깊이 영상 기반 영상 합성을 위한 병렬 프로그래밍

정재일, 호요성
광주과학기술원 정보통신공학부
e-mail : jijung@gist.ac.kr, hoyo@gist.ac.kr

Parallel Programing for Depth Image-based Rendering

요 약

원하는 시점의 영상을 합성하기 위해서 깊이 영상 기반의 영상 합성 기술이 크게 각광받고 있다. 하지만, 많은 행렬 연산과 복잡한 필터링 등으로 인해 고해상도 영상을 합성하는 데는 많은 시간이 필요하다. 본 논문은 그래픽 프로세싱 유닛을 이용하여 행렬 연산을 병렬화하고 복잡한 필터링을 단순화하여 고속으로 가상 시점을 합성할 수 있는 기술을 제안한다. 가상 시점과 실제 촬영된 영상간의 호모그래피 행렬을 깊이별로 계산한 뒤, 각 스레드를 통해서 열 단위로 워핑 과정을 수행하고, 깊이 값을 고려한 필터링과 깊이 확장을 화소 단위로 수행하여 속도와 화질을 동시에 개선한다. 실험을 통해 제안한 방법이 32dB 이상의 화질을 유지하면서 Full-HD 영상을 초당 12 프레임 이상 합성할 수 있음을 확인하였다.

Abstract

A depth image-based rendering (DIBR) technique takes center stage, but it takes a long time to synthesize a new image due to many matrix calculations and complex filtering. In this paper, we accelerate DIBR by parallelizing each process on a graphic processing unit (GPU) and simplifying the filters. We calculate homography matrices for all depth values, and conduct row-wise 3D warping via threads in the GPU. In addition, we use a pixel-wise depth filtering and expending in order to improve visual quality of synthesized images and speed. From the experimental results, we confirm that our method is able to synthesize 12 full-HD images in a second with greater than 32dB visual qulaity.

Keywords : 깊이 영상 기반 영상 합성(depth image-based rendering), 병렬 프로그래밍(parallel programming), 쿠다(CUDA), 3차원 영상 서비스(3D image service), 자유시점 텔레비젼(freeview point TV)

Ⅰ. 서 론

흑백과 컬러, 그리고 고해상도 영상 서비스를 거쳐서 최근에는 3차원 영상 서비스가 크게 각광받고 있다. 특히, 카메라 기술과 영상 처리 기술이 발전하면서 사용자가 원하는 시점의 영상을 골라서 시청할 수 있는 자유시점 영상에 대한 관심이 증가하고 있다. 이를 위해서는 실제로 많은 위치에서 촬영된 영상이 필요한데, 모든 시점의 영상을 물리적인 카메라로 촬영한다는 것은 실제로 불가능하다.

따라서 깊이 영상을 기반으로 하는 영상 합성 기술(DIBR)이 제안되어 널리 사용되고 있다. 이 기술은 객체들의 거리 정보를 포함하고 있는 깊이 영상을 이용하여 2차원 영상을 3차원 공간에 투영한 뒤, 원하는 시점 위치로 재투영하여 원하는 영상을 합성한다 [1].

하지만, 이 기술은 많은 행렬 연산과 복잡한 필터링 등을 사용하기 때문에 영상 합성에 많은 시간을 소요하게 된다. 따라서 본 논문에서는 그래픽 프로세싱 유닛(GPU)을 이용하여 영상 합성 과정을 병렬화하고 복잡한 필터링 구조를 단순화하여 시점 합성에 소요되는 시간을 획기적으로 줄이는 방법에 대해서 제안한다.

Ⅱ. 본 론

기존에는 중앙 처리 장치(CPU)를 이용한 순차적인 프로그래밍이 널리 사용되었으나, 최근에는 GPU를 이용한 병렬 프로그래밍이 많은 관심을 받고 있다. 특히, nVidia사에서 CUDA라는 C언어 기반의 병렬처리 라이브러리를 배포하면서 연산 시간 단축에 크게 기여하고 있다 [2]. 본 논문

에서는 이를 이용하여 DIBR의 각 부분을 화소 혹은 열 단위로 병렬화하고, 필터들을 병렬처리에 맞게 단순화하여 영상 합성을 고속화하였다.

1. 3차원 워핑

DIBR에서 원하는 시점으로 객체를 이동시키기 위해서 사용되는 기술이 3차원 워핑이다. 3차원 워핑을 통해서 2차원 영상을 3차원 공간으로 투영시키는데, 이때 (1)과 같은 수식이 이용된다.

$$(x,y,z)^T = \overline{RA}^{-1}(u,v,1)^T d + \bar{t} \quad (1)$$

여기서 d는 깊이값을, A는 카메라 내부 변수를, R과 t는 카메라 외부 변수를 의미한다. 이 과정은 모든 화소별로 계산돼야 하므로 많은 시간이 소요된다. 따라서 좌우 깊이 영상과 색상 영상을 GPU의 전역 메모리로 로딩 시킨 뒤, 스레드로 나눠서 행렬 계산을 수행한다. 행렬 연산의 수를 줄이기 위해서 각 깊이 별로 호모그래피를 계산하여 깊이 값에 따라 행렬곱 연산 한번으로 투영 화소의 위치를 얻을 수 있도록 한다. 이때 깊이 값에 따른 화소의 중첩을 피하기 위해서, 화소 단위가 아닌 열 단위의 병렬화를 사용한다. 이를 위해 시점의 위치에 따라 병렬화 수행 방향을 결정한다.

2. 깊이 값 필터링

DIBR의 경우 깊이 값의 정확도에 따라서 합성 영상의 화질이 크게 달라지기 때문에, 본 논문에서는 깊이 영상의 필터링을 통해서 합성 영상의 화질을 개선한다. 이 과정 또한 GPU 상에서 화소 단위로 병렬화되어 수행되며, 각 깊이 값은 다음과 같은 수식에 의해서 필터링된다.

$$D_{Dilated} = \begin{cases} Depth \dfrac{D_{cur} + \sum_{N} \begin{cases} D_{nb} & if(|D_{cur} - D_{nb}| < Th) \\ 0 & else \end{cases}}{N_{vnb} + 1} \end{cases} \quad (2)$$

여기서 D_{cur}과 D_{nb}는 현재 화소와 인접 화소의 깊이 값을 의미하고, N은 인접 화소의 개수 N_{vnb}는 현재 화소와의 깊이 값 차이가 Th 이하인 유효화소의 개수를 의미한다.

Ⅲ. 실험 결과

성능을 평가하기 위해서 Geforce 580 GTX 그래픽 카드를 이용하여 제안한 알고리즘을 수행하였다. 가상 시점 영상은 주어진 좌우 영상의 중간 시점으로 선택하였다. 그림 1과 2에서와 같이 합성된 영상의 화질은 미리 촬영된 동일 시점의 영상과의 PSNR 비교를 통해 정량화하였다.

시점 합성을 위해 소요된 시간을 비교하기 위해서, Intel i7 CPU X980에서 순차적 DIBR을 사용한 경우와 수행 시간을 비교하여 표 1에 나타내었다. GPU를 사용한 경우 97%이상의 수행 시간 단축 효과가 있음을 확인할 수 있었다.

(a) 원본 중간 시점 영상 (b) 합성 영상 (34.14dB)
그림 1. Bookarrival 영상

(a) 원본 중간 시점 영상 (b) 합성 영상 (32.03dB)
그림 2. Cafe 영상

표 1. CPU와 GPU 기반의 알고리즘 수행 시간 비교

영상 크기	CPU (s)	GPU (s)	증감율 (%)
640 x 480	1.648	0.021	-98.72
1024 x 768	2.244	0.033	-98.53
1280 x 960	2.801	0.052	-98.14
1920 x 1080	3.318	0.082	-97.53

Ⅳ. 결 론

본 논문에서는 GPU를 이용하여 DIBR을 가속화하는 기술을 제안하였다. 병렬화된 3차원 워핑과 필터링을 통해서 32dB 이상의 화질을 유지하면서 97% 이상의 시점 합성 시간을 단축시킬 수 있었다.

감사의 글

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NO. 2011-0030822).

참 고 문 헌

[1] E. Lee and Y. Ho, "Generation of High-quality Depth Maps using Hybrid Camera System for 3-D Video," Journal of Visual Communication and Image Representation, vol. 22, no. 1, pp. 73-84, 2010.
[2] CUDA Reference Manual, Nvidia, 2011.