타임 스탬프를 이용한 타일드 디스플레이 기록기의 타일 영상 병합 알고리즘

(A Tile-Image Merging Algorithm of Tiled-Display Recorder using Time-stamp)

최 기 석 † 낭 종 호 ††
(Giseok Choe) (Jongho Nang)

요 약 타일드-디스플레이 시스템은 다수의 디스플레이 디바이스를 그리드 형태로 연결하여 큰 화면과 높은 해상도를 제공해줄 수 있는 시스템이다. 타일드-디스플레이 시스템은 공동 협업 분야에서 다양하게 응용할 수 있는데, 그러한 시스템들은 일반적으로 사용 로그 정보를 기록한다. 이러한 로그 정보는 시스템의 유지 및 관리 보수를 위한 용도 뿐만 아니라, 공동 협업 상에서의 진행상황을 다시 열람할 수 있는 회의록이 된다. 타일드-디스플레이 기록기는 크게 세 단계를 거치게 되는데, 첫 번째는 각 타일에서 기록에 필요한 데이터를 기록 및 전송하는 과정이다. 두 번째는 각 타일에서 전송한 데이터를 조합하여 완성된 하나의 캡쳐 프레임을 만들게 되며, 마지막 단계에서는 조합한 결과물을 인코딩하여 저장하거나 스트리밍과 같은 서비스를 제공하게 된다. 이 과정에서 로그 정보를 저장할 때 화면 캡쳐 시간 오차로 인해 전체 로그 영상의 품질이 떨어지게 되는데, 본 논문에서는 각각의 타일 이미지를 병합하여 만들어진 전체 로그의 품질을 측정하는 타임 스탬프 기반의 평가함수를 정의하고, 정의한 평가 함수를 이용하여 로그 품질을 향상 시킬 수 있는 타일 이미지 병합 알고리즘을 제안한다.

키워드 : 타일드-디스플레이, 병합 알고리즘

Abstract The tiled-display system provides a high resolution display which can be used in different applications in co-working area. The systems used in the co-working field usually save the user logs, and these log information not only makes the maintenance of the tiled-display system easier, but also can be used to check the progress of the co-working. There are three main steps in the proposed tiled display log recorder. The first step is to capture the screen shots of the tiles and send them for merging. The second step is to merge the captured tile images to form a single screen shot of the tiled-display. The final step is to encode the merged tile images to make a compressed video stream. This video stream could be stored for the logs of co-working or be streamed to remote users. Since there could be differences in capturing time of tile images, the quality of merged tiled-display could be degraded. This paper proposes a time stamp-based metric to evaluate the quality of the video stream, and a merging algorithm that could upgrade the quality of the video stream with respect to the proposed quality metrics.

Key words : Tiled-display, Merge Algorithm

· 이 논문은 2008 한국컴퓨터종합학술대회에서 '타일드 디스플레이 기록기를 위한 병합 알고리즘'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서강대학교 컴퓨터공학과
brix@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터공학과 교수
jhnang@sogang.ac.kr

논문접수 : 2008년 9월 3일
심사완료 : 2009년 6월 13일

정보과학회논문지: 시스템 및 이론 제36권 제5호(2009.10)

1. 서 론

타일드-디스플레이 시스템은 다수의 디스플레이 디바이스를 그리드 형태로 연결하여 큰 화면과 높은 해상도를 제공해줄 수 있는 시스템이다[1]. 타일드-디스플레이 시스템이 제공해 줄 수 있는 높은 해상도는 의료영상이나 정밀기기 설계도 등의 초고해상도 영상을 표시를 가능하게 하며, 여러 가지 작업들을 동시에 표시할 수 있게 한다. 이러한 특성들을 비추어 볼 때 타일드-디스플레이 시스템은 공동 협업 분야에서 다양하게 응용할 수 있다.

공동 협업 분야에서 사용되는 시스템은 일반적으로 사용 로그 정보를 기록한다. 이러한 로그 정보는 시스템의 유지 및 관리 보수를 위한 용도뿐만 아니라 공동 협업 진행 상황을 알 수 있는 응용이 된다. 그러한 이유로 일반적인 디스플레이 시스템에서의 기록기는 이미 많은 연구와 응용을 찾아 볼 수 있다. 하지만 타일드-디스플레이 시스템 상에서의 로그 정보를 저장 기록기는 아직 그와 관련된 연구가 진행되고 있지 않다.

본 논문에서는 타일드-디스플레이 시스템의 특성이 개인용도보다는 공동협업 분야에서 응용된다는 점에 착안하여 타일드-디스플레이 기록기를 제안한다. 타일드-디스플레이 시스템 상에서 동작하는 작업들의 기록을 남기는 것은 타일드-디스플레이를 이용한 회의의 회의록이라 할 수 있다. 이 회의록을 실시간으로 외부에 전송한다면 타일드-디스플레이를 이용한 원격 회의가 가능하다.

타일드-디스플레이 기록기는 크게 세가지 단계를 거치게 되는데, 첫 번째는 각 타일에서 기록에 필요한 데이터를 기록 및 전송하는 과정이 필요하다. 이 과정에서 각 타일은 현재의 작업을 재현할 수 있는 데이터를 저장하게 되는데, 타일드-디스플레이 기록기를 응용 독립적으로 설계하기 위해서는 화면을 캡쳐하여 저장 및 전송하는 형태를 가지게 된다. 두 번째는 각 타일에서 전송한 데이터를 조합하여 완성된 하나의 캡쳐 프레임을 만들게 된다. 이 과정에서 완성된 프레임을 만들기 위한 수많은 조합이 발생하며, 이 조합 중에서 출력 품질이 가장 좋은 조합을 선택해야 한다. 마지막 단계는 두 번째 단계에서 조합한 결과물을 인코딩하여 저장, 혹은 스트림 서비스를 제공해 주는 것이다.

본 논문에서는 좀더 좋은 품질의 기록 영상을 얻기 위해 조합 영상의 품질을 측정하기 위한 기준과 조합 알고리즘을 제안한다. 제안한 알고리즘은 CPU 점유율, 네트워크 및 메모리 사용량의 제한을 받는 특성을 감안하여 설계 및 구현하였다. 또한 구현된 타일드-디스플레이 기록기에 대한 실험을 통해 조합 알고리즘에 따른 품질의 차이를 측정하였다.

2. 연구배경

2.1 기존의 타일드-디스플레이 시스템

타일드-디스플레이 시스템의 기본 구조는 그림 1과 같다. 클러스터 PC들은 각각의 모니터를 컨트롤하며 통신망으로 연결되어 있다. 이 시스템이 하나의 화면처럼 나오기 위해서는 각각의 클러스터가 동기화되어야 한다. 타일드-디스플레이는 통신망을 사용하여 동기화를 시켜 시스템을 구축하였다.

이와 비슷한 시스템은 SAGE(Scalable Adaptive Graphics Environment)[2], TeraVision[3], AG(Access Grid)[4], S/W Environments for Cluster-based Display System[5] 등이 있으며, 각각 지칭하는 명칭에는 차이가 있으나 네트워크를 통한 동기화를 사용한다는 기본적인 컨셉은 동일하다. 각각의 시스템은 고해상도의 이미지, 영상 또는 3D 렌더링을 목적으로 사용된다.

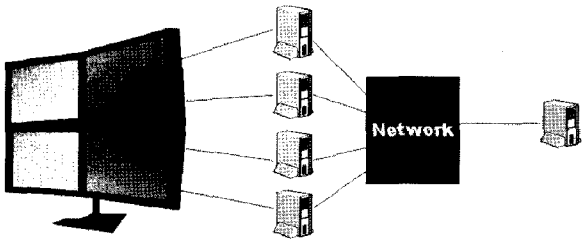

그림 1 타일드-디스플레이 시스템의 기본 구조

2.2 타일드-디스플레이 기록 시스템

기존의 타일드-디스플레이 시스템들의 용도를 살펴보면, 개인 사용자 보다는 공동 사용에 중점을 두고 있다. 공동 사용하는 시스템에서 시스템을 원격으로 열람하거나 사용 로그를 저장할 수 있는 응용은 매우 유용하게 사용될 수 있다. 타일드-디스플레이 상에서 화상 회의를 하거나 가상 공간에서 공동 협업을 하는 경우, 작업 과정을 로그로 기록해두면 회의록으로 활용할 수 있게 된다. 또한 실시간 원격 서비스를 통해 로그를 전달할 수 있고 원격지에서 로그를 재생할 수 있다면, 회의나 협업에 원격으로 참여할 수 있는 응용도 생각할 수 있다.

2.2.1 로그 저장 방식

타일드-디스플레이에 대한 로그를 저장하는 방법은 표 1처럼 크게 두 가지로 나누어질 수 있다. 화면을 직접 저장하는 방식과 이벤트 정보만을 기록하여 재현하는 방식이다. 화면을 직접 저장하는 방식은 화면을 이미지나 동영상과 같은 매체로 직접 기록하기 때문에 기록을 재생하는데 제약이 없다. 화면 저장 방식은 화면을 직접 캡쳐하기 때문에 데이터의 사용량이 상대적으로 크기 때문에 메모리와 네트워크 자원을 크게 소모한다. 이벤트 정보 저장 방식은 렌더링하는 3D의 좌표나 플레이 하는 영상의 주소 등의 정보를 기록하기 때문에 화면을 저장하는 방식의 비해 적은 데이터 사용량을 보여준다. 따라서 화면 저장 방식에 비해 메모리와 네트워크 자원의 사용량이 적다. 하지만 타일드-디스플레이 시스템에서는 수많은 응용이 동작할 수 있으며, 이벤트 저장 방식을 사용하기 위해서는 모든 응용이 이벤트 저장을 지원할 수 있도록 개발 되어야 한다. 따라서 이벤트 저장방식은 응용에 의존적인 방식이라고 할 수 있다. 또한 저장된 이벤트를 재현하기 위해서는 이벤트를 저장한 하드웨어와 동등하거나 상위의 하드웨어가 있어야 이벤트를 재현 할 수 있기 때문에 일반 PC나 PDA와 같은

기기로 저장된 이벤트를 재현 수가 없다.

본 논문에서는 타일드-디스플레이 시스템 상에서 다양한 응용 프로그램을 수행을 하며 공동 작업을 하는 과정을 가정하고, 그것을 기록하여 저장하거나 일반 PC나 PDA상에서 실시간 스트리밍을 통해 열람이 가능한 기록기를 설계 및 구현하는 것이다. 설계 및 구현하는 시스템은 기록 시스템의 재현 기기 및 응용에 대한 독립성이 요구되며, 그 요구에 부합되는 로그 기록 방식은 화면 저장 방식이다. 따라서 본 논문에서는 화면을 직접 캡처하는 화면 저장 방식을 사용한 기록 시스템을 고려한다.

표 1 로그 기록 방식의 분류

	화면 저장 방식	이벤트 저장 방식
재현 품질	해상도에 따름	좋음
데이터 사용량	비교적 크다	비교적 작다
재현 기기	독립적	의존적
응용	독립적	의존적

2.2.2 화면 캡처의 성능과 특성

화면을 직접 캡처하는 방식은 화면을 캡처하는 주기와 캡처된 화면의 해상도에 따라서 영상의 퀄리티가 크게 달라질 수 있다. 타일드-디스플레이 시스템의 기록기는 각 클라이언트가 주기적으로 화면을 캡쳐 후 네트워크를 통해서 화면을 전송하는 형태를 가지고 있다.

먼저 시스템이 다른 작업을 하지 않은 상태에서 시간을 측정해 보았다. 실험에 사용한 시스템 환경은 Intel Core2 2.4GHz, 2GB RAM, Windows XP이며, 1280×1024의 해상도를 가지고 있다. 실험 시스템에서 전체 영상을 캡쳐하여 메모리로 가져오는데 약 13ms의 시간이 소요되며, 320×240으로 리사이즈를 하는데 약 1ms, 네트워크를 통해 전송하는데 약 30ms가 소요되었다. 기록기는 이러한 작업을 주기적으로 반복하게 되므로, 실험 환경에서 얻을 수 있는 최대 FPS는 약 20fps 정도라고 할 수 있다. 이 실험은 다른 응용 프로세스가 동작하지 않는 환경에서 실험하였으며, 다른 응용이 함께 동작하는 상황을 가정한다면 fps는 좀더 낮아질 수 있다.

그림 2는 화면을 캡쳐하는 도중 다른 프로세스를 구동시켰을 때의 시간 기록이다. 먼저 전체 과정의 시간이 50ms를 초과한 경우도 볼 수 있다. 이 결과는 안정적으로 20fps의 영상을 얻기 어렵다는 것을 알 수 있다. 캡쳐, 리사이즈, 전송 3가지 과정 중에서 캡쳐 과정이 가장 큰 변동폭을 보였는데, 실험 환경에서는 약 11ms-23ms 정도의 변동폭을 보이고 있다. 이 실험을 통해 같은 시간에 캡쳐 명령을 내리더라도 화면 오차가 발생할 수 있다는 것을 알 수 있다.

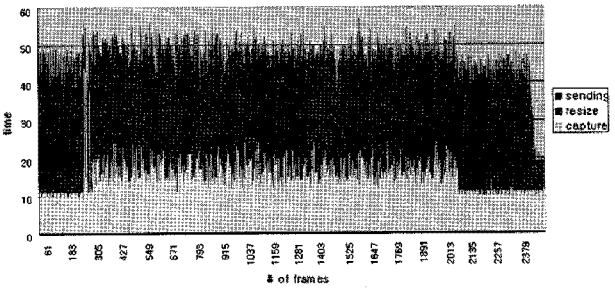

그림 2 다른 프로세스가 수행 중일 때의 화면 캡쳐 및 전송 실험

글로벌 클럭을 이용한 캡쳐를 시행하였을 때, 캡쳐 프로세스가 정확히 원하는 시간에 화면을 캡쳐할 수 있다면, 병합을 위해 네트워크로 이미지를 전송하는 과정이 불안정 하더라도 뒤틀어짐 없는 화면을 얻을 수 있다. 각각의 캡쳐 클라이언트가 다른 속도로 이미지를 전송하더라도 이미지와 함께 타임 스탬프를 찍어서 보낸다면 병합 서버에서 같은 시간에 찍힌 화면들을 병합하면 될 것이다. 하지만 그림 2에서 볼 수 있듯이 다른 프로세스가 동시에 동작할 경우 캡쳐 프로세스는 정확한 시간에 화면을 캡쳐하지 못하였다. 이렇게 생기는 시간 오차는 병합 서버에서 최종적인 병합 이미지를 생성 할 때 품질 저하의 원인이 될 수 있다. 캡쳐 프로세스의 우선 순위 실시간 프로세스 등급으로 높인다면 이러한 화면 오차가 덜해질 수는 있으나, 근본적인 해결 방안이 될 수 없다. 게다가 그렇게 된다면 캡쳐 프로세스가 다른 프로세스에 영향을 미쳐 타일드-디스플레이 시스템의 품질에 영향을 주게 된다.

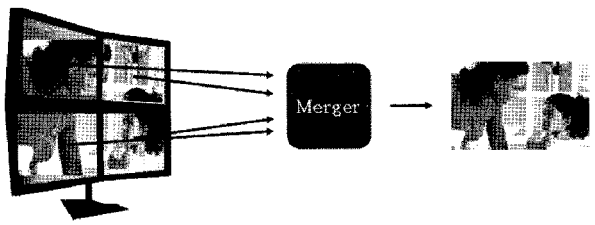

그림 3 기록 시스템의 기본 개념

3. 설계

타일드-디스플레이 상에서 동작하는 기록 시스템의 기본 구조는 그림3과 같다. 각 클라이언트는 설정된 프레임-레이트에 맞추어 화면을 캡쳐한 후 캡쳐 이미지를 전송하고, 병합 서버는 각각의 클라이언트로부터 캡쳐 이미지를 전송받아서 이미지를 병합하는 과정을 거쳐 최종 영상에 필요한 병합 이미지를 생성한다. 이 과정에서 클라이언트에서 동작하는 캡쳐 프로세스는 각 클라이언트에서 동작하는 다른 응용에 최대한 영향을 끼치

지 않도록 백그라운드로 동작하게 되며, 각 클라이언트가 글로벌 클럭을 사용하여 동시에 캡쳐 프로세스를 동작시킨다. 그러나 2.2.2에서의 실험을 통해 본 캡쳐 프로세스의 특성을 살펴보면, 동시에 캡쳐 프로세스를 동작시키더라도 캡쳐 프로세스가 동작하는 환경에 따라서 실제로 캡쳐가 되는 타이밍이 달라질 수 있다. 이러한 오차는 영상을 단순 병합하였을 때 영상의 품질이 심각하게 떨어질 수 있는 가능성을 가지게 된다. 따라서 병합 서버는 각 클라이언트에서 보내온 영상을 선택하여 조합하는 과정이 필수적이다. 이때 클라이언트의 수를 N이라고 하면 각각의 클라이언트로부터 받은 데이터를 $N(slave_i)$와 같은 형태로 표현할 수 있다. 병합 서버는 출력물의 framerate에 해당되는 시간 T_k에 맞추어 이미지를 병합하여 출력하게 된다. 이를 간단히 그림으로 표현하면 그림 4와 같다.

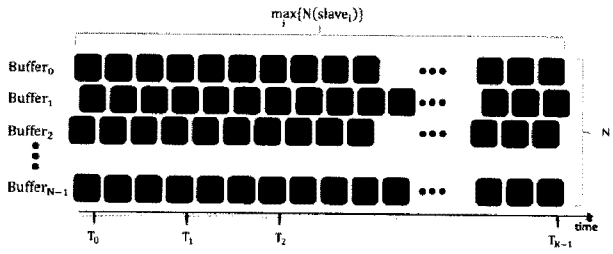

그림 4 병합 서버가 전송 받은 데이터

3.1 평가 함수의 정의

N개의 클라이언트로 이루어진 타일드-디스플레이 시스템을 가정한다. i 번째 클라이언트로부터 받은 이미지는 받은 순서대로 $Buffer_i$ 에 저장된다. i 번째 버퍼에 저장된 이미지 중에서 j 번째 이미지를 $image_i^j$ 로 표현하며, 이 이미지에 붙어있는 타임 스탬프를 TS_i^j 로 표현한다. 출력 이미지를 만들기 위해서는 각 버퍼에 있는 이미지 중 하나씩을 고르는 과정을 거쳐야 한다. 이때 $Buffer_i$ 에서 선택한 이미지 $image_i^j$ 의 버퍼 인덱스 j 를 s_i 로 표현한다. 즉 s_i 는 $Buffer_i$ 에서 선택한 이미지의 버퍼 인덱스이며, 선택한 이미지는 $image_i^{si}$ 로 표현할 수 있다. 이러한 선택을 $Buffer_0$ 부터 $Buffer_{N-1}$ 까지 하게 되면, $s_0, s_1, ..., s_{N-1}$과 같은 인덱스 리스트를 생성할 수 있다. 이 인덱스 리스트를 S 로 표현한다.

조합에 포함된 각각의 이미지들은 서로 다른 타임스탬프(Time Stamp)를 가지고 있다. 병합 서버는 N 개 버퍼에서 뽑아온 입력 이미지를 조합하여 출력 이미지를 만들게 된다. N 개의 입력 이미지 타임스탬프로부터 출력 이미지의 타임스탬프를 만든다. 이렇게 하여 만들어진 출력 이미지의 TimeStamp를 ATS(Actual Time Stamp)라고 정의한다. ATS는 각 입력 이미지의 타임스탬프의 평균을 이용하여 구하게 되며, 이를 식으로 표현하면 다음과 같다.

$$ATS(S) = avg(TS_0^{s_0}, TS_1^{s_1}, ..., TS_{N-1}^{s_{N-1}}) \quad (1)$$

병합 서버에 목적 시간 T_k가 주어지면 타일드-디스플레이의 목적 시간에 나타난 화면을 얻는 것을 기대할 수 있다. 하지만 2.2.2에서 설명한 캡쳐 프로세스의 특성에 의해 정확한 목적 시간의 화면을 얻을 수는 없다. 또한 각각의 클라이언트는 서로 다른 캡쳐 딜레이가 발생하게 된다. 따라서 주어진 목적 시간 T_k로 얻어낸 출력 이미지의 ATS는 서로 다를 수 있는데, 이 때 발생한 목적 시간과 ATS의 차이를 Delay로 정의한다. Delay는 목적 시간 T_k와 출력 이미지의 ATS간의 시간 거리를 이용하여 구하게 되며, 목적 시간 T_k에 선택한 조합 S_k에 대한 Delay를 식으로 표현하면 다음과 같다.

$$Delay(S_k, T_k) = |T_k - ATS(S_k)| \quad (2)$$

입력으로 주어지는 목적 시간은 framerate에 맞게 일정하게 주어진다. 따라서 매 프레임마다 Delay가 일정하다면, Jitter가 적다고 할 수 있다. 이러한 원리로 현재 이미지의 Delay와 이전 이미지의 Delay인 $Delay(S_{k-1}, T_{k-1})$를 이용하여 해당 프레임의 순간 Jitter를 정의할 수 있다. 따라서 목적 시간 T_k에 선택한 조합 S_k에 대한 Jitter를 식으로 표현하면 다음과 같다.

$$Jitter(S_k, T_k) = |Delay(S_k, T_k) - Delay(S_{k-1}, T_{k-1})| \quad (3)$$

Jitter의 값을 사용할 때에는 순간적인 Jitter 보다는 최근 1초간의 Jitter와 같은 형식의 구간별 Jitter를 사용하는 것이 좀더 일반적이다. 본 장에서는 매 이미지마다의 선택에서 Jitter를 사용하기 용이하게 하기 위해서 순간적인 Jitter를 정의하였다. 특정 구간에서의 Delay는 매 프레임마다의 Delay의 평균을 이용해 구할 수 있으며, 특정 구간에서의 Jitter는 매 프레임마다의 Delay의 표준 편차를 이용해 구할 수 있다.

출력 이미지는 입력 이미지들의 조합이므로, N 개의 타임스탬프를 가진다. 출력 이미지를 대표하는 타임스탬프인 ATS는 이 N 개의 타임스탬프를 이용하여 만들어진다. 이 때 만들어진 ATS와 N 개의 타임스탬프간의 차이를 이용하여 Skew를 정의하였다. 즉, Skew는 조합에 사용되는 이미지의 타임스탬프들이 얼마나 응집되어 있는가를 나타낸다. 식으로 표현하면 다음과 같다.

$$Skew(S_k) = \sum_{i}^{N-1} |TS_i^{S_i} - ATS(S_k)| \quad (4)$$

좋은 출력 이미지를 만드는 조합은 먼저 Delay가 적을수록 좋은 조합이라 할 수 있고, 출력 이미지의 간격은 일정할 수록 좋으므로 Jitter가 적을수록 좋은 조합이 될 것이다. 또한 출력 이미지를 이루는 입력 이미지들은 서로의 타임스탬프가 가까울수록 좋은 조합이 된다. 따라서 조합에 대한 평가 함수는 다음과 같이 정의

할 수 있다.

$$f(S_k,T_k)=$$
$$\alpha\cdot Delay(S_k,T_k)+\beta\cdot Jitter(S_k,T_k)+\gamma\cdot Skew(S_k) \quad (5)$$

3.2 병합 알고리즘

제안한 평가 함수 $f(S_k, T_k)$가 0에 가까울수록 선택한 조합 S_k은 좋은 조합이 되므로 평가 함수가 최소가 되는 조합을 선택하는 방법으로 최선의 선택을 할 수 있을 것이다. 따라서 가장 쉽게 조합을 선택하는 방법은 모든 선택 조합에 대해서 최소의 평가 함수 값을 가지는 조합을 선택하는 방법이 있다. 그러나 N개의 캡쳐 클라이언트가 각각 M개의 이미지를 전송한다면, 선택 가능한 집합 S는 M^N가짓수가 되므로, 각 프레임을 선택하는 시간 복잡도는 $O(M^N)$이 되며, 이 작업을 프레임의 수인 K만큼 반복해야 하므로 시간 복잡도는 $O(K\cdot M^N)$이 된다. 즉, 모든 선택 조합에 대해서 평가 함수 값을 비교하는 방법은 실시간 처리에 적합하지 않으며, 이러한 문제를 해결할 별도의 알고리즘이 필요하다.

3.2.1 검색의 시작점과 종료점

목적 시간 T_0일 때 가장 낮은 $f(S, T_0)$를 가지는 S를 S_0라고 가정한다. 새로운 목적 시간 T_1은 T_0다음에 제공되는 목적 시간이다. 시간의 순차성에 의해 $T_0 \le T_1$이 성립한다. 이 경우에 동일한 조합 S_0에 대한 Skew의 값은 같고 $f(S, T_0)$의 값은 최소값이므로 다음과 같은 식이 성립한다.

$$f(S_k, T_k) \le f(S_k, T_{k+1}) \quad (6)$$

목적 시간 T_{k-1}의 최적의 조합인 S_{k-1}을 알고 있다면, 목적 시간 T_k에 최적인 조합은 S_k은 S_{k-1}보다 이전 시간에 캡쳐된 이미지를 갖지 않는다. 따라서 S_k을 찾기 위해서 처음부터 검색할 필요는 없으며, S_{k-1}의 조합을 이용하여 검색의 시작 지점을 알 수 있다.

평가 함수 f는 Delay, Jitter, Skew로 이루어져 있는데, 이중 Delay와 Jitter로만 이루어진 함수 f'을 다음과 같이 정의한다.

$$f'(S_k,T_k)=\alpha\cdot Delay(S_k,T_k)+\beta\cdot Jitter(S_k,T_k) \quad (7)$$

선택 조합 리스트 S의 $S_0, S_1, ..., S_{N-1}$의 모든 리스트 값에 대해 같거나 큰 선택 조합 리스트 S'을 다음과 같이 정의할 수 있다.

$$S' = s'_0, s'_1, ..., s'_{N-1} \quad (8)$$

데이터는 시간 순서대로 도착하기 때문에 $TS_i^{s_i} \le TS_i^{s'_i}, (i<N)$가 성립하며, 따라서 $ATS(S)\le ATS(S')$가 된다. ATS가 변화하면, Delay와 Jitter에 영향을 주게 되는데, $f'(S_k,T_k)$은 최소점이나 최소구간을 가지는 특성이 있으므로, 만약 $f'(S_k,T_k)<f'(S'_k,T_k)$가 성립한다면 최소 구간을 통과했다고 생각할 수 있으며, 다음 선택인 S''역시 $f'(S'_k,T_k)<f'(S''_k,T_k)$와 같은 식이 성립함을 알 수 있다.

검색 시작 지점부터의 검색에서 평가함수 $f(S_k, T_k)$의 값을 가장 작게 하는 S를 S^*k라고 가정한다. 모든 S에 대하여 $Skew(S)\ge 0$이기 때문에 검색 종료 지점 이후의 S에서는 S^*k가 존재할 가능성이 없다. 이 검색 종료 지점 $S^{END}{}_k$는 다음의 식을 만족한다.

$$f(S^*_k,T_k) \le f'(S^{END}{}_k,T_k) \quad (9)$$

3.2.2 이미지 조합 알고리즘

앞서 서술한 데이터의 특성 및 평가 함수의 특성을 이용하여 검색 범위를 줄인 알고리즘 MakeImageSet을 작성하였다. 작성한 알고리즘은 그림 5와 같다.

```
1: Initialize S^Start to all 0 list.
2: Copy list S^Start to S^Prev
3: for k ← 0 to K − 1 do
4:   Initialize S^End to all ∞ list.
5:   min ← ∞
6:   Copy list S^Prev to S^Start
7:   Copy list S^Prev to S^Cur
8:   while true do
9:     if min > f(S^Cur, T_k) then
10:      min ← f(S^Cur, T_k)
11:    end if
12:    if f'(S^Prev, T_k) < f'(S^Cur, T_k), focus^Prev = focus^Cur then
13:      if min < f'(S^Cur, T_k) then
14:        s^End_focus ← s^Cur_focus
15:      end if
16:    end if
17:    S^Prev ← S^Cur
18:    if S^Cur = S^End then
19:      break
20:    end if
21:    S^Cur ← S^Cur + 1
22:  end while
23:  Merge(S^Prev)
24: end for
```

그림 5 이미지 셋 선택 알고리즘

MakeImageSet 알고리즘이 검색하는 범위는 검색의 시작점과 끝점을 포함하므로 최적해를 출력한다고 할 수 있다. 모든 경우를 검색해야 하는 경우는 $O(K\cdot InputFrame^N)$의 복잡도를 가지고 있다. 6개의 캡쳐 클라이언트로부터 초당 15프레임의 입력을 받고 초당 5프레임의 출력을 하는 경우를 가정하여 입력 각각 150프레임, 출력 50프레임의 10초에 해당하는 데이터를 가지고 테스트한 결과, 모든 경우를 검색해야 하는 경우는 약 570Tera의 범위를 검색하였고 MakeImageSet은 약 13Mega의 범위만을 탐색하였다.

4. 구현 및 분석

4.1 타임 스탬프의 동기화 방법

타일드-디스플레이 상의 서로 다른 PC들은 각각의 동기화에 맞추어 응용을 수행하고 있다. 이렇게 서로 다른 PC사이의 동기화를 맞추는 방법으로는 동기화 정보를 주기적으로 전송하는 방법이 있다. 그러나 그러한 방식은 별도의 동기화 정보를 전송하는데 대한 오버헤드를 발생시킨다.

본 논문에서는 영상의 재생 동기화를 위한 오차 허용치(±120ms)가 하드웨어 클럭의 오차보다 상대적으로 매우 크다는 것을 이용하여 시스템을 구동 중에 별도의 동기화 정보를 전송하지 않는 방법을 사용하였다. 시스템을 초기화 할 때 각 PC 사이의 하드웨어 클럭을 그림 6의 방식에 의하여 동기화 한다. 그림 6에서 ST_A는 첫 번째 메시지를 보낼 때의 전송 측의 하드웨어 클럭의 값이고, RT_A는 첫 번째 메시지를 받았을 때의 수신 측의 하드웨어 클럭 값이며, RT_B는 두 번째 메시지를 보낼 때의 수신 측의 하드웨어 클럭 값이고, ST_B는 두 번째 메시지를 받았을 때의 전송 측의 하드웨어 클럭 값이다. 메시지의 왕복 시간은 $ST_B - ST_A - (RT_B - RT_A)$이며 이 시간을 2로 나눈 값으로 전송 지연 시간 d_{prop}를 구할 수 있다. 따라서 RT_A 때의 전송 측의 시간을 다음과 같이 표현할 수 있다.

$$RT_{Base} = ST_A + \left(ST_B - ST_A - (RT_B - RT_A)\right)/2 \quad (10)$$

여기서 RT_{Base}와 현재 시간과의 차이를 $Hardware\text{-}Clock - RT_A$로 표현할 수 있으므로, 글로벌 클럭을 다음과 같이 구할 수 있다.

$$GlobalClock = HardwareClock - RT_A + RT_{Base} \quad (11)$$

병합 서버는 이와 같은 방식으로 병합 서버와 모든 캡쳐 클라이언트 사이의 클럭 동기화를 맞춘 후, 이후에 모든 캡쳐 클라이언트는 글로벌 클럭을 사용하여 캡쳐된 이미지의 타임스탬프를 생성한다.

4.2 병합 서버의 구현 및 분석

본 논문에서는 원격 참여를 위해 실시간 스트리밍과 클라이언트의 자원 사용량 최소화를 고려하여 Push 형태의 병합 서버를 구현하였다. 구현한 시스템의 구조는

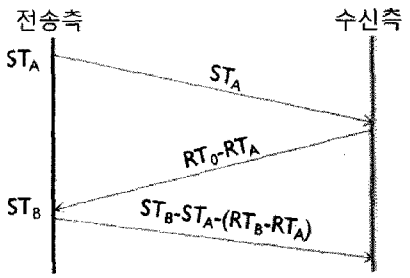

그림 6 전송측과 수신측의 클럭 동기화 알고리즘

그림 7과 같다. 각 클라이언트는 시스템의 상황에 따라 drop frame 및 jitter가 발생 할 수 있다. 따라서 각각의 버퍼는 각기 다른 속도로 채워지므로 각각 버퍼의 수신을 담당하는 스레드가 필요하다. 병합 프로세스는 N개의 공유버퍼를 확인하여 이미지 조합을 선택, 병합한다. 출력 이미지는 출력 버퍼에 저장되며 이 출력 버퍼를 열람하여 기록으로 저장하거나 화면에 출력, 혹은 외부에 실시간 스트리밍을 제공해 줄 수 있다.

알고리즘의 특성을 분석하기 위해서는 같은 입력 데이터가 필요하다. 하지만 제안한 기록기의 경우는 실시간으로 다른 프로세스가 동작하는 환경에서 동작하므로, 매번 실험마다 다른 입력 데이터로 비교하게 되어 실험을 하기에 적합하지 않았다. 제안한 병합 알고리즘은 적절한 조합을 선택하기 위해 캡쳐 클라이언트에서 생성한 타임 스탬프만을 사용하므로 캡쳐 클라이언트에서 생성한 타임 스탬프를 저장한 뒤, 그 데이터를 이용하여 실험하였다. 표 2와 같이 5fps와 15fps의 frame rate와 시스템에서 아무 작업도 하지 않는 상태를 비혼잡 상태, 여러 타일드 디스플레이용 영생 재생기가[7] 동작하고 있는 상태를 혼잡 상태로 가정하여 실험하였다.

제안한 알고리즘은 평가 함수의 세가지 요소인 Delay, Jitter, Skew의 가중치를 조절하여 이미지 조합 전략에 변화를 줄 수 있다. 만약 평가 함수의 세가지 요소 중에서 Delay를 중시한다면 그림 8의 이미지 셋 A를 선택하게 될 것이다. 마찬가지로 Jitter를 중시한다면 이미지 셋 B를 선택하게 될 것이다. 극단적으로 Skew를 중시

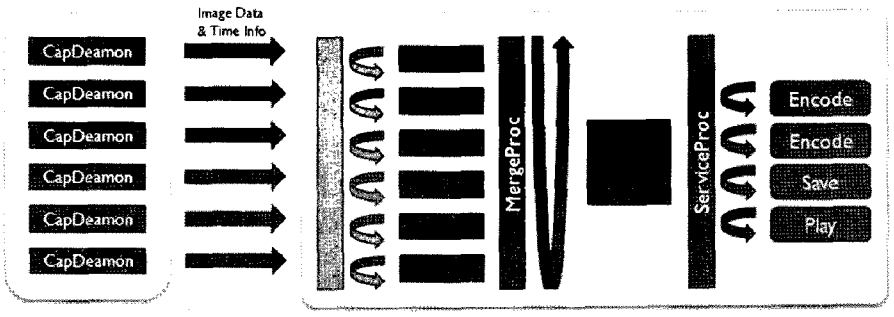

그림 7 구현 시스템의 구조

표 2 실험 데이터

	초당 캡쳐 횟수	시스템의 상태
데이터1	5fps	비혼잡
데이터2	5fps	혼잡
데이터3	15fps	비혼잡
데이터4	15fps	혼잡

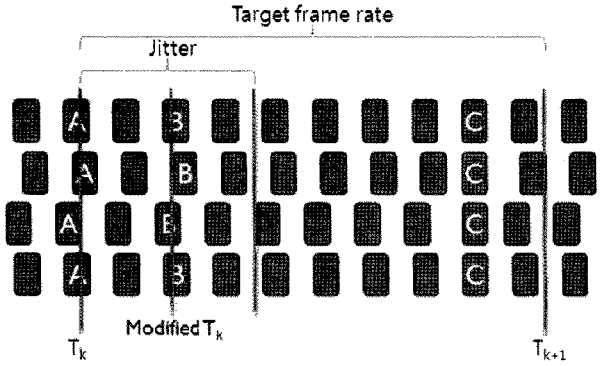

그림 8 평가 함수의 가중치에 따른 선택의 변화

하는 평가 함수를 사용한다면, 목적 시간 T_k와 멀리 떨어져있는 이미지 셋 C를 선택할 수도 있다. 각각의 조합 전략은 로깅 정보를 저장하는 컨텐츠의 성질에 따라 다를 수 있기 때문에 어떤 가중치가 가장 적합한 가중치라고 말하기는 어렵다.

실험을 위해서는 두 가지 평가 가중치를 결정해야 한다. 첫 번째 평가 전략은 Delay만을 고려하는 전략으로서 평가 함수의 가중치를 $\alpha = 1, \beta = 0, \gamma = 0$으로 설정하였다. 두 번째 평가 전략은 세 가지 평가 기준을 전부 고려하는 전략으로서 가중치 값을 $\alpha = 1, \beta = 1, \gamma = 15$으로 설정하였는데, 여기서 γ의 크기가 유독 다른 것은 Delay와 Jitter는 목적 시간 기반의 기준인 것에 비해 Skew는 조합에 의한 평가 기준이기에 평가 함수에 영향을 미치는 크기가 다르기 때문이다. $\gamma = 15$의 의미는 실험 환경을 기준으로 목적 시간에서 벗어나더라도 Skew에 의해 조합을 선택할 수 있는 범위가 최소 앞뒤 3프레임 이상이라고 해석 할 수 있다. 입력 데이터는 표 2의 데이터를 각각 사용하였으며, 각각의 입력 데이터 frame rate와 동일한 출력 데이터 frame rate를 설정한 뒤 실험을 하였다. 실험 결과는 표 3과 같다. 데이터1과 데이터3의 실험 결과는 두 가지 전략이 동일한 결과를 보여주었다. 데이터1과 데이터3의 데이터는 시스템의 혼잡도가 거의 없는 상황에서 수집된 데이터이므로 캡쳐 프로세스에 큰 오차가 없기 때문에 결과에 차이가 없는 것으로 해석된다. 반면 데이터2와 데이터4는 각각의 전략 간에 차이가 드러난다. "모두 고려한 전략"은 "Delay만을 고려한 전략"에 비해 Delay와 Jitter는 높은 모습을 보여준다. 이는 어긋나지 않은 병합 이미지를 만들기 위하여 출력 이미지 간의 간격이 약간 불규칙해졌기 때문이다. 하지만 Skew가 낮아진 부분을 보았을 때, "Delay만을 고려한 전략"보다 "모두 고려한 전략"은 이미지들 사이가 조금 불안정해졌지만, 각각의 이미지는 좀더 완전하다고 기대할 수 있다.

표 3 입출력의 frame rate가 같은 환경에서의 실험결과

	평균 delay		평균 jitter		평균 skew	
	평가1	평가2	평가1	평가2	평가1	평가2
데이터1	2.81	2.81	1.26	1.26	4.21	4.21
데이터2	3.54	6.88	2.13	3.44	6.63	6.59
데이터3	3.11	3.11	1.87	1.87	4.60	4.60
데이터4	3.47	11.23	2.37	5.12	5.74	4.18

표 3의 가장 큰 차이를 보인 데이터4의 실험 선택을 살펴보면, 통계 값 중에서 Delay가 가장 큰 차이를 보인다. 입력 이미지 간의 예상 간격이 66ms이기 때문에, Skew의 영향으로 멀리 떨어진 이미지 셋을 선택하였을 경우 60ms 이상의 Delay가 발생하여 생긴 것이다. 그러한 이유로 통계 값의 차이와는 달리 출력 품질을 눈으로 직접 살펴본 결과 큰 차이를 느끼지 못하였다. 그러한 차이는 입력 데이터와 출력 데이터에 frame rate가 같다는 데서 생긴 결과라고 생각하였다. 입출력의 frame rate가 같아지면, 알고리즘이 지능적으로 수행되더라도 실질적인 선택의 폭이 매우 제한되기 때문이라고 생각한다. 따라서 그러한 차이를 확인하기 위해 데이터4의 입력을 가지고 5fps의 출력 이미지를 만드는 실험을 수행하였다.

출력 데이터의 frame rate가 낮아지면 목적 시간의 간격에 들어가는 입력 데이터가 많아져서 좀더 자유로운 선택이 가능하다. 실험 결과는 표 4와 같다. 이 실험에서 제안한 알고리즘의 평균 Delay는 크게 상승하였다. 하지만 이것은 출력 frame rate 5fps의 주기인 198ms와 비교하였을 때 상대적으로 낮은 수치이다. 또한 Jitter를 고려한 영향으로 프레임 간격이 급격하게 빨라지기 보다는 서서히 빨라지거나 느려져서 체감상으로는 큰 차이가 없었다. 반면 Skew는 크게 낮아져서 이 실험에서의 영상 출력 결과는 알고리즘간의 성능 차를 보여주었다.

Delay만을 사용하는 알고리즘을 사용할 경우 캡쳐하는 시스템의 혼잡도가 커져서 캡쳐의 오차가 발생하게 되면, 출력 영상의 품질이 크게 저하된다. 제안한 알고리즘을 사용한 시스템 역시 캡쳐의 오차가 발생하면 출

표 4 입출력의 frame rate가 다른 환경에서의 실험결과

	평균 delay		평균 jitter		평균 skew	
	평가1	평가2	평가1	평가2	평가1	평가2
데이터4	3.16	48.23	2.98	6.67	5.66	1.18

력 영상의 품질이 저하되지만, 병합 알고리즘을 통해 품질이 저하되는 것을 막을 수 있다. 특히 이 병합 알고리즘은 입력 영상의 frame rate 보다 출력 영상의 frame rate가 낮을 경우 더 높은 효율을 보인다. 기록기를 사용하는 여러 가지 시나리오 중에서 PDA와 같은 제한적인 기기로 타일드-디스플레이 시스템을 열람하게 된다면, 네트워크 속도나 저장 용량과 같은 기기의 제한으로 인해 출력의 frame rate를 낮춰야 하는 상황이 있을 것이다. 그러한 경우 이 병합 알고리즘은 크게 효율이 있을 것이라 기대한다.

5. 결론 및 향후 연구 과제

타일드-디스플레이 시스템은 큰 화면과 높은 해상도를 제공해주기 위한 시스템이다. 타일드-디스플레이 시스템이 제공해 줄 수 있는 높은 해상도는 여러 전문 분야에서 활용할 수 있으며, 특히 공동 연구나 협업과 같은 분야에서 주로 활용된다. 그러한 점에 착안하여 회의록이나 원격 참여와 같은 응용으로 활용할 수 있는 타일드-디스플레이 기록 시스템이 필요하며, 하드웨어와 응용에 독립으로 사용하기 위해서는 화면 저장 방식의 로그 저장 방식을 택하여야 한다. 화면 저장 방식은 화면을 캡처하기 위한 작업이 필요한데, 이 작업은 다른 응용 작업의 프로세스에 따라 오차가 발생하기 때문에 정확히 동기화된 시간에 캡처 화면을 생성할 수 없는 단점을 가지고 있다. 이러한 환경에서 좋은 품질의 로깅 화면을 얻기 위해서는 조합할 수 있는 무수히 많은 조합들을 서로 비교할 수 있는 평가 함수가 필요하다. 따라서 Delay, Jitter, Skew라는 기준에 맞추어 평가 함수를 정의하고 이에 따른 병합 알고리즘을 제안하였다. 제안한 알고리즘은 모든 조합을 비교하는 방법과 동일한 최적 조합을 갖게 되며, 실시간 처리가 가능하다.

본 논문에서 제안한 알고리즘은 타임 스탬프에 기반한 방법을 사용하고 있다. 타임 스탬프는 영상을 조합하기 위한 좋은 기준이지만, 원본 화면이 정적인 모습에서 갑자기 동적인 모습을 보이거나 화면 전환을 하는 경우는 예외가 생길 수 있다. 이럴 때에 화면의 비주얼 정보까지 고려한다면 좀더 나은 결과를 얻을 수 있을 것이라 생각된다. 향후에는 타임 스탬프 이외의 정보를 함께 고려한 알고리즘을 연구할 예정이며, 좀더 정확한 화면 캡처를 위한 방법도 함께 연구할 예정이다.

참 고 문 헌

[1] M. Herald, I. Judson and R. Stenvens, Introduction to Building Projection-based Tiled Display Systems, *IEEE Computer Graphics and Applications*, pp.22-28, Jul./Aug. 2000.

[2] University Illinois, Scalable Adaptive Graphics Environment, http://www.evl.uic.edu/cavern/sage, 2007.

[3] R. Singh, B. Jeong, L. Renambot, A. Johnson and J. Leigh, TeraVision: a Distgributed, Scalable, High Resolution Graphics Streaming System, in *proceedings of the 2004 IEEE International Conference on Cluster Computing*, pp.391-400, 2004.

[4] Y. Zhao and X. Zhang, Design a secure and scalabel CVE: The Access Grid, in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, pp.59-59, 2001.

[5] H. Chen, D. Clark, Z. Liu, G. Wallace and K. Che, Software Environments For Cluster-Based Display Systems, in *proceedings of the 1st International Symposium on Cluster Computing and the Grid*, p.202, 2001.

[6] Giseok Choe, Jeongsoo Yu, Jeonghoon Choi, Jongho Nang, Design and Implementation of a Real-time Video Player on Tiled-Display System, in *Proceedings of the IEEE 7th International conference on Computer and Information Technology*, pp.621-626, October 2007.

최 기 석

2006년 서강대학교 컴퓨터공학과 학사
2008년 서강대학교 컴퓨터공학과 석사
2008년~서강대학교 컴퓨터공학과 박사과정. 관심분야는 멀티미디어 시스템, 이미지 및 동영상 검색

낭 종 호

1986년 2월 서강대학교 전자계산학과 졸업(학사). 1988년 2월 한국과학기술원 전산과 졸업(석사). 1992년 2월 한국과학기술원 전산과 졸업(박사). 1992년 3월~1992년 8월 한국과학기술원 정보전자 연구소(연구원). 1992년 9월~1993년 8월 일본 Fujitsu 연구소(방문연구원). 1993년 9월~현재 서강대학교 컴퓨터학과 교수. 관심분야는 멀티미디어 시스템, 동영상 검색, 동영상 분석, 병렬/분산 처리, 인터넷 컴퓨팅