OpenCV 이진화처리와 히스토그램 그래프를 이용한 제스처인식

백영태[O], 이세훈[*], 김지성[*]
[O]김포대학 멀티미디어과
[*]인하공업전문대학 컴퓨터시스템과
e-mail : hannae@kimpo.ac.kr, seihoon@inhatc.ac.kr, succubus93@naver.com

Gesture Recognition using binary processing and histogram graph with OpenCV

Yeong-Tae Baek[O], Se-Hoon Lee[*], Ji-Seong Kim[*]
[O]Dept. of Multimedia, Kimpo University
[*]Dept. of Computer Systems & Engineering, Inha Technical College

● 요 약 ●

NUI는 마우스, 키보드 등의 HID장치를 사용하지 않고 사용자의 신체를 사용하여 장치를 제어하는 모션 인터페이스이다. 본 논문에서는 소형 임베디드보드 라즈베리파이와 라즈베리파이에 연결된 파이카메라를 통해 영상인식과 컴퓨터 비전을 위하여 최적화된 알고리즘을 제공하는 OpenCV를 이용하여 손의 제스처를 인식하여 기존 HID장비와 비교하여 더욱 인간 친화적이며 직관적인 NUI장치 인터페이스를 구현하고자 한다.

키워드: OpenCV, 제스처인식(Gesture Recognition), 영역검출(Area Detection)

I. 서 론

삶속에 인터넷 연결이 빈번해지고 사용자가 접하게 되는 정보가 더욱 다양하고 방대해짐에 따라 사용자는 습득한 정보를 빠르게 이해하고 활용하는 것이 매우 중요하다. 이러한 정보 전달 및 가공에 있어서 NUI(Natural User Interface) 기술이 더욱 중요해지고 있다. 이 NUI는 마우스, 키보드, 펜 등의 HID장치를 이용하지 않고 사용자의 제스처를 인터페이스로 활용하기 때문에 Natural User Interface를 줄여 명명되었다[1]. 정보 획득 및 가공 기술이 복잡해질수록 기술과 사용자 사이의 간격은 점점 넓어지는데, 이러한 간격은 더 직관적이고 쉬운 인터페이스 기술을 통해 해결이 가능하다. 본 논문은 다양한 NUI 응용 시스템의 기반 기술 중 하나인 손 제스처 인식기술을 이용한 NUI응용 인터페이스 개발 및 구현을 제안한다.

II. 관련 연구

1. Related works

1. 피부 영역 검출 손가락 카운팅 방식

영상에서 피부영역 검출 방식으로 손을 검출하기 위해서 피부영역과 피부영역 이외의 영역을 이진화 영상에서 제거하기 위해 사람의 피부색과 유사한 색 값을 추출하여 이진화 마스킹과정을 통해 손을 검출해야한다. 이 때 RGB 색 공간에서 마스킹할 경우 영상에서 비치는 손의 조도에 따라 가변하는 손의 색을 판별할 수 없는 문제가 발생한다. 따라서 본 논문에서는 이 문제를 해결하기 위해 RGB정보를 인코딩한 색 정보인 YCbCr 색 공간에서 마스킹연산을 수행한다.

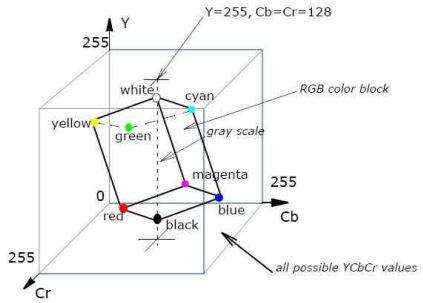

Fig. 1. Relationship between RGB and YCbCr[2]

그림1에 따르면 빛의 3원색을 정보로 이용하는 RGB 색 공간과는 달리 YCbCr 색 공간에서는 휘도와 색차 값을 정보로서 이용하게 되며, Y는 휘도, Cb와 Cr는 색차 성분 정보를 갖고 있다. 사람의 피부색을 조사하기 위해 RGB 색 공간에서 영상에 나타난 피부색의 최대, 최소값을 조사한 후, 그림 2의 RGB-YCrCb 변환 식에 따라 색 정보를 YCbCr 색 공간으로 변환해 준다. 영상의 조도에 따라 변하는 휘도Y값을 배제한 후 Cr Cb 색차 값만을 통해 피부색을 조명에 상관없이 검출해 낼 수 있다.

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B + 0$$
$$C_b = -0.169 \times R - 0.331 \times G + 0.499 \times B + 128$$
$$C_r = 0.499 \times R - 0.418 \times G - 0.0813 \times B + 128$$

$$R = \text{clamp}(Y + 1.402 \times (Cr - 128))$$
$$G = \text{clamp}(Y - 0.344 \times (Cb - 128) - 0.714 \times (Cr - 128))$$
$$B = \text{clamp}(Y + 1.772 \times (Cb - 128))$$

Fig. 2. RGB-YCrCb conversion formula[3]

위 식을 통해 YCrCb 색 공간에서의 빛에 따라 가변 할 수 있는 가변 치 범위의 피부색을 도출할 수 있으며, 이에 따라 본 논문에서 도출해낸 피부색 범위는 Cr 최소값128, 최댓값 170, Cb 최소값 73, 최댓값 158이다. 위 과정을 통해 도출해낸 정보를 토대로 피부색을 판별해 마스킹 할 수 있지만 피부색 판별 과정에서 피부색과 유사한 영역이 노이즈로서 나타날 수 있다는 문제점이 있다. 이러한 문제를 해결하기 위해 모폴로지 침식 연산을 통해 테두리 영역부터 안으로 픽셀을 소거해 규모가 작은 픽셀집단을 제거할 수 있으며 이는 직접적으로 노이즈를 최소화 할 수 있는 수단이 되며, 손가락영역을 비교적 정확하게 구분할 수 있는 장점도 있다.

Algorithm 1. Get Hand Mask

```
Begin
    minCr=128; maxCr=170; minCb=73; maxCb=158;
    cvtColor(image, YCrCb, CV_BGR2YCrCb);
    vector<Mat> planes; split(YCrCb, planes);
    mask(image.size(), CV_8U, Scalar(0));
    nr=image.rows; nc=image.cols;
    for(i=0; i<nr; i++){
        CrPlane=planes[1].ptr<uchar>(i);
        CbPlane=planes[2].ptr<uchar>(i);
        for(j=0; j<nc; j++){
            if((minCr<CrPlane[j])&&(CrPlane[j]<maxCr)&&(minCb<
CbPlane[j])&&(CbPlane[j]<maxCb))
                mask.at<uchar>(i, j)=255;
        }
    }
    erode(mask, mask, Mat(3, 3, CV_8U, Scalar(1)), Point(-1, -1), 2);
End
```

Fig. 3. Morphological Erosion Operation

위와 같이 검출된 손 영상에서 손가락의 개수를 인식하고 제스처 정보로 활용하기 위해 손의 중심을 찾는 과정이 필요하다. Algorithm 2의 거리변환행렬연산을 통해 가장 큰 픽셀집단의 중심으로부터 픽셀집단의 가장자리까지의 가장 짧은 거리를 구할 수 있는데, 이 값은 손바닥을 둘러싸는 원의 반지름으로 간주할 수 있으며, 이 반지름으로 손바닥의 중심을 도출해낼 수 있다[4].

Algorithm 2. Distance Transform Matrix

```
Begin
    distanceTransform(mask, dst, CV_DIST_L2, 5);
    minMaxIdx(dst, NULL, &radius, NULL, maxIdx, mask);
End
```

앞에서 도출한 손바닥의 중심점과 반지름을 이용해 손가락의 개수를 알아내 제스처 인식 정보로 사용해야한다. 그러므로 중심점 기준으로 약 radius * 2 의 크기로 원을 그린 후 원의 경계면을 지나는 픽셀집단의 개수를 세어 손가락의 개수를 알 수 있다. Algorithm3에서 원의 외곽을 따라 이동하면서 이진화 픽셀 값이 1로 바뀌는 지점을 체크해 손가락의 개수를 센다.

Algorithm 3. findContours Operating

```
Begin
    cImg(mask.size(), CV_8U, Scalar(0));
    circle(cImg, center, radius*scale, Scalar(255));
    findContours(cImg, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);
    fingerCount=0;
    for(int i=1; i<contours[0].size(); i++){
        p1=contours[0][i-1];
        p2=contours[0][i];
        if(mask.at<uchar>(p1.y, p1.x)==0 && mask.at<uchar>(p2.y, p2.x)>1)
        fingerCount++;
    }
End
```

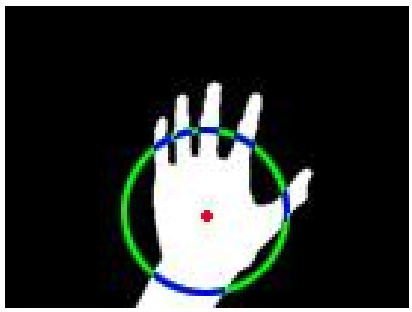

Fig. 4. Count Finger

2. 히스토그램 비교연산 방식

피부영역 검출 방식으로 제스처를 체크할 경우 손가락의 개수로 제스처를 인식하므로 손의 모양을 통해 제스처를 인식하는 것은 불가능하다. 이러한 문제를 해결하기 위해 본 논문에서는 히스토그램 비교연산을 사용한다. 그림5는 히스토그램 그래프를 나타내고 있다. 히스토그램이란 영상의 픽셀 값의 분포를 그래프로서 표현한 것인데, 그래프의 세로축은 픽셀 값의 빈도를 나타내며, 가로 축은 영상의 픽셀 값을 나타낸다. 이 때 가로 축의 픽셀 값이 RGB색 공간을 바탕으로 이루어진다면 비교적 많은 연산 시간이 소요 되므로 처리속도가 늦어질 수 있다. 따라서 색을 단순화 시킨 HSV 색 공간의 이미지로 변환한 후 히스토그램 연산을 시행한다. 그림 5는 RGB색 공간에서 HSV색 공간으로 변환한 이미지를 나타내고 있다[5].

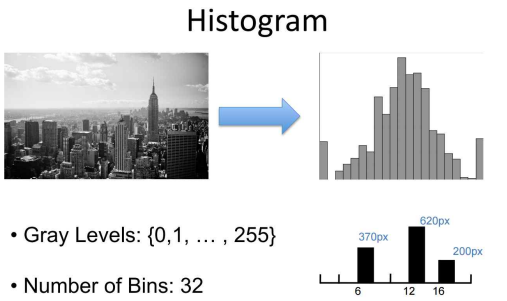

Fig. 5. Histogram Graph[6]

Fig. 6. RGB Color To HSV Color

영상에서 추출해낸 히스토그램 값을 비교 대상의 히스토그램 값과 비교하여 두 영상의 유사도를 측정할 수 있다. 따라서 히스토그램 비교연산을 바탕으로 이전에 기억해둔 손의 모양 히스토그램을 카메라에 비치는 손의 모양과 비교하여 현재 손의 제스처가 어떤 형태를 나타내는지 분석해낼 수 있다. 그림 7은 히스토그그램 비교연산 결과를 나타낸 것이다. 왼쪽 영상은 비교대상, 오른쪽 영상은 현재 카메라에 비치는 영상을 나타낸다. 커맨드 창에 출력된 비교 연산 결과는 표 1과 같다.

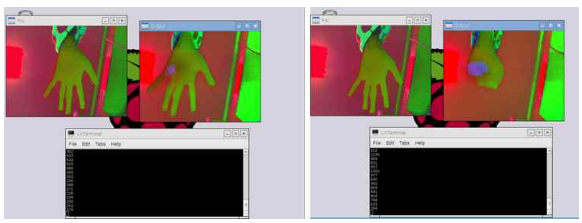

Fig. 7. Histogram Matching

Table. 1. Histogram Matching Result

회차	유사이미지	비 유사이미지
1	362	822
2	437	1134
3	422	983
4	403	851
5	389	957
6	300	1021
7	282	977
8	230	830
9	268	982
10	271	864
11	228	891
12	239	804
13	230	758
14	262	623
15	274	784

표1의 결과에 따르면 유사 영상은 히스토그램 비교연산 결과 값이 약 450 이하를 나타내며, 비 유사 영상은 결과 값이 약 750 이상을 나타낸다. 따라서 위 실험을 통해 히스토그램 비교연산으로 유사 영상과 비 유사 영상을 판별할 수 있음을 알 수 있다.

III. 제스처 인식

1. System Overview

전체 시스템 구성도는 그림 8과 같다. 그림 9은 PC와 라즈베리파이의 사용 준비를 마친 후 PC가 서버, 라즈베리파이가 클라이언트 역할을 시작하는 시점에서의 시스템 흐름과 구조를 표현한 다이어그램이다. 처음 프로그램을 실행하게 되면 먼저 라즈베리파이가 카메라를 사용할 수 있는지 기본적인 체크 후 TCP통신으로 PC서버에 접속하게 된다. 서버에 접속 후 사용자 설정정보를 PC로부터 전송받게 되는데, 이 설정정보는 어떤 방식으로 손을 검출한 후 제스처를 인식할 것인지에 대한 정보이다. 이 정보는 피부검출 손가락 카운팅 방식 또는 히스토그램 비교연산 방식 둘 중하나를 의미한다.

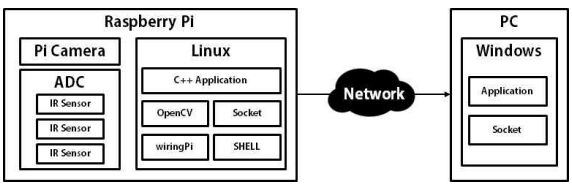

Fig. 8. Sequence Diagram

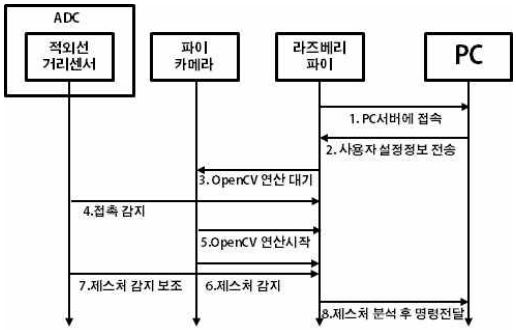

Fig. 9. Sequence Diagram

2. Using Infrared distance sensors

3개의 적외선 거리센서를 통해 손의 접촉과 두드림 및 손의 이동을 감지하여 제스처 인식을 보조한다. Algorithm 4에서 센서의 거리 감지를 통해 사용자의 손이 카메라 밖 영역에서 좌 - 우로 이동하였는지를 감지한 후 결과를 서버로 전송한다.

Algorithm 4. Left and right motion detection

```
Begin
    if(read_mcp3208_adc(0)〉700){
        st = clock();
        while(clock()-st〈1500){
            if(read_mcp3208_adc(1)〉700){
                strcpy(msg,"[TYPE=MOVE:VAL=LEFT:]");
                write(sock,msg,strlen(msg));
                delay(600);
    }}}
    if(read_mcp3208_adc(1)〉700){
        st = clock();
        while(clock()-st〈1500){
            if(read_mcp3208_adc(0)〉700){
                strcpy(msg,"[TYPE=MOVE:VAL=RIGHT:]");
                write(sock,msg,strlen(msg));
                delay(600);
    }}}
End
```

3. Histogram gesture recognition

사용자 설정정보가 히스토그램 비교연산으로 설정되었다면 비교할 제스처가 표현된 영상을 로드한 후 HSV색 공간으로 변환 후 각 비교 대상 영상의 히스토그램 그래프를 생성까지의 초기화를 마친 다음 손이 감지되길 기다린다. 손을 카메라에 비추어 인터럽트가 발생한다면 현재 파이카메라에 비친 제스처를 히스토그램 그래프화 하여 초기화한 히스토그램과 비교한다. 비교 결과가 표1의 결과를 참조하여 450이하의 결과 값이 나온다면 비교대상 영상과 일치하다고 판단 후 서버에 일치하는 영상번호를 전송한다.

4. Skin area detection finger counting

사용자 설정정보가 피부 영역 검출 손가락 카운팅 방식으로 설정되었다면 손에 대한 인터럽트가 발생하였을 때 파이카메라에 비치는 영상을 RGB 색 공간에서 YCbCr 색 공간으로 인코딩한다. 이 후 휘도값을 제외한 색차 값을 통해 손 영역을 검출해낸 후 2진화 마스킹 처리한 다음 거리변환 행렬을 통해 손바닥의 반지름과 중심을 찾아낸다. 이진화 마스킹 결과에서 손바닥 반지름*1.5의 원의 테두리를 따라 연산하며 결과가 0에서 1로 바뀌는 지점을 체크하여 손가락의 개수를 카운팅한 후 서버로 손가락 개수를 전송한다.

Ⅳ. 결 론

본 논문에서는 라즈베리파이와 거리센서, 파이카메라를 이용하여 손가락의 개수 및 모양을 판별해 PC 서버로 명령을 전달하여 손 인식 NUI 장비를 개발하고자 하였다. OpenCV 라이브러리를 사용해 구현했으며 색정보 인코딩, 모폴로지연산, 히스토그램 그래프 등을 사용하여 피부 영역 검출 손가락 카운팅 방식과 히스토그램 비교연산 방식 두 가지로 제스처를 인식하는 방식을 나누어 사용자가 선택해 사용할 수 있게끔 설계하였다. 본 논문에서는 상대적으로 양호한 조건에서 실험하였으며, 향후 이를 실험환경과 다른 열악한 환경에서 이용하기 위해서는 보다 정교하게 개선된 이진화 처리, 영상비교알고리즘이 필요할 것으로 사료된다.

Reference

[1] Gwang-hyung Lee, Dong-Kyoo Shin, Don-gil Shin, "NUI/NUX framework based on intuitive hand motion" Korea Network Information Society pp .12-13, Jun. 2014

[2] Intel , "Color Models" Intel Developer Zone , https://software.intel.com/en-us/node/503873

[3] Logic Devices, "RGB to YCbCr Report" Logic Devices pp .2, Jan 2001, http://www.logicdevices.com/support/appnotes/lf3370rgb.pdf

[4] Ju-Hwang Kim, jeong-hun Park, Jong-seo Jeon, Min-Goo Kang, Seung-Hyun Lee, "Recognition of Hand Interface for PC-Mouse using OpenCV" Korea Network Information Society pp .303-306, 2009

[5] Einführungspraktikum Computer Vision, "Digital Image Processing" R.C Gonzalez & R.E Woods , pp. 7, 2008

[6] Fabrizio Dini, giuseppe lisanti, "ImageProcessing with OpenCV" PPM2010 Seminar, pp. 11, 2010