캡쳐 간격 평활화에 기반한 비디오 동기화 알고리즘의 구현

권오성*

요약

통신, 화면 캡쳐 등에서 연속적인 이미지 입력과 음성 신호를 받아들여 비디오를 제작하는 경우에 종종 오디오와 비디오 스트림 간의 타이밍 불일치 현상이 발생한다. 본 논문에서는 이러한 AV 동기화 문제를 해결하기 위하여 오디오 시간을 기준으로 캡쳐 이미지 입력 간격을 일정하게 평활화하는 알고리즘을 제안하고 구현하였다. 제안하는 알고리즘은 연속적인 화면 캡쳐와 오디오 채널을 입력으로 하는 비디오 제작 상황에 한정하여 적용할 수 있다. 본 논문의 제안 방법의 성능을 평가하기 위하여 출력 단위 내에서 비디오와 오디오 신호의 동기화, 비동기화 구간의 평활화 등을 실험하였다. 실험은 비디어 작성 시에 요구되는 삽입 프레임 수의 변화율, 삽입 타이머 작동에 따른 보정율 등으로 이루어 졌으며 제안하는 평활화 방법이 AV 동기화 문제를 최소화하고 컴퓨터 화면 녹화 등 동영상 제작 등에서 실용성을 갖춘 적합한 적용 방법임을 확인하였다.

Implementation of Video Synchronization Algorithm Using Capture Interval Smoothing

Kwon Oh Sung*

ABSTRACT

When we are making a movie file from screen captures and an audio stream, it happens some inconsistency problem between audio and video streams. The audio stream comes from a sound card equipped at a stand-alone PC, In this paper, we propose a new smoothing algorithm between frame intervals of video stream and audio stream. Our proposed method can control the number of frames of video dynamically to solve the inconsistency of one video and one audio stream. We explain implementation method for the algorithm and experiment the efficiency of AV-Timing. We examined the ratio of inserted frames of resulted video files in a unit time, frame insertion interval and correction ratio, etc. In experimental, we confirmed that our proposed algorithm is the method minimizing the lip-synch AV-Timing problems in recording computer screen and movie making.

Key Words : A-V Synchronization, FPS, Audio Timing, Real-Time Adaption, Movie Making, Time Stamp

* 공주교육대학교 초등컴퓨터교육과(✉oskwon@gjue.ac.kr)
· 제1저자(First Author) : 권오성 · 교신저자(Correspondent Author) : 권오성
· 접수일(2011년 7월 7일), 수정일(1차 : 2011년 8월 5일), 게재확정일(2011년 8월 8일)

Ⅰ. 서 론

일반적으로 동영상 제작, 오디오 믹싱 과정에서 제작 환경과 시스템적인 요인 때문에 오디오와 비디오 스트림 비동기화 문제가 발생하기도 한다. 본 논문에서는 이러한 비동기화 문제를 효과적으로 해결하기 위한 방법을 제안하고자 한다. 제안하는 알고리즘은 연속적인 화면 캡쳐를 통하여 이미지 프레임과 구별된 채널로부터 입력되는 사운드 신호를 결합하여 출력하는 상황에 한정하여 적용할 수 있다.

일반적으로 AV 동기화는 캠코더 촬영이나 네트워크를 이용한 미디어 전송 시에도 발생하는 데, 이러한 동기 불일치 문제를 해결하기 위한 다양한 연구가 있었다[1]. 일반적으로 AV-동기를 위한 타이밍을 유지하는 방법은 인터레이스된 비디오와 오디오의 형식을 유지하는 방법이 있을 수 있고, 또 다른 방식은 명확한 자료의 동기화를 표시하는 신호 정보를 이용하는 방법 등을 이용한다[1-6,9]. 이 경우 전송 데이터 스트림 패킷의 처리는 패킷의 타이밍을 준수해야 하는데 보통 접수 데이터의 보간 혹은 확장 등을 이용한다.

본 논문에서 제안하는 알고리즘은 연속적으로 입력되는 정지 영상을 출력 비디오의 프레임 이미지로 사용하도록 하였다. 이 경우 정지 영상은 화면 캡쳐 동작으로 추출하는 데, 화면 캡쳐의 주기, 이미지 압축률, 시스템 리소스 상황 등에 기인한 문제 때문에 AV-동기화 문제가 종종 발생한다.

본 논문에서는 먼저 AV-동기화와 관련한 기존의 연구 결과들을 살펴본다. 또한, 캡쳐 간격 평활화를 통한 오디오와 비디오 스트림의 동기화 불일치 해소 방안과 이를 구체화하는 알고리즘을 설명한다. 다음에 제안하는 알고리즘의 효율성을 증명하기 위한 다양한 실험과 결과를 보이고자 한다. 끝으로 결론을 제시한다.

Ⅱ. AV-동기화 관련 연구

동영상 제작 과정에서 흔히 오디오와 비디오 스트림 동기에 관련한 문제가 발생하며, 이와 관련한 다양한 연구가 있었왔다.

오디오 동기화 불일치는 다양한 원인으로 발생하는 데, 크게 다음 두 가지 이유로 요약될 수 있다. 첫 번째로 내부적인 AV-동기화 오류를 들 수 있다. 예를 들어 비디오 카메라와 마이크로폰에서 이미지와 사운드 사이의 싱크 지연을 들 수 있다. 두 번째 경우는 외부적인 AV-동기화 오류의 경우인데 마이크로폰이 음원으로부터 원거리에 있는 경우, 소리가 광속보다 느리기 때문에 싱크 불일치 문제가 발생한다. 이 경우 거리에 비례하여 불일치의 정도가 심해진다.

또한, 동기화 문제는 네트워크 상태의 비디오 재생에서도 발생한다. 오디오 및 비디오 데이터는 연속적이고 일정한 간격으로 생성되므로 목적지에서 이것들을 동기화시켜서 올바른 재생이 보장되도록 배열해야만 한다[5,7]. 그러나, 네트워크 상의 전송 지연으로 인하여 연속적인 순서화가 담보되기 어렵다.

연구[5]에서는 멀티미디어 통신의 미디어 통기화를 위하여 NPT(Normal Play Time)를 이용하여 동기화하므로써 네트워크에 유입되는 제어 트래픽의 양을 줄이는 방법을 실험하였다.

연구 [7]에서는 버퍼를 두어 완충재를 두고 연속 재생을 보장하는 알고리즘을 설계하였다. 이 경우, 동기화는 오디오와 비디오 데이터 스트림 간의 외부적 동기화와 단일 스트림 내에서 패킷의 순서화를 위한 내부적 동기화로 나누어 볼 수 있으며, 이 두 가지가 모두 만족되어야 한다.

연구 [2]에서는 MPEG-1 디코더의 구현에서 필요한 동기화 방법을 제안하였다. 서로 다른 불규칙한 압축율로 생성된 MPEG-1 스트림을 보다 작은 쓰래드 단위로 나누어 처리하는 방식이다. MPEG-1 디코더에서 쓰래드들 간의 데이터 전달에 사용되는 공유 버퍼를 이

용하여 오디오와 비디오 스트림의 동기화를 수행하였다[2]. 이 연구에서는 연구[7]와 유사한 공유 버퍼를 두어 동기화하는 방법을 사용하였으며, MPEG-1 압축과 전송의 경우로 보다 구체화된 연구 성과를 보였다.

또 다른 동기화 연구는 비디오와 오디오에 대한 NPT(Normal Play Time)를 이용하여 동기화하는 방법이다. 기본원리는 출력 예정인 비디오 화면의 NPT와 오디오 데이터 간의 NPT를 비교하여 비디오 화면의 출력 간격을 조절하는 것이다[3]. 수신된 RTP(Real Time Clock) 패킷으로부터 화면별 타임스탬프, B-픽처를 고려하여 화면 순서를 정렬하는 방법을 사용한다. RTP 패킷 헤더로부터 오디오 프레임의 RTP 타임스탬프를 추출한다. 화면과 동기화를 이룰 예정의 오디오 데이터 블록(s 번째)을 비교하여 그 차이를 계산하고 비디오 화면간의 출력 시간 간격을 조정하여 동기화하는 방식을 사용하였다[3].

AV-동기화 문제는 비디오를 편집하는 경우에도 발생한다. 그러므로 편집기는 이를 보정하기 위한 보완 방법이 요구된다. 연구 [8] 에서는 동기화용 타임 스탬프을 생성하는 방법을 구현하였다. 타임스탬프 정보는 사용자가 지정한 상영 시간 정보와 입력 Mpeg 스트림 간의 시간 정보의 분석에 의해 진행한다[8]. 이 방법에서 가장 우선적으로 비디오와 오디오 스트림을 패킷(적당한 크기)으로 나누어 전송하며, 이 패킷의 헤더에는 오디오/비디오 데이터 여부, 동기화를 위한 시간 참조 정보를 포함한다.

연구 [6]에서는 스마트 카메라에서 발생하는 AV-비동기 문제를 해결하기 위하여 RTC을 이용하는 방법을 설계하였다.

III. 캡쳐 간격 평활화

본 논문은 화면 캡쳐로 들어오는 연속적인 스틸 이미지를 입력으로 비디오를 출력하기 위한 방법에 관한 것이다. 이때 입력되는 캡쳐 주기는 여러 가지 원인으로 인하여 규칙적이지 않을 수 있다.

본 논문에서는 이러한 캡쳐 주기의 불규칙함을 평활화하여 균등화하는 알고리즘을 제안한다. 이러한 평활화 알고리즘의 수행은 AV-동기화의 문제를 해결하기 위한 것이다. 그림 1 은 비디오 캡쳐로부터 입력되는 이미지 생성 간격이 일정하지 않음을 나타낸 것이다. 입력 이미지 img0, ... ,img n 은 동일 디멘젼 크기의 화면 캡쳐 파일이다. 이러한 이미지 캡쳐 스트림 내의 프레임 수는 상대적으로 안정적인 오디오 스트림을 기준으로 조정된다. 예를 들어 오디오 경과 시간에 비하여 비디오 캡쳐 수가 충분하지 못할 경우에 부족한 분량만큼 바로 전단계 캡쳐 이미지를 보충하는 방식이다. 이러한 전략은 녹화 단위 시간별 저장 프레임의 수를 균일하게 하는 평활화 효과를 만든다.

캡쳐 간격의 불일치는 주로 다음의 원인으로 발생한다.

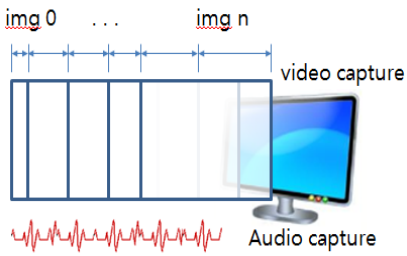

그림 1. 비디오와 오디오 캡쳐
Fig 1. Capture of Video and Audio Sources

- 비디오 코덱(Codec)의 프레임 압축과정의 시간 지연
- 가용 시스템 리소스의 제약
- 타 응용 작동에 의한 인터럽트 발생으로 인한 시간 지연
- 변환 중간 결과의 저장에 따른 시간 지연

결국, 캡쳐 간격 활성화 알고리즘은 화면 녹화 시 시간별로 일정한 수의 프레임이 저장되도록 하는 실시간 프레임 추가 보정 기능을 수행한다.

```
nCurFrameNumber = fTime * FRAME_RATE;
nNewFrameNum=nCurFrameNumber - nAppendedNum;
for(long i=0; i<nNewFrameNum; i++) {
 AppendNewFrame();
 nAppendedNum++;
}
```

위 알고리즘에서 *fTime* 은 기준이 되는 현재 녹화 시점을 의미하고, *nCurFrameNumber* 은 현재 시점에 맞는 올바른 프레임 수이다. 이 값은 출력 무비가 유지하고자 하는 초당 프레임 수인 (*FRAME_RATE*)에 의해서 계산되는 누적 프레임 수이다. *nNewFrameNum* 는 현 시점의 이상적인 프레임 수와 현재까지 누적된 프레임 수를 비교하여 차이값을 구한 것이다. 이 값은 AV-동기화를 위해서 사용된다. 이러한 과정의 반복은 구간별 프레임수를 동일하게 유지하여 결국은 캡쳐 간격을 균등하게 하는 평활화 효과를 만든다. 결국 본 알고리즘을 적용하면 여러 가지 이유로 캡쳐 간격이 균등하지 못한 상황에서 이를 해소하는 평활화 효과를 얻을 수 있고, 실시간 AV-동기화 보정이 가능하다.

Ⅳ. 단위 녹화 파일 생성

본 알고리즘의 구현은 Ⅲ 장에서 언급한 다양한 원인으로 인하여 1개의 녹화 파일로 장시간 출력하는 경우 종종 문제를 발생시킨다. 이를 해결하기 위하여 그림 2 처럼 녹화 시간 전체를 1 개가 아닌 복수개로 나눈 뒤 별도의 파일로 저장하였다.

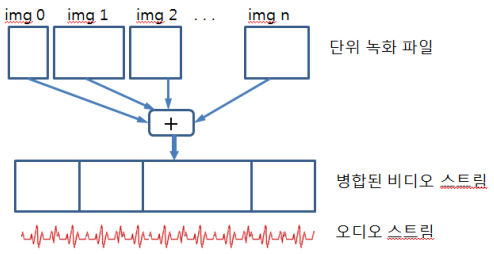

그림 2. 단위 녹화 파일의 병합
Fig 2. Merge of the Unit Recorded Files

그림 3 은 녹화 진행 중의 오디오와 비디오 간의 동기화 과정을 보여주며 y축은 적용된 프레임 저장 보정 개수를 보여준다. 이 그림에서 주기적으로 나타나는 불연속 부분은 저장 단위 파일이 변경되는 부분을 보여주고 있다.

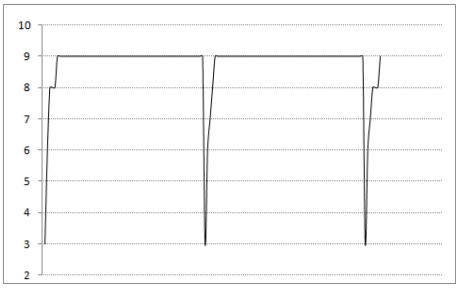

그림 3. 녹화파일의 구간별 오디오 동기화 보정
Fig 3. Correction of Audio Syncronization by the Intervals of Recorded File

그림 3 에서 보듯이 새로운 단위 녹화 결과의 생성은 초반 안정화 시간을 필요로 한다. 이러한 초반의 짧은 진동에서 집중적인 보정과정이 필요하다.

단위 녹화 파일의 크기는 작게 유지하는 경우 구간별 초반 불완전 부분이 많아지는 부작용이 있어서 비디오 재생 시 끊김 현상을 발생시킬 수 있다. 반면에 지나치게 크게 유지하는 경우는 구간 후반에서 누적에 의한 미세한 오디오 비동기화가 발생할 수 있다.

본 논문의 알고리즘은 효율적인 AV 동기화를 위하

여 단위 시간 내의 녹화 결과를 생성하고 후처리로 이러한 비디오 녹화 조각들을 병합하는 방식을 사용하였다. 이러한 녹화 결과의 결합을 위해서 본 논문에서는 Avisynth 도구를 사용하였다. AviSynth 는 오픈 소스 도구로서 비디오 편집을 위하여 일반적으로 사용되고 있다. AviSynth 스크립트 언어는 다양한 비디오 필터링 기능을 제공한다[10,11].

V. 실험 및 성능 평가

본 논문에서는 제안하는 알고리즘의 효율성 평가를 위하여 다양한 측면의 성능 평가와 이러한 평가를 바탕으로 실제 알고리즘을 구현하였다.

5.1 알고리즘의 성능 평가

그림 4 는 비디오 생성을 위한 프레임 저장 시간 간격(50, 100, 200 millisec.)을 달리하며 제안하는 평활화 알고리즘을 적용한 결과이다. 그림 4 에서 y축은 평활화 수행시 필요한 보정용 프레임 개수를 표시하며, 이 값을 시간 간격에 비례하여 회수를 재계산한 것이다. 예를 들여 주기 50 milli 의 경우는 200 milli 경우로 환산하기 위하여 4회 누적하여 표시하였다.

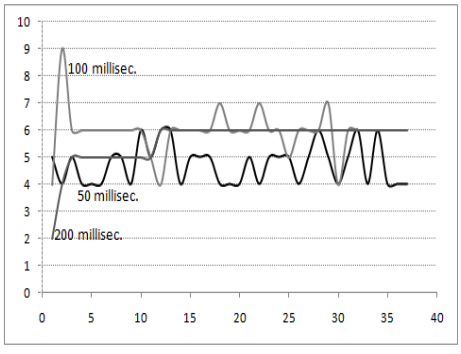

그림 4. 프레임 저장 주기별 보정값 변화 흐름 비교
Fig 4. Flow Comparision of Correction Value by the Frame saving Cycle

주기의 변화율을 비교한 결과 주기가 작은 경우(50 milliseconds)가 주기가 큰 경우(200 milliseconds)보다 변화율이 작은 것을 알 수 있다. 변화율이 작다는 것은 보정해야 할 프레임의 개수가 상대적으로 적어진다는 것을 의미하므로 보다 자연스러운 비디오의 흐름을 만들 수 있다. 그러나, 주기값이 작은 경우는 필요한 프레임을 저장하는 경우 충분한 시간을 확보하지 못하여 시스템 자원 제약 등 외부적 인터럽트 시 끊김 현상 등이 드물게 발생하는 것이 관찰되었다.

표 1 은 시간 주기별로 보정되는 프레임수를 나타낸 것이다. 필요한 프레임 수의 변동 폭이 진동하다가 안정화되는 것을 알 수 있다. 이러한 불완전 부분 때문에 전체적인 오디오 비동기화가 발생한다. 초반부의 오디오 동기화 실패는 비디오 전체 구간에 영향을 주고 치명적인 립 싱크 문제를 만든다. 앞서 설명하는 평활화 알고리즘을 구간별 초반 구간에 적용하여 이러한 문제점을 해결할 수 있었다.

표 1. 시간별 이미지 프레임 요구
Table 1. Requirements of Image Frames during Program Running

Sec.	Cap 0	Cap 1	Cap 2	Cap 3	Cap 4	Cap 5	Cap 6	Cap 7	Cap 8	Cap 9
0.24	3	3	3	3	3	3	3	3	3	2
0.56	6	6	6	6	6	6	6	6	6	6
0.88	8	7	7	8	8	7	8	8	7	7
1.18	8	8	8	8	8	8	8	8	8	8
1.49	8	9	8	9	9	8	8	8	9	9
1.81	9	9	8	9	9	8	9	9	9	9
2.13	9	9	9	9	9	9	9	9	9	9
2.42	9	9	9	9	9	9	9	9	9	9
2.74	9	9	9	9	9	9	9	9	9	9
3.06	9	9	9	9	9	9	9	9	9	9
3.36	9	9	9	9	9	9	9	9	9	9
3.67	9	9	9	9	9	9	9	9	9	9
3.99	9	9	9	9	9	9	9	9	9	9
...	...	...	...	...	...	...	...	...	...	...
18.96	9	9	9	9	9	9	9	9	9	9
19.27	9	9	9	9	9	9	9	9	9	9
19.59	9	9	9	9	9	9	9	9	9	9
19.91	9	9	9	9	9	9	9	9	9	9
20.00	9	9	9	9	9	9	9	9	9	9

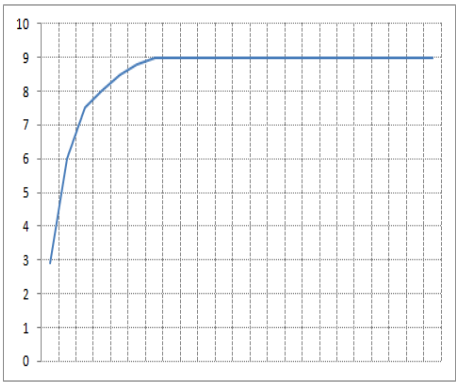

그림 5 . 시간별 이미지 프레임의 평균값
Fig 5. Average Value of Image Frames by the Seconds

그림 5 는 표 1의 AV 동기화를 위해 사용된 프레임 삽입수를 평균하여 나타낸 것이다. 그림 5 에서 보듯이 프로그램 실행 초기에는 프로그램이 녹화를 위한 준비작업 등의 원인으로 심한 동기화 불일치를 발생시킬 수 있음을 알 수 있다.

5.2 제안 알고리즘의 구현

캡쳐 이미지를 비디오 파일로 저장하기 위한 코덱으로는 xvid 를 사용하였다.

오디오카드 검색을 통하여 오디오 카드의 특성을 조사하고 녹음 기능의 유무를 우선 확인하도록 구현되었다. 제안하는 알고리즘은 오디오 녹음 시간을 기준으로 삼아 비디오 파일 생성을 진행하기 때문에 오디오 카드의 녹음 지원이 필수적이다.

본 알고리즘은 VC++로 구현되었고 윈도우즈7과 xp에서 성능을 검증하였다. 화면 녹화를 진행하기 위해서는 캡쳐 영역의 설정, 캡쳐의 시작 등을 설정할 수 있어야 한다. 이를 위하여 그림 6 과 같은 인터페이스를 구현하였다. 그림 6 에서 보는 사각틀은 녹화 영역의 경계를 표시하며, 이 영역 안의 모든 변화가 비디오 파일로 저장된다. 사각틀 내부는 투명으로 처리하여 녹화 내용을 볼 수 있도록 하였다. 녹화 완료시를 위한 이벤트 발생을 위하여 화면 하단의 작업표시줄 우편에 '녹화완료' 버튼을 위치시켰다.

그림 6. 구현 프로그램의 시작화면
Fig 6. Starting Screen of the Implemented Program.

사용자가 녹화 완료시 이 버튼을 클릭하면 '녹화완료' 이벤트를 발생되고 그림 7과 같은 변환작업창이 나타난다. 이 변환 과정에선 오디오와 비디오의 동기화, 단위 녹화 파일들의 병합 등의 작업을 수행한다.

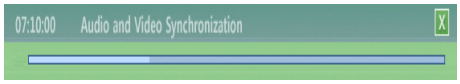

그림 7. 단위 출력 파일의 결합과정
Fig 7. Merge Process of Unit Video Files

녹화 결과는 자동 재생되도록 프로그래밍하였다. 그림 8 은 무비 플레이어의 화면이다. 플레이어는 다양한 재생 기능을 제공한다.

본 구현 프로그램의 실행결과 표 1 에서 보듯이 당초 균일하지 못했던 구간별 이미지 프레임 수를 평활화 알고리즘을 통하여 균일하게 유지할 수 있었고, AV-동기화가 가능하였다.

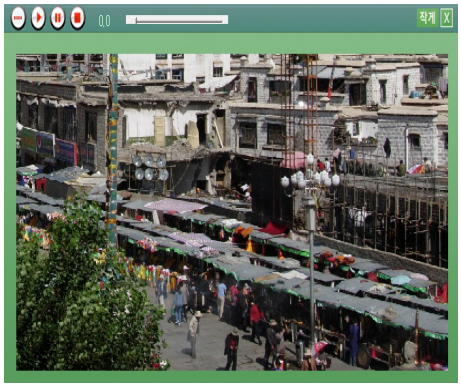

그림 8. 출력 비디오 재생기
Fig 8. Player for Output Video

Ⅵ. 결 론

본 논문은 비디오 제작, 특히 화면 캡쳐의 반복으로 이미지 프레임을 생산하여 비디오를 제작하는 경우에 발생할 수 있는 AV-동기화 알고리즘에 관한 것이다. 이를 위하여 본 논문에서는 전체 녹화를 단위 구간 녹화 단위 파일로 생성하고 이를 후처리로 병합하여 출력하는 방법을 제안하였다. 단위 녹화 결과의 비디오와 오디오의 동기화를 위하여 본 논문에서는 녹음 시간을 기준으로 동적으로 화면 캡쳐 간격을 조정하는 캡쳐 구간 평활화 방법을 제안하였다.

본 구현 알고리즘은 초당 프레임 수를 일정하게 유지하기 위하여 시간별 프레임수를 계산하고 보정하는 평활화 방법을 사용하였다. 알고리즘의 성능 평가를 위한 다양한 실험을 시행하였다. 실험은 비디어 작성 시에 요구되는 삽입 프레임 수의 변화율, 삽입 타이머 작동에 따른 보정율 등으로 시행하였는 데, 실험결과 간혹 발생하는 비동기적 사건의 부작용을 보완할 수 있었고 AV동기화와 오디오 싱크가 보장되는 비디오 출력이 가능하였다.

현재 제안하는 단위 파일별 평활화 방법은 장시간의 녹화 파일 생성과 AV 동기화를 위한 효율성이 실험결과 확인되었지만, 단위 비디오 파일의 연결 과정에서 약간의 오디오 잡음이 나타나는 문제점이 나타나기도 하였다. 이를 해결하기 위한 추가적인 연구가 필요하다고 본다.

본 알고리즘은 자유롭게 화면 녹화 결과를 생성할 수 있으므로 PC 상에서 유용한 유틸리티 도구로 활용이 가능하다. 또한, 컴퓨터 화면 재생을 교육자료로 제작하거나, 비디오 녹화 등의 컨텐츠 제작 등에 활용될 수 있을 것이다.

참고문헌

[1]http://en.wikipedia.org/wiki/ Audio_video_sync, “Audio to video synchronization”,2011

[2] 박태강, 이호석, 다중 쓰래딩 기법의 MPEG - 1 디코더에서 공유버퍼를 이용한 오디오 / 비디오 스트림의 동기화,학술발표논문집, 한국정보과학회, 제26권,제2호, pp.221-223, 1999.

[3] 서광덕, 지원섭, 정순홍“,RTP Packet 기반의 정밀한 오디오/비디오 동기화 기법”,종합학술대회논문집, 제35권,제1호, 한국정보과학회, pp.201-202, 2008.

[4] 이승연, 김정국, “실시간 운영체제 TMO-Linux의 IEEE1394 분산 IPC 기반 다중 채널 오디오 동기화 스트리밍 시스템”, 한국정보과학회 2008 가을 학술발표논문집,한국정보과학회, 제35권 제2호(B), pp. 421-425, 2008.

[5] 서광덕, 지원섭, 정순홍, "멀티미디어 통신을 위한 RTP 패킷 기반의 정밀한 오디오/비디오 동기화 기법", JOURNAL OF KOREA MULTIMEDIA SOCIETY, Vol.12 No.5, pp. 617-776, 2009.

[4] 최종근, 이병은, 정선태,“다빈치 프로세서 기반 스마트 카메라에서의 비디오/오디오 스트림간 동기화 설계 및 구현“,하계종합학술대회,대한전자공학회,pp.597-598, 2009.

[5] 박홍진, 이시진, 김영찬,"멀티미디어 동기화를 위한 버퍼 크기와 재생 보정 알고리즘 설계", 봄 학술발표논

문집, 제22권, 제1호 한국정보과학회, pp. 411-414, 1995.
[6] 신명준, 낭종호, "MPEG 시스템 스트림 편집기의 설계 및 구현", 한국정보과학회, 학술발표논문집, 제24권, 제1호, pp.611-614, 1997.
[7] 이창욱, 박용진, "멀티미디어 그룹통신을 위한 오디오 믹싱",학술발표논문집,한국정보과학회, 제23권, 제1호, pp.481-484, 1996.
[8] "The full AviSynth grammar", http://avisynth.org/mediawiki/AviSynth_grammar, 2011.
[9] "The full AviSynth grammar : Simple Examples", http://avisynth.org/mediawiki/Script_examples, 2011.

저자소개

권오성 (Kwon, Oh-Sung)

1994년 중앙대학교 컴퓨터공학과
공학박사

1995년~현재 공주교육대학교 초등컴퓨터교육과 교수,
관심분야 : 멀티미디어 처리, 영상처리 및 패턴 인식