한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 25, No. 3, P. 105~113

CNN을 이용한 Quad Tree 기반 2D Smoke Super-resolution

홍병선[O] 박지혁 최명진 김창헌*

고려대학교

bshong2850@korea.ac.kr[O] wlgur01014@naver.com blackship@korea.ac.kr chkim@korea.ac.kr*

Quad Tree Based 2D Smoke Super-resolution with CNN

Byeongsun Hong[O] Jihyeok Park Myungjin Choi Changhun Kim*

Korea University

요 약

물리 기반 유체 시뮬레이션은 고해상도 연산을 위해 많은 시간이 필요하다. 이 문제를 해결하기 위해 저해상도 유체 시뮬레이션의 한계를 딥 러닝으로 보완하는 연구들이 있으며, 그중에서는 저해상도의 시뮬레이션 데이터를 고해상도로 변환해주는 Super-resolution 분야가 있다. 하지만 기존 기법들은 전체 데이터 공간에서 밀도 데이터가 없는 부분까지 연산하므로 전체 시뮬레이션 속도 면에서 효율성이 떨어지며, 입력 해상도가 큰 경우에는 GPU 메모리가 부족해 연산할 수 없는 경우가 발생할 수 있다. 본 연구에서는 공간 분할 법 중 하나인 쿼드 트리를 활용하여 시뮬레이션 공간을 분할 및 분류하여 Super-resolution 하는 기법을 제안한다. 본 기법은 필요 공간만 Super-resolution 하므로 전체 시뮬레이션 가속화가 가능하고, 입력 데이터를 분할 연산하므로 GPU 메모리 문제를 해결할 수 있게 된다.

Abstract

Physically-based fluid simulation takes a lot of time for high resolution. To solve this problem, there are studies that make up the limitation of low resolution fluid simulation by using deep running. Among them, Super-resolution, which converts low-resolution simulation data to high resolution is under way. However, traditional techniques require to the entire space where there are no density data, so there are problems that are inefficient in terms of the full simulation speed and that cannot be computed with the lack of GPU memory as input resolution increases. In this paper, we propose a new method that divides and classifies 2D smoke simulation data into the space using the quad tree, one of the spatial partitioning methods, and performs Super-resolution only required space. This technique accelerates the simulation speed by computing only necessary space. It also processes the divided input data, which can solve GPU memory problems.

키워드: 쿼드 트리, 슈퍼 레졸루션, 가속화, 연기 시뮬레이션

Keywords: Quad Tree, Super-resolution, Acceleration, Smoke Simulation

1. 서론

물리 기반 유체 시뮬레이션에서 해상도에 비례하여 연산량이 많아지는 문제를 해결하는 것은 중요한 이슈 중 하나이다. 시뮬레이션에서 해상도가 높아지면 연산량이 급격히 증가하여 시뮬레이션 속도가 매우 느려지게 되는데, 최근에는 딥 러닝을 활용하여 이러한 문제를 해결하려는 연구가 진행

*corresponding author: Changhun Kim/Korea University(chkim@korea.ac.kr)

Received : 2019.06.11./ Review completed : 1st 2019.06.18. / Accepted : 2019.06.23.
DOI : 10.15701/kcgs.2019.25.3.105
ISSN : 1975-7883(Print)/2383-529X(Online)

되고 있다. 딥 러닝을 활용하는 연구는 활용 부분에 따라 크게 두 가지로 나뉘게 된다. 첫 번째로 시뮬레이션 자체를 딥 러닝으로 풀어내려는 분야가 있다. 두 번째로는 시뮬레이션 데이터를 입력 데이터로 하여, 스타일을 변경하거나 해상도를 높여 디테일을 표현하는 등의 연구 분야가 있다. 하지만 이런 연구들은 고해상도로 갈수록 데이터의 크기가 커지게 되어, GPU 메모리 부족으로 초고해상도의 데이터에는 활용하기 어렵다. 또한, 전체 공간에 대하여 스타일을 변경하거나 해상도를 높이므로, 불필요한 부분까지 연산하게 되어 전체 시뮬레이션이 비효율적일 수밖에 없다.

따라서 우리는 이런 문제들을 해결하기 위해 공간 분할법 중 하나인 쿼드 트리를 활용하여 Super-resolution을 효율적으로 할 수 있는 기법을 제안한다. 본 논문에서 제안하는 기법을 이전의 기법들과 비교했을 때, 결과 차이는 없으면서 Super-resolution이 필요 없는 부분의 연산이 줄어들어 시뮬레이션 속도가 향상되었다. 또한, 쿼드 트리를 활용하기 때문에 어떠한 해상도의 입력 데이터가 들어와도 공간을 분할하여 연산을 진행하므로 초고해상도의 데이터에 대한 GPU 메모리 부족 문제도 해결하였다. 따라서 초고해상도의 데이터도 효율적인 Super-resolution이 가능하다. 앞으로 2장에서는 딥 러닝 기반 유체 시뮬레이션과 공간 분할법, Super-resolution에 관한 관련 연구를 소개하고, 3장에서는 제안하는 알고리즘의 전체적인 과정을 설명한다. 4장에서는 본 논문의 기법에 대한 실험 및 분석 결과를 소개한 뒤, 결론 및 향후 연구 방향을 5장에서 제시한다.

2. 관련 연구

유체 시뮬레이션과 딥 러닝을 결합하는 시도는 최근의 딥 러닝의 발전과 함께 많이 이루어지고 있다. 서론에서 이야기했듯이 먼저 솔버 부분을 딥 러닝으로 해결하려는 연구가 있었는데, Tompson et al.[1]과 Xiao et al.[2]는 CNN을 사용하여 Naiver-Stokes 방정식의 연산을 해결하는 기법을 제안하였다. 다른 한편으로는 시뮬레이션 자체를 해결하는 것이 아닌 시뮬레이션 된 결과를 Super-resolution 하는 연구도 진행되었는데, Chu et al.[3]과 Xie et al.[4]은 2D 및 3D 연기 시뮬레이션 데이터에 대해서 CNN과 GAN을 사용한 Super-resolution 기법을 제안하였다.

하지만 이런 연구들은 전체 데이터를 한 번에 연산하기 때문에 GPU 메모리가 부족해지는 문제나 시뮬레이션이 비효율적으로 연산 되는 문제가 있을 수 있다. 따라서 본 논문에서는 이런 문제들을 해결하고자 쿼드 트리를 활용하여 2D 연기 밀도 데이터를 효율적으로 분할 하고 필요한 부분만 Super-resolution 하는 기법을 제시하여 GPU 메모리 부족 문제를 해결하고, 전체 시뮬레이션을 가속화 한다.

딥 러닝을 활용한 Super-resolution은 이미지 영역에서 많은 연구가 이루어지고 있는 분야이다. Dong et al.[5]은 처음으로 Single Image Super-resolution을 딥 러닝 기반으로 해결하는 기법을 제안하였고, 적대적 신경망(GAN)의 발전과 함께 Christian et al.[6]은 적대적 신경망을 활용한 SRGAN을 제안하였다. 최근에는 Chu et al.[7]이 연속적인 이미지를 Super-resolution 하는 연구를 진행하였으며, CNN과 GAN을 활용한 이미지 Super-resolution 연구도 계속 진행되고 있다[8].

공간 분할법 중에는 2D 공간과 3D 공간에서 효율적으로 사용할 수 있는 쿼드 트리[9]와 옥 트리[10]가 있다. 공간 분할법을 사용하는 것이 효율적인 이유는 원하지 않는 공간은 집중하지 않고, 원하는 공간에 집중하여 연산할 수 있기 때문이다. Zhou et al.[11]은 GPU 기반 옥 트리를 활용하여 3D 모델의 표면을 재구축하는 알고리즘을 제시하였다. 또한, 최근에는 공간 분할법과 딥 러닝을 함께 활용하는 연구가 진행되고 있다. Gernot et al.[12]와 Wang et al.[13]이 옥 트리와 CNN을 활용하여 3D 모델을 Classification과 Retrieval 및 Segmentation 하는 기법을 제안하였고, Maxim et al.[14]은 3D 모델을 생성하는 등의 공간 분할법을 통해 메모리 문제와 속도 문제를 해결하는 연구를 진행하였다. 하지만 이런 연구들은 유체 시뮬레이션보다는 모델링과 렌더링에 초점을 두고 있다.

3. Method

본 논문에서 제안하는 알고리즘은 크게 3가지 과정으로 구성된다. 첫 번째는 쿼드 트리를 활용하여 Super-resolution이 필요한 영역을 분류하는 전처리 과정이다. 다음으로 두 번째는 저해상도의 데이터와 고해상도의 데이터를 학습 데이터로 하여 CNN 기반의 Super-resolution 모델을 학습한 뒤, 전처리 과정을 거친 데이터를 학습된 모델에 넣어 고해상도의 데이터로 변환하는 Super-resolution 과정이다. 마지막으로 변환된 여러 고해상도 데이터를 하나의 데이터로 만들어주는 후처리 과정이 있다.

3.1 쿼드 트리를 활용한 전처리

공간 분할법 중 쿼드 트리는 2차원 데이터를 분할 하기에 적합한 알고리즘 중 하나이다. 본 논문에서 제안하는 쿼드 트리를 활용한 전처리 과정은 먼저 입력된 밀도 데이터를 임의의 패치 크기에 맞게 분할 하고(Figure 1a), GPU를 활용한 병렬 연산을 통해 각 패치 데이터의 최댓값으로 구성된 행렬을 생성한다. 생성된 최댓값 행렬을 가지고 분할된 패치의 Super-resolution 필요 유무를 상태 값으로 결정하게 되며(Figure 1b), 각각의 패치가 최하층 노드가 되어 상태 값과 노드에 관한 정보를 가지고 Bottom-Up 방식으로 트리 구조를 생성하게 된다(Figure 1c). 마지막으로, 생성된 쿼드 트리를 통해 Super-resolution이 필요한 공간의 데이터를 추출하게 되는데, 가시화하면 Figure 1d와 같다.

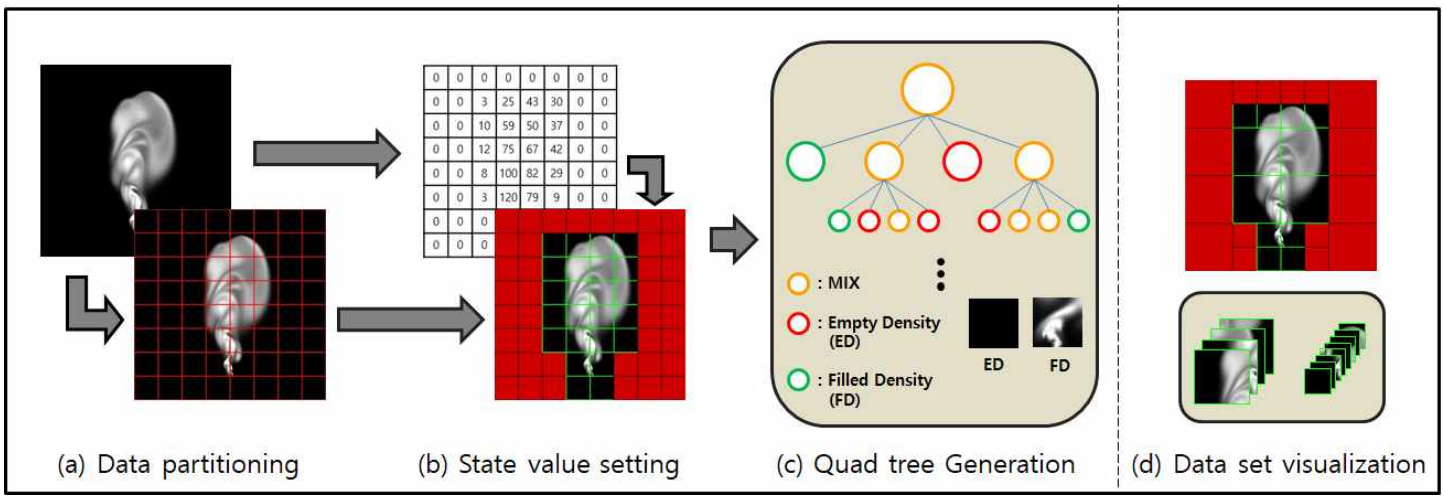


Figure 1 Pre-process using space partitioning method

3.1.1 패치의 상태 값

패치의 상태 값은 쿼드 트리 생성에 중요한 역할을 하게 되는데, 상태 값은 패치별 Super-resolution 필요 유무를 의미하고 다음과 같이 정의된다.

먼저 시뮬레이션 공간을 미리 정의한 크기의 패치로 분할하고, 각 패치에서의 최댓값이 임의로 정한 임곗값보다 크면 해당 패치는 Super-resolution이 필요한 공간이라고 판단하여 상태 값을 Filled Density 즉 “FD”로 정의하였다. 반대로, 임곗값보다 작은 경우에는 Super-resolution이 필요하지 않다고 판단하여 상태를 Empty Density 즉 “ED”로 정의하였다. 따라서 상태 값이 FD라면 Super-resolution을 수행하고, ED라면 Super-resolution을 수행하지 않는다. 본 논문에서는 임곗값을 0.01로 두었다.

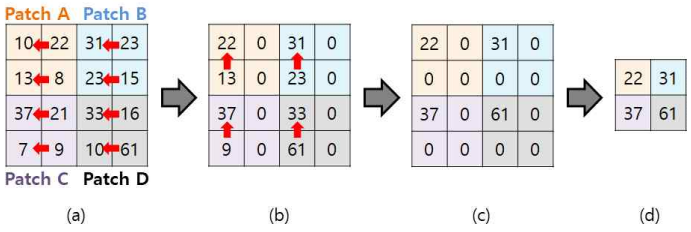


Figure 2 GPU based maximum value matrix calculation

각 패치의 최댓값을 구하는 기법은 다음과 같다. 예를 들어 Figure 2와 같은 4x4의 데이터가 있을 때, 패치 크기를 2x2로 설정하면 전체 시뮬레이션 공간은 크기가 2x2인 4개의 패치로 분할된다(Figure 2a). 이때 전체 데이터를 GPU 기반 병렬 연산을 통해 Reduction을 하면서 최댓값을 찾아가다 보면(Figure 2b, c), 결과로 나온 2x2의 행렬이 각 패치의 최댓값 모음이 되는 것을 알 수 있다. 이렇게 나온 최댓값 행렬을 가지고 임곗값과 비교하여 각 패치의 상태 값을 결정한다. 이와 다르게 최댓값을 찾을 때, 패치의 모든 데이터를 차례로 순회하며 찾는 CPU 기반의 기법도 있다. 본 논문에서 제시하는 GPU 기반 기법과 CPU 기법 간의 속도 차이를 알아보기 위해 4장에서 비교 실험을 진행하였다.

3.1.2 쿼드 트리 생성

쿼드 트리는 전체 시뮬레이션 공간을 미리 정의한 패치 크기로 분할한 각각의 패치 데이터와 앞서 계산한 패치에 해당하는 상태 값을 가지고 최하단 노드를 만들고(Figure 4a), 그 노드들을 기반으로 쿼드 트리를 Bottom-Up 방식으로 구성한다.

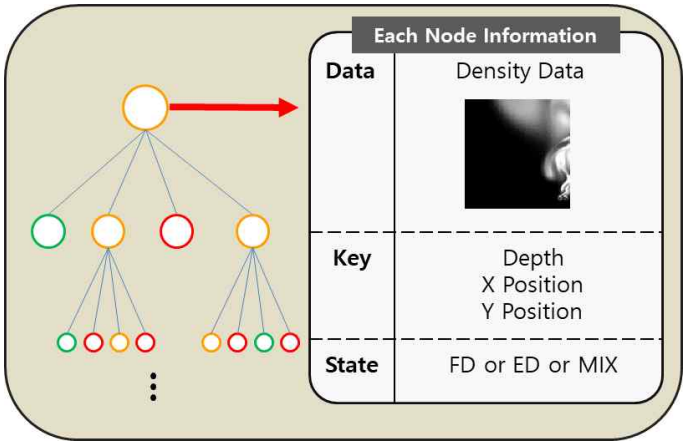


Figure 3 Each node information

쿼드 트리의 각 노드는 데이터, 키, 상태 값을 갖는다(Figure 3). 먼저 데이터값은 그 노드에 해당하는 밀도 데이터이다. 다음으로 키값은 (깊이, x_좌표, y_좌표)로 구성되어 있다. 깊이는 쿼드 트리를 만들 때, 최상위 노드의 깊이를 0으로 설정하고 최상위 노드의 자식 노드부터 깊이가 1씩 증가하는 구조이다(Figure 4c). 전체 데이터 크기와 깊이를 알게 되면 본 논문에서 제안하는 쿼드 트리 특성상 깊이가 1 깊어질수록 크기가 0.5배 되므로 현재 노드의 데이터가 어떤 크기를 가졌는지 알 수 있게 된다. x_좌표와 y_좌표는 각각의 노드가 가진 데이터가 전체 데이터에서 위치한 좌표

를 저장하는데, 깊이와 좌표값은 나중에 Super-resolution 과정이 완료된 뒤 후처리 과정에서 필요하게 된다. 3.1.1에서 정의한 상태 값은 쿼드 트리 구성 중에 FD, ED 이외에도 MIX 값을 가질 수 있는데, 이는 이 항목의 후반부에 자세히 설명하도록 한다.

Figure 4a를 보면 각 패치의 밀도 데이터를 최하단 노드의 데이터로 설정해 주었다. 최하단 노드가 가진 키값의 깊이는 수식 (1)처럼 구할 수 있다. 여기서 d 는 현재 노드의 깊이가 되고, D_{width} 는 전체 입력 데이터의 너비 값, N_{width} 는 현재 노드 데이터의 너비 값이 된다.

$$d = \log_2(D_{width} / N_{width}) \qquad (1)$$

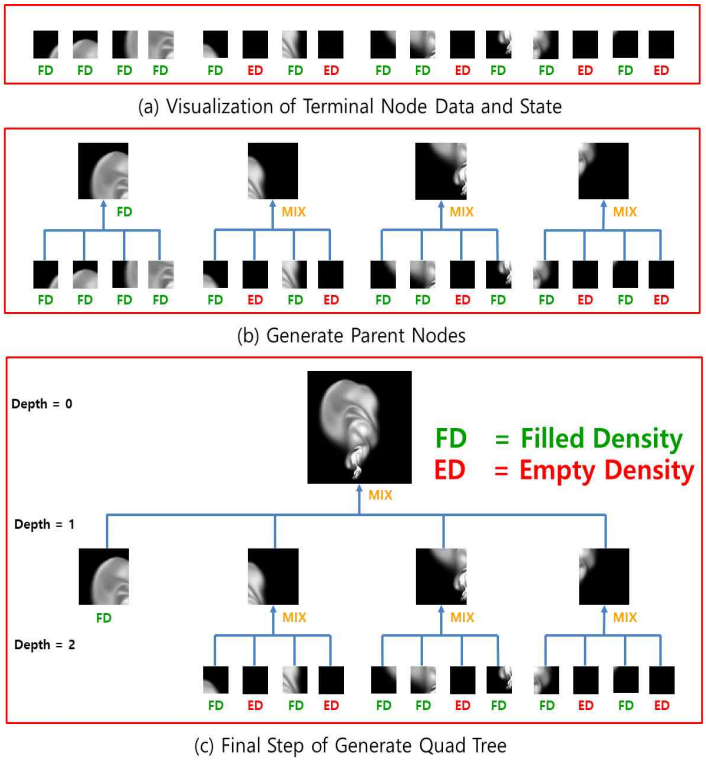
그다음 노드를 4개씩 모아서 부모 노드를 생성하게 되는데 (Figure 4b), 부모 노드의 데이터는 자식 노드 4개의 데이터를 합친 것이 되고, 키값의 깊이는 1 줄어들게 되고, x_좌표와 y_좌표는 자식 노드 중 제일 작은 값을 가지게 된다. 마지막으로 상태 값은 자식 노드들의 상태 값을 확인하여 Table 1과 같이 결정되게 된다.

Table 1 The state of the parent node is decided according to the state of the child node

Child node state value	Parent node state value
All child node is FD	FD
All child node is ED	ED
Three state are mixed	MIX

또한, 만약 자식 노드 4개의 상태 값이 전부 “FD” 또는 “ED”라면 자식 노드를 삭제하게 된다. 왜냐하면 “FD”의 경우 자식 노드의 상태 값이 전부 같으므로 자식 노드 4개를 각각 연산하는 것보다 부모 노드에서 한 번에 연산하는 것이 결과가 같으면서 속도가 더 빠르기 때문이고, “ED”라면 Super-resolution을 할 필요가 없기 때문이다. 이 작업을 깊이가 0이 될 때까지 반복하게 되면 Figure 4c에서 보이는 것처럼 최종 쿼드 트리가 생성된다.

쿼드 트리가 생성되면 깊이가 0인 최상단 노드에서부터 상태 값을 확인하며 모든 “FD”인 노드의 데이터와 키값을 모은다. 상태 값이 “ED”라면 Super-resolution이 필요하지 않은 공간이므로 버린다. 마지막 노드까지 탐색을 마친 뒤 모아둔 데이터를 Super-resolution 입력 데이터로 사용한다.



(c) Final Step of Generate Quad Tree

Figure 4 Generate Quad Tree.

3.2 Super-resolution

Super-resolution은 학습과 사용 두 단계로 나뉜다. 첫 번째로 학습 단계에서는 Figure 5의 왼쪽 위에서 보이는 것과 같이 입력 데이터인 저해상도 데이터와 표적 데이터인 고해상도 데이터로 구성된 학습 데이터가 필요하다. 학습 데이터를 생성하는 과정은 다음과 같다.

먼저 2D 연기 시뮬레이션 데이터를 얻기 위해 512x512 해상도의 격자 기반 연기 시뮬레이션을 수행하여 임의의 5가지 장면을 300프레임씩, 총 1500프레임의 2D 연기 시뮬레이션 데이터를 생성하였다. 본 논문에서는 Stam의 Stable Fluids[15]를 사용하였다. 생성한 데이터를 가지고 패치 크기 16x16으로 쿼드 트리를 만들어 상태 값이 “FD”인 데이터만 모아서 학습용 표적 데이터를 구성하였다. 학습용 입력 데이터는 표적 데이터를 Blur 처리한 뒤 Resize를 하여 Down Sampling 된 저해상도의 데이터로 구성하였다.

다음으로 Super-resolution을 하는 모델을 구성하였다. 모델은 다음 Figure 6과 같이 구성되어 있다. Figure 6에서 LR은 저해상도의 데이터(Low Resolution), SR은 Super-resolution 된 데이터, n은 필터의 개수, s는 Stride 크기를 나타낸다. 하나의 Convolution Layer는 3x3 크기의 Convolution 필터와 Activation Function인 Leaky Relu를 2개씩 사용하였고, 인코더에서의 Pooling은 Max Pooling을 사용하였으며, 디코더에서는 Deconvolution을 통하여 Convolution Layer를 구성하였다. Convolution Layer를

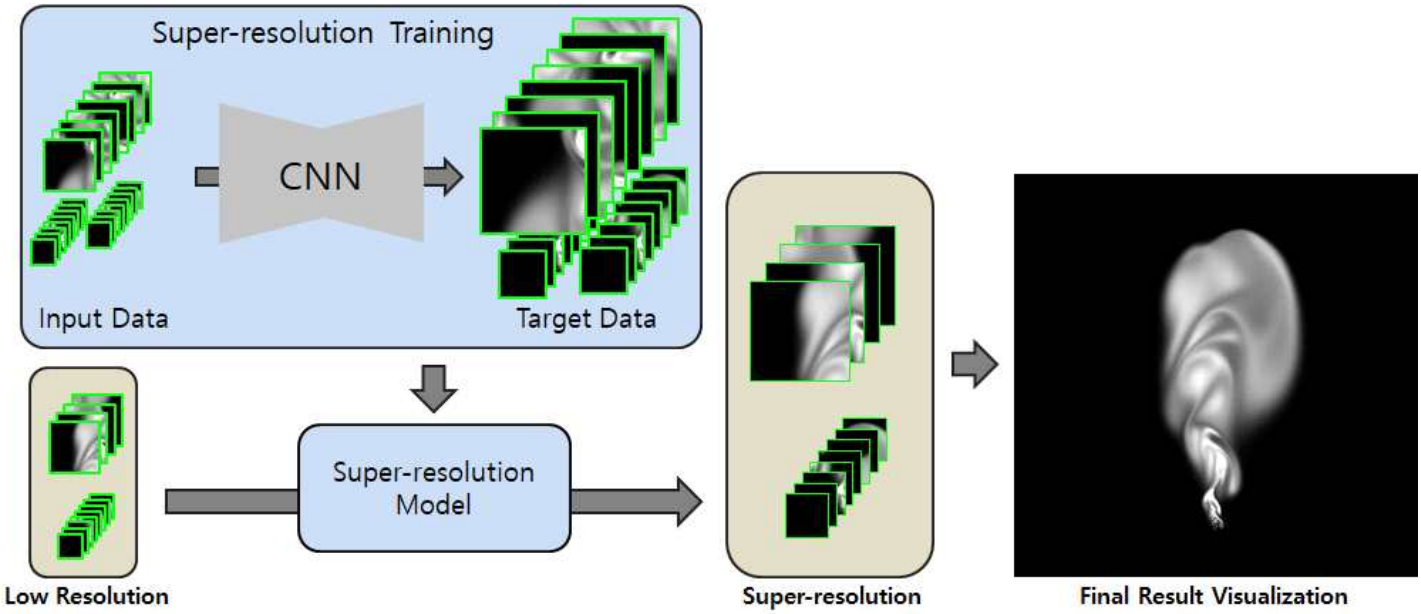


Figure 5 Super-resolution and Post-processing

Figure 6과 같이 쌓아서 전체 모델을 구성하였고, 중간에 크기가 같은 앞의 Activation Map과 뒤의 Activation Map을 He et al.[16]이 제안한 Skip Connection으로 연결하여, 이전의 정보를 잃지 않도록 구성하였다.

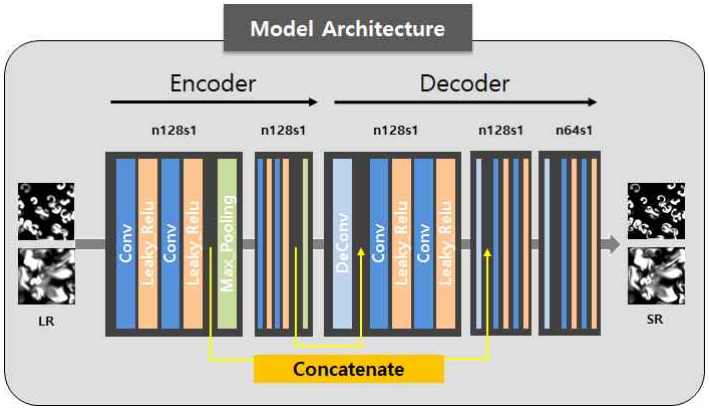


Figure 6 Model Architecture

이 모델로 학습을 진행하면서 사용한 Loss Function은 Mean Squared Error(MSE)를 사용하였고, 식은 (2)와 같다.

$$L = \frac{1}{W \times H} \sum_{i}^{W} \sum_{j}^{H} (SR_{ij} - HR_{ij})^2 \qquad (2)$$

여기서 L 은 Loss Function을 의미하고, W, H는 각각 전처리 된 데이터의 너비와 높이를 의미한다. SR, HR은 Super-resolution 된 데이터와 표적 데이터인, 고해상도의 데이터이다. Optimizer는 Adam을 사용하였고, Learning rate는 0.0001, 배치 크기는 512로 두고 총 50 epoch의 학습을 진행하였다.

학습 과정에서 Figure 7을 보면 입력 데이터에는 없고, 표적 데이터에는 있는 디테일한 부분들이 살아나는 것을 확인할 수 있었다.

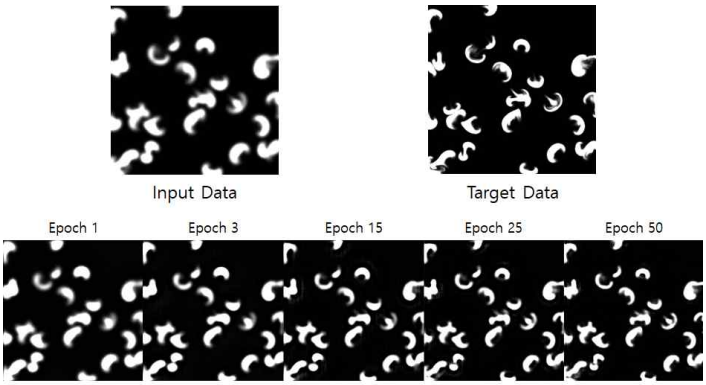


Figure 7 Visualization of several training epoch

사용 단계에서는 학습이 완료된 모델에 쿼드 트리를 이용한 전처리에서 생성된 데이터를 입력으로 넣어주면, Super-resolution이 완료된 고해상도의 데이터를 얻을 수 있다.

3.3 데이터 후처리 과정

마지막으로 앞에서 얻은 고해상도의 데이터들을 후처리 작업으로 하나로 모아주어 최종 데이터를 만들어낸다. 3.1.2에서는 쿼드 트리를 생성할 때 각각의 노드가 데이터와 키값을 저장했었다. 키값은 깊이와 x, y 좌표를 갖고 있으므로 해당 노드의 패치 크기와 위치를 특정할 수 있다. 이러한 특징을 이용하여 여러 크기로 나누어진 패치 데이터를 하나로 모아주는 작업이 가능하다.

먼저 0으로 채워진 표적 데이터 해상도의 공간을 만들고,

Super-resolution 된 고해상도의 데이터들을 키값에 있는 깊이와 x,y 좌표를 가지고 각각의 공간에 맞게 모으면 나머지 공간은 자연스럽게 0으로 채워진 상태 값이 “ED” 였던 데이터들의 공간이 된다. 본 논문에서 제시한 기법은 학습이 적게 되었을 때 후처리 작업을 하게 되면, Figure 8a에서 보이는 것과 같이 패치와 패치 사이의 경계 부분에서 Artifact가 생기는 것을 확인하였다. 하지만 이는 모델이 학습을 계속 진행함에 따라 해결됨을 확인할 수 있었다 (Figure 8b).

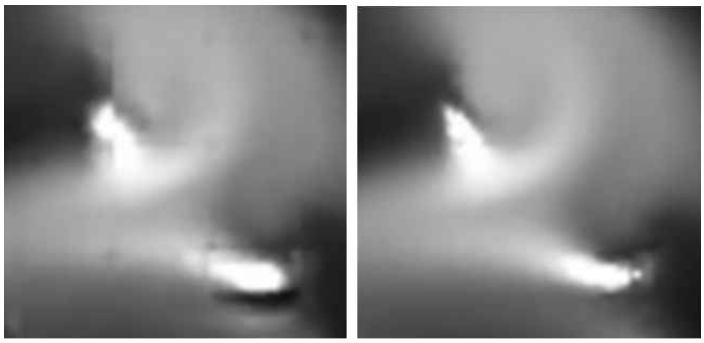


Figure 8 Visualization of boundary problem.

4. 실험 및 분석

본 논문에서 제안하는 기법을 실험 및 분석해보기 위해 여러 가지 다른 기법들과 비교해보았다. 먼저 본 논문에서 제안하는 기법은 다음과 같은 환경에서 구현되었다. Intel Core i7 8700K CPU, 16GB RAM, GTX 1080 8GB 그래픽 카드를 사용하였고 Stam의 Stable Fluids[15]를 2D 연기 시뮬레이션 데이터로 사용하였다. 실험 데이터는 Figure 5에서 가장 우측에 나오는 것과 같은 기본적인 장면의 데이터를 사용하였다.

실험 및 분석은 크게 두 가지로 나뉘는데, 첫 번째로 모델을 통해 나온 고해상도 데이터가 디테일한 부분을 표현하면서 표적 데이터와 유사한지 실험하였다. 두 번째는 속도 비교로 총 4가지 기법을 비교하였는데. 첫 번째로 Whole SR(Super-resolution)은 입력으로 받은 전체 데이터를 통째로 Super-resolution 하는 것이다. 이 기법은 이전 연구들이 사용했던 기법으로써 해상도가 커지면 GPU 메모리 문제가 생겨서 대용량의 데이터에서는 결과를 확인할 수 없다 (Figure 11 bottom). 두 번째는 Patch SR이다. 이 기법은 임의의 패치 크기로 데이터를 분할 한 뒤, 쿼드 트리를 생성하는 것이 아니라 분할된 패치 데이터를 입력 데이터로 Super-resolution을 진행한 것이다. 세 번째와 네 번째 기법은 둘 다 쿼드 트리를 사용하고 각각의 상태 값을 통해 Super-resolution을 진행하는 방식은 같지만, 다른 점은 최하단 노드의 상태 값 결정 방식을 CPU 기반 데이터 순회 방식으로 했는지, 아니면 본 논문에서 제안하는 GPU 기반 병렬 처리 방식으로 했는지의 차이이다. SR with CPU Quad Tree는 CPU 기반 데이터 순회 방식이고, SR with GPU Quad Tree는 GPU 기반 병렬 처리 방식이다.

4.1 Super-resolution

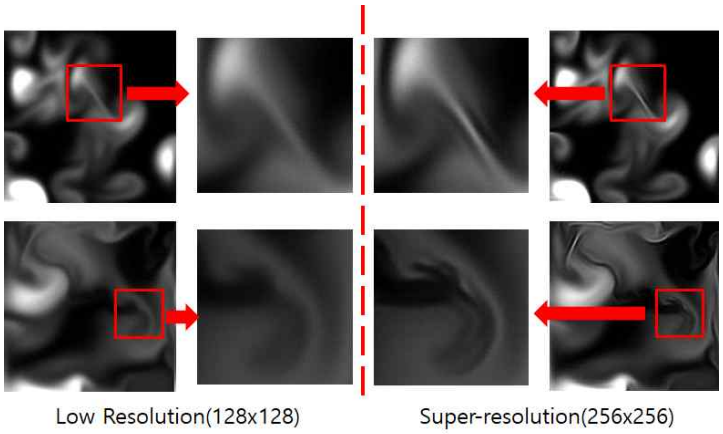


Figure 9 Super-resolution result

먼저 제안하는 모델이 만든 Super-resolution 결과를 비교해보았다. 초고해상도 데이터는 세부적인 결과들이 논문의 Figure로는 표현이 되지 않아서 저해상도의 입력 데이터로 실험을 진행하였다. Figure 9를 보면 저해상도(128x128)의 입력 데이터를 본 논문에서 제안하는 모델에 넣어 줬을 때, Super-resolution 후에 고해상도(256x256)가 되어 나온 결과가 디테일한 부분을 표현하면서 표적 데이터와 유사한 것을 확인하였다.

4.2 속도 비교

4.2.1 CPU, GPU 기반 상태 값 결정 속도 비교

본 논문에서 제안한 GPU 기반 병렬 연산을 통한 상태 값 결정과 CPU 기반 데이터 순회 상태 값 결정에 대한 속도 비교는 Figure 10과 같다. x축은 전체 알고리즘의 각 함수의 이름이고, y축은 각 함수의 소요 시간을 표현하였다. 빨간색은 데이터 로드, 초록색은 쿼드 트리 생성, 파란색은 Super-resolution, 분홍색은 데이터 후처리, 마지막으로 검은색은 데이터 저장에 걸린 시간을 표현하였다. 상태 값 결정은 쿼드 트리 생성함수 안에 포함되어있다. 결과를 보면 입력 해상도로 2048 x 2048 크기의 밀도 데이터를 사용하였는데, 다른 함수들의 소요 시간은 거의 일정하였으나 쿼드 트리 생성함수가 CPU 기반일 경우에는 5.056초, GPU 기반일 경우에는 0.296초로 GPU 기반일 때 약 17배 정도 빨라지는 것을 확인할 수 있었다. 또 전체 과정에 소요된 시간은 쿼드 트리 생성이 CPU 기반일 경우 6.231초, GPU 기반일 경우에는 1.448초로 약 4.3배 정도 빨라지는 것을 확인할 수 있었다.

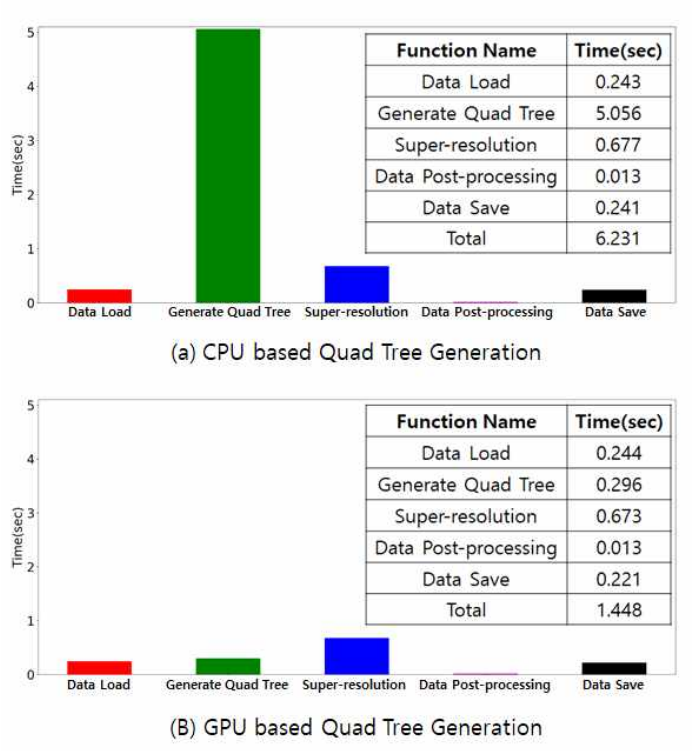


Figure 10 Result of the function running time for input resolution 2048 x 2048

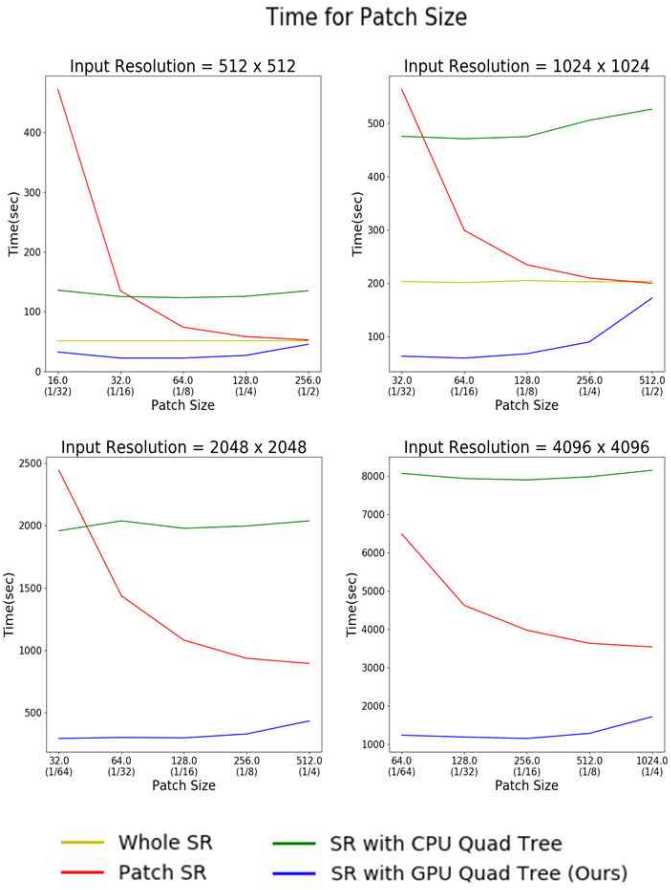


Figure 11 Result of the running time for several input resolutions and patch sizes

4.2.2 패치 크기에 따른 속도 비교

다음은 본 논문에서 제안하는 기법을 활용하여 여러 가지 입력 해상도와 패치 크기를 가지고 2D 연기 시뮬레이션을 Super-resolution 하는 실험을 진행하였다. Figure 11에서 볼 수 있듯, 총 4가지의 입력 해상도에서 5가지 패치 크기를 비교하였다. Figure 11의 결과를 보면 입력 해상도와 무관하게 모두 본 논문이 제안하는 기법이 가장 빠른 것을 알 수 있다. 하단의 두 결과에서 Whole SR이 없는 이유는 입력 해상도가 2048 x 2048 이상인 데이터에서는 GPU 메모리 문제로 인해 Super-resolution이 불가능하였기 때문이다. CPU, GPU 기반 상태 값 결정에서 비교 실험한 것과 같이 쿼드 트리를 생성할 때 CPU 순회 방식을 사용하는 것은 GPU 병렬 처리를 사용한 것보다 느릴 뿐만 아니라, Whole SR 방식보다도 느리고, 일정 패치 크기에서는 Patch SR보다도 느린 것을 확인할 수 있었다. 또 패치 크기에 따라서는 Patch SR은 패치 크기가 작아지면 작아질수록 걸리는 시간이 급증하는 것을 확인할 수 있었다. 또한, 쿼드 트리를 사용했을 시에는 최하단 노드의 패치 크기를 무조건 작게 하는 것이 항상 속도가 빨라지는 것이 아님을 확인하였고, 패치 크기를 입력 해상도의 16분의 1 크기로 했을 때 속도가 가장 빠른 것을 확인하였다.

4.2.3 입력 해상도에 따른 속도 비교

마지막으로 Figure 12는 입력 해상도를 다르게 했을 때, 300프레임의 데이터를 Patch SR과 CPU 버전의 쿼드 트리, 본 논문이 제안하는 GPU 버전의 쿼드 트리 방식을 비교해 본 결과이다. 패치 크기는 각각의 기법에서 가장 속도가 빠른 크기로 하였다. 입력 해상도의 크기가 커짐에 따른 시간의 기울기가 본 논문에서 제안하는 방식이 제일 작은 것으로 보아 해상도가 계속 커져도 본 논문에서 제안하는 기법이 가장 속도가 빠르다는 것을 유추해 볼 수 있었다.

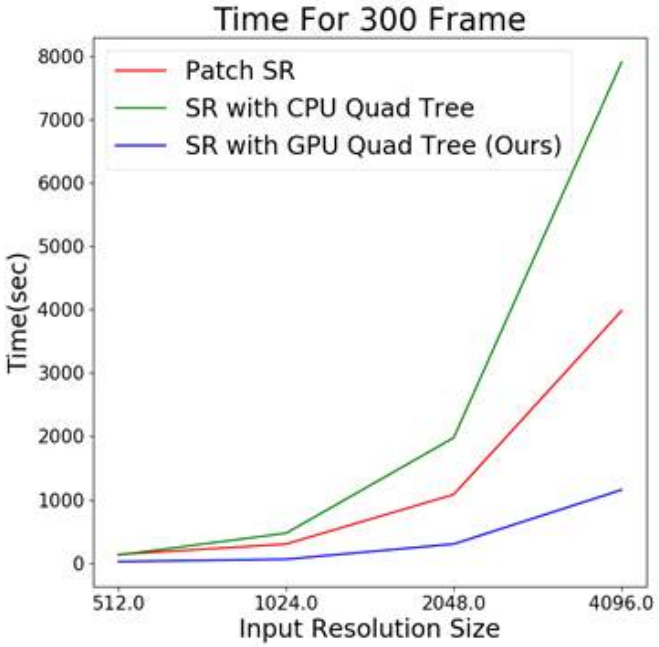


Figure 12 Result of the running time for different input resolutions

5. 결론

우리는 본 논문에서 2D 연기 시뮬레이션을 쿼드 트리와 Super-resolution을 통해 가속화 하고 메모리를 효율적으로 사용하는 기법을 제안하였다. 분할된 패치와 상태 값을 기반으로 쿼드 트리를 생성하였으며, 생성된 쿼드 트리를 사용하여 Super-resolution이 필요한 공간을 분할 및 분류하고, Super-resolution과 후처리를 통해 결과 데이터까지 얻을 수 있는 기법을 제안하였다.

실험을 통해 알 수 있었던 사실은 각각 나눠진 패치의 Super-resolution의 필요 유무를 확인할 때, CPU를 활용하는 것보다 GPU를 활용하는 것이 훨씬 큰 폭으로 속도가 향상된다는 것이다. 또한, 이 기법을 통해 이전 기법들의 입력 해상도의 크기에 따른 GPU 메모리 부족 문제를 해결하였고, Super-resolution을 필요한 공간에만 진행함으로써 전체 시뮬레이션 속도가 향상되었음을 확인할 수 있었다. 하지만 입력 데이터의 해상도에 따른 적합한 패치 크기는 현재 실험을 통해서만 확인할 수 있는 한계점을 가지고 있다. 적합한 패치 크기를 자동으로 찾을 수 있다면 더 효율적인 기법이 될 것이다.

향후 3D 연기 시뮬레이션 데이터에서도 본 논문이 제시하는 기법이 적용되는지 옥 트리를 활용하여 확인할 것이다. 또한, 제안하는 기법을 다른 기법에도 적용해 보기 위하여 Sato et al.[17]이 제안한 Style Transfer나, Xie et al.[12]이 제안한 GAN을 활용한 Super-resolution 등의 공간을 활용한 여러 가지 기법에도 적용해 볼 것이다.

References

[1] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, “Accelerating eulerian fluid simulation with convolutional networks”, *International Conference on Machine Learning* (ICML), pages 3424–3433, 2017.

[2] Xiangyun Xiao, Yanqing Zhou, Hui Wang, Xubo Yang “A Novel CNN-based Poisson Solver for Fluid Simulation” *IEEE Transactions on Visualization and Computer Graphics* (TVCG), page 1-1, 2018

[3] Mengyu Chu and Nils Thuerey, “data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors”, *ACM Transactions on Graphics* (TOG), Volume 36, issue 4, Article No. 69, 2017

[4] You Xie, Erik Franz, Mengyu Chu, Nils Thuerey, “tempoGAN: a temporally coherent, volumetric GAN for super-resolution fluid flow”, *ACM Transactions on Graphics* (TOG), Volume 37, issue 4, Article No. 95, 2018

[5] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang, “Image Super-Resolution Using Deep Convolutional Networks”, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (TPAMI), Volume 38, Issue 2, page 295-307, 2016

[6] Christian Ledig, Lucas Theis , Ferenc Huszar , Jose Caballero , Andrew Cunningham , Alejandro Acosta , Andrew Aitken , Alykhan Tejani , Johannes Totz , Zehan Wang , Wenzhe Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2017

[7] Mengyu Chu, You Xie, Laura Leal-Taixé, Nils Thuerey, “Temporally Coherent GANs for Video Super-Resolution (TecoGAN)”, *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2019.

[8] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, Christopher Schroers, “A Fully Progressive Approach to Single-Image Super-Resolution”, *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pp. 864-873, 2018

[9] Gregory M. Hunter, Kenneth Steiglitz, “Operations on Images Using Quad Trees”, *IEEE Transaction on Pattern Analysis and Machine Intelligence* (TPAMI), vol. PAMI-1, No. 2, 1979.

[10] D. Meagher, “Octree encoding: A new technique for the

representation, manipulation and display of arbitrary 3d objects by computer", *Technical Report* IPL-TR-80-111, 1980.

[11] Kun Zhou, Minmin Gong, Xin Huang, Baining Guo, "Data-Parallel Octrees for Surface Reconstruction", *IEEE Transactions on Visualization and Computer Graphics* (TVCG), Volume 17, Issue 5, page 669-681, 2011

[12] Gernot Riegler, Ali Osman Ulusoy, Andreas Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions", *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), arXiv:1611.05009v4, 2017

[13] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, Xin Tong, "O-CNN: octree-based convolutional neural networks for 3D shape analysis", *ACM Transactions on Graphics* (TOG), Volume 36, Issue 4, Article No. 72, 2017

[14] Maxim Tatarchenko, Alexey Dosovitskiy, Thomas Brox, "Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs", *IEEE International Conference on Computer Vision* (ICCV), pp. 2088-2096, 2017

[15] J. Stam "Stable Fluids," *ACM SIGGRAPH*, Annual Conference Series. 121-128, 1999

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pp 770-778, 2016

[17] Syuhei Sato, Yoshinori Dobashi, Theodore Kim, Tomoyuki NishiTa, "Example-based Turbulence Style Transfer" *ACM Transactions on Graphics* (TOG), volume 37, issue 4, Article No. 84, 2018

〈저자소개〉

홍병선
- 2018년 충북대학교 전자공학부 학사
- 2018년-현재 고려대학교 영상정보처리협동과정 석사과정
- 관심분야: 컴퓨터그래픽스, 기계학습
- https://orcid.org/0000-0003-0495-6133

박지혁
- 2017년 인하대학교 컴퓨터공학과 학사
- 2019년 고려대학교 영상정보처리협동과정 석사
- 관심분야: 컴퓨터그래픽스, 기계학습
- https://orcid.org/0000-0002-8175-7413

최명진
- 2013년 고려대학교 컴퓨터전파통신공학부 학사
- 2013년- 현재 고려대학교 컴퓨터전파통신공학과 석박통합과정
- 관심분야: 물리 기반 시뮬레이션, 가상현실, 인공지능
- https://orcid.org/0000-0002-0389-3911

김창헌
- 1979년 고려대학교 경제학과 학사
- 1987년 한양대학교 전산학과 석사
- 1993년 Tsukuba University 전자정보학과 박사
- 2014년 고려대학교 정보통신대학 컴퓨터통신공학부 교수
- 2014년-현재 고려대학교 정보대학 컴퓨터학과 교수
- 관심분야: 컴퓨터 그래픽스, 컴퓨터 비전, 물리 기반 시뮬레이션
- https://orcid.org/0000-0002-9630-9031