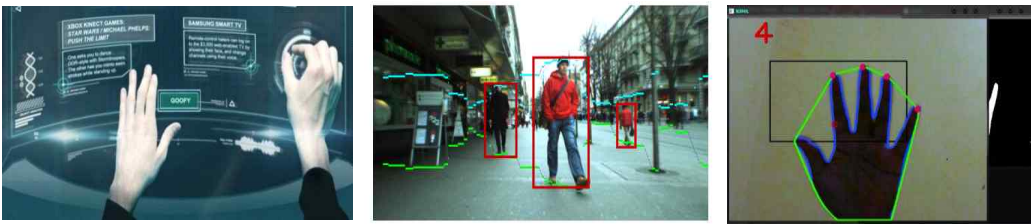


프로젝트 명	OpenCV를 활용한 Hand Mouse				
개발인원	5명	본인역할	팀원	개발기간	2016.04.25~ 2016.05.20
개발언어	C, C++				
개발 툴	Visual Studio 2013				
표준 기술	OpenCV				
개발 목적	사용자의 손동작만으로 마우스를 컨트롤이 가능한 Hand Mouse 구현.				
프로젝트 진행 목차	(1) 주제 선정 (2) Hand Mouse 벤치마킹 (3) 구현 가능한 기술적 방법 접근 (4) OpenCV 기능 및 함수 이해 (5) OpenCV를 활용한 코드 작성 및 테스트 (6) 최종 프로젝트 시연				
프로젝트 개요	(1) 주제 선정 - 사용자의 손동작을 인식하여 마우스를 컨트롤을 편리하게 하고자 영상인식을 통한 HandMouse 구현을 주제로 선정. (2) Hand Mouse 벤치마킹  (3) 구현 가능한 기술적 방법 접근 - (Intel)사가 개발한 컴퓨터비전 오픈소스 라이브러리 OpenCV 및 Window WebCAM을 사용하여 손 영역 추출 및 마우스커서 매핑. (4) OpenCV 기능 및 함수 이해 - WebCAM 영상(이미지) 출력. - 영상(이미지)의 1차원(Gray)~3차원(BGR) pixel 이해. - BGR → YCrCb로 특정 색상 검출. - 영상(이미지) Threshold()로 이진화. - erode() (5) OpenCV를 활용한 코드 작성 및 테스트 - YCrCb로 색 추출하여 손 인식. - 인식 된 pixel좌표값으로 영역 그리기 및 마우스포인트(커서,L버튼,R버튼) 부여. - 마우스이벤트(L버튼,R버튼,더블클릭,드래그) 구현. (6) 최종 프로젝트 시연 - 손가락 2개(검지, 중지)에 마우스포인트 부여 후, 마우스이벤트 작동 시연.				

프로젝트
상세
개발 내용

(가) 색상 검출

- (1) cvtColor를 이용하여 YCrCb로 변환
- (2) split를 사용하여 YCrCb로 변환된 영상을 y, cr, cb로 분리하여 저장.
- (3) 비어있는 Mat 형식의 변수에 영상의 사이즈만 넣어주어 생성.
- (4) 분리한 영상의 색상 배열에 원하는 색만 뽑아 오도록 설정.
- (5) erode 와 dilate를 사용하여 가져 온 색상 영상에 원하는 부분만 사용.

```

Mat FrameConvert::getRedColorMask(Mat _defaultFrame)
{
    int _minCr;
    int _maxCr;
    int _minCb;
    int _maxCb;

    _minCr = 200;
    _maxCr = 255;
    _minCb = 0;
    _maxCb = 200;

    //(1)
    Mat yCrCbFrame;
    cvtColor(_defaultFrame, yCrCbFrame, CV_BGR2YCrCb);

    //(2)
    vector<Mat> planes;
    split(yCrCbFrame, planes);

    //(3)
    Mat resultFrame(_defaultFrame.size(), CV_8U, Scalar(0));
    int nr = resultFrame.rows;
    int nc = resultFrame.cols;

    //(4)
    for (int i = 0; i < nr; i++)
    {
        uchar* CrPlane = planes[1].ptr<uchar>(i); //Cr채널의 i번째 행 주소
        uchar* CbPlane = planes[2].ptr<uchar>(i); //Cb채널의 i번째 행 주소

        for (int j = 0; j < nc; j++)
        {
            if ((_minCr < CrPlane[j]) && (CrPlane[j] < _maxCr) && (_minCb < CbPlane[j]) && (CbPlane[j] < _maxCb))
            {
                resultFrame.at<uchar>(i, j) = 255;
            }
        }
    }

    //(5)
    for (int i = 0; i < 1; i++)
    {
        erode(resultFrame, resultFrame, Mat(3, 3, CV_8U, Scalar(1)), Point(-1, -1), 1);
    }

    for (int i = 0; i < 1; i++)
    {
        dilate(resultFrame, resultFrame, Mat(3, 3, CV_8U, Scalar(1)), Point(-1, -1), 1);
    }

    return resultFrame;
}

```

(나) 마우스이벤트 처리 구현

- (1) 마우스 커서 이동 - 중지의 빨간색 포인트를 기준으로 마우스 이동 기능 제작.
 - Left 클릭 down 이벤트 : 검지의 파란색 포인트가 화면에서 사라지면 즉, 빨간 circle만 그려진 상태라면 클릭 이벤트 발생.
 - Left 클릭 up 이벤트 : 검지의 파란색 포인트가 화면에서 다시 나타나면 즉, 파란 circle, 빨간 circle 둘다 그려진 상태라면 클릭 up 이벤트 발생.
 - Right 클릭 down 이벤트 : 검지의 파란색 포인트와 중지의 빨간색 포인트의 거리가 일정 거리 이하로 좁혀지면 즉, 40픽셀 아래로 내려가면 Right 클릭 down 이벤트

발생.

- Right 클릭 up 이벤트 : 검지의 파란색 포인트와 중지의 빨간색 포인트의 거리가 일정 거리 이하로 좁혀졌다가 다시 벌어지면 즉, $40 < \text{픽셀} < 200$ 의 범위에 들어간다면 Right 클릭 UP 이벤트 발생.
- 커서의 떨림 보정 : 영상의 프레임이 빠른 속도로 지나가기 때문에 마우스 커서가 많이 흔들립니다. 그래서 마우스 커서의 이전 좌표를 저장하고 다음 좌표까지의 거리를 계산 후 160픽셀 보다 적은 차이는 커서의 움직임이 없도록 함.

```
if ((blueChk == true && redChk == false) || (blueChk == false && redChk == true) || (blueChk == true && redChk == true))
{
    //Mouse Move
    redCirclePoint.x = redCirclePoint.x * 85535 / GetSystemMetrics(SM_CXSCREEN) * (1920 / 640);
    redCirclePoint.y = redCirclePoint.y * 85535 / GetSystemMetrics(SM_CXSCREEN) * (1080 / 360);

    //흔들림 방지
    if (abs(beforeRedPoint.x - redCirclePoint.x) > 160 && abs(beforeRedPoint.y - redCirclePoint.y) > 107 )
    {
        ::mouse_event(MOUSEEVENTF_MOVE | MOUSEEVENTF_ABSOLUTE, redCirclePoint.x, redCirclePoint.y, 0, ::GetMessageExtraInfo());
    }
    beforeRedPoint = redCirclePoint;

    if (blueChk == false && leftClickChk == false)//Mouse Left up
    {
        ::mouse_event(MOUSEEVENTF_LEFTDOWN, redCirclePoint.x, redCirclePoint.y, 0, ::GetMessageExtraInfo());
        leftClickChk = true;
    }

    if (blueChk == true && leftClickChk == true)//Mouse Left down
    {
        ::mouse_event(MOUSEEVENTF_LEFTUP, redCirclePoint.x, redCirclePoint.y, 0, ::GetMessageExtraInfo());
        leftClickChk = false;
    }
}

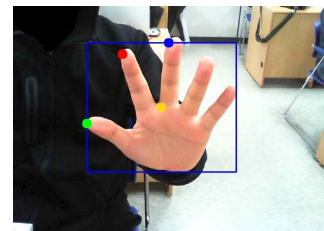
// Mouse Right up
if (rightClickChk == false && abs(tempRedPoint.x - blueCirclePoint.x) < 40)
{
    ::mouse_event(MOUSEEVENTF_RIGHTDOWN, redCirclePoint.x, redCirclePoint.y, 0, ::GetMessageExtraInfo());
    rightClickChk = true;
}

//Mouse Right up
if (rightClickChk == true && abs(tempRedPoint.x - blueCirclePoint.x) > 40 && abs(tempRedPoint.x - blueCirclePoint.x) < 200)
{
    ::mouse_event(MOUSEEVENTF_RIGHTUP, redCirclePoint.x, redCirclePoint.y, 0, ::GetMessageExtraInfo());
    rightClickChk = false;
}
```

(1) 피부영역 추출 후 이진화 -> 가장 큰 영역검출 -> 손가락검출

- 영상 이진화 후 피부색 영역을 검출하고 영역의 x축과 y축을 기준으로 세 점의 좌표를 출력한다.

시행 착오



** 피부색 검출이기 때문에 손과 배경의 피부색이 겹치는 경우 영역이 불규칙하게 지정 되는 문제가 발생