Journal of Institute of Control, Robotics and Systems (2013) 19(1):65-71
http://dx.doi.org/10.5302/J.ICROS.2013.19.1.065 ISSN:1976-5622 eISSN:2233-4335

네트워크 카메라의 움직이는 물체 감지를 위한 스마트폰 기반 영상처리 방법

Smart Phone Based Image Processing Methods for Motion Detection of a Moving Object via a Network Camera

김 영 진, 김 동 환*
(Young Jin Kim[1] and Dong Hwan Kim[1])
[1]Seoul National University of Science and Technology

Abstract: In this work, new smart phone based moving target detection is proposed. In order to implement the task, methods of real time image transmission from network camera, motion detecting algorithm and its effective implementation are also addressed. The network camera transfers image data by MJPEG format which contains various information such as data and IP address, and the smart phone separates the image data received through a WiFi module. Later, the image data is converted to a Bitmap image format, and with the help of the embedded OpenCV library on a smart phone and algorithm, it was found that the moving object was identified effectively in terms of real time monitoring and detection.

Keywords: motion detection, smart phone, network camera, PTZ, MJPEG (Motion JPEG), JNI

I. 서론

오늘날 감시카메라는 개인의 사생활 보호에서 공공기관의 보안에 이르기까지 다양한 분야에 이용된다. 최근 급증하고 있는 범죄에서도 감시카메라의 역할은 중요하게 대두된다. 전통적인 지능형 감시 시스템은 중요한 지역 보안을 위해 유선으로 단말기인 PC를 사용한다. 근래의 감시 카메라 시스템은 이동성이 요구되고 있기에 PC 기반의 감시 시스템으로는 충족 할 수가 없다.

장소 이동과 신뢰성에 대한 감시 시스템에 대한 사용자의 요구가 더해지고 있으며, 모바일 기기의 발달로 모바일 감시 시스템에 대한 사용자의 요구는 점점 더 증가하고 있다[1]. 이러한 이유로 장소이동에 대한 연구는 더욱 활발히 진행되고 있으며, 안드로이드 기반의 모바일 디바이스를 통한 영상 수신 및 처리에 대한 관심은 더욱 높아지고 있다.

안드로이드라는 리눅스 기반의 OS (Operating System)을 이용하여 언제 어디서나 지역을 모니터링 할 수 있는 감시 시스템이 제안되었다[2,3]. 스마트폰에 장착되어 있는 카메라를 통하여 영상처리를 하는 기술이나, 영상을 압축하여 전송하는 기술에 많은 관심을 가지고 있으며, 많은 연구가 활발히 진행되고 있다. 모바일 폰에서 비디오 스트림의 인코딩하고, 디코딩하는 기술의 성능은 아직까지 성과를 평가하기는 어렵다[4]. 각종 기기에 따른 특성에 따라 다르지만 영상을 인코딩 또는 디코딩하는 속도를 평가 할 때 5 fps (frame per second)이하는 낮은 속도로 10~15 fps 이상은 좋은 품질의 비디오 스트림으로 인정하고 있다[4].

안드로이드 OS를 사용하는 모바일 폰에서는 자체 카메라 영상을 인코딩하고 WiFi 망을 통하여 전송하고, 수신을 받는 멀티홉 기능을 수행 할 수 있는 영상 송/수신 기술도 개발되어 지고 있다. 이러한 기술 중 가장 중요한 부분은 영상을 인코딩/ 디코딩하는 속도이다[4].

감시카메라 시스템의 경우 MJPEG, MPEG, H.264와 같은 프로토콜을 사용하므로 무선네트워크를 통해 전송되는 속도는 결정되어 있다. 그러나 수신 받은 영상을 디코딩하는 속도는 그 방법에 따라 시스템 성능에 많은 영향을 준다.

감시카메라에서는 시스템의 목적에 따라, 움직이는 물체 인식과 정밀 추적 기능 등이 요구되고 있다. 배경과 물체의 분리를 위해 Wavelet 변환을 사용하고 물체를 추적하기 위해 물체 궤적정보를 이용하는 등의 연구가 이루어지고 있다[5]. 이러한 연구나 사례는 컴퓨터를 기반으로 하여 영상을 실시간으로 인식하는 시스템이다.

본 연구에서는 감시 카메라 시스템에서 전송하는 영상을 수신 받아 영상을 디코딩하는 속도를 개선하고, 획득된 영상을 처리하는 부분에 대한 방법을 제시하였다. 또한 네트워크를 통해 전송 받은 영상에 이미지 처리를 통하여 이동 가능한 네트워크 감시 시스템의 사용 가능 여부를 확인하였다.

스마트폰 운영체제는 복잡한 이미지를 계산하기 위한 메모리와 CPU 자원에 대한 제한이 있으므로 하드웨어가 복잡한 이미지 프로세싱을 제공하지 않는다. 멀티미디어 데이터를 처리할 수 있도록 하드웨어가 지원하는 경우에도 최적화되지 않을 경우 상당한 시스템 지연이 발생할 수 있다[6].

* 책임저자(Corresponding Author)
논문접수: 2012. 3. 27., 수정: 2012. 10. 25., 채택확정: 2012. 11. 22.
김영진: 서울과학기술대학교 나노생산기술연구소
(trocad@seoultech.ac.kr)
김동환: 서울과학기술대학교 기계시스템디자인공학과
(dhkim@seoultech.ac.kr)
※ 본 연구는 서울시 산학연 협력사업(PA100010)의 지원으로 이루어졌음.

이동이 가능한 감시 시스템에서는 수신한 영상을 디코딩하는 속도와 영상처리를 하는 속도가 가장 중요하다. 시스템의 속도지연을 최소화하면서 복잡한 영상처리를 하기기 위해서 임베디드 플랫폼에서 사용 가능한 라이브러리인 OpenCV를 널리 사용한다[7]. 그러나 안드로이드 OS는 리눅스 기반의 운영체제이며, Java 라이브러리를 통해 장치를 제어한다. OpenCV는 C언어 기반의 C++ 형태로 되어있는 라이브러리이므로 안드로이드 운영체제에서는 바로 사용할 수 없다. 안드로이드 운영체제에서 사용하기 위해서는 Java에서 인식할 수 있는 인터페이스를 사용하여야 한다[8].

본 논문에서는 무선 네트워크를 통해 수신 받은 감시 카메라의 영상을 디코딩하고, 디코딩 속도 개선 방안, 그리고 움직이는 물체를 인식할 수 있는 방안을 제안하고 그 결과를 제시 하였다.

II. 영상처리를 위한 시스템 구성

1. 감시카메라의 영상 포맷

그림 1은 일반적인 감시카메라 시스템 구성도이다. 감시카메라에서는 이더넷을 통하여 H.264, MPEG, MJPEG 등과 같은 비디오 스트림을 전송한다. 본 연구에서의 영상전송 포맷은 MJPEG의 영상 포맷을 사용하였다.

MJPEG은 Motion JPEG을 일컫는 것으로 한 개의 스틸사진들을 연속으로 처리하기 위한 JPEG 표준 비디오 형식이다. 비디오 스트림을 스틸 사진의 연속으로 취급한다. 따라서 Inter coding 압축을 사용하지 않기 때문에 편집이 용이하다. Still Frame Coding (Intra Coding)만을 사용함으로써 MPEG에 비하여 계산량이 적고, 화면단위의 편집이 가능하며 구현이 용이하다. 압축률은 고화질에서는 1:10 이며 최대 1:20이 된다. 위와 같은 특징은 저 전력과 단순한 설계가 요구하는 모바일 응용에 매우 적합하다[9]. 따라서 대부분의 감시카메라의 기본 전송 포맷으로 사용을 한다.

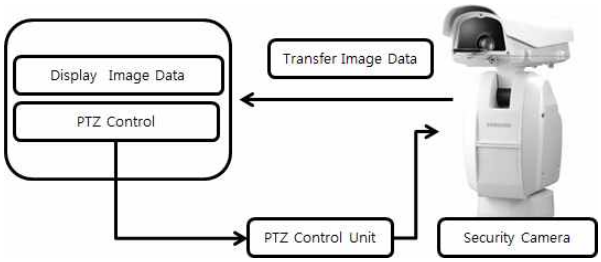

그림 1. 감시카메라의 시스템 구조.

Fig. 1. System structure of surveillance camera system.

그림 2. MPEC 비디오 시퀀스의 도식도.

Fig. 2. Schematic image of the MJPEG video sequence.

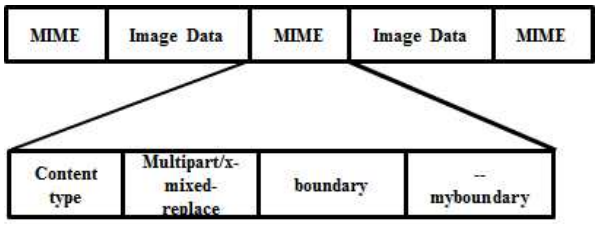

그림 3. JPEC 코드의 구조.

Fig. 3. The organization of JPEG code stream.

그림 2는 압축된 JPEG 이미지가 정지영상의 연속으로 전송되는 개념을 나타낸다[10]. 그림 3은 감시카메라에서 사용하는 JPEG 스트림을 나타낸다. 그리고 그림4는 스마트폰과 감시카메라를 연결하는 전체 구조를 보여준다.

여기서 MIME는 인터넷을 통하여 영상을 전송하는 방식을 의미하며, Start of image는 한 장의 영상 중 그 영상의 시작을 의미하며, 그 값은 0xFF, 0xD8이며, End of image는 한 장의 영상 중에 끝을 의미하며, 그 값은 0xFF, 0xD9이다. Frame 안의 정보의 Comment는 생산정보로서 Time Stamp Trigger Data를 포함하며, 양자화(Quantization Table)에서 휘도는 Y, 색차 정보인 Cb/Cr의 정보를 포함한다. Start of Frame은 이미지의 크기 및 샘플링에 관한 정보를 가지고 있으며, Huffman Table에는 휘도의 DC, AC계수, 색차의 DC, AC 계수에 대한 데이터가 있고 Start of Scan에서는 각 성분들이 어떤 Huffman 테이블을 사용하는 지를 알려주는 정보들이 포함되어 있다[11].

2. 감시카메라의 MJPEG 영상 수신

그림 4는 본 연구에서 구축한 스마트폰과 감시카메라의 시스템 구성도이다. 카메라에 WiFi AP (Access Point)가 장착되어 있으며, WiFi AP를 통하여 스마트폰을 감시카메라와 무선으로 연결한다. 스마트 폰에서는 감시카메라의 PTZ (Pan, Tilt, Zoom) 유닛을 제어 할 수 있으며, 감시카메라를 통하여 획득한 영상을 처리하여 움직이는 물체를 감지하는 일을 수행하도록 한다.

WiFi를 통하여 전송한 영상을 스마트 폰에서 수신 받은 데이터 중 MIME 부분과 이미지 Data에서 Start of image (0xFF, 0xD8)를 찾아 제거하고, 데이터를 End of image (0xFF, 0xD9) 전까지 데이터를 획득하여 데이터 처리를 하면 영상을 획득 할 수 있다. 그림 5는 이미지 데이터에서 영상을 획득하기 위한 디코딩을 나타낸다.

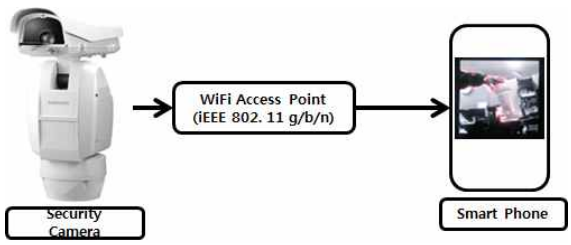

그림 4. 스마트폰과 감시 카메라의 전체 다이어그램.

Fig. 4. The overall diagram of smartphone and surveillance camera system.

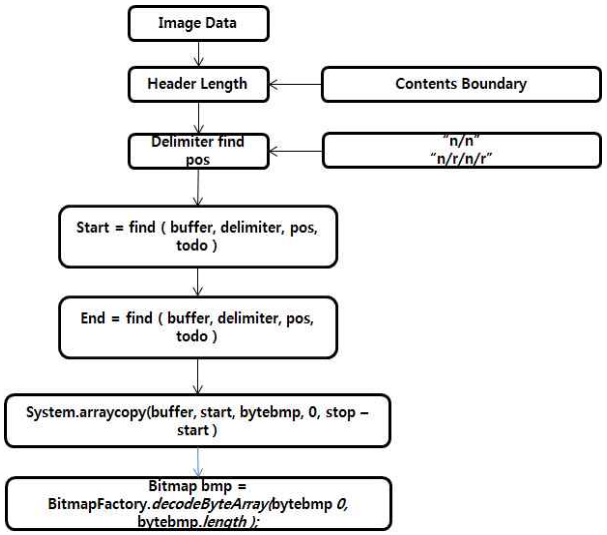

그림 5. 이미지 디코딩의 도표도.
Fig. 5. The overall diagram of image decoding.

기존의 이미지 디코딩 방식은 획득된 영상 데이터에서 Contents-boundary의 길이를 찾고, 임의의 버퍼에 저장 후 식별자(delimiter)를 찾는다. 식별자가 찾아지면 SOI의 길이를 찾고, EOF의 길이를 찾는다. 두 개의 길이는 메모리에서 시작과 끝의 길이를 이용하여 구해지고, 이 후 이미지를 저장하고자 하는 변수(bytebmp)에 저장하고, 저장된 변수를 통해 안드로이드에서 제공하는 BitmapFactory를 통하여 Bmp이미지로 가공한다. 이러한 과정의 디코딩은 정확한 이미지를 획득할 수 있는 반면 계산 시간으로 인하여 디코딩 속도를 느려지게 한다. 이러한 단점을 해결하고자 BitmapFactory 함수를 createBitmap으로 대치하였으며, 식별자 찾는 부분을 제거하고 이미지의 시작과 끝의 길이를 찾은 후 길이 만큼 skip하도록 함으로 속도를 증가 시킬 수 있었다.

안드로이드 OS에서 기존의 방식으로 이미지를 획득하여 영상처리 하는데 소요되는 시간은 평균 175 ms이다. 개선된 디코딩 방식으로 이미지를 획득할 경우 평균 88 ms로, 약 2배의 속도를 향상 시킬 수 있다. 그림 6은 디코딩하는 속도를 개선한 다이어그램이다.

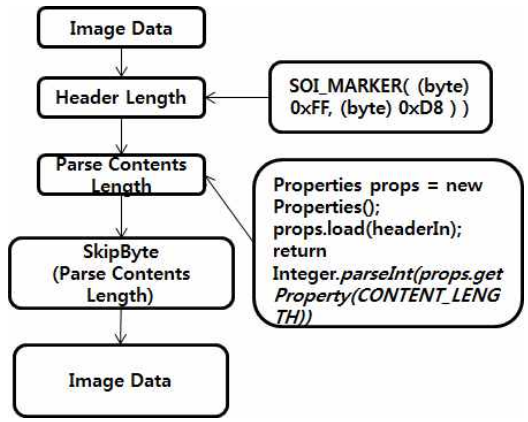

그림 6. 디코딩하는 속도를 개선한 다이어그램.
Fig. 6. The overall diagram for improving image decoding speed.

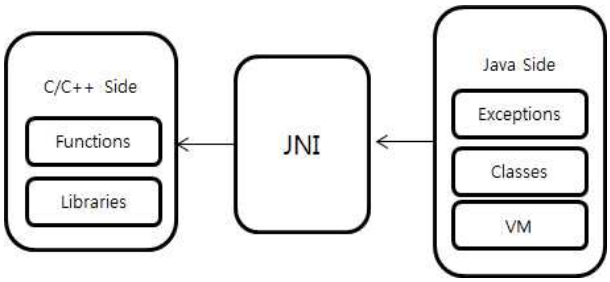

그림 7. 원 코드와 자바 사이에서 게이트로서의 JNI.
Fig. 7. JNI serves as a gateway between native code and Java.

3. OpenCV를 구현하기 위한 개발환경 구현

OpenCV (Open Source Computer Vision)는 인텔에서 만든 영상처리 라이브러리이다. 기초 영상처리서부터 고급 수준의 영상처리까지 많은 알고리즘들이 함수로 구현되어있다. OpenCV는 영상처리 분야를 다루기 위한 C++, C# 환경 상에서 가장 많이 사용되었으나, Java 환경에서도 사용하도록 개발 되었다. 즉 안드로이드와 같은 자바환경의 OS에서도 사용 할 수 있는 라이브러리이다. OpenCV를 안드로이드 OS에서 활용하기 위해서는 다소 불편하더라도 사전에 JNI 인터페이스 구현을 위한 NDK (Native Develop Kit)를 통해 C++로 되어 있는 OpenCV 라이브러리를 자바언어에서 인식 할 수 있도록 컴파일 처리한다[12]. 컴파일러는 빌드한 라이브러리를 안드로이드 어플리케이션 프로그램으로 작성하기 위한 SDK 플러그인이 설치되어 있는 이클립스에 포함한다. 본 논문에서는 FastFeatureDetect 라이브러리를 JNI로 구현하였다. 그림 7은 C++로 만들어진 FastFeatureDetect 라이브러리를 JNI를 통해서 사용할 수 있는 구조를 나타낸다.

Open CV를 PC에서 사용할 경우 기본적으로 Open CV가 C++ 또는 C# 언어 만들어져 있어 라이브러리를 프로그램 프로젝트에 포함하여 사용하면 된다. 안드로이드의 SDK 언어가 Java로 되어 있으므로 C++로 되어 있는 언어를 NDK (Native Development Kit)을 통하여 JNI (Java Native Interface)에서 사용할 수 있도록 프로그램을 재 배포해야 한다.

III. 스마트폰에서의 실시간 영상처리

1. 영상 검출을 위한 알고리즘

감시카메라에서 전송한 데이터를 획득하면 스마트폰에서는 영상을 획득하는 부분과 영상을 처리하는 과정을 거친다. 그림 8은 획득된 데이터에서 영상을 획득하는 부분을 나타내었다.

감시카메라에서 전송받은 데이터 중 MIME에 대한 컨텐츠를 Header Parse에서 제거하고 영상의 시작과 끝 부분을 제외한 길이만큼의 데이터를 Parse Content Length에 넘겨주고, 이미지 데이터를 획득한다. 이미지가 있을 경우 영상처리 프로세서로 인자를 넘겨주어 이미지 프로세싱을 진행하고, 이미지가 없을 경우 다시 이미지를 획득하도록 한다.

이동 물체 추적을 위해서 우선 움직이는 물체를 인식하기 위해 배경을 제거하는 방식을 사용하였다. 배경을 제거하기 위해서 이전 프레임에서 획득된 영상의 화소값을 $f(x,y)$에 저장하고 현재 프레임에서 획득된 영상의 화소값

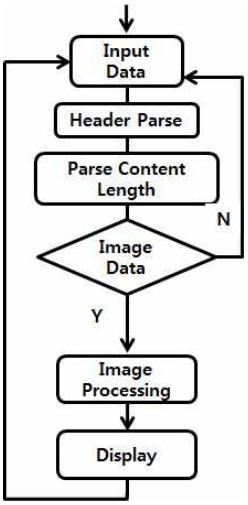

그림 8. 네트워크 카메라로부터 획득된 데이터에서 이미지 데이터 추출.

Fig. 8. Image data extraction from the data obtained from network camera.

을 $g(x,y)$에 저장하여 두 영상의 차이를 구하였다. 두 영상의 차 $h(x,y)$는 식 (1)에 정의 되며, 0~255까지 정규화를 하기 위해 절대값을 취하여 차영상을 구한다.

$$h(x,y) = |g(x,y) - f(x,y)| \tag{1}$$

여기서 $h(x,y)$는 차영상을 의미하며, $f(x,y)$는 이전 영상 프레임을, $g(x,y)$은 현재의 영상 프레임을 의미한다. 배경이 제거되면 차영상 $h(x,y)$이 남는다. 그리고 배경이 제거된 영상을 이진화 한다. 이진화 된 영상에서 특징 점을 찾아낸다. 특징 점을 찾은 영상을 현재 영상인 $g(x,y)$와 정합하면 움직이는 물체를 인식한 영상을 획득 할 수 있다. 그림 9는 움직이는 물체를 인식하기 위한 순서도를 나타내

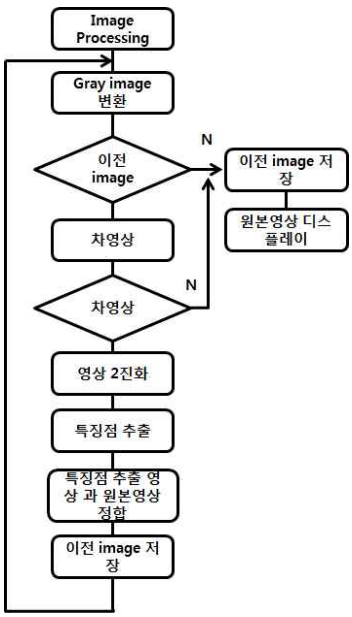

그림 9. 모션 감지를 위한 알고리즘.

Fig. 9. Algorithm of motion detection.

었다.

획득된 영상은 컬러영상이므로 영상처리를 쉽게 할 수 있도록 흑백영상으로 변환한다. 흑백영상을 통하여 영상 프레임의 차영상을 구한다. 이 방법은 움직이는 객체의 경계를 검출하는 특징을 가지고 있다. 식 (2)에 의하여 임계값(T)에 의한 이진화 된 영상을 획득한다.

$$f(h(x,y)) = \begin{cases} 255, \text{if } (h(x,y)) > T \\ 0, \quad \text{if}(h(x,y)) \le T \end{cases} \tag{2}$$

그림 10은 이진화 된 영상을 나타내었다.

이진화 영상을 획득 후 OpenCV의 알고리즘 중 특징점을 추출하는 알고리즘을 사용하여 이진화 된 영상에 존재하는 특징을 찾는다. 특징점은 코너 알고리즘을 사용하여 검출한다. 가장 널리 사용되는 코너에 대한 정의는 해리스에 의해 제안되었다[13,14]. 그림 11은 특징점을 추출한 영상을 나타낸다.

해리스 코너 검출 방법은 영상 내에 정의된 윈도우(window)를 상하좌우로 움직이며 윈도우 안의 픽셀 값의 변화를 분석하여 코너를 결정하는 방식을 사용한다.

특징점을 찾은 영상과 원본영상과 정합을 통하여 영상을 복원하였다. 식 (3)를 통하여 두 영상을 정합하였다.

$$I(x,y) = I_n(x,y) + I_f(x,y) \tag{3}$$

여기서 $I(x,y)$은 정합한 영상, $I_n(x,y)$는 원본영상, $I_f(x,y)$는 특징점을 검출한 영상이다.

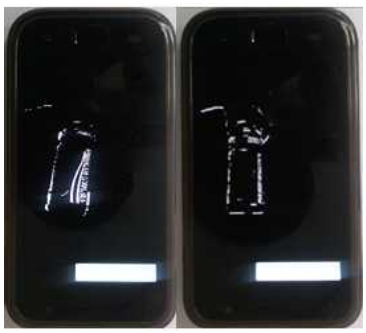

그림 10. 차 영상으로부터 이진화 영상 표현.

Fig. 10. The binary image representation from difference image.

그림 11. 움직이는 물체로 부터의 특징점.

Fig. 11. Feature points from moving object.

그림 12. 원본 이미지와 특징점을 검출한 이미지를 정합한 영상.

Fig. 12. Image combination of original image and feature-detected image.

그림 12는 원본 이미지와 특징점을 검출한 이미지를 정합한 영상을 타나내었다. 두 개의 이미지를 정합함으로 특징점이 검출되어 표시됨을 확인 할 수 있다.

IV. 실험

실험에 사용한 카메라는 AXIS-215 네트워크 돔 카메라이며, 스마트폰은 삼성 갤럭시S폰을 사용하였다.

실험은 주변에 특징점 검출이 상당히 많이 될 수 있는 복잡한 실험실에서 진행했으며, 백그라운드 영상의 특징점이 전혀 검출되지 않고, 오직 움직이는 물체만 검출되는지 확인 하였다.

그림 13은 움직이는 물체를 추적하는 영상을 보여준다.

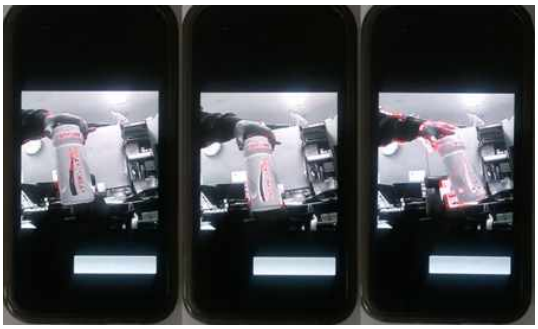

그림 13. 움직이는 물체를 특징점 추출.

Fig. 13. Feature points extraction from moving object.

그림 14. PC 기반 영상처리 프로그램에서 정지물체의 특징점 없음 표현.

Fig. 14. Representation of no feature points for stationary object on PC program.

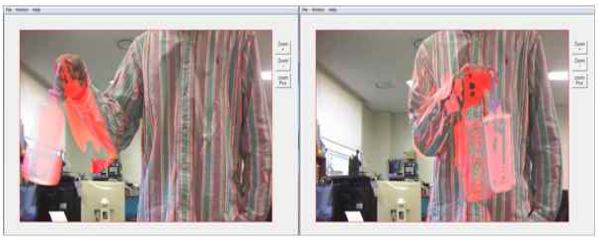

그림 15. PC에서 움직이는 물체의 특징점 추출.

Fig. 15. Feature points extraction from moving object on pc program.

그림과 같이 배경에 존재하는 특징점은 전혀 검출되지 않으며, 오직 움직이는 물체에 대한 특징점만 검출됨을 확인할 수 있다. 고정되어 있는 배경이 변화하지 않으므로 차영상을 획득하는 부분에서 고정부분은 제거되기 때문이다. 그림 14는 정지한 물체의 경우 특징점이 검출되지 않음을 보여주고 있다.

카메라의 영상 전송 속도는 30 fps이다. 무선네트워크를 통하여 전송 및 수신을 하면 영상 속도는 27 fps (frame per second)정도가 된다. 이것은 무선 네트워크의 속도에 따라 전송 및 수신 속도가 달라진다는 것을 의미한다. 스마트폰에서 수신 받은 영상을 처리할 경우 영상 프레임 디스플레이 속도는 12 fps ~ 7 fps의 정도가 된다. 영상처리를 하는 계산과정에서 60 % ~ 70 %의 속도 저하가 발생하였다.

다음으로 스마트폰과 PC의 영상처리 속도를 비교를 하였다. 그림 15는 동일한 방법으로 작성한 PC 기반 영상처리 프로그램으로 C#으로 작성하였다. 여기서는 병을 좌우로 흔들 때의 움직이는 물체의 인식과 PC의 영상처리 속도를 확인 하였다.

PC에서 영상처리를 하여 화면상에 영상을 표시한 결과 16.2 fps가 되었다. 영상처리 전과 비교하여 40 % 정도 속도 저하가 발생하였다. 스마트폰과 마찬가지로 PC에서도 영상처리 시 속도저하가 발생됨을 확인할 수 있다. 네트워크를 통하여 스마트폰으로 전송된 영상에서 움직이는 물체를 검출할 경우 움직이는 물체의 수와 특징점의 개수가 많아지면 처리 속도가 느려짐을 확인하였다. 또한 네트워크에서 전송되는 영상에 많은 정보를 가지고 있어 전송 속도가 느려지며, 전송 속도로 인하여 영상처리의 속도가 느려지는 경우도 확인하였다.

표 1은 PC와 안드로이드 운영체제를 사용하는 스마트폰 사이에 속도를 비교하였다.

표 1. PC와 안드로이드 운영체제를 사용하는 스마트폰 사이에 속도 비교

Table 1. Comparison processing speed of the PC and Android Smart Phone (Galaxy S).

Items	PC		Smart Phone	
	Max	Min	Max	Min
Frame (fps)	20	16.2	12	7
Recognition rate (%)	100		100	
Function calculation (s)	0.005	0.009	0.023	0.039

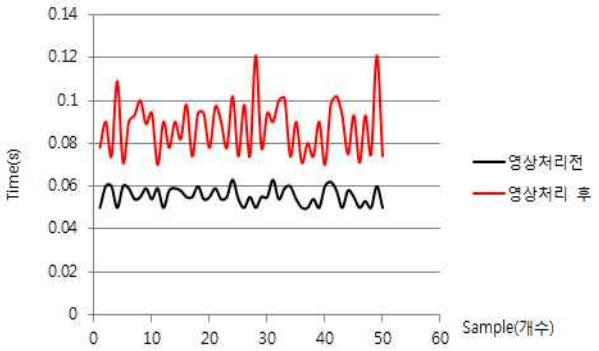

그림 16. 영상처리 전과 후의 스마트폰의 CPU 처리속도 측정.

Fig. 16. Measurement of smart phone CPU processing time before and after image processing.

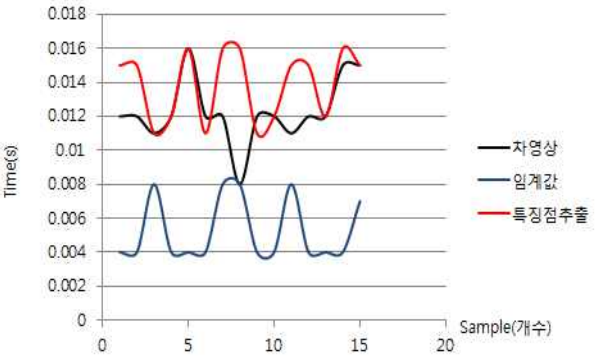

그림 17. 차영상 임계값, 특징점 추출의 함수를 계산할 때 소요되는 시간 측정.

Fig. 17. Measurement of smart phone CPU function calculation time.

그림 16은 영상처리 전과 후의 스마트폰의 CPU 처리속도를 나타낸다. 영상처리 전의 영상 디스플레이 속도보다 1/4감소한 속도이다.

영상처리를 할 때 차영상, 임계값 함수, 특징점을 찾는 함수가 호출된다. 이 함수들의 계산 시간으로 인하여 영상의 디스플레이 속도가 감소하는 것으로 나타났다. 그림 17은 차영상 임계값, 특징점 추출의 함수를 계산할 때 소요되는 시간을 나타낸다.

세 개의 함수를 수행 할 때 소요되는 평균시간이 약 36.4 ms이다. 따라서 영상처리 전에 소요되는 평균 시간이 55 ms이고 영상처리까지 소요되는 평균 시간이 88 ms 이므로 세 개 함수를 처리하는데 소요되는 시간으로 인하여 영상처리 전과 영상처리 후의 시간차가 발생을 하여, 영상 프레임 디스플레이 속도가 감소한다는 것을 알 수 있었다. 스마트폰을 통하여 영상처리를 하더라도 12 fps ~ 7 fps의 영상 프레임 디스플레이 속도를 가지고 있는 것을 확인하였다. 이 결과는 일반적인 모바일 기기에서 10 ~ 15 fps의 프레임 속도를 좋은 품질로 인정하고 있으므로 본 연구의 실험은 영상을 디스플레이하고 동시에 영상 처리를 한 상태에서도 12 fps ~ 7 fps의 프레임 속도가 나왔으므로 좋은 품질의 영상을 구현한 것이라 할 수 있다. 이것은 PC와 비교하면 상당히 많은 손실을 가지고 있다 할 수 있으나, 모바일 기기를 통하여 구현되는 영상처리에서도 상당히 품질이 좋은 영상을 제공할 수 있는 것을 보였다.

V. 결론

본 논문에서는 네트워크 감시카메라의 영상을 스마트폰에서 수신 받아 움직이는 물체를 인식하기 위한 방법을 제안 하였다. 제안된 방법은 PC를 통하여 영상처리용으로 사용하는 OpenCV를 안드로이드 OS 기반의 스마트폰에서도 탑재하여 스마트폰에서도 실시간 영상처리가 가능함을 보였다.

획득된 영상을 디코딩 할 때의 속도를 향상 시켜 시스템 전체적인 속도를 증가 시켰으며, 안드로이드 스마트폰서 영상 처리를 하기 위하여 C++로 구현된 라이브러리를 자바언어로 인식 할 수 있도록 JNI 인터페이스 구현을 위한 안드로이드 NDK를 구성해야 하는 다소 불편함이 있지만, 차영상과 이미지 정합을 통하여 움직이는 물체의 인식을 가능하게 하였다. 이러한 방법을 통하여 스마트폰으로도 영상처리가 가능함을 보였고 이를 활용하여 CCTV의 PTZ 제어와 움직이는 물체의 감시 등을 가능하게 하여 기존의 PC 기반 감시카메라 시스템과 대비하여 이동성의 편의 제공이 가능함을 보였다. 또한 스마트폰 기기의 성능이 계속 빨라지고 있어, 보다 더 복잡한 영상처리 기법을 적용하는 것이 가능할 것으로 생각된다.

향후 연구되어야 할 과제로는 스마트폰에서 영상 처리를 더 빠르게 수행할 수 있는 알고리즘에 대한 연구가 요구된다.

참고문헌

[1] H. Pang, L. Jiang, L. Yang, and K. Yue, “Research of android smart phone surveillance system,” *Int. Conf. on Computer Design And Applications*, pp. 373-376, 2010.

[2] E. Burnette, “Android basis tutorial,” B. Zhang, C. Gao, and Y. Yang, Translation. Beijing: Posts & Telecom Press, pp. 11-189, 2009.

[3] E2ECIoud Studio, “Simple terms google android,” Beijing: Posts & Telecom Press, pp. 39-256, 2009.

[4] P. Hu, N. Symons, J. Indulska, and M. Portmann, “Wireless multi-hop video streaming using android phones,” *8th IEEE PerCom Workshop on Pervasive Wireless Networking*, pp. 782-787, 2012.

[5] Y. Wang, J. F Doherty, and R. E. Vandyck, “Moving object tracking in video,” *AIPR, 29th Applied Imagery Recognition Workshop (AIPR'00)*, pp. 95-101, 2000.

[6] R. S. Deepthi and S. Sankaraiah, “Implementation of mobile platform using QT and open CV for image processing applications,” *IEEE Conf. on Open Systems*, Langkawi, Malaysia, pp. 284-289, 2011.

[7] K. M. Saipullah, A. Anuar, N. A. Ismail, and Y. Soo, “Real-time video processing using native programming on android platform,” *IEEE 8th Int. Colloquium on*

Signal Processing and its Applications, pp. 277-281, 2012.

[8] K. C. Son and J. Y. Lee, "The method of android application speed up by using NDK," *3rd International Conference on Awareness Science and Technology (iCAST)*, pp. 382-385, 2011.

[9] L. Chen and C. W. Lee, "The effects of mobility and redundancy on wireless video streaming over MANETs," *Advanced Information Networking and Applications-Workshops*, pp. 949-953, 2008.

[10] C. C. Wang, M. J. Chi, and Y. T. Chang, "New watermarking algorithm with coding efficiency improvement and authentication in video surveillance," *Fifth Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 1253-1259, 2009.

[11] C. H. Cho, K. Y. Min, and J. W. Chong, "A design of real-time MJPEG encoder for mobile communication devices," *IEEK Conference*, vol. 27, no. 1, pp. 1301-1304, 2004.

[12] A. Anuar, K. M. Saipullah, N. A. Ismail, and Y. Soo, "OpenCV based real-time video processing using android smartphone," *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, vol. 1, no. 3, pp. 58-62, 2012.

[13] OpenCV, Open source Computer Vision library. In http://opencv.willowgarage.com/wiki/, 2009.

[14] G. Bradski and A. Kaehler "Learning OpenCV," O'Reilly Media, Inc., CA, USA, pp. 429-434, 2008.

김 영 진

2000년 서울과학기술대학교 기계설계자동화공학부 졸업. 2002년 동대학원 기계설계학과 졸업, 현재 서울과학기술대학교 나노생산기술연구소 연구원. 관심분야는 프로그래밍, 로보틱스, 자동제어.

김 동 환

1986년 서울대 기계설계학과 졸업. 1988년 동 대학원 석사졸업. 1995년 Georgia Institute of Technology 박사. 현재 서울과학기술대학교 기계시스템디자인공학과 교수. 관심분야는 메카트로닉스, 로보틱스.