GPGPU 기반의 변위증분 방법을 이용한 중간시점 고속 생성

구자명¹ · 서영호² · 김동욱^{1*}

Fast Generation of Intermediate View Image Using GPGPU-Based Disparity Increment Method

Ja-myung Koo¹ · Young-ho Seo² · Dong-wook Kim^{1*}

¹ Department of Electronic Materials Engineering, Kwangwoon University, Seoul 447-1, Korea

² Department of Liberal Arts, Kwangwoon University, Seoul 447-1, Korea

요 약

자유시점 또는 오토스테레오스코픽 비디오 서비스는 3차원 영상을 제공하는 차세대 방송 시스템으로, 여러 시점의 영상들이 필요하다. 본 논문에서는 가상 시점 영상을 고속 생성하기 위해 알고리즘 병렬 구조를 최적화하고, Compute Unified Device Architecture(CUDA)를 이용한 General Propose Graphic Processing Unit(GPGPU) 기반의 중간시점 영상 고속 생성을 위한 최적화 기법을 제안한다. 제안한 방법은 좌/우 깊이영상을 병렬화시킨 스테레오 정합알고리즘을 이용하여 변위정보를 얻은 후, 깊이 당 변위증분을 계산하여 사용한다. 계산된 변위증분을 사용하여 해당 각 화소들의 깊이 값을 이용하여 좌/우 영상들을 원하는 위치의 중간시점으로 영상을 이동시킨다. 그 다음, 비폐색영역들을 서로 상호 보완하여 없앤 다음에 남은 홀들은 홀 필링으로 없애 최종 중간시점 영상을 생성한다. 제안한 방법을 구현하여 여러 실험 영상에 적용한 결과, 생성된 중간시점 깊이영상의 화질은 평균 PSNR 30.47dB이었으며, Full HD급 중간시점 영상을 초당 38 프레임 정도 생성하는 속도를 보였다.

ABSTRACT

Free-view, auto-stereoscopic video service is a next generation broadcasting system which offers a three-dimensional video, images of the various point are needed. This paper proposes a method that parallelizes the algorithm for arbitrary intermediate view-point image fast generation and make it faster using General Propose Graphic Processing Unit(GPGPU) with help of the Compute Unified Device Architecture(CUDA). It uses a parallelized stereo-matching method between the leftmost and the rightmost depth images to obtain disparity information and It use data calculated disparity increment per depth value. The disparity increment is used to find the location in the intermediate view-point image for each depth in the given images. Then, It is eliminate to disocclusions complement each other and remaining holes are filled image using hole-filling method and to get the final intermediate view-point image. The proposed method was implemented and applied to several test sequences. The results revealed that the quality of the generated intermediate view-point image corresponds to 30.47dB of PSNR in average and it takes about 38 frames per second to generate a Full HD intermediate view-point image.

키워드 : 3차원 디스플레이, 중간시점 영상 생성, 범용그래픽 처리 장치, 쿠다

Key word : 3D display, intermediate view image generation, GPGPU, CUDA

접수일자 : 2013. 01. 14 심사완료일자 : 2013. 07. 29 게재확정일자 : 2013. 08. 05

* Corresponding Author Dong-Wook Kim(dwkim@kw.ac.kr, Tel:+82-2-940-5167)

Department of Electronic Materials Engineering, Kwangwoon University, Seoul 447-1, Korea

Open Access <http://dx.doi.org/10.6109/jkiice.2013.17.8.1908>

print ISSN: 2234-4772 online ISSN: 2288-4165

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

I. 서 론

2009년부터 영화 ‘아바타’에 의해 본격화된 스테레오 3차원(3-dimensional, 3D) 영상에 대한 관심이 커지면서 세계 3D 관련 기관들이 그간 준비를 해놓은 각종 장비, 장치들을 출시하기 시작하였다. 현재의 3D 영상/비디오는 대부분 좌우의 두 시점에 해당하는 영상들을 좌, 우안에 각각 주입시키는 스테레오스코피(stereoscopy) 방식이며, 이 방식으로 능동식(셔터 글라스 방식) 또는 수동식(편광방식)의 안경식 모니터를 사용하고 있다. 하지만, 이러한 방식에 대해 시청자들이 안경을 착용하여야 하는 불편함과 두 시점의 영상을 양안에 강제적으로 주입하면서 생겨난 현기증과 같은 불편감이 발생하면서 [1], 이에 최근 시청자의 안전한 시청을 위해서 국제 표준화기구[2] 및 일본[3]에서 시청안정성을 위한 가이드라인을 발표한 바 있다. 이러한 문제들 때문에 단순한 안경식 3D를 벗어나 이제에는 시점에 제약이 없는 3D 영상에 대한 욕구로 발전되고 있다. 시점에 제약이 없는 방식으로는 자유시점(free view-point)[4]과 오토스테레오스코프(auto-stereoscope)[5]가 이 방식에 포함된다. 자유시점은 주로 2D영상으로, 시청자가 보기를 희망하는 또는 시청자의 위치에 따라 시점의 영상을 볼 수 있도록 하는 기술이다. 오토스테레오스코프는 앞서 설명한 자유시점을 스테레오 영상의 개념으로 발전시킨 것으로, 시청자의 위치에 해당하는 스테레오 3D영상을 보여주는 기술이다. 이것은 또한 다시점(multi-view) 또는 극초다시점(ultra multi-view) 스테레오스코프의 시점을 거의 무한대로 늘린 개념이라 볼 수 있다. 그리고 좌/우안에 강제로 시차를 갖는 영상을 주입하지 않고 공간상에 3D영상을 맺혀, 일반 사물을 보듯이 안경을 착용하지 않고 공간상에 3D 입체영상 볼 수 있는 홀로그래피 기술은 디지털 영상분야의 궁극적인 목표로 여겨지고 있다[6][7].

하지만, 앞서 언급한 모든 기술들의 문제점은 여러 시점들의 정보가 필요하며 이러한 모든 시점을 포함하게 되면 데이터 전송량이 엄청나게 많아진다는 것이다. 따라서 최근에는 여러 시점의 정보들을 얻기 위해서 송신측에서 필요한 정보를 제공하고 수신측에서 그 정보를 받아 필요한 시점들을 생성하게 하는 것이다. 최근, 원하는 가상시점을 생성하는데 DIBR(depth image based rendering) 방법을 가장 많이 사용하는데, 이 기술은 텍

스처 영상에 상응하는 깊이 영상을 이용하여 텍스처 영상을 원하는 가상시점으로 생성하는 기술이다. 이 기술로 생성된 필요한 시점의 화질은 깊이정보에 크게 좌우되는데, 예전에는 깊이정보를 스테레오 정합(stereo matching)으로 획득하는 것이 일반적이었지만, 최근에 깊이정보를 직접 획득할 수 있는 깊이카메라[8][9]가 대중화되면서 자유시점 및 다시점[10], 오토스테레오스코프의 가상시점 생성뿐만 아니라 깊이정보를 포함한 정보들을 송신하기 위한 코딩기법에 대한 연구도 진행되고 있다[11].

이러한 깊이정보를 사용하여 원하는 시점의 3D영상을 생성하는 DIBR은 시점차이에 의한 왜곡이 발생하거나 시점이 이동하면서 비폐색영역(disocclusion)과 같은 생성한 시점에 대한 정보 부족으로 본래 영상보다 화질이 떨어진 영상을 얻게 된다. 그래서 이러한 문제점들을 해결하기 위해 전처리 또는 후처리로 좀 더 본래 영상에 가까운 영상을 생성하기 위해서 연구가 진행 중이다[12][13]. 그러나 이러한 과정으로 인하여 생성 속도가 그만큼 늘어나게 되면서 비디오 서비스를 위한 실시간 생성은 그만큼 어려워지게 되었다. 이러한 문제들이 나타나면서 자연스럽게 임의의 가상시점 영상을 좀 더 빠르게 생성하기 위한 방법을 찾게 되고 최근에는 GPU(graphics processing unit)를 이용한 고속 생성 방법들을 연구하고 있다[14][15].

본 논문에서는 GPU 방식을 이용하여 실시간으로 Full HD 해상도의 임의의 중간시점 영상을 생성하는 방법을 제안한다. 주어진 좌/우 깊이영상으로 스테레오 정합 알고리즘을 병렬화로 최적화하여 깊이의 변화에 대한 변위(disparity)의 변화값을 구하고, 구한 깊이당 변위중분을 이용하여 좌/우 영상 상의 임의의 시점에 해당하는 영상을 고속 생성하는 방법을 제안한다.

II. 중간 시점 생성 방법

요즘 텍스처 영상뿐만 아니라 그에 상응하는 깊이 정보가 생겨나면서 가상 시점을 생성하는데 쉽게 깊이 정보를 이용하여 생성하는 DIBR 방법을 가장 많이 사용하고 있다. 임의의 중간 시점의 영상을 생성하기 위해서 원본 시점의 영상을 DIBR을 이용하여 원하는 목표 시점으로 이동하는 것이다. 이때 3D 워핑 기술을 이용

하여 원본 시점 영상을 가상 시점 영상으로 옮길 수 있다[16]. 3D 워핑 기술은 촬영한 카메라 파라미터와 깊이 정보를 이용하여 원본의 시점의 각 화소가 생성하고자 하는 가상의 시점영상의 어느 곳에 나타나는지를 정의하여 가상 시점을 생성하는 것이다. 그림 1은 3D 워핑 기술을 이용하여 원본 시점의 영상을 원하는 가상 시점의 영상으로 생성한 결과를 보이고 있다.

그림 1. 중간시점 깊이 및 텍스처 영상의 예 (a) 원본 (b) 가상 시점 (c) 3D 워핑
Fig. 1 Example of the intermediated depth and texture image (a) original (b) virtual (c) 3D warping

그림 1(a) 시점의 텍스처 영상을 깊이 정보를 이용하여 (b) 시점의 텍스처 영상을 생성하기 위해서 3D 워핑 기술을 사용한 결과가 그림 1(c)인 것이다. 3D 워핑을 한 결과 보는 것처럼 생성한 영상에 하얀 부분들이 존재하는데 원본 시점의 영상의 정보만으로 생성하고자 하는 가상 시점의 영상을 생성하는데 정보가 부족하여 생겨난 것인데 이것은 비페색영역이라고 부른다. 즉, 원본 시점의 영상에서 객체에 의해 가려져 있던 부분이 가상 시점으로 이동하면서 그 부분이 들어나면서 생기는 것이다. 이것은 DIBR 기술의 한계점으로, 이 부분을 보다 가상 시점 원본에 가깝게 복원하는 것이 이 부분

을 연구하는 사람들에게 큰 해결해야 부분이다.

이러한 비페색영역을 가장 효과적으로 해결하는 방법이 또 다른 시점의 원본 영상을 똑같이 생성하고자 하는 가상 시점의 영상을 만들어서 서로 상호 보충하여 비페색영역을 최소한으로 없애는 블렌딩 방법이 있다. 즉, 각각의 다른 시점의 텍스처 영상과 그에 상응하는 깊이 영상이 두 쌍씩 존재하며, 그것들을 각 시점의 중간 시점으로 3D 워핑을 한 후, 생겨난 비페색영역을 서로 상호 보완하는 것이다. 그림 2에서 블렌딩 방법의 예를 보이고 있다.

그림 2. 블렌딩 방법 예: (a) 왼쪽 시점에서 중간 시점, (b) 오른쪽 시점에서 중간 시점, (c) 블렌딩 영상
Fig. 2 Example of blending method: (a) intermediate view from left view, (b) intermediate view from right view, (c) blending

그림 2에서 볼 수 있듯이 원/오른 시점의 영상들을 각각 중간 시점으로 생성할 때 원본영상에 정보가 부족하여 비페색영역들이 생겨난 것을 볼 수가 있고, 이러

한 비폐색영역에 해당하는 정보들을 다른 시점에서 찾아 채울 수 있다는 것을 알 수가 있다. 즉, 예를 들어 왼쪽 시점의 원본 영상을 중간 시점으로 3D 워핑을 한 경우, 비폐색영역들이 객체의 오른쪽에 생기게 되고, 반대로 오른쪽 시점의 원본 영상을 중간 시점으로 3D 워핑을 하면 비폐색영역들이 반대로 객체의 왼쪽에 생기게 되는데, 이러한 비폐색영역들은 반대 시점에서 중간 시점으로 이동한 영상에서는 존재하는 정보들이라서 비폐색영역을 채울 수 있게 된다. 이렇게 서로 정보가 부족한 비폐색영역들을 서로 상호 보완하면서 해결한 것이 그림 2(c)인 것이다.

그러나 블렌딩을 한 후에도 채우지 못한 비폐색영역들이 존재하는데 그 이유는 양쪽 시점인 원/오른 시점에는 존재하지 않는 정보이지만 중간 시점에서는 존재하는 정보가 있기 때문에 생기는 현상이다. 이러한 나머지 비폐색영역들은 홀필링 방법으로 채우게 된다. 기존에 존재하는 홀필링 방법중 하나는 Criminisi[17]가 제안한 방법이 있는데, 홀 주변에 등과도선을 봐서 그 부분들을 우선권을 주면서 반복적으로 그 부분들을 복사를 하면서 홀들을 채우는 방법이다. 그런데 남겨진 홀들을 분석해 보면 결국에는 비폐색영역들이 남겨진 부분으로 비폐색영역은 시점이동으로 객체에 의해 가려져 있던 배경에 속하는 영상들인 것을 알 수가 있다. 즉, 남겨진 홀들은 객체에 속하는 영상보다는 배경에 속하는 영상들로 채우는 것이 맞다는 것이다. 그래서 I. Daribo[18]는 앞서 말한 Criminisi가 제안한 방법을 좀 더 가상 시점 생성에 특화시켜 남겨진 홀들을 좀 더 배경에 속하는 영상들을 우선으로 채우는 방법을 제안하였다. 또 다른 방법으로 S. Zinger[12]는 채울 홀의 주변 8방향에 있는 화소들을 찾은 후, 그 중에 배경에 속하는 화소들만을 가져와서 이 중에서 홀까지의 거리에 따른 가중치를 주어서 홀들을 채우는 방법도 제안하였다. 이렇듯 가상 시점 생성에 생기는 비폐색영역들을 효과적인 방법으로 채우는 연구들이 계속 진행되고 있다.

III. 변이값 증분을 이용한 고속 중간시점 영상 생성

본 장에서는 본 논문에서 제안하는 GPGPU기반의 중간시점 영상 고속 생성 방법에 대해 설명한다. 제안

하는 방법의 전체 흐름도를 그림 3에 나타내었다.

그림 3. 제안한 중간시점 영상 고속 생성 방법의 전체 흐름도
Fig. 3 Overall flow diagram of the proposed intermediate view-point image fast generation method

그림에서 보는 바와 같이, 제안한 방법은 크게 전처리(pre-processing) 과정, 가상 시점 영상 생성(virtual view image generation) 과정으로 이루어진다. 좌/우의 텍스처 영상과 그에 상응하는 깊이영상이 주어지고, 이 두 쌍의 영상들은 보정과정을 거쳤다고 가정한다.

3.1. 전처리 과정

앞서 말한 것처럼 3D 워핑을 하기 위해서는 깊이 정보뿐만 아니라 그것을 촬영한 카메라의 파라미터가 있어야지 가능하다. 그래서 본 논문에서는 카메라 파라미터를 사용하지 않고 중간시점 영상을 생성하는 기본 방법으로 주어진 좌/우 깊이영상의 깊이정보의 변화 당 변위정보의 변화 값을 구하여 가상시점 생성에 사용하게 된다. 깊이정보의 변화 당 변위정보의 변화 값을 구하기 위해서 먼저, 변위정보를 얻은 깊이정보를 정해야 한다. 입력된 두 영상 중 하나의 영상에서 빈도수가 높은 깊이 값 두 개를 정해야 한다. 그 방법으로, 깊이 영상의 깊이 값의 빈도 수의 히스토그램을 구하고, 그중에 상위 10%와 하위 10% 범위 내에서 빈도수가 가장 높은 값(z_{far} , z_{near})을 각각 선택한다.

그 다음, 이 두 깊이 값에 해당하는 모든 화소들에 대한 두 깊이 영상간의 스테레오 정합을 수행하여, 획득한 변위 값들 중에서도 가장 빈도수가 높은 변위 값 d_{far} 와 d_{near} 를 각각 획득한다.

그림 4. 고속 스테레오 정합을 위한 GPU를 이용한 병렬화
Fig. 4 Parallelism using GPU for fast stereo matching

이 때 연산시간을 많이 소요하는 전역적 방법이 아닌 3×3 의 고정된 정합창을 사용하는 국부적(local) 방법을 사용한다. 그런데도 변위증분을 구하는 전처리 과정에서 많은 시간이 소요되어 고속 변위 획득을 위하여 병렬화 처리를 위하여 GPU를 실행하여 좀 더 빠르게 변위들을 획득한다. 그림 4에서 GPU를 이용한 스테레오 정합 병렬처리의 예를 나타내고 있다.

그림 4에서 볼 수 있듯이 병렬화를 위하여 GPU를 구성하고 있는 다수의 스트림 코어들을 이용하여 필요한 데이터 처리들을 각각의 스트림 코어에 할당을 한 후 동시에 연산을 수행하도록 하는 것이 가능하다. 그리고 쓸데없는 수행 시간을 줄이기 위하여 데이터 저장 및 접근 속도를 최소화하는 방법이 있는데, 저장할 수 있는 데이터량은 크면서 읽기/쓰기가 가능한 데이터를 호스트와 디바이스가 서로 주고받기가 가능하며, 모든 쓰레드에 의해 접근이 가능하지만 병렬처리에 따른 동시 접근에 대한 병목 현상으로 인하여 접근 속도가 느린 전역 메모리(global memory) 접근을 제한적으로 하고 각 온 칩 프로세서 안에 존재하여 저장할 수 있는 데이터량은 적지만 접근 속도가 빠른 공유 메모리(shared memory)를 사용하여 좀 더 고속 처리를 이끌게 하였다. 공유 메모리는 같은 그룹에 속한 쓰레드들끼리 공유가 가능한 메모리로 먼저 전역 메모리에서 공유 메모리로 반복적으로 사용하는 데이터들을 미리 옮겨 놓은 후, 쓰레드들이 접근 시간이 긴 전역 메모리에 접근하지 않

고 접근 횟수를 공유 메모리에 높여 반복적으로 사용하는 데이터를 복사해 놓은 공유 메모리 안에 데이터를 자주 사용하게 함으로써 효율적으로 병렬 처리가 가능하도록 한다. 즉, 일반적으로 많이 사용하게 되는 파라미터, 본 방법에서는 찾고자 하는 깊이의 변위 정보를 공유 메모리에 복사해 놓고 사용함으로써 처리 속도를 줄이는 것이 가능하다. 이렇게 획득한 변위 값과 해당 깊이 값으로 깊이증분 당 변위증분을 식 (1)과 같이 계산한다.

$$d_{incr} = \frac{\Delta d}{\Delta z} = \frac{d_{near} - d_{far}}{z_{near} - z_{far}} \quad (1)$$

제안한 방법은 변위증분을 찾기 위한 전처리 과정은 장면이 시작할 때 첫 번째 프레임에 대해 한 번만 수행하고, 그 장면의 나머지 프레임에는 동일한 변위증분 값을 사용한다.

3.2. 가상 시점의 영상 생성

이렇게 변위증분 값이 구해지면 이 값을 이용하여 좌 영상 또는 우 영상으로부터 좌/우 영상 사이의 임의의 중간시점에 해당하는 영상을 생성할 수 있다.

그림 5. 중간시점
Fig. 5 Intermediate view point

구한 변위증분 값으로 좌/우 시점 영상에서 중간시점 영상으로 시점을 이동하는 방법은 Koo[19]가 제안한 방법을 사용하였다. 좌 영상과 우 영상, 생성하고자 하는 중간시점 영상이 각각 그림 5와 같은 위치라고 가정한다면, 좌 영상으로부터 중간시점(첨자 IV) 영상을 생성할 경우 좌 깊이영상의 i 번째 화소의 깊이 값이 $z_{i,left}$ 일 때 이 화소에 대한 중간시점 화소의 변위 값 $d_{i,IV;left}$ 은,

$$d_{i,IV,left} = d_{incr} \frac{B'}{B} z_{i,left} \quad (2)$$

으로 구할 수 있다. 따라서 이 화소에 해당하는 중간시점의 x -좌표 $x_{i,IV,left}$ 은 좌영상 화소의 x -좌표가 $x_{i,left}$ 일 때,

$$x_{i,IV,left} = x_{i,left} - d_{i,IV,left} \quad (3)$$

가 된다.

즉 좌영상의 $(x_{i,left}, y_{i,left}, z_{i,left})$ 에 해당하는 화소는 중간시점 영상에서 $(x_{i,IV,left}, y_{i,left}, z_{i,left})$ 에 해당한다. 이와 같은 과정으로 모든 좌영상의 화소에 대응하는 중간시점 화소의 위치와 깊이 값을 계산하여 좌 영상으로부터의 중간시점 영상을 생성한다. 동일한 방법으로 우 영상으로부터도 같은 중간시점 영상을 생성할 수 있는데, 이 때 우영상의 $(x_{j,right}, y_{j,right})$ 화소의 깊이 값이 $z_{j,right}$ 일 때 이로부터 생성되는 중간시점 화소의 변위 값 $d_{j,IV,right}$ 과 중간시점 영상에서의 x -좌표는 각각,

$$d_{j,IV,right} = d_{incr} \frac{B'}{B} z_{j,right} \quad (4)$$

$$x_{j,IV,right} = x_{j,right} + d_{j,IV,right} \quad (5)$$

가 된다.

따라서 우영상의 화소 $(x_{j,right}, y_{j,right}, z_{j,right})$ 에 대응되는 중간시점 영상의 좌표 값과 깊이 값은 $(x_{j,IV,right}, y_{j,right}, z_{j,right})$ 가 된다. 이러한 방법으로 좌/우 영상의 각 화소들을 원하는 중간시점의 위치로 이동시킬 수 있다.

이 때, 그림 6처럼 영상의 모든 화소들의 병렬화 연산 처리에 의한 시점 이동을 위하여 영상의 각 화소들을 GPU안에 쓰레드에 각각 할당하여 동시에 연산 처리하도록 한다. 이 때, 효율적인 GPU의 병렬 동작을 위하여 적절한 자원 할당을 하여 동작하도록 해야 한다. 즉, 효과적인 병렬 동작을 위하여 블록 당 적정 쓰레드 수를 정해야 한다. 먼저 생성하고자 하는 임의의 중간시점 영상의 해상도에 따라 적정 블록 당 쓰레드를 정한다.

그래야 블록 안의 각 쓰레드들이 영상의 화소를 하나씩 맡아서 효율적인 병렬처리가 되도록 해야 한다. 예를 들어 한 블록 당 할당 할 수 있는 쓰레드의 수가 최대 512개라고 하면, 임의의 시점으로 영상을 이동할 때, 쓰레드를 모두 사용한다면 한 블록으로 512개의 화소들을 시점 이동이 가능하다.

그림 6. 병렬화를 위한 쓰레드 분할

Fig. 6 Image divided in threads for parallelism

즉, 생성하고자 하는 중간시점 영상의 해상도에 맞게 블록 당 쓰레드 수와 적정 그리드를 생성한다면 효과적인 병렬 처리에 의한 화소 시점 이동이 가능하다는 것이다. 예를 들어, 1024X768의 해상도를 갖는 영상을 생성하고자 한다면, 블록을 (32,16)으로 구성한다면 한 블록 당 쓰레드의 개수가 512개가 되어 효율적으로 최대 쓰레드 수를 사용하게 되고, 블록 당 쓰레드 개수에 맞추어 그리드를 (32,48)로 구성하면 모든 블록 안의 쓰레드들이 영상의 모든 화소 각 하나씩을 맡아 연산 처리할 수 있게 되어 효율적으로 생성하고자 하는 중간시점으로 모든 화소들을 병렬적으로 시점 이동할 수 있다. 이 때, 전처리 과정처럼 메모리 접근 속도를 높여서 추가적인 소요 시간을 줄이기 위하여 온 칩 프로세서에 존재하여 캐시처럼 사용가능한 공유 메모리의 사용가능한 크기를 고려하며 사용하여 좀 더 고속 처리를 이끌었다.

이 때 중간시점 영상의 특정 화소에 좌 또는 우영상에서 오는 두 개 이상의 화소들이 오는 경우가 발생한다. 즉, 중간시점 영상의 특정 화소에 두 개 이상 중복이 되는 경우가 발생이 된다는 것이다. 이 경우는 중복된 화소 중에서 촬영한 카메라에서 가장 가까운 것으로 판

단되는 깊이 값을 갖는 화소 값을 선택해야 한다. 이러한 과정을 하는 것을 깊이 버퍼링 과정이라고 한다. 본 논문에서는 이 과정에서 반복적인 연산을 피하기 위해 룩업 테이블을 사용한다. 현재 위치에 있는 깊이 값이 이동하여서 온 깊이 값보다 크다면, 그 화소는 가까운 깊이 값에 해당하는 화소로 채우고 깊이 버퍼는 그것으로 다시 개선했다. 이러한 과정을 그림 7에서 예를 보이고 있다.

그림 7. 지버퍼링
Fig. 7 Z-buffering

이제는 좌/우 영상을 각각 중간 시점으로 이동시키는데 앞서 말한 것처럼 보통 블렌딩 과정을 통해 두 시점에서 각각 생성한 중간 시점 영상들로 비폐색영역들을 서로 상호 보완하여 채우고 만약에 같은 위치에 두 개의 값이 중복되게 된다면 이 경우 카메라 거리에 따른 가중치를 주어서 평균으로 채우게 된다. 그러나 제안한 방법은 이러한 가중치에 따른 평균을 넣는 과정을 하지 않고 비폐색영역은 다른 시점에서 존재하는 정보들로 채우는 과정만을 걸친다.

이 과정의 다음 단계는 두 시점에서 정보가 존재하지 않아서 남겨진 비폐색영역들을 홀로 정의하고 그것들의 화소 값을 결정하는 단계이다. 앞서 말한 것처럼 위의 과정에서 홀로 정의된 대부분의 화소는 시점이 변화하면서 객체의 가려져 있던 배경에 속하는 화소들이다. 기존에 텍스처 영상의 중간시점 생성 연구에서 홀들을

채우는 방법은 상당히 많이 연구되어 왔으나, 본 논문에서는 가장 간단하면서 빠르고, 효과적인 방법을 사용한다. 남겨진 홀에 대하여 그 홀의 수평방향으로 좌측방향과 우측방향을 각각 검색하여 각 방향에서 값이 존재하는 가장 가까운 즉, 홀이 아닌 화소 값을 찾는다. 그 다음, 홀들은 배경에 속하는 화소 값으로 채워야 한다는 것을 알기 때문에 각 방향에서 찾은 화소 값들 중에서 깊이 값이 큰 값(원거리)을 선택하여 채우고자 하는 홀의 값으로 결정한다. 그림 8에서 홀필링 예를 보이고 있다. 홀을 채우기 위해서 좌/우 각각 수평방향으로 화소 값이 존재하는 화소를 찾은 후, 깊이 영상을 통하여 찾은 화소들의 깊이 값이 비교 한다. 그림 8(a)처럼 Z와 Z'의 깊이 값을 비교 한 후, 만약에 Z보다 Z'이 더 원거리에 속하는 깊이 값이라면 그 깊이 값에 속하는 텍스처 영상의 화소들로 홀들을 수평방향으로 채우게 된다.

그림 8. 홀필링 예: (a) 깊이 영상 홀필링, (b) 텍스처 영상 홀필링
Fig. 8 Example of hole filling: (a) hole-filling depth image, (b) hole-filling texture image

IV. 구현 및 실험

본 장에서는 3장에서 설명한 제안한 중간시점 영상 고속 생성방법을 구현하고 이를 여러 실험영상에 적용하여 제안한 방법의 성능을 평가한다.

4.1. 실험환경과 실험영상

3장에서 제안한 방법을 Microsoft Window7 환경에서 Microsoft Visual Studio 2008을 사용하여 구현하였다. 이 때 Intel Core i7-3770 3.4 GHz CPU와 16GB RAM을 가지고 있는 PC를 사용하였고, GPU는 코어 개수가 1536개이며, 메모리 용량이 2GB인 GTX-680을 사용하였다. 사용한 실험영상은 표 1에 보인 바와 같이

네 가지이며 모두 MPEG에서 사용하는 것이다[20]. 그리고 모든 시퀀스는 동일한 프레임 수를 갖도록 200 프레임 씩 잘라서 사용하였다. 이들 시퀀스들은 획득한 깊이정보를 그대로 사용하지 않고 약간의 후처리를 수행한 것으로 판단된다. 즉, 동일시간의 좌영상과 우영상, 그리고 중간시점 영상의 배경이나 물체가 상당히 많은 차이를 보이는 경우가 나타난다.

표 1. 실험영상

Table. 1 Test sequence

Sequence name	Resolution	Made by
Mobile	720×540	Philips
Lovebird1	1,024×768	ETRI
Cafe	1,920×1,080	GIST
Poznan street	1,920×1,088	Poznan Univ.

4.2. 실험결과

그림 9은 표 1의 실험 영상에서 첫 프레임에 속하는 영상들에 대해 제안한 방법의 실험 결과 영상을 보이고 있다. 그림 9(a)는 실험 영상의 중간 시점에 속하는 원본 영상이고, 그림 9(b)는 제안한 방법으로 생성한 최종 중간시점 영상이다. 각각 원본 영상과 비교하면 서의 차이가 나지 않는 것을 알 수 있다.

그림 9. 중간 시점 생성 예 (mobile, lovebird1, cafe, 및 Poznon street) (a) 원본 영상 (b) 결과 영상

Fig. 9 Examples of the intermediate view generation (mobile, lovebird1, cafe, and Poznon street) (a) original and (b) result images

그림 10은 표 1의 네 개의 실험 영상들에 대하여 프레임별로 생성된 중간시점 영상과 원본 중간시점 영상과의 화질을 객관적으로 비교할 수 있는 PSNR(Peak Signal to Noise Ratio)로 나타낸 것이다.

그림 10. 영상화질에 대한 실험결과.

Fig. 10 Experimental results for the image quality.

그림 10의 PSNR 값들의 평균과 중간시점 생성 평균 수행 시간을 표 2에 나타내었다.

표 2. 평균 PSNR과 수행 시간

Table. 2 Average PSNR value and execution time

Test Sequence	PSNR (dB)	Execution time (ms)	
		Pre-Processing	Average
Mobile	33.15	80.5	8.9
Lovebird1	25.92	121.7	14.4
Cafe	30.73	141.6	26.3
Poznan street	32.08	159.1	26.9
Average	30.47	125.7	19.13

실험 영상의 화질 결과가 실험 영상마다 차이가 나는데 영향을 주는 것은 중간시점 생성 영상과 원본 영상과의 배경 부분이 정확하게 일치하는 것인데 이 부분은 가상 시점 생성 방법의 특성상 깊이 영상에 의한 텍스처 영상의 시점 이동이 정확히 원본 배경 부분과 일치하지 않으면 그만큼 화질 결과에 영향을 준다.

본 논문이 제안한 방법은 변위중분의 값을 얻기 위해서 첫 번째 프레임에서만 전처리 과정을 수행하고 나머지 프레임에서는 획득한 변위중분의 값을 사용하여 중간 시점을 생성한다. 표 2에 첫 번째 프레임에서 수행하는 전처리과정의 수행 시간을 나타내며 평균 수행 시간은 첫 번째 프레임의 수행시간을 포함한 모든 프레임의 중간 시점 생성 수행 시간의 평균을 나타낸 것이다. 표 2를 보면 Poznan street 영상이 가장 많은 시간을 소요하고, Mobile 영상이 가장 작은 시간을 소요하였다. 비슷한 해상도를 갖는 Cafe와 Poznan street 영상의 평균 생성 시간이 차이나는 이유는 전처리 과정의 수행 시간 차이인데, Poznan street 영상의 정해진 깊이 각각의 변위를 구하는 스테레오 정합의 수행 시간이 오래 걸리는 것을 볼 수 있다. 우리가 제안한 방법으로 Full HD 해상도의 중간시점 영상을 약 1초에 38 프레임씩 생성할 수 있다.

표 3은 실험 영상을 CPU에서 중간시점을 생성을 했을 때와 CUDA를 이용하여 GPU에서 연산 했을 때의 평균 중간시점 생성 소요 시간을 비교한 것이다. 표 3에서 볼 수 있듯이 약 9배의 개선이 이루어졌음을 알 수 있다.

표 3. CPU와 GPU의 소요 시간 비교

Table. 3 Execution time (CPU vs GPU)

Test Sequence	CPU time (ms)	GPU time (ms)	Ratio
Mobile	69	8.9	1 : 7.8
Lovebird1	121	14.4	1 : 8.4
Cafe	252	26.3	1 : 17.5
Poznan street	274	26.9	1 : 10.2
Average	179	19.13	1 : 9.4

본 논문이 제안한 중간시점 생성의 성능을 알기 위해서 표 4에서 이 전에 있는 GPU를 이용한 중간시점 생성 방법[21]과 비교하였다. 비교한 방법이 사용한 영상의 해상도가 1024X768이기 때문에 본 논문도 같은 해상도로 비교하였다. 표에서 볼 수 있듯이 본 논문이 제안한 방법이 기존에 있는 방법보다 약 10배 정도 빨리 중간시점 영상을 생성하는 것을 알 수 있다.

표 4. 성능 비교

Table. 4 Performance comparison

Resolution	Execution time (ms)	
	H. Shin[20]	proposed method
1024X768	147.945	14.4

V. 결 론

본 논문에서는 GPU를 이용하여 고속으로 좌/우 깊이 영상의 깊이 당 변위의 중분을 계산하고, 이를 이용하여 임의의 중간시점 영상을 고속 생성하는 방법을 제안하였다. 임의의 중간시점 영상을 촬영한 카메라 파라미터들을 사용하여 생성할 수 있지만, 본 논문에서는 그 파라미터들이 주어지지 않았다고 가정하고, 좌/우 깊이 영상에 대해 스테레오 정합을 수행하여 깊이 당 변위중분을 계산하였다. 이 과정에서 본 논문은 CUDA를 이용하여 병렬 처리를 통해 스테레오 정합 연산을 고속화하였다. 그리고 이 과정을 통해 첫 프레임에서 변위중분을 계산하면 그 다음 프레임부터는 이 변위중분을 사용하여 중간시점 영상을 생성하는 것이 가능하다. 변위중분

이 계산되면 이것을 이용하여 좌/우 영상으로부터 원하는 중간 시점의 영상의 각 좌표를 계산하고, 각 텍스처 영상들을 임의의 중간시점 영상으로 생성 가능하다. 그리고 시점 변화에 따른 정보부족으로 생기는 비페색영역들은 두 시점으로 생성한 중간시점 영상으로 서로 상호 보완하여 없앨 수 있다. 그리고 중간시점으로 화소가 이동하면서 생기는 화소 중복을 록업 테이블을 이용한 Z-Buffering 통하여 해결할 수 있다. 그리고 마지막으로 남은 비페색영역들은 홀필링을 수행하여 최종의 중간시점 영상을 생성하게 된다.

제안한 방법을 구현하여 다양한 실험 영상으로 실험한 결과 생성된 중간시점 영상의 평균 화질은 평균적으로 30.47dB의 PSNR 값을 보였으며, 수행속도는 Full HD급 영상을 생성 시 26.3ms 즉, 초당 약 38 프레임의 중간시점 영상을 생성할 수 있었다.

본 논문의 방법이 Full HD급 영상의 중간시점 영상을 실시간으로 생성하는 것이 가능하지만, 다소 생성 영상의 화질이 떨어진 것으로 보이고 있는데, 향후 보다 효과적인 프로그래밍을 연구하여 효과적으로 수행 시간을 최소한으로 유지하면서 생성 영상의 화질이 뛰어난 방법을 연구할 계획이다.

감사의 글

본 연구는 지식경제부, 방송통신위원회 및 한국 산업기술평가관리원의 산업원천기술개발사업(정보통신)의 일환으로 수행하였음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호처리 요소 기술 및 하드웨어 IP 개발]

REFERENCES

[1] Edited by O. Schreer, et al., 3D Video Communications, John Wiley & Sons Ltd., Atrium, England, 2005.
[2] ISO IWA3, Image safety reducing the incidence of undesirable biomedical effects caused by visual image sequences, 2005.
[3] 3D Consortium, "3DC Safety Guideline for Popularization

of Human-friendly 3D," 2006.
[4] Li Yu, et al., "Depth based View Synthesis with Artifacts Removal for FTV," IEEE Intl. Conf. on Image and Graphics, pp. 506-510, 2011.
[5] Chia-Ming Cheng, et al., "Spatio-Temporally Consistent Novel View Synthesis Algorithm from Video-plus-Depth Sequences for Autostereoscopic Displays," IEEE Trans. on Broadcasting, Vol. 57, No. 2, pp. 523-532, June 2011.
[6] S. A. Benton and V. M. Bove, Jr., Holographic Imaging, John Wiley and Sons Inc., Hoboken NJ, 2008.
[7] Reichelt, et al., "Holographic 3-D Displays - Electro-holography within the Grasp of Commercialization", A part of the Book named as Advances in Lasers and Electro Optics, INTECH, pp. 683-710, April 2010.
[8] J. L. Wilson, Microsoft kinect for Xbox 360, PC Mag. Com, Nov. 10, 2010.
[9] SR4000Data Sheet, <http://www.mesa-imaging.ch/prodview4k.php>, MESA Imaging, Oct. 2010.
[10] A. Smolic, et al., "Intermediate View Interpolation based on Multiview Video plus Depth for Advanced 3D Video Systems," IEEE ICIP, pp.2448-2451, 2008.
[11] G. Cheung, et al., "On Dependent Bit Allocation for Multiview Image Coding with Depth-Image-Based Rendering", IEEE Trans. on Image Processing, Vol. 20, No. 11, pp. 3179-3194, Nov. 2011.
[12] S. Zinger, L. Do, and P. H. N. de With, "Free-viewpoint depth image based rendering," J. Vis. Commun. Image Representation, vol. 21, no. 5 - 6, pp. 533 - 541, 2010.
[13] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole filling method using depth based inpainting for view synthesis in free viewpoint television and 3-D video," in Proc. Picture Coding Symp., Chicago, IL, May 2009, pp. 233 - 236.
[14] Jungsik Park, Ji-Youn Choi, In Ryu, and Jong-Il Park, "Universal View Synthesis Unit for Glassless 3DTV," IEEE Transactions on Consumer Electronics, 58(2): 706-711, May 2012
[15] Luat Do, et al., "GPU-accelerated Real-time Free-viewpoint DIBR for 3DTV," IEEE Transactions on Consumer Electronics, 58(2): 633-640, May 2012
[16] L. McMillan, Jr., "An image-based approach to three-dimensional computer graphics," Ph.D. dissertation, University of North Carolina at Chapel Hill, Chapel Hill, NC, 1997.
[17] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," IEEE Trans. Image Process., vol. 13, no. 9, pp. 1200 - 1212,

- 2004.
- [18] I. Daribo, B. Pesquet "Depth-aided image inpainting for novel view synthesis," Multimedia Signal Processing (MMSP), IEEE International Workshop on pp 167 - 170, 2010
- [19] J-M Koo, Y-H Seo, H-J Choi, J-S Yoo, and D-W Kim, "Intermediate Depth Image Generation using Disparity Increment of Stereo Depth Images", Journal of broadcast engineering, Vol. 17. Issue 2, pp. 363-373, March. 2012.
- [20] http://wg11.sc29.org/svn/repos/MPEG-4/test/trunk/3Dview_synthesis/VSRS
- [21] H. Shin, Y. Kim, H. Park, and J. Park, "Fast View Synthesis using GPU for 3D Display", In IEEE Trans. Consumer Electronics, vol. 54, no. 4, pp. 2068-2076, Nov. 2008.

구자명(Ja-Myung Koo)

2012년 2월 안양대학교 정보통신공학과 졸업(공학사)
 2012년 3월 광운대학교 전자재료공학과 석사과정
 ※관심분야 : 3D 영상처리, 디지털 홀로그램, Stereo Vision, 워터마킹

서영호(Young-Ho Seo)

1999년 2월 광운대학교 전자재료공학과 졸업(공학사)
 2001년 2월 광운대학교 일반대학원 졸업(공학석사)
 2004년 8월 광운대학교 일반대학원 졸업(공학박사)
 2005년 9월 ~ 2008년 2월 한성대학교 조교수
 2008년 3월 ~ 현재 광운대학교 교양학부 부교수
 ※관심분야 : 실감미디어, 2D/3D 영상 신호처리, 디지털 홀로그램, SoC 설계

김동욱(Dong-Wook Kim)

1983년 2월 한양대학교 전자공학과 졸업(공학사)
 1985년 2월 한양대학교 공학석사
 1991년 9월 Georgia공과대학 전기공학과(공학박사)
 1992년 3월 ~ 현재 광운대학교 전자재료공학과 정교수
 2009년 3월 ~ 현재 광운대학교 실감미디어 연구소 연구소장
 ※관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP설계, Wireless Communication