LIC 기반의 유동 가시화 기법에 대한 조사 연구

Survey on the LIC based flow visualization

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 가시화는 가시화 기법, 대상 데이터의 차원, 대상 유동의 종류 등 여러 가지 기준으로 분류가 가능하다. 가시화 기법은 크게 직접 기법, 인티그레이션(integration) 기법, 파생 데이터 기반 기법 등으로 나눌 수 있고, 데이터의 차원은 2차원, 2.5차원, 3차원 등으로 구분할 수 있으며, 유동의 종류는 일정한(steady) 유동과 불규칙한(unsteady) 유동으로 나눌 수 있다. 이러한 유동 가시화는 그 종류가 매우 많고 다양한데, 본 논문에서는 대표적인 인티그레이션 기법인 LIC 기법에 초점을 맞추고 각 기법들을 데이터의 차원으로 분류하고 각 기법의 장단점을 논하고자 한다.

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 understand a special feature of the vector data. Flow visualization can be classified into various criterions such as visualization technique, data dimension, type of the flow, and so on. Visualization technique can be categorized into direct method, integration method and derived data based method. Data dimension can be divided into 2D, 2.5D and 3D. Type of flow data may be classified into steady and unsteady. In this paper, various LIC based flow visualization methods will be introduced which is one of representative integration based techniques. Those methods will be categorized with more detailed criterions such as dimension and type of flows.

I. 서 론

1. 유동 가시화

유동 가시화(Flow Visualization)란 여러 가시화 분야 중 하나로써, 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. 유동 가시화는 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 대표적으로 널리 사용되는 여러 유동 가시화 기법들 중 LIC(Line Integral Convolution) 기반 기법에 대해 설명하고자 한다.

2. LIC 기반 기법

LIC를 이용한 유동 가시화 기법은 유동 데이터 가시화 기법 중 텍스쳐 기반 기법과 함께 가장 널리 사용되는 기법 중 하나이다. 특히, LIC 기법은 전체 유동 데이터의 특징을 한눈에 볼 수 있도록 전체 영역에 걸쳐서 스트림라인 (또는 패스라인)을 생성하도록 한다. 이러한 방법을 이용해서 LIC 기법은 기존의 기법과는 달리 중요한 피쳐(feature)를 비교적 간단하게 시각화할 수 있다. 반면 전체 영역에 걸쳐서 스트림라인을 생성하기 때문에 계산량이 많아진다는 단점이 존재한다. 2장에서는 LIC가 무엇인지 소개하고, 3장, 4장 5장에서 이를 이용한 유동 데이터의 가시화 기법을 각각 2D, 2.5D, 3D등 차원에 따라 설명하고 6장에서 결론에 대해 논하도록 하겠다..

II. LIC

1. LIC 소개

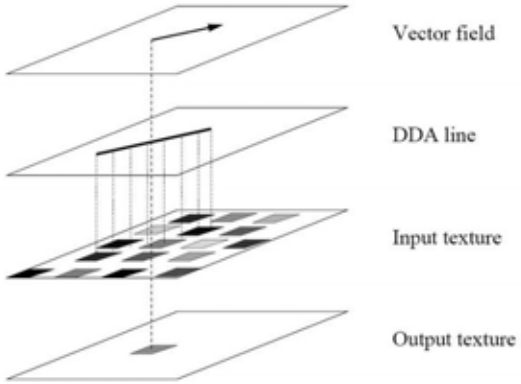
LIC(LIne Integral Convolution)는 벡터 데이터를 비교적

정확하게 표시할 수 있는 강력한 기술이지만 근본적으로는 주어진 벡터장을 따라서 입력된 이미지(텍스쳐 또는 복셀세트)를 흐리게 하는 필터링 기술을 말한다. LIC는 벡터장과 복셀세트를 입력받아 각 복셀들의 명암도가 벡터장의 인테그랄 선(Integral Line)을 따라 일치하도록 해서 벡터 데이터의 특징을 시각화한다. 이 기법은 1993년에 Cabral과 Leedom에 의해서 처음 발표된 뒤 지금까지도 벡터 데이터를 가시화하는데 빈번하게 사용되는 대표적인 벡터 데이터 가시화 기법이다[1].

▶▶ 그림 1. LIC 필터 적용 예

2. DDA 콘볼루션 연산

LIC는 DDA 콘볼루션(DDA Convolution)이라 불리는 테크닉의 변형판이다. DDA 콘볼루션이란 그림 2와 같이 벡터의 방향에 따라 DDA 선을 생성한 뒤 이 선위에 위치하는 입력 텍스쳐의 텍셀(texel)들을 콘볼루션하는 것을 말한다. DDA 콘볼루션에서 DDA 선을 생성할 때, 시작점에서부터 벡터의 진행방향과 진행 반대방향으로 L만큼 그리는데, 이 L의 길이가 길어질 수록 입력 텍스쳐 이미지가 많이 흐려지고 짧을 수록 원본 입력 텍스쳐와 비슷해진다. 이 콘볼루션은 벡터의 진행 방향 및 진행 반대방향에 위치한 텍셀들의 값을 서로 연관시켜서 그 진행 방향에 따라 텍셀들이 서로 유사한 값을 가지도록 해서 벡터장의 전체적인 모습을 가시화한다.



▶▶ 그림 2. DDA 콘볼루션 연산

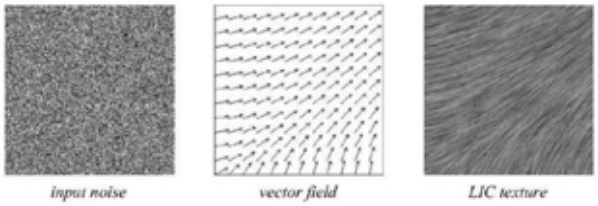
3. LIC 연산

DDA 콘볼루션을 적용한 벡터 데이터 가시화 기법은 벡터장을 직선으로 정의할 수 있다는 가정 하에 이루어졌기 때문에 실제 벡터 데이터를 정확하게 표현하지 못한다. LIC는 벡터장을 실제 데이터와 유사하게 곡선으로 표현해서 이러한 DDA의 약점을 보완했다. 이러한 곡선은 위에서 설명한 어드벡션 기법 들 중 하나를 이용해서 생성하는데, 빠른 속도가 필요한 경우에는 오일러(Euler) 기법이, 높은 정확도가 필요할 경우에는 4차원 Runge-Kutta 기법이 많이 사용된다.

주어진 곡선에서, LIC는 다음과 같이 정의된다 :

$$I(X_0) = \int_{S_0 - L}^{S_0 + L} k(s - s_0)\, T(\sigma(s))ds \quad (1)$$

여기서 $I(X_0)$는 $X_0 = \sigma(s_0)$에 위치한 복셀의 명암도이고, k는 $2L$의 길이를 갖는 필터 커널이며 T는 입력 텍스쳐를 의미한다. LIC 기법에서도 DDA 콘볼루션 기법과 마찬가지로 L의 길이에 따라 텍스쳐의 흐려짐의 정도가 달라지는데, Stalling과 Hege에 따르면 콘볼루션 길이 $2L$을 텍스쳐 이미지 길이의 1/10로 정할때 가장 좋은 결과를 냄을 알수 있었다고 한다[3]. 그림 3은 벡터 데이터와 노이즈 텍스쳐를 이용한 LIC 가시화의 예이다.



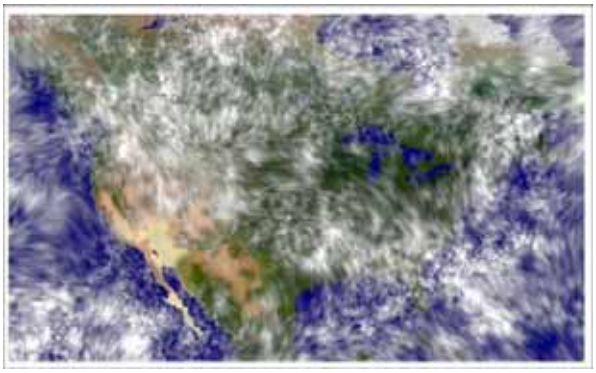
▶▶ 그림 3. 벡터 데이터와 노이즈 데이터를 이용한 2D LIC

III. 2차원 기법

1. 기본 LIC

Cabral과 Leedom은 LIC를 이용한 유동 데이터의 가시화를 처음으로 제안했다[1]. 이 기법은 II장에서 설명한 LIC와 노이즈 이미지를 이용해서 유동 데이터의 벡터의 흐름을 가시화한다. 기본적인 LIC 필터링 방법만을 이용해서 유동 데이터를 가시화할 경우 전체적인 벡터장의 모습은 가시화가 가능하지만 실제 벡터들이 어떤 방향으로 이동하는지는 알 수 없다. 이러한 점을 해결하기 위해 Cabral과 Leedom은 모양이 물결 모양의 함수를 필터로 사용했고 벡터의 진행방향으로 함수가 이동하도록 함으로써 유동의 진행 방향에 따라 텍스쳐가 움직이도록 했다. 그림 4는 이 알고리즘을 사용해서 바람의 이동을

가시화한 결과이다.



▶▶ 그림 4. LIC 기법을 이용해서 가시화한 바람장

2. Fast LIC

Stalling와 Hege는 기존의 LIC 기법을 보완해서 보다 향상된 속도의 LIC 기법을 제안했다[3]. 이를 Fast LIC라고 한다. 속도 향상은 크게 두 가지의 아이디어로 이루어졌는데, 첫 번째는 기존의 LIC 기법에서 있었던 불필요한 스트림라인 계산을 최소화한 것이고, 두 번째는 같은 스트림라인에 있는 다른 픽셀들의 계산을 재활용하는 방법으로 계산량을 줄인 것이다. 일반적으로 LIC 기법에서는 하나의 스트림라인이 여러 개의 픽셀을 차지하게 되는데, 각 픽셀들의 계산은 인접한 픽셀들의 계산과 모두 비슷하게 된다. Fast LIC 기법에서는 이러한 픽셀들에 대해 모두 독립적으로 LIC 계산을 수행하지 않고 주변 픽셀의 계산 결과를 활용하게 함으로써 속도를 향상시켰다. Fast LIC는 기존의 LIC 기법에 비해 속도가 빠를 뿐만 아니라 정확도 또한 높다. 기존의 LIC 기법은 스트림라인을 생성할 때 속도 향상을 위해 오일러(Euler) 기법을 이용했지만 Fast LIC에서는 4차원 Runge-Kutta 기법을 사용했다.

3. OLIC / FROLIC

기존의 LIC 기반 가시화 기법들은 유동 데이터의 전체적인 벡터의 흐름을 한눈에 파악할 수 있게 해주지만 애니메이션을 사용하지 않고 이미지 한 장만으로 시각화했을 경우 흐름의 방향을 파악하기엔 곤란했다. OLIC(Oriented LIC)는 Wegenkittl이 제안한 유동 데이터 가시화 알고리즘으로 이미지 한 장으로 유동 데이터의 흐름을 파악할 수 있도록 했다[5]. 이것은 노이즈 텍스처의 각 점들을 콘볼루션할 때 벡터의 진행 방향에 따라 번져보이게 하는 기법으로 가능했다.

FROLIC(Fast Rendering of OLIC)는 이름 그대로 OLIC의 속도 향상 기법이다. 이 기법은 Wegenkittl과 Groller에 의해 제안되었는데, 속도를 빨리하기 위해 정확도를 희생함으로써 가능했다. FROLIC에서는 OLIC의 물방울 형태의 벡터 추적을 단순화해서 속도를 향상시켰다.

4. GPUFLIC

Li 등은 Shen 등의 UFLIC를 GPU에서 구현했다[8]. UFLIC의 알고리즘을 GPU의 프래그먼트 쉐이더(Fragment Shader)를 이용해서 구현했는데, GPU의 특성에 맞추어 기존의 알고리즘을 변형했다. 우선, UFLIC에서는 Value Depositing을 구현하기 위해 각 픽셀마다 Ring Bucket 형태의 버퍼를 두고 LIC 계산을 했는데, GPU에는 이러한 Ring Bucket 형태의 버퍼가 없기 때문에 구현에 문제가 생긴다. 이를 해결하기 위해 GPUFLIC에서는 각 픽셀들을 Scattering 할 때 한꺼번에 그 픽셀이 영향을 미치는 모든 픽셀들에 보내지 않고 현재 프레임에서 영향을 미치는 픽셀에만 보내는 방법으로 버퍼의 필요성을 없앴다. 이 경우, LIC 계산 시에 이전 프레임에서 계산한 값을 재사용하기 힘들어지는 문제가 발생하지만 GPU 자체의 매우 빠른 SIMD 병렬 구조를 이용해서 기존 방법보다 월등히 빠른 성능을 나타낼 수 있었다.

IV. 2.5차원 기법

1. Curvilinear LIC / Unsteady LIC

Forssell과 Cohen은 Cabral과 Leedom의 LIC를 이용한 유동 데이터 가시화 알고리즘을 확장해서 이 알고리즘을 Curvilinear 좌표계의 데이터와 불규칙한(unsteady) 데이터에 대해서도 적용할 수 있도록 했다[2]. 그들은 유동 데이터의 좌표계를 물리 좌표계(physical coordinate.)와 계산 좌표계(computational coordinate.)로 나누어 가시화 작업을 처리했는데, 계산 좌표계는 유동 계산이 이루어진 curvilinear 좌표계이고 물리 좌표계는 실제 물체의 좌표계인 rectilinear 좌표계이다. curvilinear 좌표계에서는 좌표계의 특성상 LIC 계산이 어렵기 때문에 이 좌표계를 rectilinear 좌표계로 변환한 뒤 일반적인 LIC 기반 가시화 방법을 사용한 것이다. 이들 좌표계는 Jacobian 행렬을 이용해서 상호 변환이 가능하다. 또, Forssell과 Cohen은 LIC를 이용해서 불규칙한 유동 데이터를 가시화했다. 기존의 방법에서는 안정한(steady) 데이터에 대해서만 가시화가 가능했지만 이를 확장한 것이다. Cabral과 Leedom의 방법에서는 스트림라인을 생성해서 가시화했기 때문에 불규칙한 유동 데이터를 가시화할 경우엔 Temporal Coherency가 매우 작아서 문제가 있었지만 Forssell과 Cohen은 패스라인을 생성하는 방법으로 가시화해서 이러한 문제를 어느 정도 해결할 수 있었다. 마지막으로 이 방법에서는 기존의 방법과 유사한 애니메이션 기법을 사용해서 벡터의 방향을 표시하였을 뿐만 아니라, 필터 함수의 진동수를 변경하

는 방법으로 벡터의 속도까지도 표시했다. 이 방법은 특히 유동의 속도가 가변적인 불규칙한 데이터의 가시화에 적합하다.

2. 표면에서의 Fast LIC

Battke 등은 Fast LIC 기법을 확장에서 3차원상의 임의의 좌표계에서 표현되는 임의의 물체의 표면에 Fast LIC 기법을 적용시켰다. 특히, 기존의 Forssell과 Cohen의 기법은 curvilinear 좌표계에 한정되었지만 이 기법은 그러한 제약을 없앴다. Battke 등은 주어진 표면을 다수의 삼각형 타일의 모음으로 변환하고 각 삼각형 타일에 LIC 텍스쳐 계산을 하는 방식을 적용했다. 그러나 이 방법은 1,600~ 4,000 개 수준의 삼각형 타일에 대해서만 적용이 가능해서 상대적으로 작은 크기의 표면의 가시화만이 가능한 문제점이 있었다.

3. UFLIC

UFLIC는 Shen과 Kao가 제안한 LIC 기반의 불규칙한 유동 데이터의 가시화 기법이다[4]. Fossell과 Cohen이 제안한 기존의 불규칙한 데이터 가시화 기법은 복잡한 불규칙한 데이터에 적용할 경우 공간 결합성(spatial coherency)와 시간 결합성(temporal coherency)이 떨어지는 문제가 있었다. UFLIC에서는 이러한 문제를 Value Depositing과 Successive Feed-Forward를 이용해서 해결했다. UFLIC의 기존의 기법에서는 LIC 계산을 할 때 각 픽셀별로 인티그레이션 선을 따라 콘볼루션 계산을 하는 "Gather" 연산을 했다. 그러나 이러한 방법은 각기 다른 패스라인이 한 픽셀에서 만날 경우 문제가 발생한다. 이러한 문제를 해결하기 위해 UFLIC에서는 Value Depositing을 이용했다. Value Depositing이란 LIC 계산을 할 때 한 픽셀에서 그 픽셀값을 계산하는데 영향을 미치는 모든 픽셀들의 값을 가져와서 LIC 계산을 하는 기존의 방법과는 달리, 각 픽셀에서 그 픽셀이 영향을 미치는 다른 픽셀들에 값을 보내면 (Scattering) 해당 픽셀들은 그 픽셀값을 저장(Deposit)해 놓고 있다가 최종적으로 모든 픽셀들의 저장이 끝난 뒤에 LIC 계산을 하는 기법을 말한다. 이러한 기법을 이용하면 공간 결합성이 떨어지는 문제를 해결할 수 있다. 또, 시간 결합성이 떨어지는 것을 해결하기 위해 Successive Feed-Forward 기법을 사용했는데, 이는 다음 프레임을 가시화할 때 입력하는 노이즈 이미지로 일반적인 노이즈 이미지를 사용하지 않고 이전 프레임의 가시화 결과를 가져오는 것이다. 이렇게 하면 이전 이미지의 픽셀들이 패스라인을 따라 한 스텝씩 이동하기 때문에 각 프레임들의 시간 결합성에 문제가 없게 된다.

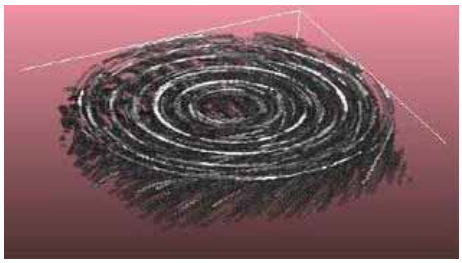
V. 3차원 기법

1. 3D LIC

3D LIC 기법은 기본적인 LIC 기법의 3차원 버전으로 Rezk-Salama 등이 제안했다[6]. 이 기법은 3차원 유동 볼륨 데이터에서 3차원 어드벡션 기법으로 인티그레이션 선을 계산하고 이를 볼륨 텍스쳐로 저장한 뒤 일반적인 3차원 텍스쳐 매핑을 이용한 볼륨 렌더링 알고리즘으로 가시화하는 기법이다. 물론 기존의 3차원 텍스쳐 매핑 기법을 이용했기 때문에 거의 실시간으로 가시화가 가능하다. 이 기법 역시 대상 데이터가 3차원 데이터이기 때문에 조밀한 인티그레이션 기법으로 가시화할 경우 문제가 발생한다. 이러한 문제를 해결하기 위해 Rezk-Salama 등은 전통적인 볼륨 렌더링 기법에서 전이함수(transfer function)과 절단면(clipping plane)을 차용했다. 즉, 전이함수을 이용해서 전체 데이터 중 덜 중요한 부분을 제거하고 절단면을 이용해서 데이터의 내부 모습을 가시화한 것이다. 전이함수는 GPU의 프래그먼트 쉐어더와 종속텍스쳐(dependent texture)로 쉽게 구현이 가능하고 절단면 역시 기본적인 OpenGL 함수로 쉽게 구현이 가능하다.

2. Seed LIC

Seed LIC는 Helgeland와 Andreassen이 제안한 3차원 유동 데이터의 가시화 기법으로 Fast LIC를 기반으로 하고 있다[7]. 3차원 유동 데이터의 경우에는 LIC의 속도를 개선한 Fast LIC에서도 오랜 시간이 걸리기 때문에 이러한 점을 개선하기 위해 고안된 기법이 Seed LIC이다. Seed LIC는 3차원 유동 데이터의 모든 복셀(voxel)에 대해서 어드벡션을 하지 않고, 미리 지정된 점들(seed)에서부터만 어드벡션을 함으로써 계산 시간을 줄였다. 일례로, 기존의 Fast LIC에서 752초의 계산 시간이 걸리는 128 X 128 X 128 크기의 3차원 데이터를 Seed LIC를 이용해서 가시화할 경우 4.7초가 걸린다고 한다. 이 가시화의 예는 그림 5에서 볼 수 있다. Seed LIC에서는 3차원 데이터에서의 깊이 정보를 용이하게 인식하게 하기 위한 방법으로 Limb Darkening을 제안했다. Limb Darkening이란 각 인티그레이션 선들의 외곽선을 어둡게 칠하는 기법인데, 기존의 쉐이딩 기법들에 비해 속도가 빠르다는 장점이 있다.



▶▶ 그림 5. Seed LIC

VI. 결 론

본 논문에서는 LIC 기반의 유동 가시화 기법들을 분류하고 각 기법들에 대해 살펴보았다. LIC 기반의 유동 데이터 가시화 기법은 유동의 전체적인 흐름을 한눈에 파악할 수 있게 한다는 장점이 있다. 또, 기존의 방법들과는 달리 Seed Point를 결정해야할 필요가 없다는 점에서 보다 빠른 가시화가 가능하다. 유동 데이터는 크게 차원으로 분류가 가능한데, 2차원과 2.5차원의 데이터에 대한 가시화는 일반적인 PC로도 실시간으로 가시화가 가능하지만 3차원 데이터의 경우에는 어드벡션 해야 하는 벡터의 양이 너무 많아 일반 PC를 이용한 실시간 가시화는 어려운 형편이다. 3차원 유동 데이터의 경우 가시화할 때 시간상의 문제뿐만 아니라 표현 상에도 문제가 있는데, 유동 데이터 내부와 외부를 한꺼번에 가시화하기 어렵다는 것이다. 이러한 점을 해결하기 위해 전이함수 적용, 회전, 절단 등의 다양한 방법들이 제시되었다. 유동 데이터는 항공, 자동차, 기상, 천체물리 등 다양한 분야에서 다루어지고 있다. 슈퍼컴퓨터 및 계측 장비의 발달로 이들 분야에서 다루는 유동 데이터의 해상도 및 크기가 기하급수적으로 커지고 있으나 이를 다루는 가시화 장비 및 기술의 진보는 더딘 편이다. 특히, 3차원의 불규칙한 유동 데이터의 경우 실시간으로 가시화할 수 있는 방법은 현재까지는 찾아볼 수 없는 실정으로 추가적인 연구가 반드시 필요하다고 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. Stalling, H. Hege, "Fast and Resolution Independent Line Integral Convolution", Proceedings of SIGGRAPH 95, pp. 249-256, 1995

[4] H.W. Shen, D. Kao, "UFLIC: A Line Integral Convolution Algorithm for Visualizing Unsteady Flows", Proceedings of IEEE Visualization 97, pp. 317-323, 1997

[5] R. Wegenkittl, E. Groller, "Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet", Proceedings of IEEE Visualization 97, pp. 309-316, 1997

[6] C. Rezk-Salama, P. Hastreiter, C. Teitzel, T. Ertl, "Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping", Proceedings of IEEE Visualization 99, 1999

[7] A. Helgeland, O. Andreassen, "Visualization of Vector Fields using Seed LIC and Volume Rendering", Proceedings of IEEE Visualization 2002, 2002

[8] 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