소켓 통신을 통한 아이폰에서의 이미지 및 동영상 컨텐츠 출력 시스템

김영두* · 조태훈**

한국기술교육대학교

Realtime multimedia contents display system in Iphone

Youngdoo Kim* · Tai-Hoon Cho**

Korea University of Technology and Education

E-mail : foxkyd@kut.ac.kr*, thcho@kut.ac.kr**

요 약

최근 애플의 아이폰과 구글의 안드로이드폰 출시로 스마트 폰에 대한 관심이 증가하고 있다. 일반 PC 어플리케이션과 마찬가지로 스마트 폰에서도 멀티미디어 컨텐츠의 활용은 중요할 것이다. 여기서 멀티미디어 컨텐츠란 대표적으로 이미지와 동영상을 말한다. 이미지와 동영상 컨텐츠를 위해 사용되는 대표적인 코덱으로 JPEG과 MPEG 그리고 최근에 동영상 코덱으로 각광 받고 있는 H.264가 있다. 본 논문에서는 JPEG와 H.264로 압축된 raw sequences를 사용한다. 본 논문은 Iphone의 LCD상에 소켓으로부터 들어오는 Jpeg과 H.264 raw sequences를 어떻게 효율적으로 디코딩 및 출력해 줄 것인지에 대해서 제시한다. 그리고 이를 실제 구현을 통해서 시뮬레이션한 결과를 보여준다.

ABSTRACT

Recently, there is increasing concern about smart phone due to the release of Iphone and Android phone. Like the normal desktop application, using multimedia contents will have important role for smart phone application. The multimedia contents mentioned here mean Image and Video. The famous codecs for Image and video are Jpeg Mpeg and h.264, which is popular recently as a vedeo codec because of its high compression ratio compared with the other video codecs. In this paper, we will use Jpeg and H.264 encoded raw sequences. This paper will suggest how to decode and display effectively Jpeg and H.264 raw sequences arrived from socket stream. And we will show you the simulation result.

키워드

Jpeg, H.264, Iphone, Socket, Server-client, TCP/IP

Ⅰ. 서 론

최근에 Iphone과 Android폰의 출시로 스마트 폰에 대한 관심이 증가하고 있다. 그리고 스마트 폰에서 멀티미디어 컨텐츠, 대표적으로 동영상 컨텐츠를 처리하는 것은 일반 PC용 어플리케이션에서처럼 자주 요구될 것이다.

멀티미디어 컨텐츠를 처리하기 위해 Iphone 은 이미지를 처리하는 SDK와 동영상을 처리하기 위한 SDK를 제공해 주고 있다. [1][2]

대표적으로 Iphone에서 제공해 되는 UIImage 클래스를 사용하면, bmp, png, gif, jpeg등의 이미지 처리를 쉽게 할 수 있다.

하지만 Iphone SDK는 동영상 컨텐츠를 처리하

기 위해 send box내의 동영상 파일과 웹 스트림으로부터 들어오는 데이터만을 처리해 줄 수 있다. 즉, 파일과 웹 스트림을 제외한 다른 형태의 스트림으로부터 들어오는 동영상 데이터를 처리하는 것은 어렵다. [3]

이를 위해 Jpeg을 이용하여 동영상과 같은 처리를 하는 방법과 H.264에 대한 디코딩을 직접 핸들링 할 수 있도록 Iphone상에 직접 디코딩 모듈을 포팅하여 동영상 데이터를 처리하기 위한 방법을 제시한다.

Ⅱ. 기본 구조

동영상 데이터를 보내고 디스플레이 하기 위해서 기본적으로 그림 1과 같이 Server-Client 구조를 이용하였다.

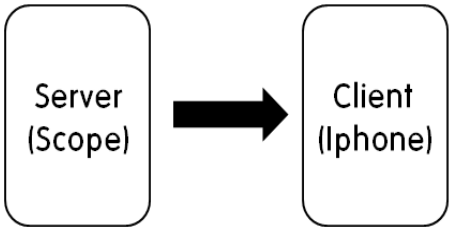

그림 1. 기본 구조

스코프라는 장비를 통해서 카메라로부터 영상정보를 획득하고 이 영상을 인코딩한다. 이 장비는 Jpeg이나 H.264으로 인코딩을 선택할 수 있다.

인코딩된 raw sequences는 소켓 스트림을 통해서 Iphone으로 전송이 된 후, 디코딩 과정을 거쳐 Iphone의 Lcd에 display된다.

여기서 스코프는 서버가 되고 Iphone은 클라이언트가 된다. 그리고 소켓스트림은 TCP/IP를 트랜스포트 계층으로 이용한다.

Ⅲ. Jpeg과 H.264

Jpeg과 H.264는 각각의 이미지와 동영상데이터를 인코딩 및 디코딩하기 위해 사용되는 코덱이다.

동영상 압축의 과정은 이미지의 압축에 비해 상당히 복잡하지만, 기본적으로 이미지 압축을 위해 사용되는 과정과 유사하다.

영상의 압축은 RGB->YUV변환, 메크로 블록화, DCT, 양자화, 엔트로피 코딩, 부호화의 단계를 거친다. 여기에 영상의 경우 프레임간의 유사성에 의한 차이를 압축하는 P프레임 부호화 방식이 사용된다. [4]

Jpeg는 이미지 데이터를 위해 널리 사용되는 코덱이다. 때문에 대부분의 플랫폼에서 기본적으로 디코딩을 위한 API를 제공해 주기 때문에 어렵지 않게 Jpeg 이미지를 처리하기 위한 구현을 할 수 있는 장점이 있다.

H.264는 최근에 각광받고 있는 동영상 코덱이다. H.264의 장점은 기존의 동영상 압축 코덱(Mpeg-2, Mpeg-4)에 비해 뛰어난 압축률을 제공해 준다는 점이다.

H.264는 같은 압축 비율에 기존의 코덱들 보다 나은 화질을 유지한다. 예로 한 장의 DVD에 Mpeg-2포멧의 압축된 파일이 2시간 가량의 영상 데이터를 저장할 수 있다면 H.264 포맷의 압축 파일일 경우 4시간 가량의 영상 데이터를 저장할 수 있다. [5]

하지만 H.264를 Iphone에서 Display하기 위해서는 웹 스트림으로 들어온 데이터나 send box내의 파일 데이터만을 처리할 수 있다. 즉, H.264를 처리하기 위해 직접 H.264 디코딩 API를 접근할 수 없다.

이를 해결하기 위해서 H.264를 디코딩 할 수 있는 모듈을 직접 Iphone상에서 구현한 후에 소켓 스트림 및 디스플레이 부분과 연결해 주어야 하는 작업이 필요하다.

Ⅳ. Jpeg 영상의 처리

일반적으로 영상데이터는 1초에 30프레임 이상의 출력이 요구된다. 이는 사람의 눈이 연속적인 영상이라고 느끼기 위해서 요구되는 프레임의 양이다.

데이터의 전송을 위해서 소켓을 사용하기 때문에 무선랜으로 통신할 경우 가능한 대역폭을 1Mbps 라고 하자.

그리고 일반적으로 스코프에서 320X240의 이미지를 최고의 Jpeg 화질로 압축할 경우 15kbytes의 이미지 사이즈를 갖는다.

이론적으로 인코딩과 디코딩 및 전송과정에서의 딜레이를 제외하면 1초에 68프레임 정도를 디스플레이 할 수 있다.

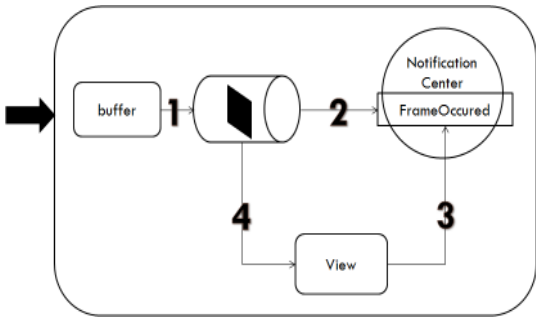

그림 2. Jpeg 영상 처리 구조의 Client부분

그림 2는 Iphone상에서 Jpeg을 처리하기 위해 사용한 구조이다.

소켓을 통해 들어온 연속된 Jpeg raw

sequences는 먼저 버퍼에 저장된 후 한 장의 Jpeg 프레임 단위로 큐에 삽입된다. 큐에 Jpeg 프레임이 존재한다면 self 객체로부터 통보 센터(Notification Center)에 한 장의 프레임이 버퍼상에 존재한다는 메시지를 보내준다. 여기서 통보를 위해 사용한 이름은 FrameOccured라고 임의로 지정해 주었다.

통보센터에는 처음에 커스텀 뷰 객체를 생성한 후 FrameOccured 메시지를 감시하는 Observer로 등록해 놓았기 때문에 FrameOccured메시지가 발생할 때마다 커스텀 뷰를 통해 버퍼에 있는 jpeg raw sequences를 디코딩한 후 출력 하게 된다. 각 과정은 그림 2의 번호 순서대로 진행된다.

이렇게 실제 구현을 통해 테스트해 본 결과 실제로 인코딩과 디코딩 과정에서의 처리 시간과 데이터 전송시에 발생하는 딜레이로 인해서 1초에 40~50프레임 사이의 jpeg 영상만을 출력할 수 있다.

그렇지만 이러한 프레임 rate 역시 사람이 시각적으로 연속된 영상이라고 느낄 수 있기 때문에 Iphone상에서 소켓으로부터 들어오는 영상데이터를 처리하기 위해서 Jpeg을 이용할 수 있다.

Ⅴ. H.264의 처리

Iphone에서 H.264를 처리하는 것은 H.264 디코딩 API에 직접 접근할 수 가 없기 때문에 직접 디코딩 모듈을 구현해야 한다.

여기서는 H.264를 위해 제공되는 오픈 소스인 JM version17.0을 사용하여 디코딩 모듈을 구현하였다. [6]

JM은 C언어로 구현되어 있기 때문에 플랫폼에 제약 없이 C를 지원해주는 곳에서 포팅해서 사용할 수 있다.

특히 Iphone같은 경우 Objective-C를 사용하지만 C언어와의 호환이 가능하기 때문에 JM의 코드 수정을 최소화 하면서 바로 디코딩 모듈을 포팅해서 이용할 수 있다.

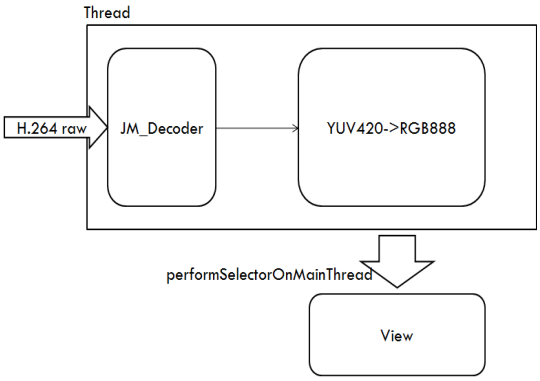

그림 3. H.264 영상 처리 부분의 Client부분

그림 3은 Iphone상에서 구현한 H.264를 처리하기 위해 사용한 구조이다.

H.264를 처리하기 위한 별도의 스레드를 하나 생성한 후 소켓에서 들어온 H.264 raw sequences를 YUV420으로 디코딩한 후 YUV420을 RGB888 색상으로 변환하는 부분까지를 처리하도록 했다.

여기서 RGB로 변환된 정보를 버퍼에 담아 바로 View로 출력하게 될 경우 while 루프 상에서 디코딩이 이루어지기 때문에 View상에 바로 디스플레이 되지 않는 문제가 발생한다.

이를 해결하기 위해 performSelectorOnMainThread 함수를 이용해서 디코딩이 처리되는 스레드 안에서 View를 갱신하기 위한 임의의 Selector를 지정한 후에 이 Selector안에서 View의 갱신이 이루어지도록 처리를 해야 한다.

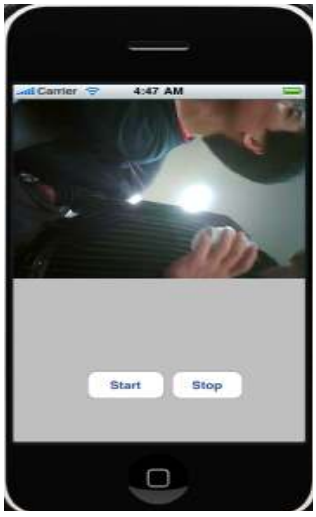

그림 4. H.264 출력 결과

그림 4에서 실제 구현을 통하여 Iphone의 시뮬레이터 상에서 소켓으로부터 들어오는 H.264 raw sequences에 대한 처리를 할 수 있음을 볼 수 있다.

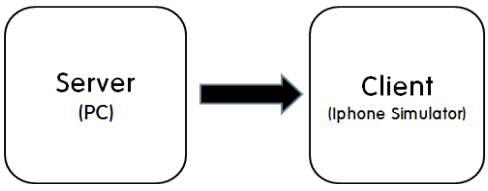

그림 5. 시뮬레이션을 위한 Server-Client 구조

본 논문에서 사용한 시뮬레이션은 일반 데스크탑 PC를 Scope를 대체한 서버로 사용 하였고 Iphone Simulator를 Iphone Device를 대체한 클라이언트로 사용하였다.

vi. 결 론

지금까지 Iphone에서 멀티미디어 컨텐츠, 동영상을 처리하기 위해서 Jpeg과 H.264를 이용한 모델을 제시했고 실제 구현을 통해서 시뮬레이션한 결과를 보였다.

이 논문을 통해서 Jpeg을 이용해서 동영상 컨텐츠를 처리할 수 있음을 보여 주었다. Iphone에서 H.264를 처리하는데 직접 소켓에서 들어오는 데이터를 처리할 수 없기 때문에 H.264 디코딩 모듈을 직접 구현하여 현제 Iphone에서 처리 가능한 H.264파일의 핸들링 범위를 넓힐 수 있는 모델을 제시했고 시뮬레이션한 결과를 보였다.

Jpeg을 사용한 모델은 Ipod touch 2세대 firmware 3.1.2에서 실제로 동작하는 것을 확인하였다. 하지만 H.264를 사용한 모델은 아직 개선해야할 것들이 많이 있다.

H.264를 이용한 모델의 가장 큰 문제는 현재 device에서 실제로 동작하지 않는 것에 있다. 이는 실제 Iphone에서 소켓에서 들어오는 H.264 영상을 처리하기 위해 반드시 개선해야 하는 문제이다.

그리고 JM을 H.264 디코딩 모듈로 채택하였기 때문에 디코딩 과정을 하드웨어의 가속없이 처리해야하는 제약이 있다. 이는 JM의 DCT를 처리하는 부분이나 모션보상을 처리하는 부분 그리고 디코딩 과정에서 연산이 많이 요구되는 블록을 Iphone에서 활용 가능한 하드웨어 가속을 이용한다면 개선의 여지가 많이 남아 있다. [7]

참고문헌

[1] 스티븐 코찬, “프로그래밍 오브젝티브C 2.0", 2009년

[2] 유성관, “iPhone 아이폰 SDK 튜토리얼”, 2010년

[3] http://developer.apple.com/iphone:Iphone 개발 apple 공식 사이트

[4] Thomas Wiegand, Gray J Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol.13, No.7, July2003, pp.560-576.

[5] 석진욱, 김범호, 이정우, 조창식, “HD급 H.264 기술의 발전 동향", 전자통신동향분석 제 21권 제 1호 2006년 2월

[6] http://iphome.hhi.de/suehring: Jm 오픈소스 공식 사이트

[7] 호요성, “H.264/AVC 알고리즘 이해와 프로그램 분석”, 2008년