GPU를 이용한 2차원 영상 기반 유동 가시화 기법의 가속

Acceleration of 2D Image Based Flow Visualization using GPU

이중연
한국과학기술정보연구원

Lee Joong-Youn
Korea Institute of Science and Technology Information

요약

유동 가시화란 가시화 기술의 한 영역으로, 벡터 데이터를 2차원 또는 3차원의 형태로 시각적으로 표출하는 것을 말한다. 즉, 일반적으로 벡터 데이터는 (x, y, z)의 형식으로 이루어져 있는 수열의 집합인데, 이를 사람이 그 특징을 쉽게 인지할 수 있도록 그림 또는 애니메이션으로 표시하는 것을 말한다. 유동 가시화 기법에는 여러 가지가 있지만 영상 기반 유동 가시화 기법(IBFV)은 현존하는 조밀한 인티그레이션 기법들 중 가장 빠른 기법 중 하나이다. 본 논문에서는 GPU를 이용해서 영상 기반 유동 가시화 기법을 가속하고 이를 구현했는데, 특히, 메쉬 어드벡션 (mesh advection)을 꼭지점 프로그램을 이용해서 가속했다.

Abstract

Flow visualization is one of visualization techniques and it means a visual expression of vector data using 2D or 3D graphics. It aims for human to easily find and understand a special feature of the vector data. The Image Based Flow Visualization (IBFV) is one of the fastest technique in the dense integration based flow visualization techniques. In this paper, IBFV is accelerated and implemented using commodity GPU. Especially, mesh advection is accelerated at the vertex program.

I. 서 론

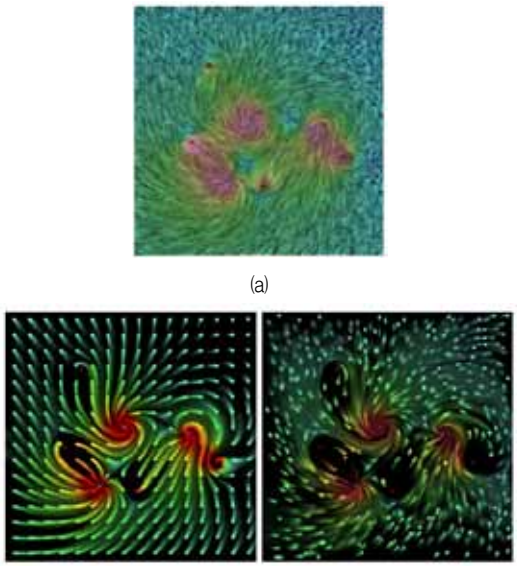
유동 가시화(Flow Visualization)란 가시화의 한 분야로, 벡터 데이터를 2차원, 또는 3차원의 형태로 시각적으로 표출하는 것을 말한다. 즉, 일반적으로 벡터 데이터는 (x, y, z)의 형식으로 이루어져 있는 수열의 집합인데, 이를 사람이 그 특징을 쉽게 인지할 수 있도록 그림 또는 애니메이션으로 표시하는 것이다. 유동가시화는 산업, 연구, 교육 등을 망라한 매우 다양한 분야에서 활용되어 지고 있으며 자동차 산업, 항공기 산업, 기상 예측, 의료 영상 등의 분야가 대표적이다. 이들 분야에서는 공통적으로 벡터 데이터를 다루는데, 이를 얼마나 빠르게, 그리고 얼마나 효과적으로 가시화하느냐는 그 분야에서 경쟁력을 유지하기 위한 필수적인 요소라고 할 수 있다. 본 논문에서는 2차원 유동 데이터에 대한 영상기반 유동 가시화 기법을 GPU를 이용해서 가속했고, CPU를 이용하는 경우와 GPU를 이용하는 경우를 서로 비교했다. 이를 통해 2차원의 유동 데이터를 실시간으로 가시화하고자 했다.

II. 영상 기반 유동 가시화

1. 개요

영상 기반 유동 가시화는 유동 데이터의 각 벡터에 사상되는 메쉬(mesh)를 생성하고 이를 벡터의 방향에 맞춰서 어드벡션(advection) 시킴으로써 유동 가시화를 구현하는 방법이다. 이때 메쉬의 움직임을 인간이 쉽게 인지할 수 있도록 노이즈 텍스쳐를 메쉬에 입힘으로써 LIC(Line Integral Convolution) 기법과 유사한 가시화 결과를 얻을 수 있다. 각 메쉬에 노이즈 텍스쳐를 매핑시키고 메쉬를 벡터의 방향으로 움직이면 노이즈 텍스쳐 역시 따라서 움직이게 되는데 이는 파티클 어드백션(particle advection)과 같은 결과를 가져온다. 스트림라인(streamline)을 생성하기 위해서는 움직이는 파티클의 궤적을 저장해야 하는데 이는 LEA(Lagrangian Eulerian Advection)의 경우와 같이 이전 가시화 결과를 모두 블렌딩하는 방법으로 구현이 가능하다. 영상 기반 유동 가시화는 LEA와 달리 텍스쳐의 모든 텍셀에 대해서 어드벡션하지 않고 메쉬를 어드벡션하고 중간의 텍셀들은 그래픽스 하드웨어의 이선형보간(bilinear interpolation)을 이용했다는 점에서 보다 빠른 가시화를 가능하게 했다. 또한, 이 기법은 텍스쳐를 이용해서 전체적인 가시화 스타일을 변경할 수 있다. 일반적인 노이즈 텍스쳐를 사용하면 LIC 스타일의 가시화 결과를 얻을 수 있고, 텍셀의 일부분만 색깔을 칠하고 블렌딩하는 연산을 바탕 영상들이 빠르게 사라지거나 또는 느리게 사라지도록 변

경함으로써 다양한 스타일의 가시화 기법을 에뮬레이션할 수 있다. 그림 1은 LIC 스타일, arrow 스타일, particle 스타일의 가시화 결과를 보여준다.



(a) (b) (c)

▶▶ 그림 1. IBFV의 여러 가시화 스타일
(a) LIC ,(b) arrow, (c) particle

2. 알고리즘

영상기반 유동 가시화의 실제 구현을 위한 알고리즘은 다음과 같다.

1. 메쉬 생성
2. 새로운 노이즈 텍스쳐 생성
3. 메쉬의 각 꼭지점을 할당된 벡터의 방향과 크기로 이동
4. 메쉬에 노이즈 텍스쳐 매핑
5. 어드벡션된 노이즈 텍스쳐를 프레임버퍼에서부터 읽음
6. 이전 프레임들에서 생성된 결과와 블렌딩

2.1 메쉬 생성

영상기반 유동 가시화는 노이즈 텍스쳐를 어드벡션시킴으로써 유동을 가시화하는 기법이다. 실제 구현시 노이즈 텍스쳐를 OpenGL 파이프라인에 넣을 수 있는 유일한 방법은 기본 메쉬를 생성하고 이 메쉬 위에 텍스쳐를 매핑하는 것이다. 이를 위해 새로운 메쉬를 생성해야 하는데, 이 메쉬는 실제 유동 데이터의 도메인 상에 정의된 메쉬와 동일하게 생성한다.

2.2 새로운 노이즈 텍스쳐 생성

이번 프레임에서 사용할 새로운 노이즈 텍스쳐를 생성한다. 매 프레임마다 동일한 노이즈 텍스쳐를 사용하면 일정한 유동(steady flow)의 경우엔 가시화 영상이 정지한 것으로 보이게 된다. 그러나 매 프레임마다 살짝 다른 노이즈 텍스쳐를 사용하면 일정한 유동의 경우에도 계속 움직이는 영상을 얻을 수 있다.

2.3 메쉬의 꼭지점 이동

메쉬의 각 꼭지점을 할당된 벡터의 방향과 크기로 이동시킨다. 즉, 어드벡션 시키는 것인데 메쉬의 꼭지점이 이동하면 이에 매핑된 텍스쳐도 함께 이동하게 됨으로써 particle advection이 구현된다. 이때, 메쉬는 이전 프레임에 어드벡션된 메쉬가 아니라 초기화된 메쉬이다. 어드벡션 계산이 된 메쉬는 OpenGL 파이프라인으로 넘어가게 된다.

2.4 메쉬에 노이즈 텍스쳐 매핑

메쉬의 각 꼭지점에 노이즈 텍스쳐를 매핑한다. 이전 단계에서 메쉬가 어드벡션되었기 때문에 노이즈 텍스쳐 역시 어드벡션되게 된다. 이 과정은 그래픽스 하드웨어에 의해 실시간으로 처리된다.

2.5 프레임버퍼 읽기

어드벡션된 노이즈 텍스쳐를 프레임버퍼로부터 읽어 온다. 이렇게 읽어온 가시화 결과는 노이즈 텍스쳐의 각 노이즈(particle)들이 어드벡션된 결과로써 아직 스트림라인이 생성되지는 않았다. OpenGL에서 glReadPixels(); 함수를 이용하면 된다.

2.6 이전 프레임들에서 생성된 결과와 블렌딩

이전 프레임들에서 생성된 결과들과 블렌딩함으로써 스트림라인을 생성한다. 이렇게 생성된 결과는 LIC 기법과 유사한 이미지가 된다. 이 이미지는 다음 프레임을 위해 저장된다. 그래픽스 하드웨어의 하드웨어 블렌딩 연산을 사용하면 실시간으로 연산이 가능해진다.

III. GPU를 이용한 가속 기법

1. 개요

II장에서는 일반적인 PC에서 OpenGL을 이용한 영상 기반 유동 가시화의 구현에 대해 설명했다. 영상기반 유동 가시화 기법은 기법 자체에서 노이즈 텍스쳐를 메쉬에 매핑시키고 벡터의 방향에 따라 메쉬의 꼭지점을 움직이는 등 OpenGL 등의 3D 그래픽스 기법을 사용하기 때문에 3D 그래픽스 하드웨어를 이용한 속도 향상이 용이하다. 이러한 유동 가시화 기법에서는 메쉬의 각 꼭지점들을 벡터의 방향에 따라 어드벡션해야 하는데 이는 CPU에서 계산하도록 구현되었다. 즉, 기본적

인 메쉬가 입력되면 CPU에서는 각 메쉬에 해당되는 벡터를 읽어서 그 방향에 따라 꼭지점을 이동하는 연산을 수행한 뒤 이를 이용해서 새롭게 어드벡션된 메쉬를 생성하여 OpenGL 파이프라인으로 넘기는 것이다. 이러한 CPU를 이용한 메쉬 어드벡션은 매 프레임마다 새로운 메쉬를 생성해야 하는 점에서 속도를 떨어뜨리게 된다.

본 논문에서는 이러한 메쉬의 어드벡션을 GPU에서 계산해서 속도를 빠르게 하는 방법을 제안한다. GPU의 꼭지점 쉐이더는 OpenGL 파이프라인으로 들어온 꼭지점들을 이동시키는 기능이 있다. 이 기능을 이용하면 CPU에서 새로운 메쉬를 생성할 필요없이 매 프레임마다 기본적인 메쉬를 이용해서 빠르게 영상 기반 유동 가시화를 구현할 수 있다.

꼭지점 쉐이더에서 꼭지점의 위치를 움직이기 위해서는 유동 데이터에서 해당되는 벡터를 가져와야 하는데, 이는 유동 데이터를 꼭지점 텍스쳐(vertex texture)로 생성함으로써 가능해진다.

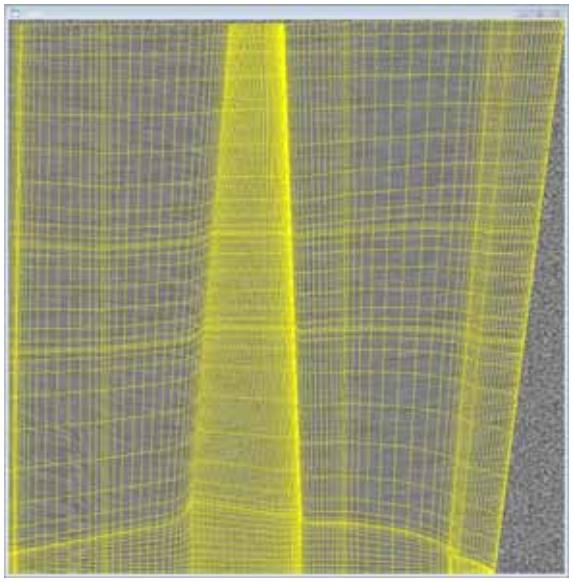
2. 꼭지점 쉐이더에서의 메쉬 어드벡션

영상기반 유동 가시화를 GPU로 구현하게 되면 유동 텍스쳐와 노이즈 텍스쳐 등 두 종류의 텍스쳐가 필요하게 된다.

꼭지점 쉐이더에서는 유동 텍스쳐를 필요로 하게 되고 프래그먼트 쉐이더에서는 노이즈 텍스쳐를 필요로 하게 되는데, 여기에 매 프레임마다 노이즈 텍스쳐의 내용이 바뀌게 됨을 감안하면 프로그램 내에서 텍스쳐 관리에 신경을 써야한다. 특히, 어드벡션 계산이 끝나고 살짝 이동된 노이즈 텍스쳐를 입히고 어드벡션된 유동 가시화 결과를 프레임버퍼로 부터 읽어올 때는 꼭지점 및 프레그먼트 쉐이더를 비활성화시키고 유동 텍스쳐 역시 비활성화 시켜야 한다. 또, 프레임버퍼로 부터 읽어온 가시화 결과를 다시 텍스쳐로 생성해야 하는데 이렇게 생성된 텍스쳐는 결국 다음 프레임때 노이즈 텍스쳐로 이용되기 때문에 현재 노이즈 텍스쳐로 대체할 수 있도록 현재 텍스쳐를 노이즈 텍스쳐로 설정해야 한다.

IV. 구현 결과

본 논문에서 구현에 사용한 PC는 Intel Core2 Quad Q6600 (2.4GHz) CPU와 4GB의 메모리 그리고 nVidia GeForce 8800 GPU를 장착한 모델이다. 이 PC에서 그림 2와 같은 고등훈련기의 날개 부분을 가시화했는데, CPU와 GPU에서 구현한 결과는 표 1과 같은데, GPU를 사용할 경우 CPU를 사용할 경우보다 약 2배가량 속도가 빠름을 알 수 있다.



▶▶ 그림 2. 고등훈련기의 영상기반 유동 가시화

[표 1] 구현 방법에 따른 영상기반 유동 가시화 성능비교

구현 방법	CPU	GPU
속도	103.4 fps	217.8 fps

V. 결 론

본 논문에서는 2차원 유동 데이터를 위한 영상 기반 유동 가시화 기법을 일반 PC에서 구현하고 GPU를 이용한 새로운 가속 기법을 제안했다. 특히, GPU의 꼭지점 쉐이더의 꼭지점 텍스쳐 기능을 이용해서 메쉬의 각 꼭지점들을 해당되는 벡터 방향에 따라 실시간으로 어드벡션함으로써 매우 빠르게 렌더링 결과를 생성할 수 있었다. 본 논문에서 구현한 영상 기반 유동 가시화 기법은 여타 다른 조밀한 인티그레이션 기반의 유동 가시화 기법과 마찬가지로 유동의 전체적인 흐름을 한눈에 파악할 수 있게 한다는 장점이 있다. 또, 기존의 방법들과는 달리 Seed Point를 결정해야할 필요가 없다는 점에서 보다 간편한 가시화를 가능케 한다. 이러한 기능은 연구자가 보고자 하는 피쳐(feature)를 찾고 이를 표출하기 위한 Seed Point를 지정해야 하는 여타 다른 유동 가시화 기법에 비해 간편하게 전체 유동 데이터의 특성을 표출하도록 하는 측면에서 e-Scinece 응용 분야나 계산 제어(computational steering) 분야에 보다 적합하다고 하겠다. 2차원의 유동 가시화는 유체역학, 대기과학 등 다양한 분야에서 중요하게 다루어지고 있는 분야로써 미국, 독일 등 여러 선진국에서 광범위한 연구가 진행 중이다. 우리나라에서도 구미 선진국과 같이 관련 연구가 보다 활발하게 이루어져야 하겠다.

▌참 고 문 헌▐

[1] B. Cabral, L. Leedom, "Imaging Vector Fields using Line Integral Convolution", Proceedings of SIGGRAPH93, pp. 263-270, 1993

[2] L. Fossel, S. Cohen, "Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable Speed Animation and Unsteady Flows", IEEE Transactions of Visualization and Computer Graphics, Vol.1, 1995

[3] D. Weiskopf, M.Hopf, T. Ertl, "Hardware-accelerated visualization of time-varying 2D and 3D vector fields by texture advection via programmable per-pixel operations", Vision, Modeling, and Visualization VMV '01 Conference Proceedings, pp. 439-446, 2001

[4] B. Jobard, G. Erlebacher, and M.Y. Hussaini, "Lagrangian -Eulerian Advection for Unsteady Flow Visualization", Proceedings IEEE Visualization 2001, 2001

[5] J. van Wijk, "Image based flow visualization", ACM Transactions on Graphics, Vol. 21, Num. 3, pp. 745-754, 2002

[6] A. Telea, J. van Wijk, "3D IBFV: Hardware-accelerated 3D flow visualization", Proceedings of IEEE Visualization 2003, pp. 233-240, 2003

[7] G. Li, X. Tricoche, C. Hansen, "GPUFLIC: Interactive and Accurate Dense Visualization of Unsteady Flows", Proceedings of Eurographics/IEEE-VGTC Symposium on Visualizations, pp. 29-33, 2006