DOI: 10.3745/KIPSTA.2009.16A.6.419

Semi-Lagrangian 이류항 계산의 추적법 개선

박 수 완[†] · 백 낙 훈[††] · 유 관 우[†††]

요 약

일반적으로 사실성 있는 유체를 시뮬레이션하기 위해 Navier-Stokes 방정식을 사용한다. Euler 구조에서 Navier-Stokes 방정식을 풀 때, 이류항은 비선형이어서 계산이 복잡하기 때문에 근사화한 모델로 Semi-Lagrangian 방법을 사용한다. Semi-Lagrangian 방법에서는 먼저 이류하는 위치를추적하고, 추적한 위치에서 값을 보간해서 사용한다. Stam이 제안한 방법으로 계산할 경우, 이 과정에서 수치적 소실이 많이 발생하기 때문에 수치적 소실을 보정하려는 노력들이 있어 왔다. 그러나 대부분의 경우에 보간하는 과정에서의 소실을 줄이려는 노력이거나, 입자를 같이 사용하는 방법이었다. 따라서 본 논문에서는 Euler 구조에서 다른 추가나 변형을 가하지 않고 이류항의 연산에서 추적법을 개선함으로 수치적 소실을 줄이는 방법을 제안한다. 우리의 방법에서는 현재 격자의 속도로 역추적하는 기존의 방법이 아니라, 현재의 격자로 오게 될 속도를 가진 격자를 찾아서, 그 격자의 물리량들을 선형 보간하여 사용한다. 이는 직관적으로 생각할 때, 어느 지점의 물리량은 그 지점의 속도로 인해 다음 단계에 다른 지점에 있게 된다는 사실을 그대로 적용한 것이다. 본 논문에서 제안한 방법으로 기체를 시뮬레이션 했을 때 수치적 소실이 줄었으며, 그로 인해 사실성을 높이면서도 실시간 처리가 가능했다.

키워드 : 유체 시뮬레이션, 이류항, 수치적 소실

Improved Trajectory Calculation on the Semi-Lagrangian Advection Computation

Su Wan Park[†] · Nakhoon Baek[††] · Kwan Woo Ryu[†††]

ABSTRACT

To realistically simulate fluid, the Navier-Stokes equations are generally used. Solving these Navier-Stokes equations on the Eulerian framework, the non-linear advection terms invoke heavy computation and thus Semi-Lagrangian methods are used as an approximated way of solving them. In the Semi-Lagrangian methods, the locations of advection sources are traced and the physical values at the traced locations are interpolated. In the case of Stam's method, there are relatively many chances of numerical losses, and thus there have been efforts to correct these numerical errors. In most cases, they have focused on the numerical interpolation processes, even simultaneously using particle-based methods. In this paper, we propose a new approach to reduce the numerical losses, through improving the tracing method during the advection calculations, without any modifications on the Eulerian framework itself. In our method, we trace the grids with the velocities which will let themselves to be moved to the current target position, differently from the previous approaches, where velocities of the current target positions are used. From the intuitive point of view, we adopted the simple physical observation: the physical quantities at a specific position will be moved to the new location due to the current velocity. Our method shows reasonable reduction on the numerical losses during the smoke simulations, finally to achieve real-time processing even with enhanced realities.

Keywords : Fluid Simulation, Advection Term, Numerical Loss

1. 서 론

계산유체역학(CFD, Computational Fluid Dynamics)과 수치해석 등의 분야에서는 이미 오래 전부터 다양한 상황에서 유체의 흐름을 사실적으로 시뮬레이션 하기 위한 연구가 있어 왔다. 복잡한 이론과 수식을 동반하는 유체 역학을, 사실성뿐만 아니라 계산에 소요되는 시간까지 고려해야 하는 그래픽스에서는 직접 적용하기 힘들었다. 그러나 최근에는 컴퓨터 연산 능력 향상, 메모리의 향상, 그래픽 카드의 향상, 각종 수식의 간소화하는 모델이 등장하여 실시간 처리에 관한 연구가 진행 중이다[1-3]. 이는 약간의 물리적 사실성을

※ 이 논문은 2009년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(과제번호: 2009-0088544).
† 준 회 원 : 경북대학교 컴퓨터공학과 박사과정
†† 종신회원 : 경북대학교 전자전기컴퓨터학부 부교수(교신저자)
††† 정 회 원 : 경북대학교 컴퓨터공학과 교수
논문접수 : 2009년 5월 12일
수 정 일 : 1차 2009년 8월 4일
심사완료 : 2009년 8월 17일

희생하더라도 시각적인 사실성을 유지하는 범위 내에서 효율성을 높이도록 하는 것이다.

유체를 사실적으로 시뮬레이션 할 때는 일반적으로 Navier-Stokes 방정식을 활용한다. 컴퓨터 그래픽스 분야에서, Foster와 Metaxas는 3차원 유체의 움직임을 시뮬레이션 하기 위해 Navier-Stokes 방정식을 유한차분법(finite difference method)을 이용하여 계산하였다[2]. 이 방법은 명시적 솔버(explicit solver)를 사용하였기 때문에 상대적으로 긴 시간 간격에 대해서는 시스템이 불안정해지는 현상이 발생하였고, 대화형이고 실시간에 처리하는 유체 시뮬레이터로 사용하기는 힘들었다. 후에 Stam은 긴 시간 간격에서도 안정성이 있으며 상대적으로 빠른 시뮬레이션 방법을 제안하였다[3]. 여기서는 Navier-Stokes 방정식을 풀 때 유체의 비압축성 성질을 가지도록 압력을 계산하는 방법(Pressure projection scheme)과 이류항(advection term)의 계산에서 Semi-Lagrangian 방법을 사용하였다.

일반적으로 Semi-Lagrangian 방법은 두 단계로 연산된다. 첫 단계는 현재의 격자로 이동해 올 위치를 찾는 것이고, 다음 단계는 Euler 구조(framework)에서 격자(grid)들은 이산적인 모델이기 때문에 찾은 위치가 격자들 사이에 놓였을 때 값을 보간(interpolation)하는 것이다. 이 과정에서 정확도를 높이기 위해서는 정확한 위치를 찾는 것과 찾은 위치의 값을 정확하게 보간하는 것이 필요하다[4]. Stam은 Semi-Lagrangian 방법에서 기존에는 반복법(iteration method)으로 위치를 찾던 것을 현재 격자의 속도로 역추적했고, 찾은 위치의 값을 보간할 때 선형 보간을 사용했다. 이 두 단계는 많은 근사화를 한 것으로 수치적 소실이 발생할 수 있다.

Fedkiw는 이런 소실을 줄이기 위해 Navier-Stokes 방정식을 풀 때 유체의 회전하는 움직임을 발생시키는 소규모의 소용돌이(vorticity)를 추가하였고, 3차 함수를 이용한 보간법을 소개하였다[5]. Song은 MCIP(Monotonic CIP)를 사용하였는데[6], 이는 각 격자에 물리량을 저장하는 것에 더하여, 그의 공간적 미분 값도 저장하여 활용하는 방법으로, 시뮬레이션의 이류항을 계산할 때 3차의 정확성을 가지는 CIP 방법을 적용하였다. 그러나 이 방법은 차원 별로 나누어 계산함으로 인해 구현이 어렵고 연산 시간이 증가하였으며 수치적 에러가 있다. 후에 Kim은 연산 시간이 많은 것과 수치적 에러가 있는 단점을 보완하기 위해 USCIP(Unsplit Semi-Lagrangian CIP) 방법을 제안하였다[7]. 이는 Yabe와 Aoki의 원래의 2차와 3차 CIP 다항식들[8, 9]에 변형을 가한 것으로 차원을 나누어서 연산하지 않는다. 이는 기존의 CIP 솔버들에 비해 보다 안정적이며, 연산량이 상대적으로 적으면서도 정확성을 높일 수 있다.

Kim은 이류항을 계산할 때 2차 정확도를 주기 위해 BFECC(Back and forth error compensation and correction)이라는 방법을 소개하였다[10, 11]. 이는 이류항의 계산에서 Stam이 제안한 Semi-Lagrangian의 방법에서 정방향으로 물리량을 구한 후에 그 값을 가지고 다시 역방향으로 물리량을 구해서, 원래의 물리량과 비교하여 오차를 보정하는 방법이다. 또한 Fedkiw은 BFECC에서 수정된 물리량을 사용하는 방법을 수정한 MacCormak의 방법을 적용하였는데, 이는 연산량을 줄이는 동시에 정확도를 다소 높였다[12].

위의 방법들은 이류항의 계산에 있어서 Semi-Lagrangian 방법을 사용할 경우 이류항의 값을 가져 오기 위해 보간을 할 때 정확도를 높이거나, 각 격자에 있는 물리량의 오차를 수정하는 방법이다. 즉 이 방법들은 읽어오는 값 자체에 정확도를 높이는 과정이었으며, 추적할 위치를 찾는 것에서는 기존의 방법들을 그대로 사용하였다. 또한 위의 방법들은 극단적인 값들이 나오는 경우가 있어서 클램핑(clamping)을 하지 않으면 안정성을 잃게 된다. 또한 물리량을 구할 때 각 격자에 있는 수치적 소실은 줄였지만 다음과 같은문제가 발생할 수있다. 역추적 방법을 통해서 이류항을 구할 때 현재 속도로 역추적하기 때문에 현재 속도가 0인 격자에서는 이류항의 속도가 0이 되거나 해당 물리량이 격자 자신의 값이 된다. 현재 속도로 역추적해서 찾은 위치의 속도나 물리량이 0인 경우엔 갑작스러운 값의 변화로 소실이 더 커진다. 특히 유체의 흐름이 빠르거나 급격한 곳에서는 자신의 속도로 역추적한 곳의 속도가 사실은 현재의 격자 위치를 향하지 않고 완전히 다른 곳을 향한 것일 수도 있다.

이에 대해 Lagrangian과 Euler 구조를 같이 사용하는 하이브리드(hybrid) 방법들이 제시되었다. 즉 이류항을 계산할 때 입자가 물리량을 가지고 이동하면 각 격자는 자기에게 위치한 입자의 값을 사용하는 것이다. 이런 예로는 입자 레벨 셋 방법(particle level set method)[13], 소용돌이 입자 방법(vortex particle method)[14], 미분값 입자들(derivative particles)[15]이 있다. 그러나 이 방법들은 추적하는 과정에 있어 정확도는 높였지만 입자 자체의 연산, 격자들끼리의 연산, 입자와 격자 간에 값을 주고 받는 연산이 필요하여 복잡하며, 또한 표면이나 소용돌이 등의 특정 값들에만 적용을 하거나 유체의 흐름이 복잡해지는 곳에서만 제한적으로 사용하는 형태였다.

본 논문에서는 Semi-Lagrangian 방법으로 이류항 연산을 할 때 추적 과정에서의 수치적 소실을 줄이는 새로운 방법을 제안한다. 각 격자가 입자인 것처럼 생각하고 다음 단계에 물리량이 있게 될 위치를 구한 후에, 구한 위치에서 이류항의 값을 계산할 때는 자기에게 오게 될 격자의 물리량을 사용한다. 이는 직관적으로 생각할 때, 어느 지점의 물리량은 그 지점의 속도로 인해 다음 단계에 다른 지점에 있게 된다는 사실을 그대로 적용한 것이다. 값을 보간할 때는 일차 정확도의 선형 보간법을 사용한다. 본 논문의 방법은 수치적인 정확도를 높이기보다는 Euler 구조에서 Navier-Stokes 방정식을 풀 때 필연적으로 발생하는 소실을 효과적으로 줄이고자 한다. 이는 Stam의 방법에서 이류할 지점을 찾는 방법에 변형을 가한 것으로, 긴 시간 간격(time step)에서도 안정적이며, 구현이 쉽고, 처리 속도도 빠르다.

본 논문의 구성은 다음과 같다. 2장에서는 우리가 사용할 기본적인 유체 시뮬레이션 모델을 살펴보고, 3장에서는 추적에서 소실을 줄이는 이류항의 계산 방법을 보인다. 4장에서는 구현 결과를 기술하고, 5장에서는 결론을 기술한다.

2. 기본적인 유체 시뮬레이터 모델

본 논문에서는 무압축성 무점성 기체를 가정하며, Euler 구조에서 Navier-Stokes 방정식을 사용하여 기체의 역학적 움직임을 시뮬레이션하고자 한다. 격자의 중심에 속도와 밀도 값을 저장하며, 각 격자는 자기에게 이류할 물리량의 위치 정보를 가지고 있다. 격자 모델과 시뮬레이션 과정은 이류항 계산의 부분을 제외하고는 Stam의 방식[16]을 따른다.

속도 벡터장을 $\mathbf{u}$로 나타낼 때, 비압축성 흐름에 대한 Navier-Stokes 방정식은 다음과 같다.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

여기서 ρ는 기체의 밀도이고, p는 속도 장에서의 압력이며, ν는 점성 계수이고, $\mathbf{f}$는 중력이나 부력과 같은 외부 힘이다. 식 (2)는 비압축성일 때 유체의 질량이 보존되는 것을 의미한다.

무점성인 경우에, Navier-Stokes 방정식은 확산항(diffusion term)이 0으로 되어, 다음과 같이, Euler 방정식이 된다.

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f} \tag{3}$$

본 논문에서는 비압축성 기체를 구현하기 때문에 점성과 관련된 확산항은 고려하지 않고, 위의 Euler 방정식을 사용한다.

3. 소실을 줄이는 이류항의 계산

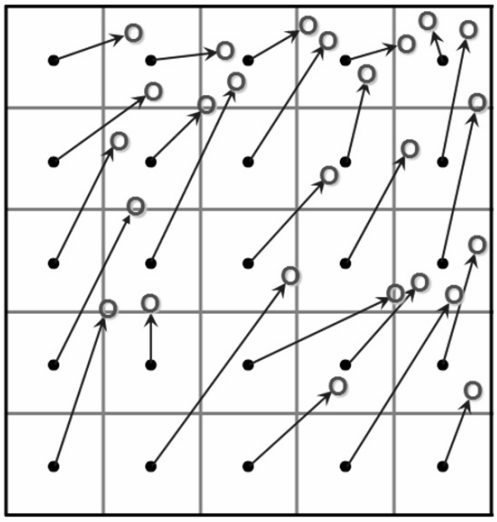
Semi-Lagrangian 방법으로 이류항을 구하기 위해서는 두 단계의 계산이 필요하다. 첫 번째는 현재 각 격자로 이동해 올 위치를 찾는 것이고, 두 번째는 Euler 격자들은 이산적인 모델이기 때문에 찾은 위치가 격자들 사이에 놓였을 때의 값을 보간하는 것이다. Stam의 경우에 첫 번째 과정인 현재의 격자로 이동해 올 위치를 찾을 때 현재의 격자 속도를 이용하는 역추적방법을 사용했으나 본 논문에서는 다음과 같은 방법을 사용한다. 먼저 각 격자가 입자인 것처럼 생각하고 다음 단계에 어느 격자가 그 위치에서 자신의 속도로 연산하고자 하는 격자의 위치로 이동하게 될 지를 찾는다. 구해진 위치의 격자에다가 다음 단계의 속도와 밀도 등 물리량의 이류항 성분을 구할 때 사용하도록 추적할 위치 정보를 준다. 이를 통해 다음 단계의 각 격자는 이류항 연산에서 어느 위치에서 물리량을 가져와야 하는 지에 대한 정보를 가지게 된다. 후에 두 번째 과정으로 선형 보간법을 통해 값을 가져온다.

먼저 추적할 위치를 찾는 방법은 다음의 과정들로 이루어진다. 각 격자는 자기가 가진 속도로 $\triangle t$ 후에 있게 될 위치를 계산한다. 이는 자신의 속도로 (그림 1)과 같이 위치로 이동하게 될 것이다.

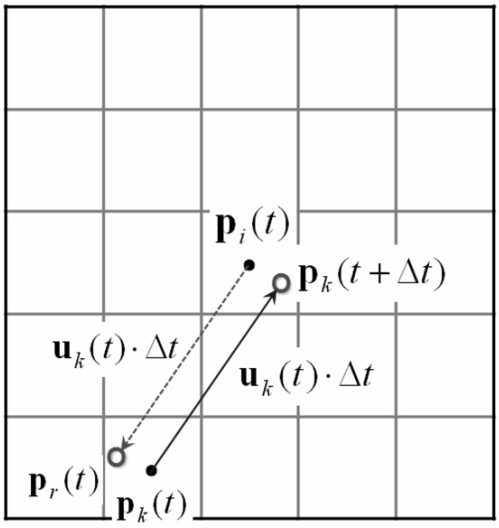
$$\mathbf{p}_k(t+\Delta t) = \mathbf{p}_k(t) + \mathbf{u}_k(t)\cdot\Delta t \tag{4}$$

$$\mathbf{p}_r(t) = \mathbf{p}_i(t) - \mathbf{u}_k(t)\cdot\Delta t \tag{5}$$

한 격자를 예로 보면, (그림 2)에서 격자를 입자처럼 생각할 때 k 번째 격자의 중심 위치 $\mathbf{p}_k(t)$는 식 (4)를 통해 다음 단계에 $\mathbf{p}_k(t+\triangle t)$ 위치에 있게 되고[(그림 2)에서 실선], $\mathbf{p}_k(t+\triangle t)$는 i 번째 격자의 내부에 있게 된다. 이로써 이류항을 연산할 때 i 번째 격자는 어느 위치의 데이터를 사용해야 하는 지를 결정할 수 있게 된다. i 번째 격자에서 추적할 위치($\mathbf{p}_r(t)$)는 i 번째 격자의 중심점 $\mathbf{p}_i(t)$에서



(그림 1) 각 격자의 물리량이 $\triangle t$ 시간 후에 이류에 의해 이동할 위치



(그림 2) 한 격자가 이동하게 될 위치(실선)와 추적할 위치(점선) 찾음

식(5)를 통해 k 번째 격자의 속도로 추적해서 찾는다[(그림 2)에서 점선]. 그리고 한 격자에 여러 위치에서 물리량이 올 경우에는 그 물리량들을 평균하여 취한다.

각 격자는 이 과정을 통해 다음 단계에 어느 위치에서 이류할 성분들이 오게 되는지를 알게 된다. 그러나 이렇게 했을 때 다음과 같은 문제가 발생할 수 있다. 특정 격자에서 속도와 물리량들을 가지고 있었고 다음 단계에 다른 자리로 이동하게 되는데, 이럴 경우에 다음 단계에서 이 위치로 오게 될 물리량들이 없을 경우에 시뮬레이션에서 불연속적인 모습을 가지게 된다. 이를 막기 위해서 자신에게 이류할 성분이 없지만 자신이 속도와 물리량을 가진 격자일 경우엔, 자신에게 가장 근사한 위치로 오는 속도를 가진 격자를 가지고 근사화한다.

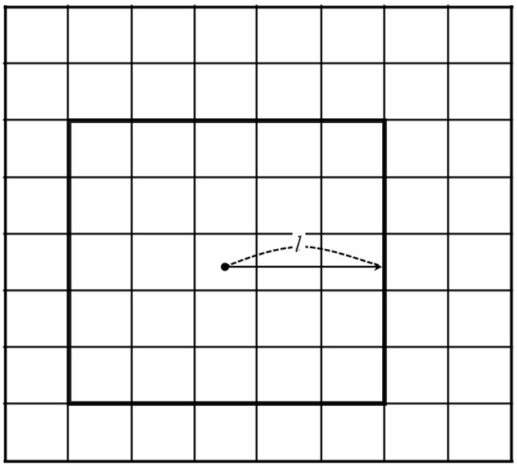
근사에 사용할 격자들을 탐색하기 위한 범위는 다음과 같이, 각 격자의 속도로부터 단위시간당 이동거리를 계산하여 사용한다.

$$l_{average} = \sum |\mathbf{u}_i(t)|\,\Delta t / T \tag{6}$$

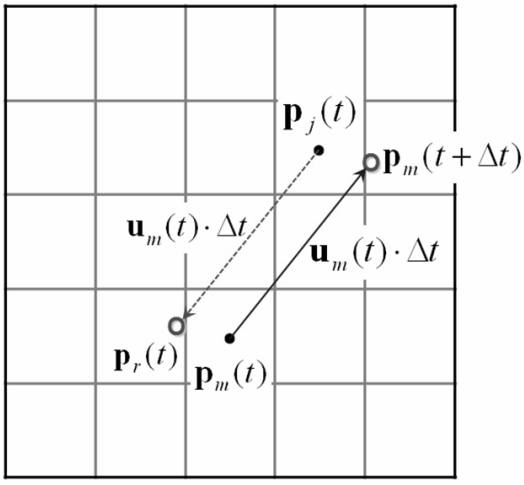
여기서 $\mathbf{u}_i$는 i 번째 격자의 속도이고 T는 속도를 가진 격자들의 개수이다. 계산된 탐색 범위가 너무 커서 시스템의 속도가 현격하게 느려지거나, 너무 작아서 탐색이 무의미한 경우를 피하기 위해, 다음과 같이 사용자가 정의하는 상하한선을 둔다.

$$l = \begin{cases} \alpha, & \text{if } l_{average} > \alpha \\ \beta, & \text{if } l_{average} < \beta \\ l_{average}, & \text{otherwise} \end{cases} \tag{7}$$

한 격자를 예로 보면 j 번째 격자에 오는 물리량들이 없다고 가정했을 때, (그림 3)과 같이 탐색 범위 내에서 식 (4)를 이용해서 j 번째 격자 위치에 가장 가깝게 오는 m 번째 격자를 찾고[(그림 4)에서 실선], 식 (5)를 이용하



(그림 3) 탐색 범위



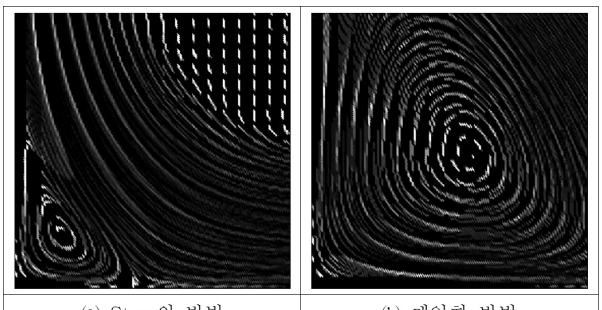
(그림 4) 근사 위치를 찾음

여 j 번째 격자의중심 $\mathbf{p}_j$ (t)에서 m 번째 격자의 속도 $(\mathbf{u}_m(t))$로 추적한 위치($\mathbf{p}_r$ (t))를 찾는다[(그림 4)에서 점선].

이 과정들을 마치면각 격자는 이류할 물리량들이 어느 위치에서 오는 지를 저장하게 된다. 이 정보를 통해 각 격자는 물리량들의 이류항을 선형 보간법을 통해 계산한다.

4. 구현 결과

본 논문에서 제시하는 방법은 CoreTM2 Quad CPU Q9550 2.83GHz 3G 메모리, GeForce GTX 260(896M) 그래픽카드를 가지는 시스템 상에서 C++, OpenGL을 사용하여 구현되었다. (그림 5)는 격자 수 128×128에서 한 스텝의 간격 $\triangle t$를 0.01로 두고, lid driven cavity를 시뮬레이션하였을 때 Stam의 방법[6]과 제안한 방법의 스트림라인 플롯(streamline plot) 결과이다. 각각의 방법은 오랜 시간이 흐른 후에는 (그림 5)와 같은 패턴의 스트림라인을 반복한다. 두 방법 모두가 근사화된 방법이어서 정확도는 다소 떨어지지만, 동일한 코드에서 이류항 부분만을 다르게 했을 때 제안한 방법[(그림 5)-(a)]의 경우엔 lid driven cavity에서 형성하는 스트림라인에 가까운 모습을 보이지



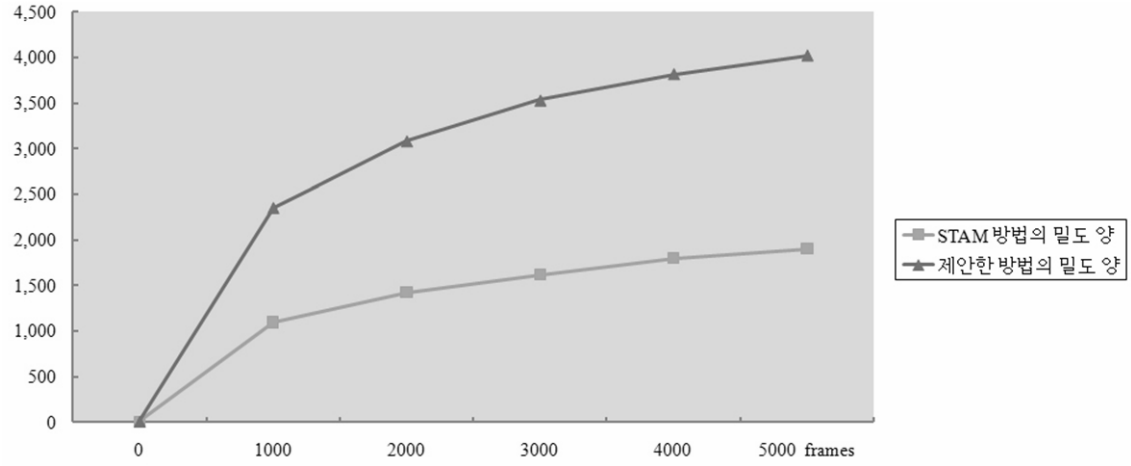
(a) Stam의 방법 (b) 제안한 방법

(그림 5) 128×128 격자, $\triangle t$=0.01에서 lid driven cavity 시뮬레이션한 스트림라인 플롯 결과

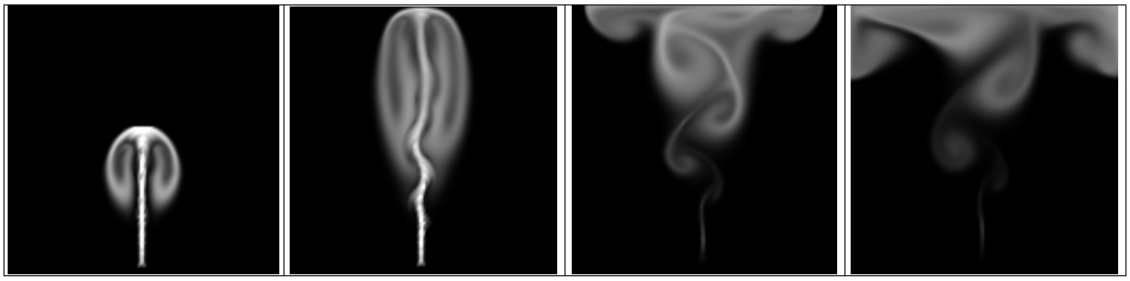
만 Stam의 방법[(그림 5)-(b)]에서는 그렇지 못한 그림을 나타낸다.

밀폐된 곳에서 동일한 조건으로 일정량의 기체를 쏘아 올렸을 때의 동작을 대상으로 Stam의 Semi-Lagrangian 방법과 제안하는 방법을 비교하였다. 격자 수 128×128에서 한 스텝의 간격 $\triangle t$를 0.1로 두고 연기를 연속으로 분사한 후 모든 격자들의 밀도 양의 변화를 그래프로 나타내면 (그림 6)과 같다. 제안한 방법으로 시뮬레이션한 경우에 더 빠른 속도로 밀도가 증가하며 양도 더 많은데, 이는 속도의 양이 잘 보존되고 소실되는 양이 적어 밀도가 이웃 격자들로 더 잘 전해지며, 시뮬레이션의 반복 과정에서 누적되며 밀도 자체의 소실 양도 적기 때문이다. 반복 횟수가 많아진 후에는 밀도의 추가가 없었는데, 그 때 증가 속도가 완만해지는 정도도 제안한 방법이 적음을 볼 수 있다.

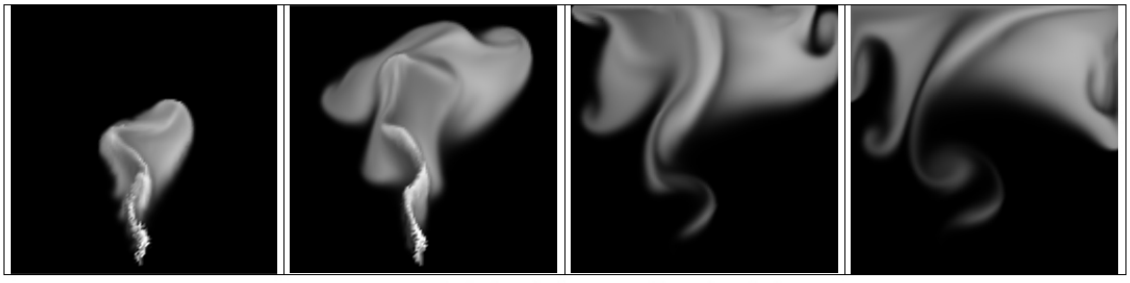
(그림 7)과 (그림 8)은 격자 수 128×128에서 한 스텝의 간격 $\triangle t$를 0.1로 두고 연기를 연속으로 분사한 후 시뮬레이션한 결과이다. (그림 7)은 소용돌이 효과를 주지 않은 경우이고, (그림 8)은 소용돌이 효과를 준 경우이다. (그림 7)과 (그림 8)에서 보는 것처럼 제안한 방법을 사용한 경우에 속도와 밀도가 잘 이류하여, 밀도가 빠른 시간에 잘 퍼져 나가므로 역동적인 연기의 모양을 관찰할 수 있다. 소용돌이를 주지 않은 (그림 7)에서는 Stam의 방법[(그림 7)-(a)]과 제안한 방법[(그림 7)-(b)]의 꺾이는 모양이 다른데, 이는 속도의 소실이 적고 물리량들의 전달 속도도 빨라서 다른 격자들로 물리량들을 잘 전달하기 때문이다. 소용



(그림 6) 전체 격자들의 밀도 양의 변화



(a) STAM 방법으로 이류항 계산

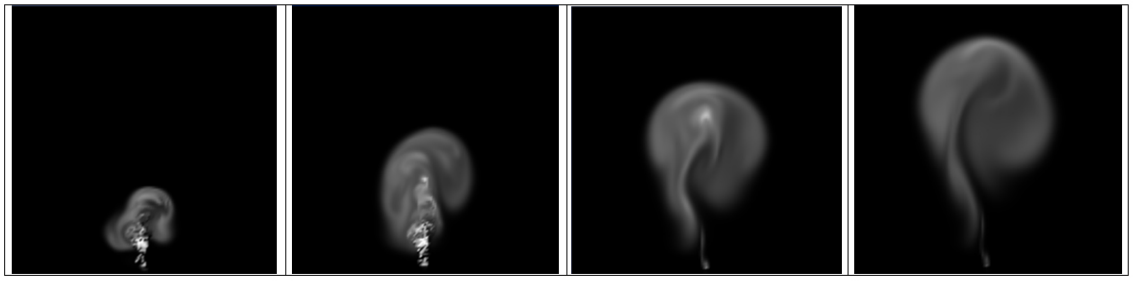


(b) 제안한 방법으로 이류항 계산

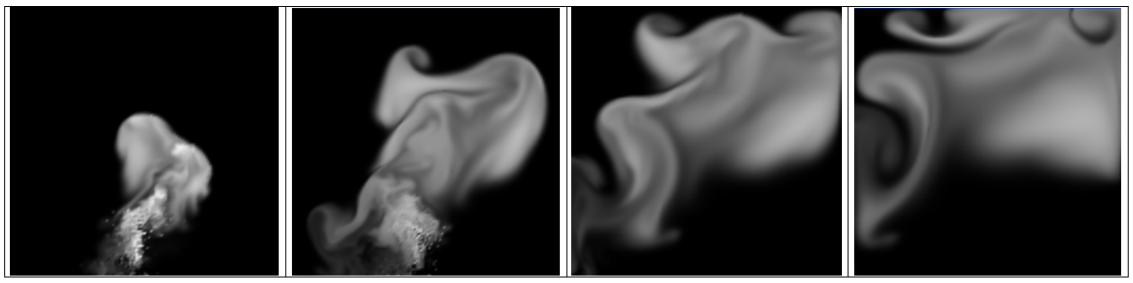
(그림 7) $\triangle t$=0.1이고 소용돌이를 주지 않을 때 시뮬레이션 결과

돌이를 준 (그림 8)에서 Stam의 방법[(그림 8)-(a)]은 소용돌이가 희미하게 사라지는 모습을 보이지만, 제안한 방법[(그림 8)-(b)]은 작은 소용돌이가 보다 큰 모양의 소용돌이로 발전하는 모습을 볼 수가 있다. 격자들에 있는 속도 벡터와 밀도가 잘 전달되어 회전 또한 소실되지 않고 잘 퍼져나가기 때문이다. (그림 9)는 격자 수가 256×256인 경우의 시뮬레이션 그림이며 소실이 적음으로 보다 사실적인 모습을 보인다. (그림 10)은 제안한 방법을3차원으로 확장하여 시뮬레이션 한 결과이다.

2차원으로 각 격자 크기에 따라 시뮬레이션하였을 때의 수행 결과는 <표 1>과 같았다. CPU 구현을 하였을 경우에는 본 논문에서 제안하는 방법이 다소 오래 걸렸으나 큰 차이는 나지 않았다. 이 시간 차이는 각 격자를 한번 검색하는 것에 해당하는 시간인데, GPU의 병렬 처리 연산을 할 경우에는 별다른 차이가 없을 것으로 예상된다.

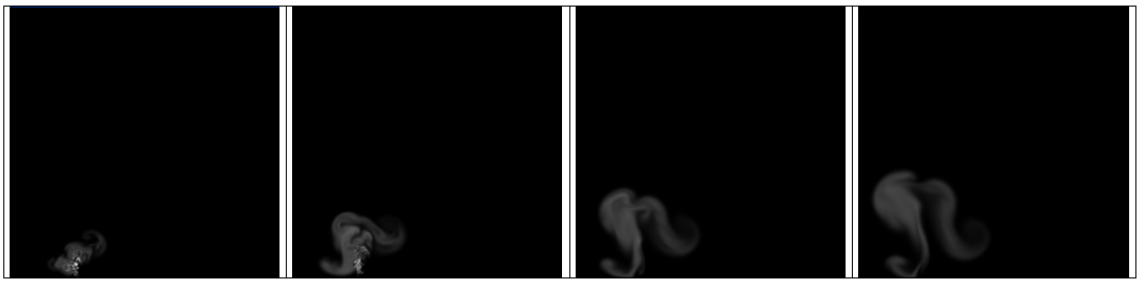


(a) STAM 방법으로 이류항 계산

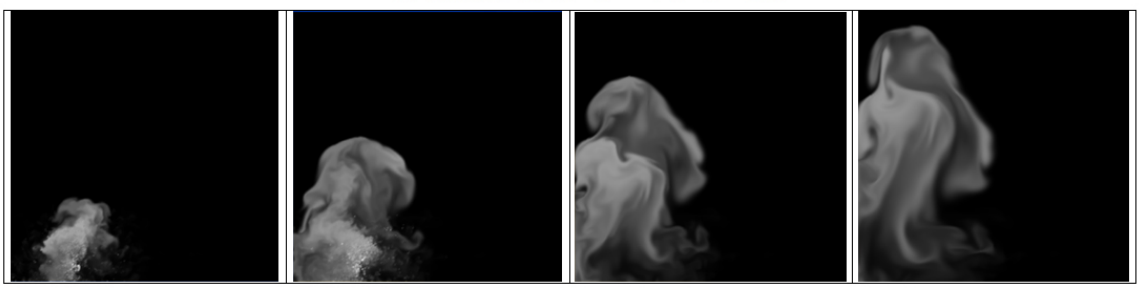


(b) 제안한 방법으로 이류항 계산

(그림 8) $\triangle t$=0.1이고 소용돌이를 주었을 때 시뮬레이션 결과

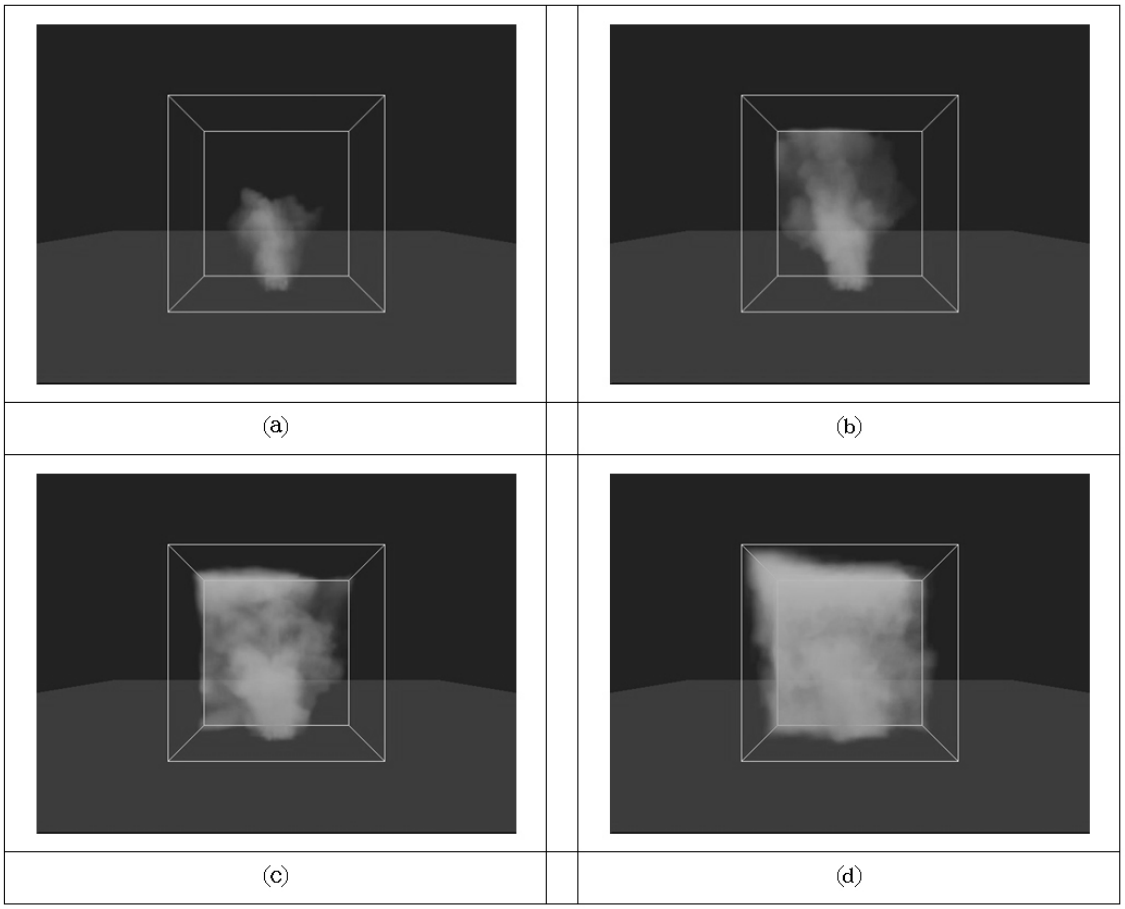


(a) STAM 방법으로 이류항 계산



(b) 제안한 방법으로 이류항 계산

(그림 9) $\triangle t$=0.1이고 소용돌이를 주었을 때 256×256 격자에서의 시뮬레이션 결과



(그림 10) 3차원에서 연기 시뮬레이션

〈표 1〉 Stam의 방법과 제안한 방법의 시뮬레이션 수행 시간 비교
(단위:msec)

	number of grids			
	64	128	256	512
Stam의 방법	4.2	17.1	92.4	337.2
제안한 방법	4.7	18.9	107	375.9

5. 결 론

본 논문에서는 연기의 움직임을 사실적으로 표현하기 위해, Semi-Lagrange 방법의 이류항 연산 과정에서 물리량을 읽어올 격자를 찾는 과정을 개선하였다. Semi-Lagrange 방법의 이류항 연산과정은 필요한 물리량을 가진 격자를 찾는 것과 그 격자 내에서 해당되는 값을 보간법으로 계산하는 과정으로 구성된다. 그래픽스 분야에서는 보간법의 개선을 통하여 이류항 연산의 정확도를 높이는 방법은 다양하게 연구되어 왔으나, 필요한 정보를 가진 격자를 효과적으로 찾아내는 추적법에 대한 연구는 상대적으로 취약했다.

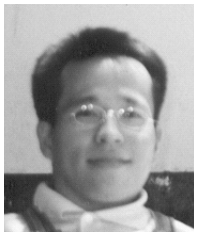
본 논문에서는 이 문제점을 개선하기 위해 이류항 연산에서 기존의 역추적법 대신에, 보다 직관적이고 효율적인 방법으로 현재의 격자로 오게 될 속도를 가진 격자를 찾았다. 찾은 후에 그 격자의 물리량들을 선형 보간하여 사용함으로써 기존의 방법보다 소실을 줄이는 방법을 제안했다. 그 결과 밀도와 속도의 소실량이 감소하여 역동적인 연기를 세밀하게 표현할 수 있었다. 또한 제시한 방법은 Stam의 방법에서 추적법을 개선한 것이기 때문에 긴 시간 간격에서도 안정적인 시뮬레이션이 가능하다. 처리 속도의 가속을 위해서는 GPU 구현이 가능할 것이고, 다음 단계에서 진행될 예정이다.

본 논문이 제안하는 방법은 수치적인 정확도를 직접적으로 높이려는 시도 대신, Euler 구조에서의 Navier-Stokes 방정식 계산 과정에서의 소실량을 줄임으로써 사실성을 높일 수 있음을 보였다. 격자 사이의 값들을 보다 정밀하게 보간하는 방법들을 적용하여 정확도를 더 높일 수 있을 것이다.

참 고 문 헌

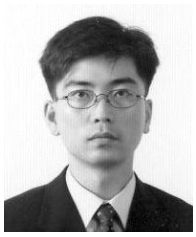
[1] E. Wu, Y. Liu, and X. Liu, “An improved study of real-time fluid simulation on GPU,” *Comp. Anim. Virtual Worlds*,

pp.139-146, 2004.
[2] N. Foster and D. Metaxas, "Modeling the Motion of Hot, Turbulent Gas," *SIGGRAPH'99*, pp.181-188, 1997.
[3] J. Stam, "Stable fluids," *SIGGRAPH'99* , pp.121-128, 1999.
[4] P. Bartello and S. J. Thomas, "The Cost-Effectiveness of Semi-Lagrangian Advection," *Monthly Weather Review*, pp.2883-2897, 1996.
[5] R. Fedkiw, J. Stam, and H. W. Jensen, "Visual simulation of smoke," *SIGGRAPH'01*, pp.15-22, 2001.
[6] O.-Y. Song, H. Shin, and H.-S. KO, "Stable but Non-Dissipative Water," *ACM Transactions on Graphics*, pp.81-97, 2005.
[7] D. Kim, O.-Y. Song, and H.-S. KO, "A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting," Proceedings of EUROGRAPHICS 2008, pp.467-475, 2008.
[8] T. YABE and T. AOKI, "A universal solver for hyperbolic equations by cubic-polynomial interpolation i. one dimensional solver" *Computer Physics Communications 66*, 2-3, 219-232, 1991.
[9] T. YABE, T. ISHIKAWA, P. Y. WANG, T. AOKI, Y. KADOTA, and F. IKEDA, "A universal solver for hyperbolic equations by cubic-polynomial interpolation ii. Two and three-dimensional solvers" *Computer Physics Communications 66*, 2-3, 233-242. 1991.
[10] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Using BFECC for fluid simulation" *In Eurographics Workshop on Natural Phenomena*, 2005.
[11] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac, "Advections with significantly reduced dissipation and diffusion," *IEEE Trans. Vis. Comput. Graph. 13(1)*, pp.135-144, 2007.
[12] A. Selle , R. Fedkiw, B.-M. Kim, Y. Liu and J.Rossignac, "An unconditionally stable MacCormack method," *Journal of Scientific Computing*, pp.350-371, 2008.
[13] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph. 21*, pp.736-744, 2002.
[14] A. Selle, N. Rasmussen, and R. Fedkiw, "A vortex particle method for smoke, water and explosions," *ACM Trans. Graph. 24*, 2005.
[15] O.-Y. Song, D. KIM, H.-S. KO, "Derivative particles for simulating detailed movements of fluids," *IEEE Transactions on Visualization and Computer Graphics 13*, pp.711-719, 2007.
[16] J. Stam, "Real-Time Fluid Dynamics for Games," *Proceedings of the Game Developer Conference*, 2003.



박 수 완

e-mail : ipsuwan@gmail.com
1999년 경북대학교 유전공학과(학사)
2002년 경북대학교 컴퓨터공학과(공학석사)
2002년~현 재 경북대학교 컴퓨터공학과 박사과정
관심분야 : 컴퓨터 그래픽스, 알고리즘



백 낙 훈

e-mail : oceancru@gmail.com
1990년 한국과학기술원 전산학과(학사)
1992년 한국과학기술원 전산학과(공학석사)
1997년 한국과학기술원 전산학과(공학박사)
2004년~현 재 경북대학교 전자전기컴퓨터학부 부교수
관심분야 : 모바일 그래픽스, 리얼타임 그래픽스

유 관 우

e-mail : kwryu@bh.knu.ac.kr
1980년 국립경북대학교 전자공학과(공학사)
1982년 한국과학기술원 전산공학과(공학석사)
1990년 메릴랜드대학교 전산공학과(공학박사)
1991년~현 재 경북대학교 컴퓨터공학과 교수
관심분야 : 멀티패러다임 프로그래밍, 병렬 알고리즘, 컴퓨터 그래픽스등