타임 스탬프와 샷 검출을 이용한 타일드 디스플레이 기록기의 병합 알고리즘

최기석° 낭종호
서강대학교 컴퓨터공학과
brix@sogang.ac.kr, jhnang@sogang.ac.kr

A Tile-Merging Algorithm of Tiled Display Recorder using Time-stamp and Shot Detection

요약

타일드-디스플레이 시스템은 다수의 디스플레이 디바이스를 그리드 형태로 연결하여 큰 화면과 높은 해상도를 제공해줄 수 있는 시스템이다[1]. 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 진행 상황을 다시 열람할 수 있는 훌륭한 회의록이 된다. 이러한 로그 정보를 저장할 때 각각의 타일들의 화면 저장 오차에 의해 전체 로그 영상의 품질이 떨어지게 되는데, 이를 보완하기 위해 이전 연구에서 타임 스탬프 기반의 영상 병합 알고리즘을 제안하였다. 하지만 타임 스탬프 기반의 영상 병합 알고리즘은 샷 경계 부분이나 움직임 크게 변화하는 영상에서 품질이 나빠지는 경우를 대응하기 어렵다. 본 논문에서는 이러한 타임 스탬프 기반 알고리즘에서 대응할 수 없었던 샷 경계 부분의 품질 저하를 줄이기 위해 타임 스탬프와 샷 검출을 함께 이용하는 알고리즘을 제안한다.

1 서론

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 표시할 수 있으며, 여러가지 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 용도 뿐만 아니라, 그 자체로서도 공동 협업 진행 상황을 알 수 있는 응용이 된다. 그러한 이유로 일반적인 디스플레이 시스템에서의 기록기는 이미 많은 연구와 응용을 찾아 볼 수 있다. 타일드-디스플레이 시스템 상에서 동작하는 작업들의 모습을 기록을 남기는 것은 타일드-디스플레이를 이용한 회의의 회의록이라 할 수 있으며, 그것을 실시간으로 외부에 전송 할 수 있다면 타일드-디스플레이를 이용한 회의에 원격으로 참여할 수 있게 된다.

타일드-디스플레이 기록기는 크게 세가지 과정를 거치게 되는데, 각 타일에서 기록에 필요한 데이터를 기록 및 전송하는 과정과 각 타일에서 전송한 데이터를 조합하여 완성된 하나의 캡쳐 프레임을 만드는 과정, 그리고 이전 단계에서 조합한 결과물을 인코딩하여 저장, 혹은 스트림 서비스를 제공해 주는 과정이 있다.

이전 연구에서는 타일드-디스플레이 기록기를 위한 타임 스탬프 기반의 평가 함수와 알고리즘을 제안하였으나 이 알고리즘은 움직임이 크게 변화하는 구간이나 샷 경계 구간에서의 품질 저하를 해결 할 수 없었다. 본 논문에서는 이러한 타임 스탬프 기반의 병합 알고리즘의 단점 중 샷 경계 구간의 품질 저하를 개선하기 위하여 기록 영상의 샷 검출을 이용한 알고리즘을 제안한다.

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 동일하다고 할 수 있다. 각각의 시스템은 고해상도의 이미지, 영상 또는 3D 렌더링을 목적으로 사용된다.

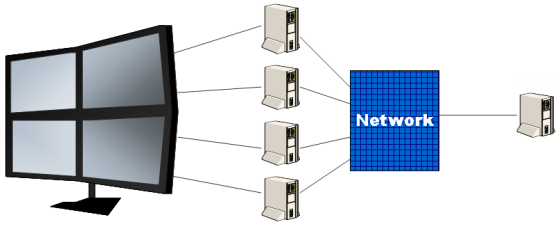

그림 1: 타일드-디스플레이 시스템의 기본 구조

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 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 1 처럼 크게 두가지로 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 플레이하는 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 것을 가정하고, 그것을 기록하여 저장하거나 일반 PC나 PDA상에서 실시간 스트리밍을 통해 열람이 가능한 기록 시스템을 고려한다. 따라서 타일드-디스플레이 기록 시스템은 기록 시스템의 재현 기기 및 응용에 대한 독립성이 요구되며, 그 요구에 부합되는 로그 기록 방식은 화면 저장 방식이다. 따라서 본 논문에서는 화면을 직접 캡쳐하는 화면 저장 방식을 사용한 기록 시스템을 고려한다.

2.2.2 화면 캡쳐의 성능과 특성

화면을 직접 캡쳐하는 방식은 화면을 캡쳐하는 주기와 캡쳐된 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. 영상을 캡쳐하여 메모리로 가져오는데 약 13ms의 시간이 소요되며, 320×240으로 리사이즈를 하는데 약 $1ms$, 네트워크를 통해 전송하는데 약 $30ms$가 소요되었다. 기록기는 이러한 작업을 주기적으로 반복하게 되므로, 기록기에서 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 2은 화면을 캡쳐하는 도중 다른 프로세스를 구동시켰을 때의 시간 기록이다. 먼저 전체 과정의 시간이 $50ms$를 초과한 경우도 볼 수 있다. 이는 기록기가 안정적으로 20fps의 영상을 출력이 어렵다는 것을 알 수 있다. 또한 캡쳐, 리사이즈, 전송 3가지 과정 중에서 캡쳐 과정이 가장 큰 변동폭을 보이는 것을 알 수 있다. 실험에서는 약 $11ms - 23ms$ 정도의 변동폭을 보이고 있으며, 이것을 통해 같은 시간에 캡쳐 명령을 내리더라도 최소한 $12ms$ 이상의 화면 오차가 발생 할 수 있다는 것을 알 수 있다.

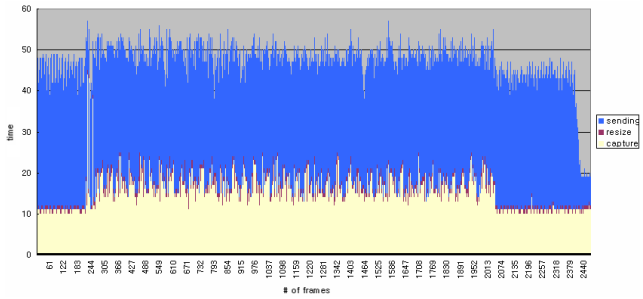

그림 2: 다른 프로세스가 수행 중일때의 화면 캡쳐 및 전송 실험

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 타임스탬프를 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 퍼포먼스에 영향을 주게 된다.

2.3 병합 알고리즘

타일드-디스플레이 상에서 동작하는 기록 시스템의 기본 구조는 그림 3과 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 캡

쳐 프로세스는 각 클라이언트에서 동작하는 다른 응용에 최대한 영향을 끼치지 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 있으며, 이 오차는 최소 13ms 이상이 된다. 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 frame-rate에 해당되는 시간 T_k에 맞추어 이미지를 병합하여 출력하게 된다. 이를 간단히 그림으로 표현하면 그림 4와 같다.

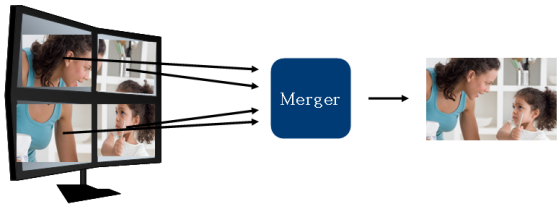

그림 3: 기록 시스템의 기본 개념

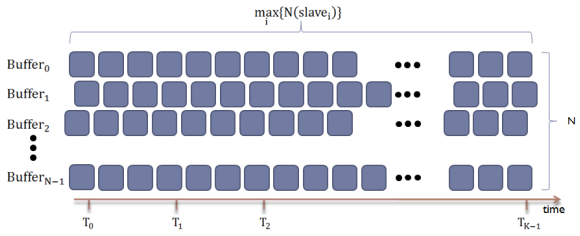

그림 4: 병합 서버가 전송 받은 데이터

3 설계

3.1 평가 함수의 정의

N개의 클라이언트로 이루어진 타일드-디스플레이 시스템을 가정한다. i번째 클라이언트로부터 받은 이미지는 받은 순서대로 $Buffer_i$에 저장된다. i번째 버퍼에 저장된 이미지 중에서 j번째 이미지를 $Image_i^j$로 표현하며, 이 이미지에 붙어있는 타임 스탬프를 TS_i^j로 표현한다. 출력 이미지를 만들기 위해서는 각 버퍼에 있는 이미지중 하나씩을 고르는 과정을 거쳐야 한다. 이때 $Buffer_i$에서 선택한 이미지 $Image_i^j$의 버퍼 인덱스 j를 s_i로 표현한다. 즉 s_i는 $Buffer_i$에서 선택한 이미지의 버퍼 인덱스이며, 선택한 이미지는 $Image_i^{s_i}$로 표현 할 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 S로 표현한다.

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

$$ATS(S) = avg(TS_0^{s_0}, TS_1^{s_1}, TS_2^{s_2}, ..., TS_{N-1}^{s_{N-1}})$$

출력 이미지의 $TimeStamp$인 ATS를 이용하여 $Delay$, $Jitter$, $Skew$를 다음과 같이 정의할 수 있다.

$$Delay(S_k, T_k) = |T_k - ATS(S_k)|$$

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

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

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 가까울 수록 좋은 조합이 된다. 이 특성들을 고려한 평가 함수는 다음과 같이 정의할 수 있다. [7]

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

입력으로 들어오는 모든 데이터를 원하는 출력시간 T_k에 맞추어 평가 함수로 평가를 할 경우, 가장 작은 평가 함수값을 가지는 조합이 평가 함수를 이용한 최적의 출력 조합이 된다.

3.2 타임 스탬프 기반의 평가 함수를 이용한 알고리즘과 그 한계

최적의 출력 조합을 찾아내기 위하여 고려해야 하는 경우의 수는 실시간으로 처리할 수 없을 정도로 많다. 하지만 입출력의 중요한 파라메터로 쓰이는 타임 스탬프가 순차적으로 이동한다는 것을 이용하면 고려해야하는 경우의 수를 크게 줄여 실시간 처리를 할 수 있다. [7]

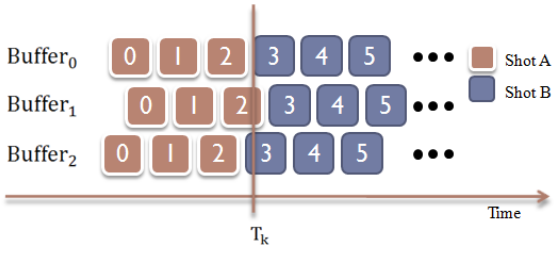

그림 5: 타임 스탬프 정보만으로 해결할 수 없는 경우

병합 알고리즘이 필요한 근본적인 이유는 원하는 출력 시간에 맞는 화면이 없는 경우가 많기 때문이다. 이 때, 병합 알고리즘은 원하는 화면과 시각적으로 최대한 비슷한 화면을 찾아서 채워야 한다. 타임 스탬프 기반의 평가함수를 이용한 병합 알고리즘은 가까운 시간의 화면이 시각적으로 비슷할 것이라는 가정을 포함하고 있다. 대부분의 경우에 각각의 타일드-디스플레이 클러스터들은 30ms내외의 오차를 보이고, 그 오차는 원하는 화면과 앞뒤로 $1-2$ 프레임의 범위 내에 존재한다. 이러한 범위내에서 시간을 기반으로 시각적 오차를 최소화 한다는 가정은 움직임이 극단적으로 크지 않는 한 크게 무리가 없다. 그러나 영상의 샷 경계 부근으로 가면 극단적

인 반례가 존재한다. 샷과 샷 사이에서는 단 1ms의 차이라도 완전히 다른 영상이 될 수 있기 때문이다. 그림 5을 보면 목적시간 T_k가 샷 경계 부근에 위치하였고, 이러한 상황에서는 (3,2,3)과 같은 선택을 할 가능성이 존재한다. (3,2,3)과 같은 선택을 한 경우 서로 다른 두개의 샷이 하나의 이미지가 되어 기록 영상의 품질이 크게 떨어질 것이다. 따라서 문제가 발생하는 샷과 샷의 경계를 검출하여 이 부분의 로그 품질을 보완하는 방법이 필요하다.

3.3 명암(Pixel Intensity)을 이용한 샷 검출

본 논문에서는 영상의 샷 검출을 위해 명암(Pixel Intensity)을 이용한 방법을 사용하였다.[8] 먼저 픽셀의 명도를 $I(x,y)$로 정의하고 픽셀 좌표 (x,y)에 k 프레임과 $k+l$ 프레임 사이의 명도차를 다음과 같이 표현한다.

$$D_{k,k+l}(x,y) = |I_k(x,y) - I_{k+l}(x,y)|$$

이렇게 정의된 명도차로 k 프레임과 $k+l$ 프레임 사이의 불연속성값 $z(k)$를 다음과 같이 표현할 수 있다.

$$z(k) = \frac{1}{XY}\sum_{x=1}^{X}\sum_{y=1}^{Y} D_{k,k+l}(x,y)$$

여기서 구한 불연속성값 $z(k)$의 역치값 T를 정의한다. 정의한 역치값 T를 넘으면서 꼭대기점이 되는 k 프레임을 찾아서 샷의 경계로 인식한다.

본 논문에서는 위와 같은 방법으로 샷 검출을 하였으나, 샷 검출을 하는 방법은 많은 연구가 되어 있다. 프레임간의 히스토그램 비교를 통한 방법이나 에지를 이용하여 검출하는 방법, 모션 벡터를 이용하여 검출하는 방법 등이 있다. 본 논문에서 제안할 알고리즘은 이러한 방법들을 통한 샷 경계 정보만이 필요하므로 샷 경계를 검출하는 다른 방법을 써도 무방하다.

3.4 샷 검출을 이용한 보정 방법

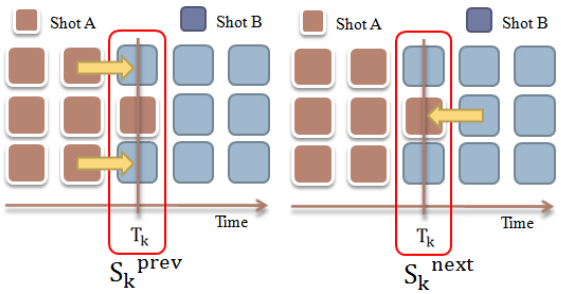

그림 6: 두 가지 이미지 셋 후보

타임 스탬프 기반의 알고리즘을 사용하였을 경우 샷 경계에서 그림과 같은 경우가 생길 수 있다. 이 경우 평가 함수로 선택된 이미지 셋 S_k를 하나의 샷만을 포함하도록 하기 위해 이미지 셋 후보 S_k^{prev}, S_k^{next}를 생각할 수 있다. S_k^{prev} 는 Shot A를 기준으로 하여 수정한 셋 후보이고, S_k^{next} Shot B를 기준으로 하여 수정한 셋 후보이다. 이 두 가지 이미지 셋 후보 S_k^{prev}, S_k^{next}는 서로 다른 샷의 조각이 포함된 기존의 이미지 셋 S_k보다 보기 좋다고 할 수 있지만 $Jitter$를 발생시키는 요인이 된다. 따라서 샷 검출을 이용한 보정이 $Jitter$에 미치는 영향을 최소화 하기 위해서 두 이미지 셋 후보 S_k^{prev}, S_k^{next} 중에서 작은 평가 함수 값을 가지는 이미지 셋을 선택한다. 이렇게 하여 선택한 이미지 셋을 S_k^*라고 한다면 이 이미지 셋 S_k^*는 반드시 $ATS(S_{k-1}) < ATS(S_k^*) < ATS(S_{k+1})$이 되어야 한다. 샷 경계의 이미지가 깨지는 것을 보완하기 위해 다소의 $Jitter$ 손실은 생각할 수 있겠지만, 프레임의 시간 순서는 뒤바뀌어서는 안되기 때문이다.

4 실험 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 7의 방식에 의하여 동기화 한다. 그림 7에서 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 수신측에서의 글로벌 클럭은 다음과 같이 구해진다.

$$GlobalClock = HardwareClock - RT_A + (ST_A + (ST_B - ST_A - (RT_B - RT_A)/2)$$

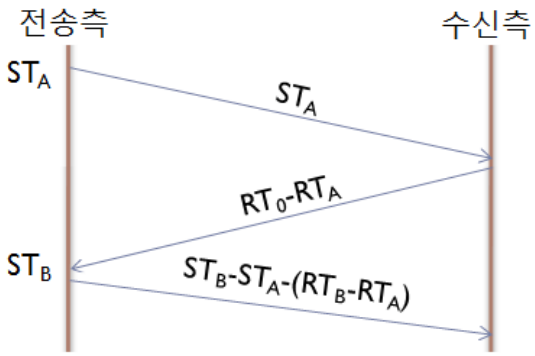

그림 7: 전송측과 수신측의 클럭 동기화 알고리즘

4.2 병합 알고리즘의 분석

타일드 디스플레이 기록기는 실시간으로 동작을 하여야 한다. 하지만 네트워크 전송을 통한 실시간 처리를 하였을 경우 동일한 데이터를 생성하는 것이 불가능 하기 때문에, 오프라인상에서 동작하는 실험용 타일드 디스플레이 기록기를 구현하였다. 구현한 시스템은 네트워크를 통해 기록기로 전송하고, 타일드 디스플레이 기록기는 해당 데이터를 오프라

인으로 저장한다. 각각의 알고리즘은 오프라인으로 저장된 완전히 동일한 데이터를 가지고 테스트 하였다. 실험 환경은 3×2의 타일드 디스플레이 상에서 테스트하였으며, 데이터를 저장할 때에 타일드-디스플레이 동영상 플레이어[1]를 동작시켰다.

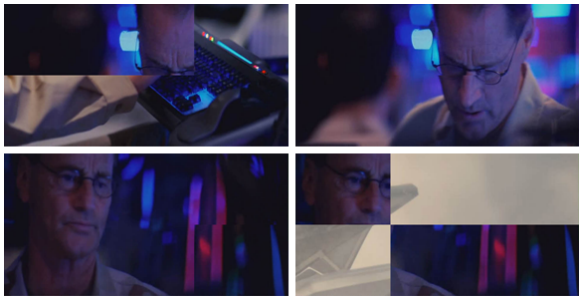

그림 8: 샷 경계에서 잘못 병합되는 예시

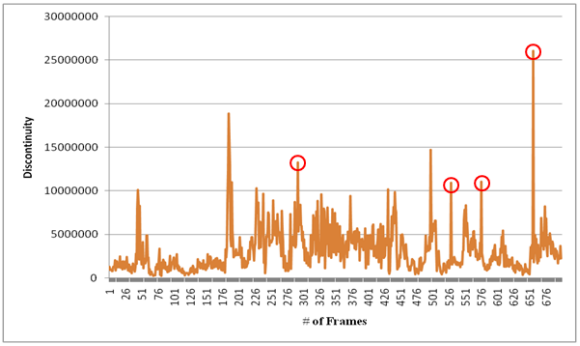

그림 9: 잘못 병합된 위치와 샷 경계 추출

이 데이터를 가지고 타임 스탬프 기반의 병합 알고리즘을 돌린 결과 몇몇 군데에서 그림 8과 같은 잘못 병합된 이미지를 볼 수 있었다. 이러한 문제는 사람의 눈으로 직접 보기 전까지는 문제가 생겼는지의 여부조차 알 수 없다. 실험한 결과를 직접 하나하나 판단해서 세어봤더니, 그림 9에서 동그라미로 표시된 4군데임을 확인 하였다. 이 문제점이 일어난 4곳은 모두 불연속 함수의 peak 부분에서 일어났으며, 알고리즘 설계때 예상한대로 비교적 손쉽게 찾을 수 있었다.

샷 검출과 타임 스탬프를 함께 사용한 병합 알고리즘은 총 6000 프레임을 눈으로 확인하였으나, 타임 스탬프 기반의 병합 알고리즘에서 생겼던 잘못 병합된 이미지를 찾을 수 없었다.

5 결론

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 캡쳐하기 위한 작업이 필요한데, 이 작업은 다른 응용 작업의 프로세스에 따라 오차가 발생하기 때문에 정확히 동기화된 시간에 캡쳐 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 타임 스탬프 기반의 평가 함수를 통해 이미지 조합을 얻을 수 있다. 이 방법은 샷의 경계에서 취약점을 보이기 때문에 이를 보완하기 위해 샷 경계를 검출하여 보완하는 방법을 통해 알고리즘을 개선하였다.

참고문헌

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

[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*, 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*, pp. 202, 2001.

[6] E. Magana, J. Aracil and J. Villadangos, Packet Video Broadcasting with General-Purpose Operating Systems in an Ethernet, *in Proceedings of Multimedia Tools and Applications*, pp. 5-28. Sep. 2004.

[7] 최기석, 낭종호, 타일드 디스플레이 기록기를 위한 병합 알고리즘, *한국정보과학회 종합 학술대회 논문집*, 제25권 제1호(A), 2008.

[8] Kikukawa T., Kawafuchi S., Development of an automatic summary editing system for the audio visual resource, *Transactions of the Institute of Electronics, Information and Communication Engineers*, Vol. J75-A, No.2, 1992.