# 다시점 및 다중 클러스터 환경에서 네트워크를 이용한 효율적인 실시간 영상 합성 기법

유강수* · 임은천** · 심춘보***

# An Efficient Real-Time Image Reconstruction Scheme using Network in Multiple View and Multiple Cluster Environments

KangSoo You*, Eun-Cheon Lim**, Chun-Bo Sim***

본 연구는 지식경제부 및 정보통신산업진흥원의 대학IT연구센터 지원사업의 연구결과(NIPA-2009-(C1090-0801-0047)) 및 지식경제부 및 정보통신산업진흥원의 지원을 받아 수행된 연구결과임.(09-기반, 산업원천기술개발사업)

## 요 약

네트워크를 기반으로 하나의 클러스터가 4개의 카메라로 구성된 4개의 다중 클러스터로부터 2D 영상을 조합하여 3D 입체 영상을 생성하는 알고리즘 및 시스템을 제안한다. 제안하는 기법은 다중 클러스터 환경에서 동작하고 실시간 대용량의 데이터 처리로 인한 시스템의 부하를 분산시키기 위해 네트워크를 이용한 서버-클라이언트 구조를 가진다. 아울러 성능 향상을 고려해 JPEG 압축과 램 디스크 방식을 적용한다. 4채널 16개의 카메라로부터 입력되는 입력 영상에 대해서 이진화 영상을 구하고, Sobel 및 Prewitt 등의 에지 검출 알고리즘을 적용시킨 후 영상들 간의 시차를 구한 후에 3D 입체 영상을 생성한다. 성능 분석 결과, 클라이언트에서 서버로 전송하는 전송시간은 약 0.05초가 소요되며, 4채널 16개의 카메라로부터 2D 영상을 조합하여 3D 입체 영상을 생성하는 알고리즘에 소요되는 시간은 약 0.84초가 소요된다. 이를 통해 실시간으로 다시점 및 다중 클러스터 환경에서 3D 입체 영상을 생성하는 효율적인 시스템임을 확인할 수 있었다.

## ABSTRACT

We propose an algorithm and system which generates 3D stereo image by composition of 2D image from 4 multiple clusters which 1 cluster was composed of 4 multiple cameras based on network. Proposed Schemes have a network-based client-server architecture for load balancing of system caused to process a large amounts of data with real-time as well as multiple cluster environments. In addition, we make use of JPEG compression and RAM disk method for better performance. Our scheme first converts input images from 4 channel, 16 cameras to binary image. And then we generate 3D stereo images after applying edge detection algorithm such as Sobel algorithm and Prewiit algorithm used to get disparities from images of 16 multiple cameras. With respect of performance results, the proposed scheme takes about 0.05 sec. to transfer image from client to server as well as 0.84 to generate 3D stereo images after composing 2D images from 16 multiple cameras. We finally confirm that our scheme is efficient to generate 3D stereo images in multiple view and multiple clusters environments with real-time.

키워드

QR코드 해석기, 트리즈, u-맵, 유비쿼터스 응용, u-팜플렛

Key word

Multiple View, Multiple Cluster, 3D Image Reconstruction, Disparity

* 전주대학교 교양학부 교수
** 순천대학교 멀티미디어공학과 석사과정
*** 순천대학교 멀티미디어공학과 조교수(교신저자)

접수일자 : 2009. 08. 25
심사완료일자 : 2009. 09. 08

## Ⅰ. 서 론

3차원(이하, 3D) 입체 방식의 화면 출력 기술이 발전하게 된 것은 보다 더 현실적이고 입체감 있는 영상을 보기 위한 사용자의 꾸준한 요구가 있었기 때문이다. 그리고 이는 게임, 에니메이션, 방송, 건축, 설계, 군사, 항공, 천문, 의료 등 다양한 응용 분야[1]에 활용되기 시작했다. 대부분의 첨단 기술의 시작이 군사용에서 시작된 것처럼 3D 시뮬레이션 시스템의 시초는 1940년의 MIT의 WhirlWind로 기록되어 있다. 이후 40 여년이 지난 1980년대부터 3D 데이터 처리의 활용은 다양한 분야로 파급되게 되었고, 특히 게임 분야는 일반 사용자에게 3D를 가장 널리 알리게 된 계기가 되었다. 게임의 3D 영상은 크게 하드웨어와 소프트웨어로 나뉘어서 생성되게 된다. 즉, 하드웨어 쪽에서 3D 가속기를 통해서 3D 영상을 생성하게 되고, 이 3D 가속기를 다루기 위한 API를 OpenGL[2], DirectX[3]를 통해 제공받고 있다. 그러나 기존의 2D 출력 장치를 사용한 3D 영상 기술은 입체감을 뛰어나게 표현하지 못하기 때문에 입체감 및 현실감을 더욱 증가시키기 위한 3D 영상 출력을 위한 표시 장치 기술과 알고리즘 개발이 진행 중이다[4].

3D 영상 출력을 위한 과정은 크게 영상 획득, 획득 영상의 모델링, 인코딩 방식, 전송, 3D 영상 전환, 3D 표시 장치의 6가지 과정으로 나뉘며[5-7]. 획득하는 카메라의 방식에 따라서 크게 병렬(parallel) 방식, 컨버전스(convergence) 방식, 짝 컨버전스 방식 등이 있다[8-9]. 3D 영상을 생성하는 카메라는 사람의 눈의 역할을 하며 좌 영상과 우 영상의 시차(disparity)를 통해 새로운 3D 영상을 생성한다. 이 영상을 획득하는 과정에서 높은 대역폭이 필요하기 때문에 대부분의 카메라는 IEEE 1394와 같은 높은 대역폭을 가지는 인터페이스와 연결하여 구현되게 된다[10]. 조금 더 현실적인 화면을 얻기 위해서 오차를 보정한 알고리즘이 사용되거나, 대역폭을 줄이기 위해서 시차 예측을 수행하여 중간 영상을 생성한다[11]. 대량의 대역폭을 차지하는 영상을 바로 사용하는 것은 비효율적이기 때문에 데이터를 압축해서 사용하게 된다. 압축된 데이터를 수신한 후에 중간 영상(IVR, Intermediate View Reconstruction)을 생성한다. 이 때 영상 처리 기술이 사용되며 이 과정을 통해서 시차 정보를 얻을 수 있다. 큰 규모의 영상을 위해서는 서버/클라이언트 구축, 압축 코덱의 구현이 요구된다. 비디오 스트리밍 시스템은 보통 UDP 기반의 RTP 프로토콜을 사용하며[12], DSS, GPAC, VideoLAN, Apple QuickTime 등이 알려져 있다.

본 논문에서는 네트워크를 기반으로 한 클러스터가 4개의 카메라로 구성된 4개의 다중 클러스터로부터 2D 영상을 조합하여 3D 입체 영상을 생성하는 알고리즘 및 시스템을 제안한다. 제안하는 기법은 다중 클러스터 환경에서 동작하고 실시간 대용량의 데이터 처리로 인한 시스템의 부하를 분산시키기 위해 네트워크를 이용한 서버-클라이언트 구조를 가진다. 아울러 성능 향상을 고려해 JPEG 압축과 램 디스크 방식을 적용한다. 4채널 16개의 카메라로부터 입력되는 입력 영상에 대해서 이진화 영상을 구하고, Sobel 및 Prewitt 알고리즘을 적용시킨 후 영상들 간의 시차를 구한 후에 3D 영상을 생성한다.

본 논문의 구성은 다음과 같다. 2장에서는 연구의 배경 및 관련 연구를 살펴보고 3장에서는 제안하는 다시점 및 다중 클러스터 환경에서 네트워크를 이용한 효율적인 실시간 영상 합성 기법을 기술한다. 4장에서는 구현 결과 및 성능평가에 대해서 소개하고 마지막으로 5장에서 결론 및 향후 연구를 제시한다.

## Ⅱ. 관련연구

본 논문과 관련된 배경 연구로 3D 브라우징 시스템의 구성요소인 카메라 구성 방식과 영상 보정 방법에 대해서 소개한다. 먼저, 3D 브라우징 시스템의 입력 장치에 해당하는 다시점 카메라 구성 방식은 양안 카메라(Stereo Camera) 방식을 사용하며, 이 방식은 크게 교차축 방식과 수평식 방식으로 나뉜다. 먼저, 교차축 방식은 관심물체에 대해 모든 카메라의 광축을 회전시켜 한 점에서 수렴하도록 하고, 각 카메라가 물체로부터 동일한 거리만큼 떨어지도록 설계하며, 카메라를 평행하게 설치하고 관심물체에 수렴점을 형성하기 위해서 카메라의 렌즈의 위치를 수평이동 시키는 방식이다[11]. 교차축 방식의 경우에 영상들 사이에서 오차가 발생하게 되어 왜곡된 입체 영상을 만들기 때문에 반드시 이를 보정해야하기 때문에 일반적인 3D 콘텐츠 제작에서는 전혀 사용하지 않는다. 이와 달리 수평식 방식은 카메라의 렌즈 위치가 수평 이동되어 오차를 수정하기 때문에 영상 왜곡이 적기 때문에 대부분의 3D 콘텐츠 제작에 사용된

다. 그러나 카메라 수가 늘어날수록 넓은 범위를 볼 수 있는 렌즈를 사용해야 한다는 단점이 존재한다. 참고문헌 6에서는 다중 뷰 방식의 3D 비디오 영상을 표현하기 위한 합성 알고리즘을 제안하고 있다. 이 방식은 다수의 카메라를 통해 얻은 시차를 적용하여 입체감을 얻을 수 있는 방식이나 전체적으로 시스템 구성 및 촬영, 전송 등에서 효율적이지 못하다.

다음으로 영상 보정 방법은 카메라가 위치한 장소에서 바라보는 시점에 따라 물체 자체의 색상, 밝기, 균일도가 달라지며, 카메라와의 거리 차이로 인한 왜곡 현상, 카메라 오차, 미세한 크기 차이, 카메라 자체의 밝기, 색상 차이 문제로 인해 일반적으로 카메라로부터 수신한 데이터는 직접 사용할 수 없다. 따라서 이러한 문제를 해결하기 위해 두 가지 방법을 이용한다. 첫 번째 방법은 차이가 발생할 수 있는 단계별로 보정을 적용하고, 최종 영상을 합성하는 방법이다. 이 방법은 정확하게 원하는 데이터를 얻을 수 있는 장점이 있지만, 많은 양의 데이터를 위한 큰 대역폭이 요구되며, 현재 시점이 아닌 이전 시점의 데이터는 필요가 없는 실시간적인 응용에서는 적용하기 어렵다. 그래서 각 단계 중에 최종 영상에 영향을 크게 줄 수 있고, 연산량이 적은 연산만 수행하여 최적화를 수행해야 한다. 두 번째 방법은 중간에 보정 작업을 수행하지 않고, 영상에 대해서 이진화를 수행한 후에 차이값을 계산하고, 이를 각 채널에서 얻은 영상에 적용하는 방법이다. 이는 첫 번째 방법에 비해서 연산량이 거의 없지만, 정확도와 화질 면에서 문제가 생긴다.

## Ⅲ. 제안하는 기법

### 3.1 전체 시스템 구조

본 논문에서 제안하는 다시점 및 다중 클러스터 기반의 실시간 영상 합성 기법에 대한 전체 시스템 구성은 그림 1과 같다. 카메라는 4채널 즉, 4개의 카메라를 묶어 하나의 클러스터를 구성한다. 하나의 클러스터는 영상 데이터를 수신하여 시스템에서 정의한 프로토콜에 따라 네트워크를 통해 영상을 전송한다. 영상은 정의된 프로토콜에 의해 패킷을 분할하고, 영상을 조합하여 3D 장치에 전송한다. 카메라는 그림 1과 같이 수평 지지대를 이용하여 하나의 PC에 2대씩 IEEE 1394 호스트 카드를 사용하여 연결되어 있고 전체적으로 본 논문에서는 4개의 클러스터를 이용하고 있으며, 4대의 PC는 1G급의 스위치 허브에 연결되어 서버로 전송된다.

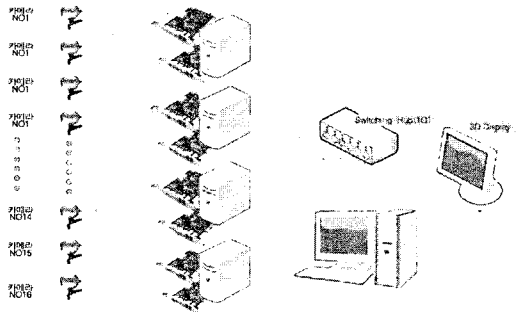


그림 1. 전체 시스템 구조
Fig. 1. Overall system architecture

한 클러스터마다 영상을 입력받은 후에 JPEG 압축을 수행한다. 카메라로부터의 입력 영상은 압축 전 상태이기 때문에 대역폭을 그대로 차지하게 된다. 그러므로 클러스터 당 최대 대역폭은 식 (1)을 통해 계산된다.

$$TB() = pixels \bullet bits \bullet FPS \quad \text{...(식 1)}$$

최대 대역폭이 400 MBps이므로 이를 통해 초당 프레임 수를 구할 수 있다. 최대 프레임 수는 클러스터 당 14.2 FPS이다. 영상 획득 시의 대역폭과 별도로 네트워크를 통해 데이터를 전송해야 한다. 이 때 JPEG 압축을 통해서 대역폭을 많이 줄일 수 있다. 생성된 압축 이미지는 메모리상에 참조 형태로 존재하게 되면 동기화 문제가 발생하므로 이를 위해 램 디스크(RAM disk) 방식의 파일 형태로 저장한다. 영상 데이터는 순차적으로 전송하게 되며, 클러스터마다 번호를 부여하고 이 번호에서 클러스터 내 채널 번호를 할당한다. 클러스터 번호는 기계 번호라고 부르며, 영상 데이터가 순차적으로 전송되는 것이 보장되어야 차후 3D 입체 영상 합성 알고리즘에서 데이터 동기화를 획득할 수 있다. 이와 별도로 한 시점의 데이터를 다른 시점의 데이터와 구분하기 위해 버전(Version) 개념을 도입하여 적용하고 있으며, 계속해서 생성되는 영상에 대해서 버전 번호(Version Number)를 부여하지 않게 되면, 알고리즘 적용 시에 현재의 상태를 담고 있는 16개의 채널 상의 영상의 동기를 보장할 수 없게 된다. 그림 2는 전체 데이터 흐름도를 나타낸다.

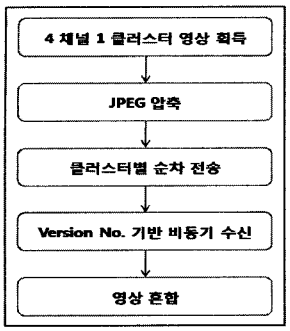



그림 2. 전체 데이터 흐름도
Fig. 2. Data flow-work

### 3.2 서버 네트워크 구조

제안하는 시스템의 네트워크 서버는 이벤트 기반으로 구현되어 있으며, 연결 생성, 연결 소켓에 대한 정보 검색을 위한 맵 관리, 동기 읽기/쓰기, 비동기 읽기/쓰기, 작업 쓰레드 생성 및 할당, 버퍼 할당, 자원 해제 등의 기능을 가지고 있다. 연결 생성 후에 이벤트 방식의 처리를 위해서 연결 소스에는 단일 이벤트 핸들러를 추가할 수 있다. 단일 이벤트 핸들러 내부는 상태에 따라서 처리 로직이 바뀌게 된다. 아울러 작업자 수, 연결 이벤트 핸들러, 지연 읽기 횟수, 미리 준비된 연결 정보 저장 Context의 수 등의 속성 정보를 설정할 수 있도록 구현되어 있다. 서버는 그림 3과 같은 단계에 따라서 구동된다.

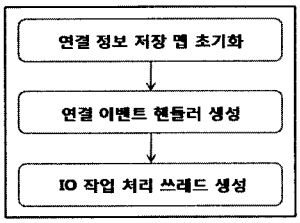



그림 3. 서버 측 네트워크 생성
Fig. 3. Server-side network generation

연결 정보 저장 맵은 해시 테이블로 단일 패킷을 표현하는 ClientContext를 저장하게 된다. 즉 ClientContext는 소켓 객체와 동기화 객체, 전송이 지연된 횟수, 패킷 순서 등을 저장하며, 파일을 전송할 경우에는 추가적으로 파일에 대한 연결점과 전체 파일 크기, 파일의 현재 크기, 파일이 전송 중인지 송신 중인지를 나타내는 플래그 등의 값을 가진다. 데이터를 수신했다는 이벤트를 다른 곳으로 전파하는 이벤트 소스가 생성되게 되면, 이 이벤트 소스에 대한 리스너를 생성하게 된다. 이벤트 자체가 동기화 방식이 아닌 비동기 방식으로 겹쳐서 발생하기 때문에 단순한 I/O가 아닌 겹쳐진 I/O를 처리하도록 버퍼와 EventSelector를 생성하여 이벤트 리스너를 구현하고 있다. 리스너 쓰레드는 서버 소켓이 중지되지 않은 동안 계속 동작하게 되며, EventSelector에 의해 다중으로 발생하는 이벤트 중에서 Accept 이벤트만을 수신하여 동작하게 된다. 아울러 I/O 작업 처리를 위한 쓰레드를 생성하는 단계에서는 쓰레드를 정지 상태로 생성해 두고, 차후에 쓰레드 생성에 소요되는 시간을 줄이고 성능을 높이도록 설계한다. 속성으로 정의된 최대 쓰레드 수만큼 쓰레드를 생성하고 쓰레드 풀에 추가하게 된다. 다음으로 서버는 그림 4와 같은 단계로 데이터를 처리하게 된다. 클라이언트에 대해 영상 데이터를 요청하면 비동기적으로 데이터 수신이 완료되며, 완료이벤트를 받은 후에 영상 데이터를 비트맵으로 다시 복원한다. 복원 후에 3D 변환 알고리즘을 적용한 후, 3D 전용 출력 장치로 전송한다.

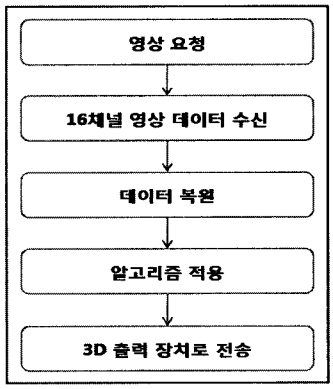



그림 4. 서버 측 데이터 처리 과정
Fig. 4. Server-side data process

### 3.3 클라이언트 네트워크 구조

클라이언트에서는 서버로의 접속을 안정적으로 유지하기 위해 단일 소켓이 아닌 채널수만큼의 소켓을 생성한다. 소켓이 생성되면 이벤트가 발생하게 되고, 이벤트를 수신하여 ServerContext를 ContextMap에 추가하게 된다. 동일한 서버에 대한 정보를 채널수만큼 가진다고 볼 수 있지만, 동일한 서버 정보이더라도 서로 전혀 다른 Context이기 때문에 Context Map을 사용하지 않고 단일 참조에 ServerContext를 저장할 수도 있는데, 이러한 방

식은 다수의 쓰레드에서의 경쟁조건으로 인해 클라이언트 측 처리 부하가 높아지게 된다. 서버에서 영상에 대한 요청을 하게 되면 기존에 저장한 ServerContext를 통해 서버와 통신하게 되고, 이 때 패킷을 생성하게 되며 ServerContext는 GUI 쓰레드와 로직 쓰레드가 동시에 접근할 수 있기 때문에 동기화 락(Lock)을 통해 동시 접근을 막고 사용 후에 락을 해제한다. 그림 5는 클라이언트 측 데이터 처리 과정을 나타낸다.

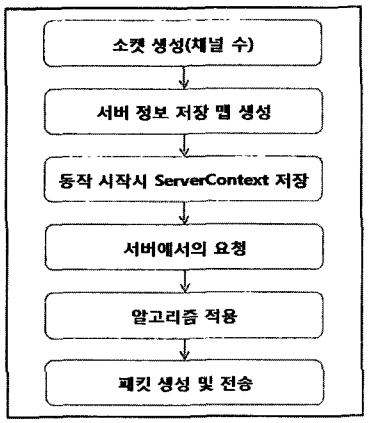



그림 5. 클라이언트 측 데이터 처리 과정
Fig. 5. Client-side data process

클라이언트는 서버와 연결이 되고 서버에서 명시적인 전송 요청이 있고 난 후에 데이터를 전송한다. 이것은 하드웨어 트리거(Trigger)가 생성한 이벤트로 인해 소프트웨어 Context에서 발생하는 쓰레드 경쟁 상황을 피하기 위함이며 서버에서 동기화에 대한 권한을 가지고 락을 조정하기 때문에 동기화를 통해 높은 수준의 격리성을 얻을 수 있는 장점을 가진다. 4채널 영상 데이터를 구분하기 위해서 기계 번호와 채널 번호를 패킷 헤더에 추가하게 되며, 버전을 구분하기 위해서 버전 번호를 추가한다. 이 때 버전은 무한정 커질 필요가 없기 때문에 식 (2)을 통해 버전 번호 최대값을 결정한다.

$$2^n \fallingdotseq fw(TB(), ps)/ch \quad ...(식\ 2)$$

n은 최대값이며, TB()는 최대 대역폭, ps는 패킷 사이즈이고, ch는 채널 수이다. 이를 계산하게 되면 32를 얻을 수 있다. 버전 번호는 32를 넘지 못하기 때문에 32 이상이 되면 0이 되어야 하므로 모듈러 연산을 필요로 한다. 그러나 모듈러 연산을 통해 버전 번호 증가에 대한 락을 보장할 수 없기 때문에 내부 락을 위해 모듈러 연산 대신에 락 지원 연산을 사용한다. 채널 번호 역시 마찬가지로 증가시킨다.

### 3.4 네트워크 패킷 구조

명령 패킷 외에 영상 데이터를 전송하기 위한 패킷이 있다. 초기 영상 데이터 패킷은 그림 6과 같이 세 부분으로 구성된다.

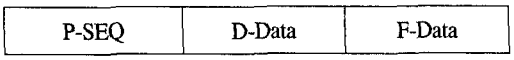



그림 6. 초기 영상 데이터 패킷 구조
Fig. 6. Packet structure of initial image data

P-SEQ 필드는 패킷의 순서를 정렬시키기 위해서 존재한다. P-SEQ는 ClientContext 내에서 유일한 값을 가지기 때문에 동일한 패킷 번호를 가지고 있다고 하더라도 서버에서 겹치지 않는다. 패킷이 비동기 형태로 전송되기 때문에 이를 수신하는 측에서 순서를 정렬하여 데이터를 조합하려면 이러한 순서 필드가 필요하다. D-Data 필드는 Display Data의 약자로 3D 데이터를 표현하는 데이터 필드이다. 가장 앞 부분의 V-SEQ는 Display Data가 동일한 영상에 대한 것인지에 대한 구분을 위해서 존재하며, 동일한 시점에 최소한 채널 개수만큼 동일한 값을 가질 수 있다. 왜냐하면 패킷이 전송되다가 잘리게 되면 P-SEQ는 증가되지만 V-SEQ는 그대로 현재 값을 가지게 되기 때문이다. 영상 데이터가 생성된 기기와 채널에 대한 정보와 알고리즘을 통해서 얻은 영상간의 시차(Disparity) 값을 전달하는 역할을 한다. 여기서 시차(Disparity)란 두 채널의 영상 사이의 차이점을 의미한다. 이는 가상의 3D 가상 세계 좌표를 일치시키기 위해 사용한다.

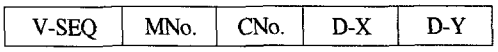



그림 7. Display Data(D-Data) 필드 세부 구조
Fig. 7. Detailed structure of Display Data field

File Data 필드는 그림 8과 같다. 영상은 JPEG으로 압축이 된 상태로 램 디스크에서 복사되고 전송되어야 하기 때문에 파일명을 포함하고 있다. 아울러 File 전송 시

동시에 전체 바이너리를 전송하는 것이 아니라 겹친 I/O를 통해서 조금씩 더 진행된 바이너리를 전송하기 때문에 동기화 문제가 발생할 수 있다. 따라서 현재까지 전송한 데이터 양과 목적 데이터 양이 다른 쓰레드에서 읽히는 것이 아니라 현재 받은 패킷을 통해 읽히도록 하여 동기화 문제를 방지한다. 또한 파일은 읽기와 쓰기 모드가 있으며, 클라이언트 입장에서는 쓰기 모드이고, 이를 보수화 하여 읽기 모드로 연산하면 된다.

| F-Name | C-FS | T-FS | F-Mode |
|---|---|---|---|

그림 8. File Data(F-Data) 필드 세부 구조
Fig. 8. Detailed structure of File Data field

마지막으로 각 영상 데이터 사이의 패킷은 D-Data를 동시에 보낼 필요가 없기 때문에 그림 9와 같이 P-SEQ와 파일 데이터만 가진 패킷을 보내면 된다. 서버는 위의 두 패킷을 조합하여 전체 3D 영상을 생성하게 된다.

| P-SEQ | F-Data |
|---|---|

그림 9. 영상 데이터 사이의 패킷 구조
Fig. 9. Packet structure between image data

### 3.5 3D 변환 알고리즘

알고리즘은 연산 부하를 줄이기 위해 클라이언트 쪽과 서버 측에서 공통적으로 그림 10과 같은 연산 과정을 거친다. 아울러 정확한 구현을 위해서는 깊이 테스트가 필요하지만, 성능을 위해 깊이에 대한 좌표가 같다고 가정하고 구현한다. 클라이언트에서는 데이터를 모두 처리하고 압축하여 보내게 된다. 서버에서 처리하게 하면 부하가 매우 크기 때문이다.

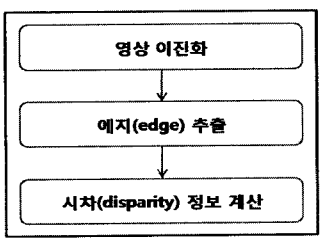



그림 10. 공통 처리 알고리즘
Fig. 10. Common algorithm for 3D reconstruction

클라이언트 측에서 알고리즘을 적용한 후에 이 데이터를 바탕으로 패킷을 생성하여 전송하게 된다. 서버에서는 추가적으로 다음과 같은 처리를 하게 된다.

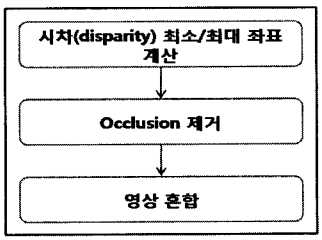



그림 11. 서버 측 추가 적용 알고리즘
Fig. 11. Server-side additional algorithm

단순히 단일 클러스터에 대한 시차가 아닌 클러스터간 시차를 계산하게 된다. 식 (3)에 의해서 최소, 최대 시차값을 계산하고 Occlusion 제거를 위한 좌표를 얻게 된다.

$$mPos(x,y) = (\min(|dispX(n)|\frac{\max(X)}{n=0}), \min(|dispY(m)|\frac{\max(Y)}{m=0})) \quad \text{...(식 3)}$$

$$MSize(x,y) = (\max(|dispX(n)|\frac{\max(X)}{n=0}), \max(|dispY(m)|\frac{\max(Y)}{m=0})) - mPos(x,y) \quad \text{...(식 4)}$$

식 (3)과 식 (4)는 정확한 좌표를 계산할 수 있지만, 렌즈나 빛에 의한 왜곡이 발생할 경우 화면이 축소, 확대를 반복하게 되기 때문에 이를 막기 위해서 최대 좌표는 몇 번의 테스트 영상을 거쳐 최대값 중에 가장 작은 좌표로 값을 고정하여 사용해야 한다. 계산된 값을 통해 영상을 자르게 되면, 모든 채널의 데이터는 상하 좌우의 잘려진 부분이 없게 된다[13].

## Ⅳ. 구현 및 성능평가

### 4.1 구현 환경 및 결과

제안하는 시스템은 Windows XP에서 .NET Visual Studio 2005를 사용하여 C++ 및 Windows 32 API와 MFC를 이용하여 구현하였다. 아울러 3D 출력 장치는 3840*2400의 해상도와 22.2″의 크기를 가지고 있는 IBM

T221 모델을 사용했으며, 16개의 IMITech의 IMC-40FT 카메라를 사용했다. 시스템에서 사용하는 데이터는 하드 디스크가 아닌 램 디스크(RAM Disk)에 저장되며, RAM은 DDR 333 메모리를 사용했으며, Frequency가 166Mhz, 2의 Data Rate, 64비트의 버스를 가지고 있다. 그림 12는 IEEE 1394 호스트 카드 8개를 이용하여 4개의 PC에 각각 4개의 카메라를 한 클러스터로 4개의 클러스터를 구성한 것이다. 카메라는 그림 12와 같이 수평 지지대에 위치하여 일정한 간격과 수평 위치를 유지하도록 하였다. 수평 거리가 커지게 되면 알고리즘에서 검색 윈도우가 늦게 일치점을 발견하기 때문에 속도는 더 떨어지게 된다. 주 컴퓨터는 여러 클러스터들이 연결된 스위칭 허브와 연결되었으며, 특히 3D 영상을 출력하기 위한 3D 출력 모니터와 연결되어 있다.

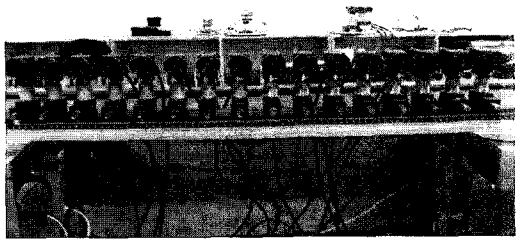


그림 12. 16개 영상 카메라(IMC-40FT 모델)
Fig. 12. 16 image camera(IMC-40FT model)

그림 13. 4개 클러스터(16개 카메라)를 통해 획득한 영상의 예
Fig. 13. Example of acquisition image with 4 clusters(16 camera)

각 영상은 서버에서 요청이 온 후에 카메라로부터 획득을 시작하여 서버로 전송하게 된다. 클라이언트는 영상이 전송되기 전까지는 카메라에서 생성되는 신호를 수신하지 않고 무시한다. 에지 검출 알고리즘으로는 Prewitt 필터와 Sobel 필터 알고리즘을 사용하였으며, 그림 13은 4개의 클러스터 즉, 16개의 카메라로부터 동시에 획득한 샘플 영상이며, 에지 검출 알고리즘을 적용한 결과 화면이다.

서버 시스템은 그림 14와 같은 사용자 인터페이스(GUI)를 가지고 있다. 최상단은 원본이미지이며, 두 번째 영상은 원본 훼손을 막기 위한 복사본 이미지 이며, 세 번째는 이진화 영상, 네 번째는 에지 추출된 영상이다. 다섯 번째 영상은 Disparity에 맞게 Translate 한 영상이며, 여섯 번째 영상은 Occlusion 제거 영상이다. 값을 고정했기 때문에 간혹 약간씩의 Occlusion이 발생하지만, 주 시각 지점이 아니고, 여러 채널이 존재하기 때문에 큰 문제는 발생하지 않는다. 이 화면에서 전체화면 버튼을 클릭하면 마지막 단계인 3D 조합 알고리즘을 거쳐 그림 15의 오른쪽 모니터와 같이 3D 입체 영상을 확인할 수 있다.

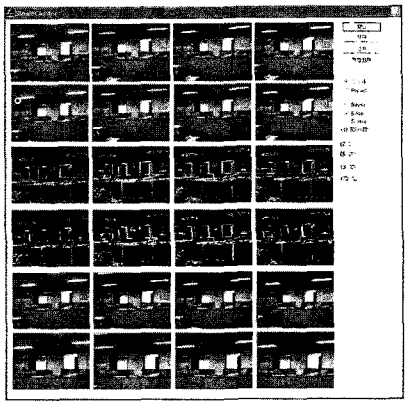


그림 14. 서버 시스템의 사용자 GUI
Fig. 14. User GUI of server system

그림 15의 오른쪽 모니터는 본 논문에서 사용한 3D 출력 장치인 IBM T221 모델로 3840*2400 해상도를 가지고 있으며, 눈에 보일 정도로 빗금이 존재한다. 이 빗금은 혼합된 각 채널 영상을 표현하며 채널 영상이 16개이기 때문에 자세히 보면 세로 방향으로 16개의 줄을 볼 수 있다. 3D 표시 장치에서 확인해 보면 렌티큘러 렌즈

를 통해 보게 되기 때문에 관찰자가 옆으로 이동하게 되면 다른 쪽 빗금 쳐진 영상은 안 보이게 되어 렌즈에서 16 채널 중 한 채널의 영상만 보이게 된다. 그림에서 확인할 수 있듯이 일반 영상과 3D 영상은 다르게 보인다. 일반 모니터에서 보는 화면은 뭉개진 영상처럼 보이게 되지만, 3D 영상에서는 한 시점에는 하나의 영상만 보이도록 구현이 된다.

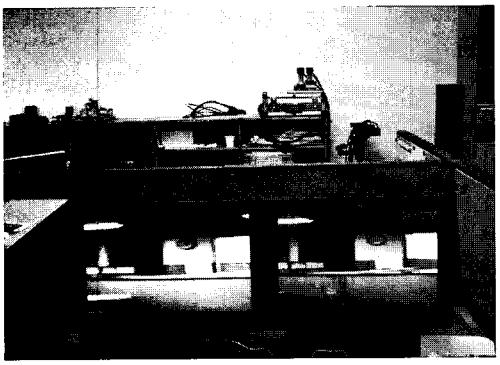


그림 15. 3D 전용 모니터의 결과 화면
Fig. 15. Result of 3D monitor

**4.2** 성능평가

서버에서의 데이터 처리 시간은 하위 클러스터와 동일하다. 물론 하위 클러스터가 증가되면 네트워크 소모 시간은 더 증가하게 된다. 그러나 그림 16에서와 같이 16 채널에서의 네트워크 소요 시간은 많이 소요되지 않는다. 여기서 단일 클러스터의 평균 처리 시간은 약 0.84초이다. 대부분의 처리 시간은 시차를 계산하는 데 소요된다. 평균적으로 전체 시간의 약 70% 정도를 차지하고 있음을 알 수 있다.

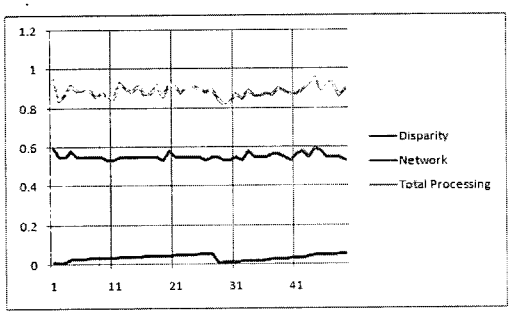



그림 16. 전체 프로세스 소요시간(초)
Fig. 16. Whole process elapsed time(sec.)

네트워크를 통해서 영상 데이터를 수신할 때, 초기에는 소모 시간이 적었지만, 뒤로 갈수록 더 많은 시간을 소모하게 된다. 그러나 알고리즘에서 처리하는 양과 균형을 이루게 되면 특정 값으로 수렴하게 된다. 전송 시간의 평균값은 16개 채널에서 약 0.05초이다. 이 시간을 뺀 나머지 시간이 4채널 16개의 카메라로부터의 영상을 조합하여 3D 영상을 생성하는 알고리즘에서 소요하는 시간이다. 이는 평균 0.84초이며, 이 시간에는 영상 처리를 통합하지 않고, 테스트를 위해 분산하여 처리했기 때문에 생기는 0.07초 정도의 오버헤드가 포함되어 있다.

## V. 결 론

본 논문에서는 네트워크를 기반으로 한 클러스터가 4개의 카메라로 구성된 4개의 다중 클러스터로부터 2D 영상을 조합하여 3D 입체 영상을 생성하는 알고리즘 및 시스템을 제안하였다. 제안하는 기법은 다중 클러스터 환경에서 동작하고 실시간 대용량의 데이터 처리로 인한 시스템의 부하를 분산시키기 위해 네트워크를 이용한 서버-클라이언트 구조를 가진다. 아울러 성능 향상을 고려해 JPEG 압축과 램 디스크 방식을 적용한다. 4채널 16개의 카메라로부터 입력되는 입력 영상에 대해서 이진화 영상을 구하고, Sobel 및 Prewitt 알고리즘을 적용시킨 후 영상들 간의 시차를 구한 후에 3D 영상을 생성한다. 성능 분석 결과, 클라이언트에서 서버로 전송하는 전송시간은 약 0.05초가 소요되며, 4채널 16개의 카메라로부터 2D 영상을 조합하여 3D 입체 영상을 생성하는 알고리즘에 소요되는 시간은 약 0.84초가 소요된다. 이를 통해 실시간으로 다시점 및 다중 클러스터 환경에서 3D 입체 영상을 생성하는 효율적인 시스템임을 확인할 수 있었다.

알고리즘의 수행 시간에서 가장 큰 비중을 차지하는 것은 3D 영상 조합 메서드와 시차 계산 메서드이기 때문에 이 시간을 단축할 수 있는 새로운 알고리즘이나 하드웨어 개발에 대한 연구가 필요하다. 아울러 보다 현실적인 3D 영상을 얻기 위해서는 영상 내에 있는 객체들의 깊이를 계산하여, 관찰자로 부터의 거리를 고려하여 이미지 축소 및 음영 조정을 수정해야 할 것이다.

## 참고문헌


[1] 오길록 외, 3차원 입체영상(3DTV) 방송중계 시범서비스, 정보통신부, pp.2-7, 2002.

[2] 이환용, 백낙훈, "OpenGL을 이용한 OpenGL ES 1.1 구현", 한국정보처리학회논문지, 제16권, 제3호, pp. 159-168, 2009.

[3] 정선태, "DirectX 기반 다채널 영상 감시 시스템 구현 및 성능 평가", 한국콘텐츠학회논문지, 제5권, 제1호, pp. 217-227, 2005.

[4] S. Laveau, and O.D. Faugeras, "3-D Scene Representation as a Collection of Images", Proc. of Int'l. Conf. on Pattern Recognition, pp.689-691, 1994.

[5] 조숙희, 윤국진, 서정일, 안충현, 이수인, "지상파 DMB 기반 3차원 AV 서비스 시스템", Telecommunication Review, 제14권, 제4호, pp.652-665, 2004.

[6] J.-R. Ohm and K. Muller, "Incomplete 3D-Multiview representation of video objects", IEEE Transaction on Circuits and Systems for Video Technology, special issue on SNHC, pp. 389-400, 1999.

[7] Klaus Hope, "An Autostereoscopic Display Providing Comfortable Viewing Conditions and a High Degree of Telepresence", IEEE Ttrans. on Circuits and Systems for Video Technology, Vol.10, No.3, pp.359-365, 2000.

[8] Kiyohide Satoh, Itaru Kitahara, and Yuichi Ohta, "3D Image Display with Motion Parallax by Camera Matrix Stereo", IEEE Proc. of MULTIMEDIA '96, pp.349-357, 1996.

[9] 허경무, 박영빈, "A Viewpint-Dependent Autoster-eoscopic 3D Display Method", Proc. of ISIE 2001, 2001.

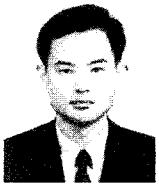
[10] O. Schreer, N. Brandenburg, and P. Kauff, "Real-Time Disparity Analysis for Applications in Immersive Tele-Conference Scenarios - A Comparative Study", Proc. of Int'l Conf. on ICIAP 01, pp.346-351, 2001.

[11] 이광순, 김형남, 허남호, 안충현, "평행축 스테레오 카메라에서의 주시각 제어 방법", 한국방송공학회 학술대회 논문집, pp.297-301, 2002.

[12] 오재학, 차호정, 최영근, "스트리밍 미디어 캐쉬 서버를 위한 RTSP/RTP 스트림 제어 기법", 한국정보과학회논문지:컴퓨팅의 실제, 제9권,제3호, 2003.

[13] Ioannis Pitas, "Digital Image Processing Algorithms and applications", Wiley-IEEE, 2000.


## 저자소개

**유 강 수(KangSoo You)**

1991년 전북대학교 컴퓨터공학과 (공학사)
1994년 전북대학교 컴퓨터공학과 (공학석사)
2005년 전북대학교 영상공학과(공학박사)
2006년~현재 전주대학교 교양학부 교수
※관심분야 : 영상처리, 컴퓨터 비전, 멀티미디어 시스템

**임 은 천(Eun-Cheon Lim)**

2007년 순천대학교 정보통신공학부(공학사)
2007년~2009년 순천대학교 멀티미디어공학과(공학석사)
※관심분야 : 멀티미디어 정보검색, 멀티미디어 시스템

**심 춘 보(Chun-Bo Sim)**

1996년 전북대학교 컴퓨터공학과 (공학사)
1998년 전북대학교 컴퓨터공학과 (공학석사)
2003년 전북대학교 컴퓨터공학과(공학박사)
2005년~현재 순천대학교 멀티미디어공학과 조교수
※관심분야 : 멀티미디어 DB & IR, 멀티미디어 시스템, 멀티미디어 응용, 유비쿼터스 컴퓨팅