Journal of The Korea Society of Computer and Information
Vol. 23 No. 10, pp. 95-101, October 2018

www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.10.095

Sprite Animation Based Fire Effects Using Spark Textures and Artificial Buoyancy Field

Jong-Hyun Kim*

Abstract

In this paper, we propose an image-based synthesis method that can effectively represent the spark effect in fire simulation. We use the real flame image or animated image as inputs and perform the following steps : 1) extract feature vectors from the image, 2) calculate artificial buoyancy, and 3) generate and advect spark textures. We detect the edge from images and then calculate the feature vectors to calculate the buoyancy. In the next step, we compute the high-quality buoyancy vector field by integrating the two-dimensional feature vector and the fluid equation. Finally, the spark texture is advect by buoyancy field. As a result, our method is performed much faster than the previous approach and high-quality results can be obtained easily and stably.

▸Keyword: Sprite sheet animation, Fire effects, Spark textures, Artificial buoyancy field

I. Introduction

불을 이용한 특수효과는 영화, 애니메이션, 게임, 이미지 프로세싱 분야에 자주 사용되고 있다. 이런 불 효과를 제작하기 위해 전문 디자이너는 불 색상과 스타일을 조정하여 게임이나 영상에서 활용되는 불 효과를 매혹적이게 표현한다. 최근에는 게임엔진(예: Unity) 내 불 스타일의 템플릿과 에셋을 사용하여 영화나 게임 배경에 불 효과를 표현한다(Fig. 1 참조).

Fig. 1. Fire effects in game.

불 효과로 가장 많이 사용하는 방법은 스프라이트 애니메이션이며, 이 방법은 Unity와 Unreal과 같은 게임엔진에서도 사용되고 있다. 가장 일반적인 방법은 원본 영상에 불 템플릿 영상을 붙여 넣는 방식이다. 디자이너는 배경 이미지 위에 불을 입히고, 원하는 화염 모양과 방향, 세부 정보를 잘라내기, 붙여넣기, 삭제, 회전, 크기 조정 작업 등 수동으로 편집한다. 이 방법은 쉽게 제작을 할 수 있지만, 현실적인 불과는 다소 거리가 있어 보이는 결과가 만들어진다. 이 방법을 통해 사실적인 불 효과를 만들어 낼 수는 있지만 전문화된 기술과 오랜 시간에 걸친 값 비싼 노력이 필요하다. 스틸 프레임 기반의 불 효과는 절차적 접근법으로 결과를 자동 생성 할 수 있다[1,2]. 그러나 이 방법을 사용하면 사전에 정의된 불 효과 이외의 효과를 추가적으로 삽입하거나 편집하는 것이 어렵다.

시뮬레이션 기반의 접근법은 사실적이며 매우 상세한 화염 효과를 만들어낸다. 목표 형상을 따르도록 연료를 삽입하고 연료를 연소시킨 후 원하는 방향으로 불꽃을 유도하는 시뮬레이션을 수행함으로, 화염의 모양과 흐름을 직관적으로 제어하는 방법이다. 인터랙티브한 화염 시뮬레이션은 저해상도, 2차원 시뮬레이션으로 개발 될 수 있지만 유체의 디테일이 저하되기 때문에 결과의 품질이 떨어지고, 고품질의 불 효과가 나타나는 고해상도, 3차원 시뮬레이션은 계산양이 크기 때문에 인터랙티브 시스템에 적절하지 않다. 영상 특수효과 분야에서 시뮬레이션을 활용함에 있어서 가장 중요한 요소는 품질과 속도의 균형

• First Author: Jong-Hyun Kim, Corresponding Author: Jong-Hyun Kim
*Jong-Hyun Kim (jonghyhunkim@kangnam.ac.kr), Dept. of Software Application, Kangnam University
• Received: 2018. 08. 23, Revised: 2018. 10. 15, Accepted: 2018. 10. 18.

이다. 필수적이면서도 상호 충돌이 발생하는 품질과 속도, 두 가지를 모두 만족시키기 위해 본 논문에서는 실제 불을 촬영한 영상 또는 고해상도 시뮬레이션을 통해 만들어진 결과 영상을 분석하여 불 효과를 효과적으로 표현해낼 수 있는 새로운 프레임워크를 제안한다. 본 연구는 3차원 유체 방정식을 사용하지 않고, 입력 영상을 분석하여 계산한 인위적인 부력장을 이용하여 불꽃 텍스처를 이류 시킨다. 각 단계는 개별적인 레이어에서 계산되며 최종 결과는 모든 레이어의 합성을 통해 제작된다. 입력 영상이 실제 불 또는 고해상도 시뮬레이션을 통해 만들어진 영상이기 때문에 최종적으로 고품질의 불 스프라이트 애니메이션 결과를 만들 수 있으며, 이를 위해 본 논문에서 제안하는 기법은 아래와 같다 :

- 계산양이 큰 격자 기반의 시뮬레이션이 아닌, 이미지 시퀀스로부터 특징벡터를 추출하여 벡터장을 계산하는 방법을 제안한다.
- 특징벡터로부터 안정적으로 인위적인 부력장을 계산하고 정제하는 기법을 제안한다.
- 2차원 유체 방정식을 이용하여 시간에 따라 변화하는 부력장 생성 기법을 제안하고, 이 벡터장을 이용한 불꽃 텍스처 생성과 이류 방법을 제안한다.

본 논문은 위에서 요약한 기여도를 기반으로, 2차원 이미지 시퀀스로부터 추출된 특징벡터를 이용하여 인위적인 부력장을 모델링하고, 이 방향 벡터를 이용하여 불꽃 텍스처를 생성하고 이류시키는 새로운 프레임워크를 소개한다.

II. Preliminaries

1. Related works

유체 시뮬레이션은 영화, 게임, 영상, 미술 등 다양한 분야에서 기술 수요가 높은 분야이다. Stam은 Semi-Lagrangian 기법을 제안하여 유체 시뮬레이션의 수치적 안정성을 개선시켰다[3]. Foster와 Fedkiw는 Semi-Lagrangian 방법과 효과적 선형풀이 해법인 공액경사법(Conjugate gradient method)을 사용하여 물을 시뮬레이션 하였고, 물의 표면을 표현하기 위해 레벨셋을 이용하였다[4]. Enright 등은 레벨셋 표면에 입자들을 배치하여 시뮬레이션이 진행되면서 손실되는 유체 볼륨의 양을 최소화하는 입자-레벨셋 기법으로 유체 시뮬레이션의 정확성을 개선시켰다[5].

물리 기반 시뮬레이션 분야에서는 입자 기반으로 유체의 움직임을 효율적으로 계산하는 기법인 SPH(Smoothed particle hydrodynamics) 방법도 많이 활용되고 있다[6]. SPH 기법의 계산 양을 효율적으로 줄이기 위해 상태 방정식(Equation of state)을 활용하는 WCSPH(Weakly compressible SPH)기법이 제안되었고[7], 그 후 Solenthaler 등은 입자의 압력을 계산할 때 예측-조정 기법을 사용하여 유체의 비압축성을 보장하는 PCISPH(Predictive corrective incompressible SPH)기법을 제안하였다[8]. 이 기법은 시뮬레이션 과정에서 압력을 계산하기 전에 입자의 속도와 위치로부터 반복적으로 밀도를 예측하여 그 수치로 비압축성 압력을 계산하였다.

Nguyen 등은 그래픽스 분야에서는 처음으로 레벨셋 기반 불 시뮬레이션 및 렌더링 기법을 제안하였다[9]. 이 기법은 연료, 불의 표면, 연소 후 매질 등을 사용하고 연소 전/후의 속도를 계산하여 불의 움직임을 사실적으로 묘사하였다. Hong 등은 DSD(Detonation shock dynamics) 기법을 유체 방정식과 결합하여 불 표면의 주름 패턴을 상세하게 개선시켰다[10]. 그 후 Horvath 등은 입자와 격자의 혼합 시뮬레이션을 다중 GPU로 가속화하는 프레임워크를 제안했다. 이 기법은 시뮬레이션 공간을 여러 개의 2차원 슬라이스 조각으로 나누어 시뮬레이션 하는 방법으로, 고해상도의 불 시뮬레이션을 빠르게 수행할 수 있도록 하였다[11].

Kim 등은 화염 시뮬레이션의 시각적인 디테일을 개선시키기 위해 무발산 와동 입자와 슈퍼샘플링 기반의 레벨셋 기법을 제안하였고[16], 이 기법을 분산처리로 확장하여 영화 “7 광구”에서 사용하였다[17]. Jung 등은 다중그리드 푸아송 해법을 제안하여 불과 연기와 같은 비압축성 유체의 시각적인 디테일을 개선시켰다[14]. 뿐만 아니라, 화염의 난류를 다항식 기반으로 계산하여 난류의 흐름을 개선한 연구들도 제안되었다[15]. 불 연구 분야에서는 화염뿐만 아니라 유체의 온도장과 속도장에 따라 불똥의 움직임을 표현할 수 있는 기법이 제안되었다[12]. 렌덤 워크 기반을 사용하는 이 기법에서의 불똥은 화염의 움직임과는 다소 다른 노이즈한 움직임을 표현했다. 불똥이란 특성을 처음 표현했다는 부분은 훌륭하지만 랜덤 워크 기반이기 때문에 사실적인 불똥 효과를 표현하기엔 충분하지 않다.

최근에는 텍스처 합성 기법을 기반으로 물 시뮬레이션의 표면적 디테일을 개선시키는 연구도 진행되었다[13]. 2차원 텍스처 합성기법을 시뮬레이션에 활용하려는 노력들이 있었지만, 이류 과정으로 인해 텍스처가 블러링 되거나 색상이 혼합되어 디테일이 감소되는 문제가 발생했다. 그러나 이 연구에서는 곧 이 고질적인 문제를 해결하여 물 표면의 디테일을 개선시켰다.

III. The Proposed Scheme

제안하는 방법의 알고리즘 개요는 스프라이트 이미지 시퀀스를 입력으로 아래와 같은 순서로 수행된다(Fig. 2 참조).

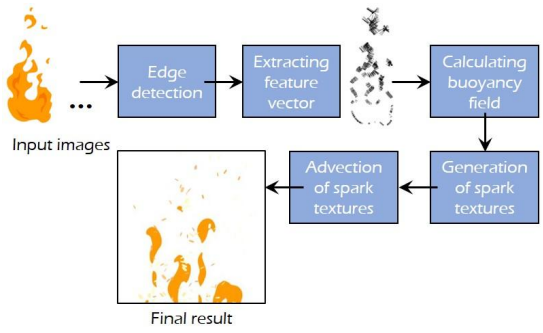


Fig. 2. Algorithm overview

1) 입력 이미지들로부터 화염의 움직임을 계산하기 위해 특징벡터를 계산한다. 특징벡터는 불의 표면에서 화염의 움직임을 추출해내기 위한 과정이다.
2) 특징벡터를 기반으로 부력장을 계산한다. 이 과정은 현 시점에 대한 부력뿐만 아니라 이류 과정을 포함시켜 시간에 따라 움직이는 부력의 디테일한 움직임을 추출한다.
3) 마지막으로는 부력 벡터장에 불꽃 텍스처를 생성시키고 부력의 흐름에 따라 불꽃 텍스처를 이류시킨다.

1. Extraction of feature vectors

입력 이미지로부터 특징벡터를 추출하기 위해 이미지 시퀀스 데이터로부터 에지를 추출하고 에지의 미분을 이용하여 특징벡터를 계산한다(Fig. 3 참조). 이 벡터는 에지 기반으로 모델링되었으며 다양한 에지 검출 기법 중 DoG 필터를 사용한다.

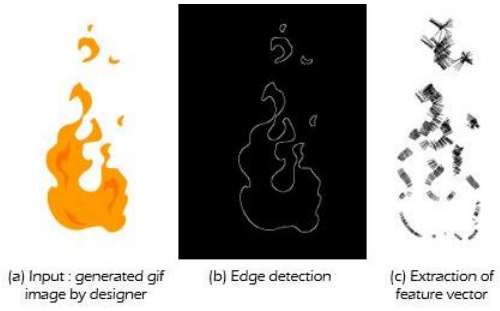


Fig. 3. Calculating feature vectors.

이산화에 따른 오류 누적을 피하기 위해 2차 미분을 이용하는 DoG(Difference of Gaussian) 필터는 아래와 같은 수식으로 계산된다(Equation 1 참조).

$$DoG(x,y) = \frac{e^{\frac{-(x^2+y^2)}{2\sigma_1^2}}}{2\pi\sigma_1^2} - \frac{e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}}{2\pi\sigma_2^2} \quad (1)$$

여기서 σ_1과 σ_2는 DoG 함수의 커널을 결정하는 값으로 이 값을 변화시켜 검출할 에지의 넓이를 조절할 수 있으며, 본 논문에서는 σ_1과 σ_2를 2.5와 2.15로 각각 설정하였다. DoG 필터는 원본 이미지에서 가우시안 필터링된 이미지를 배면 LoG(Laplacian of Gaussian)[18]로 얻은 에지 이미지와 굉장히 비슷한 이미지를 얻을 수 있다는 것이 핵심이다. 여기서 가우시안 필터링된 이미지는 다른 σ값들로 계산된 이미지이다. 이 때 두 개의 σ값들은 1.6:1의 비율을 가져야 한다.

$$\nabla f\kappa = \underbrace{\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)}_{\nabla f}\underbrace{\left(\frac{w^y}{w^{height}}\right)}_{\kappa} \quad (2)$$

에지가 검출된 이미지에서 인위적인 부력 방향을 계산하기 위해 1차 미분을 활용한다(Equation 2 참조).

여기서 ∇f는 1차 미분 수식이며, κ는 부력을 조절하는 변수이다. 일반적으로 화염에서 난류는 연소 후에 강하게 나타나기 때문에 Y축을 기준으로 난류의 조절 변수 κ를 조절한다. 위 수식에서 w^y는 픽셀의 y좌표이고, w^{height}는 입력 이미지의 세로길이이다. 결과적으로 κ값이 아래쪽은 0에 가깝고, 위로 올라갈수록 1에 가까워진다. Fig. 4은 다양한 κ에 따라 생성된 특징벡터이다. 이 그림에서처럼 연료가 생성되는 아래 부분보다 연소 후의 윗부분이 더욱더 큰 특징의 벡터를 갖는 것을 볼 수 있다. 만약 κ가 없다면 동일한 부력 크기로 인해 불똥이 아래 방향으로 움직여 실제 화염이 흐르는 방향과 패턴이 달라지는 문제가 발생하기 때문에 본 연구에서는 κ값을 조절함으로 이 문제를 완화한다.

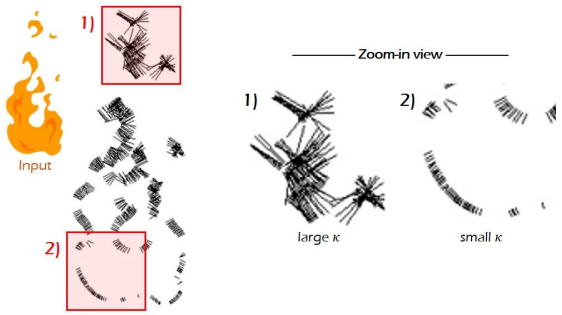


Fig. 4. Controlling feature vectors by κ.

2. Calculation of artificial buoyancy field

본 연구에서는 불꽃의 자연스러움을 표현하기 위해 특징벡터 Fv와 유체 방정식을 통합하여 부력을 새롭게 모델링한다. 우선 유체의 운동을 나타내는 운동량 방정식의 미분 형태를 계산하기 위해 질량 dm을 갖는 유체 입자에 뉴턴의 제2법칙을 아래와 같이 적용한다(Equation 3 참조).

$$F = \frac{P}{dt} \quad (3)$$

여기서 선형 운동량 P는 아래 수식에 의해 계산된다 (Equation 4 참조).

$$P = \int_{mass} udm \qquad (4)$$

연속적인 시스템에서 질량 dm에 대해 뉴턴의 제2법칙은 아래와 같이 쓸 수 있다(Equation 5 참조).

$$dF = dm\frac{du}{dt} \qquad (5)$$

속도장에서 움직이는 질량 dm의 가속도 수식을 얻은 다음 뉴턴의 제2법칙을 벡터 방정식으로 아래와 같이 다시 쓸 수 있다(Equation 6 참조).

$$dm\frac{Du}{Dt} = dm\left(\frac{\partial u}{\partial t} + u \bullet u\right) = dF \qquad (6)$$

여기서 질량 dm과 부피 $dV = dxdydz$의 미분요소에 작용하는 힘을 고려하여 dF를 계산한다. 일반적으로 유체 역학에서 dF는 아래 수식과 같이 표면력(Surface force)과 체력(Body force)으로 구성된다(Equation 7 참조).

$$dF = dF_{surface} + dF_{body} \qquad (7)$$

표면에 작용하는 표면력은 아래와 같이 점성(Viscous force)과 압력의 차(Pressure difference)로써 다시 쓸 수 있다 (Equation 8 참조).

$$dF_{surface} = \{\mu\nabla \bullet (\nabla u) - \nabla p\}dV \qquad (8)$$

체력은 유한한 전체 요소에 가해지는 힘이며, 일반적인 Navier-Stokes 방정식에서 중력은 유체에 작용하는 유일한 체력이다. 중력 대신, 앞에서 설명한 부력을 이용하여 체력을 아래와 같이 다시 쓸 수 있다 (Equation 9 참조).

$$dF_{body} = U_{potential}dV \qquad (9)$$

마지막으로 체력은 중력과 함께 아래와 같이 계산한다.

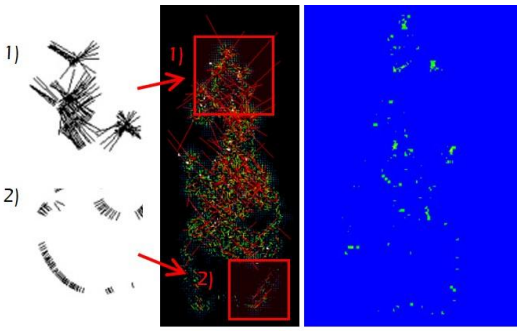
$$U_{potential} = U_{gravity} + U_{buoyancy} \qquad (10)$$

여기서 $U_{buoyancy}$는 사용자가 정의한 인위적인 부력이며, 이 수식을 이용하여 Navier-Stokes 방정식을 아래와 같이 다시 쓸 수 있다(Equation 11 참조).

$$\frac{\partial(\rho u)}{\partial t} + \nabla \bullet (\rho uu) = \mu\nabla \bullet (\nabla u) - \nabla p + \rho g + U_{buoyancy} \qquad (11)$$

Equation 11과 일반적인 Navier-Stokes 방정식의 유일한 차이점은 사용자가 정의한 부력인 $U_{buoyancy}$의 존재이다. $U_{buoyancy}$는 입력 이미지로 시퀀스로부터 계산되기 때문에 계산양이 크게 증가되지 않고, 유체 방정식과 쉽게 결합된다.

Fig. 5a에서 1번 특징벡터는 2번에 비해 상대적으로 큰 크기의 특징벡터를 갖으며, 결과적으로 특징벡터의 크기를 조절하여 부력을 안정적으로 계산하였다. Fig. 5b는 불꽃 텍스처가 생성될 위치이며 특징벡터의 크기가 사용자가 정의한 임계값보다 큰 곳을 생성 위치로 계산하였다. 3.1장에서 특징벡터를 추출하고, 3.2장에서는 앞에서 계산한 특징벡터를 이용하여 2차원 유체 방정식을 계산한다. 이때 특징벡터는 유체 방정식의 외력에 추가하여 풀었으며(수식 11 참조), 특징벡터에 따라 부력장을 계산하고(Fig. 5a의 오른쪽 결과), 이 부력장을 이용하여 불꽃 텍스처를 움직인다. 속도의 크기가 다음 조건을 만족하는 부분에 대해서 불꽃 텍스처를 생성한다: $\| u \| > \lambda$, 여기서 λ는 불꽃 텍스처의 양을 조절하는 임계값이다(Fig. 5b 참조). 다음 장에서 불꽃 텍스처를 이류하는 방법에 대해 자세히 설명한다.



(a) Buoyancy field by feature vectors (b) Candidate positions of spark textures (green)

Fig. 5. Calculating artificial buoyancy field and candidate position of spark texture.

3. Generation and advection of spark textures

불꽃 텍스처가 생성될 위치에 텍스처를 추가하고, 부력 벡터장에 의해 매 프레임에 불꽃 텍스처를 이류시킨다. 이 과정에서 Stam의 Semi-Lagrangian 기법을 사용하였다[3]. 본 연구에서 모델링한 부력은 입력 애니메이션 데이터로부터 특징벡터를 계산하고 이것을 이용하여 유동을 분석했기 때문에 사실적인 움직임으로 불꽃 텍스처를 이류시킬 수 있다. Fig. 6는 불꽃 텍스처를 추가한 결과이며 강한 난류 부분에 불꽃 텍스처가 생성되고 유동에 따라 자연스럽게 불꽃 텍스처가 움직였다.

Fig. 6. Improved visual effects by spark textures.

IV. Results and Conclusions

제안한 방법의 우수성을 판단하기 위해 디자이너가 제작한 다양한 스프라이트 애니메이션을 입력해서 실험했으며 안정적으로 불꽃 효과를 만들어냈다. 이와 같은 결과를 만들기 위해 사용한 입력 애니메이션 데이터는 불꽃이 없고 화염만 있는 애니메이션 데이터(*.gif)이며, Fig. 7~10는 제안한 기법으로 추가된 불꽃 효과들이다.

Fig. 7. Fire sprite animation1.

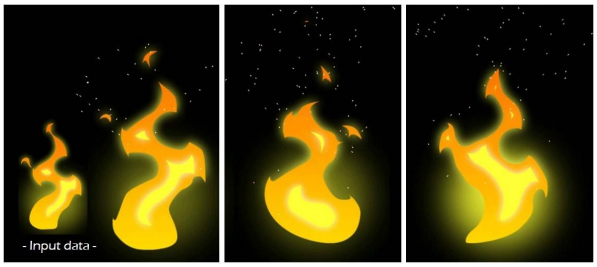


Fig. 8. Fire sprite animation2.

Fig. 7과 8은 카툰 형식의 애니메이션 장면에 제안하는 방법을 적용해본 결과이다. 화염의 방향에 맞게 불꽃이 생성되고 움직이는 결과를 얻었다. 뿐만 아니라 일반적인 화염의 형상과는 다른 특정 형태의 화염 형상에도 우리의 방법은 불꽃을 상세하게 표현하였다. Fig. 9에서 보듯이 'A'자 형태에 맞게 불꽃이 생성되고 움직이는 결과가 만들어진 것을 볼 수 있다.

Fig. 9. Fire sprite animation3.

Fig. 10은 파라미터인 λ값을 변경으로 좀 더 많은 양의 불꽃을 생성한 결과이다. λ는 부력장의 크기로부터 불꽃 텍스처를 생성하는 임계값이 이 값이 작으면 더 많은 불꽃 텍스처가 생성되며, Fig. 10과 비교 했을 때 상대적으로 더 많은 양의 불꽃 효과가 생성된 것을 쉽게 볼 수 있다.

Fig. 10. Different results by changing threshold λ.

일반적으로 이러한 불꽃 효과는 화염보다 표현하기 더 어렵다. 그 이유는 불꽃 입자들의 개수가 많고 화염의 움직임을 고려해서 복잡한 불꽃의 움직임을 표현해야 되기 때문이다. 우리는 이런 불꽃 표현을 효율적으로 표현해 낼 수 있는 새로운 프레임워크를 제안하였고, Table 1은 본 연구에서 제시한 결과에 대한 환경을 요약한 표이며, Table 2는 결과에서 사용한 시뮬레이션 파라미터들이다.

최근 컴퓨터 그래픽스 분야에서 불똥 시뮬레이션을 표현하기 위한 수치 기법이 제안되었는데[12], 그 기법과 비교해봤을 때 본 연구에서 제안한 기법의 성능이 크게 개선된 것임을 알 수 있다. 우리의 기법은 격자 기반 수치 시뮬레이션을 기반으로 한 것이 아닌, 2차원 이미지 합성을 기반으로 수행한 훨씬 빠른 결과를 나타내는 기법이기 때문이다. Kim 등[12]은 불똥 효과를 표현하는데 프레임 당 평균 100초가 수행되었고(Fig. 11 참조), 장면마다 계산 시간의 차이는 있지만 비교해봤을 때 모든 결과에서 시각적으로 비슷한 품질의 불똥을 만들어 내는데 성능이 크게 향상되었다.

Fig. 11. Fire-flake animations with previous method[12].

격자 기반 시뮬레이션 데이터라면 격자 해상도가 높은 고해상도 시뮬레이션에서 수행되어야 하지만, 본 연구는 이미지 프로세싱과 합성을 기반으로 수행되기 때문에 이미지 상에서 화염의 움직임만 보인다면 바로 적용이 가능하다. Fig. 8에서 보듯이 저품질 화염 애니메이션에서도 불꽃 효과를 빠르게 생성할 수 있기에 화재 시뮬레이션뿐만 아니라 실시간 상호작용을 요구하는 애플리케이션에서도 활용이 가능할거라 예상한다.

본 연구 방법의 한계점으로는 화염 스프라이트 애니메이션을 잘 표현해내지만, 2차원 입력 이미지는 정확한 깊이 정보가 없기 때문에 불꽃과 오브젝트의 인터랙션이 다소 어색하다. 향후, 깊이 정보를 이용하여 화염과 불꽃의 시각적인 디테일을 한 층 더 개선시킬 방법에 대한 연구를 진행 할 계획이다.

Table 1. Sizes of our example scenes.

Figure	Size of input image	Grid resolution	Number of spark textures
7	656x656	164x164	avg. 120
8	512x512	128x128	avg. 210
9	500x500	125x125	avg. 260
10	500x500	125x125	avg. 1,120

Table 2. Simulation parameters.

Figure	σ_1	σ_2	λ	Time-step
7	2.5	2.15	0.6	0.02
8	2.5	2.15	0.6	0.02
9	2.5	2.15	0.6	0.02
10	2.5	2.15	0.3	0.02

REFERENCES

[1] Fernando, Randima, “GPU gems: programming techniques, tips and tricks for real-time graphics”, Pearson Higher Education, 2004.

[2] Fuller, Alfred R and Krishnan, Hari and Mahrous, Karim and Hamann, Bernd and Joy, Kenneth I, “Real-time procedural volumetric fire”, Proceedings of the 2007 symposium on Interactive 3D graphics and games, pp.175-180, 2007.

[3] Stam, Jos, “Stable fluids”, Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp.121-128, 1999.

[4] Foster, Nick and Fedkiw, Ronald, “Practical animation of liquids”, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp.23-30, 2001.

[5] Enright, Douglas and Losasso, Frank and Fedkiw, Ronald, “A fast and accurate semi-Lagrangian particle level set method”, Computers & structures, pp.479-490, 2005.

[6] Muller, Matthias and Charypar, David and Gross, Markus, “Particle-based fluid simulation for interactive applications”, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.154-159, 2003.

[7] Becker, Markus and Teschner, Matthias, “Weakly compressible SPH for free surface flows”, Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.209-217, 2007.

[8] Solenthaler, Barbara and Pajarola, Renato, “Predictive-corrective incompressible SPH”, ACM Transactions on Graphics, pp.40-46, 2009.

[9] Nguyen, Duc Quang and Fedkiw, Ronald and Jensen, Henrik Wann, “Physically based modeling and animation of fire”, ACM Transactions on Graphics, pp.721-728, 2002.

[10] Hong, Jeong-Mo and Shinar, Tamar and Fedkiw, Ronald, “Wrinkled flames and cellular patterns”, ACM Transactions on Graphics, pp.47-52, 2007.

[11] Horvath, Christopher and Geiger, Willi, “Directable, high-resolution simulation of fire on the GPU”, ACM Transactions on Graphics, pp.41-49, 2009.

[12] Kim, TaeHyeong and Hong, Eunki and Im, Jaeho and Yang, Dohyeon and Kim, Youngbin and Kim, Chang-Hun, “Visual simulation of fire-flakes synchronized with flame”, The Visual Computer, pp.1209-1038, 2017.

[13] Gagnon, Jonathan and Dagenais, Francois and Paquette, Eric, “Dynamic lapped texture for fluid simulations”, The Visual Computer, pp.901-909, 2016.

[14] Jae-Gwang Lim, Bong-Jun Kim, Jeong-Mo Hong, "An Adaptive FLIP-Levelset Hybrid Method for Efficient Fluid Simulation", Journal of The Korea Computer Graphics Society pp.1~11, 2013.

[15] Sun-Tae Kim, Jeong-Mo Hong, “Visual simulation of turbulent fluids using MLS interpolation profiles", The Visual Computer, 2012.

[16] Sun-Tae Kim, Jeong-Mo Hong, “Visual simulation of turbulent fluids using MLS interpolation profiles", The Visual Computer, 2012.

[17] Sun-Tae Kim, Jeong-Hyun Lee, Dae-Yeong Kim Yeong-Su Park, Seong-Ho Jang, Jeong-Mo Hong, "A Case Study of Fluid Simulation in the Film 'Sector 7'", Journal of The Korea Computer Graphics Society, volume 18, number. 3, 2012.

[18] Marr, David, and Ellen Hildreth. "Theory of edge detection." Proc. R. Soc. Lond. B, pp.187-217, 1980.

Authors

Jong-Hyun Kim received the B.A. degree in the department of digital contents at Sejong University in 2008. He received M.S. and Ph.D. degrees in the department of computer science and engineering at Korea University, in 2010 and 2016. Prof. Kim is an assistant professor in the department of software application in Kangnam University. His current research interests include fluid animation and virtual reality.