보통신공학과 교수
- 2020년~현재 인하대학교 인공지능융합연구센터/인공지능융합대학원 사업단장
- 관심분야 : 컴퓨터비전, 그래픽스, 딥러닝
- https://orcid.org/0000-0003-4774-7841

김 종 현
- 2008년 세종대학교 컴퓨터학과 학사
- 2010년 고려대학교 컴퓨터학과 석사
- 2016년 고려대학교 컴퓨터학과 박사
- 2013년~2016년 (주)텐일레븐 선임연구원
- 2023년~현재 인하대학교 소프트웨어융합대학 디자인테크놀로지학과 부교수
- 관심분야 : 물리 기반 시뮬레이션, 가상/증강현실, 지오메트리 프로세싱, 게임 물리, 게임 AI
- https://orcid.org/0000-0003-1603-2675