Digital Image Processing

LabWork #CV7

Professor : Jin-Woo Jung

1. 각 단계별 결과를 첨부하고 무엇을 하는 단계인지 기술하세요.

|  |  |
| --- | --- |
| 단계별 결과 | 단계별 설명 |
|  | VideoCapture 클래스의 인스턴스를 생성한 후, 웹 캠으로부터 이미지를 얻는다. 이때 인스턴스의 파라미터로 자신이 사용할 카메라에 해당하는 index 값을 넣는다. |
|  | 마스크로 사용할 이미지를 imread 함수를 이용하여 불러들인다. |
|  | 마스크의 사이즈를 웹 캠으로부터 촬영한 이미지의 사이즈와 같게 만든 후 더하기 연산을 이용해 마스크를 적용시킨다. 이때 원 바깥에 있는 배경 부분은 사라지게 된다. |
|  | 마스크가 적용된 이미지를 YCrCb 컬러 모델로 변환시킨다. 이 컬러 모델을 적용시키는 이유는 색의 특점 범위를 잡아 피부색과 유사한 색을 잡아야 하기 때문이다. |
|  | YCrCb 컬러 모델에서 특점 범위에 대하여 이진화 작업을 수행한다. |
|  | 마스크가 적용된 이미지를 gray\_scale 이미지로 변환시킨다. |
|  | 위에서 이진화 작업한 이미지를 이용하여 gray\_scale 이미지에서 손 영역만 추출한다. |
|  | 추출된 손 영역 이미지에서 이진화 작업을 수행한다. 이때 threshold 값으로 200을 넣었다. |
|  | Erode 연산을 이용하여 손 영역 이미지의 노이즈를 제거한다. |
|  | Center of Mass 함수를 이용하여 손바닥 영역을 찾는다. |
|  | Gray\_scale 이미지에 손바닥 영역을 표시한다. |
|  | 손바닥 영역이 표시된 gray\_scale 이미지에서 원 gray\_scale 이미지를 subtract하녀 손바닥 영역만 추출한다. |
|  | Canny Edge detecting을 이용하여 손금으로 추정되는 edge를 추출한다. 이때 원 영역까지 같이 표시하게 된다. |
|  | 추출된 Canny Edge에서 원형 노이즈를 제거하여 최종적으로 자신의 손금을 추출하게 된다. |

1. 더 좋은 방법이 있다면 직접 코드를 수정하고 코멘트와 함께 수정된 코드를 첨부하세요.

|  |
| --- |
| 설명 |
| #include <opencv2/opencv.hpp>  using namespace cv;  #define CAM\_WIDTH 800  #define CAM\_HEIGHT 480  Mat getHandMaskWithBGR(const Mat& image) {  Mat mask(image.size(), CV\_8U, Scalar(0));  const int nr = image.rows; //이미지의 행 길이를 저장한다.  const int nc = image.cols; //이미지의 열 길이를 저장한다.  for (int i = 0; i < nr; i++) {  const Vec3b\* data = image.ptr<Vec3b>(i);  for (int j = 0; j < nc; j++) {  uchar R = data[j][2]; //R frame 저장  uchar G = data[j][1]; //G frame 저장  uchar B = data[j][0]; //B frame 저장  if (R > 95 && G > 40 && B > 20 && max(R, max(G, B)) - min(R, min(G, B)) > 15 && abs(R - G) > 15 && R > G && R > B)  mask.at<uchar>(i, j) = 255;  //YCrCb로 변환하지 않고, BRG 모델에서 각각의 픽셀에 대하여 바로 피부색에 해당하는 값을 가지고 있는지 확인한다.  //만약 비교하는 값을 변화시키면 배경과 같이 추출될 수도 있고, 반대로 손이 제대로 추출되지 않을 수 있다.  }  }  return mask;  }  int main()  {  VideoCapture cam(0);  cam.set(CV\_CAP\_PROP\_FRAME\_WIDTH, CAM\_WIDTH); //웹 캠에서 촬영하는 이미지의 너비를 지정한다.  cam.set(CV\_CAP\_PROP\_FRAME\_HEIGHT, CAM\_HEIGHT); //웹 캠에서 촬영하는 이미지의 폭을 지정한다.  Mat Origin\_frame; //웹 캠에서 촬영하는 이미지를 저장하기 위한 변수를 선언한다.  Mat HAND\_MASK = imread("1.png", 1); //mask 이미지를 불러온다.  if (!cam.isOpened())  {  printf("--(!)Error opening videio cam\n"); //open 되지 않았다면 오류메세지 출력  return -1;  }  while (cam.read(Origin\_frame))  {  if (Origin\_frame.empty())  {  printf("--(!) No cam frame -- Break!"); //웹 캠으로 촬영하는 이미지가 제대로 저장되지 않는 경우 오류메세지 출력  break;  }  resize(HAND\_MASK, HAND\_MASK, Size(Origin\_frame.cols, Origin\_frame.rows), 0, 0, CV\_INTER\_LINEAR); //mask의 size를 웹 캠으로 촬영한 이미지의 size와 같도록 조정한다.  //CV\_INTER\_NN : 최근접 이웃 보간법 - 원본의 가장 가까운 픽셀값으로 할당  //CV\_INTER\_LINEAR: 양선형 보간법 - 주위 4개의 픽셀값을 이용  //CV\_INTER\_CUBIC : 큐빅 보간법 - 주위 16개의 픽셀값을 이용  //CV\_INTER\_AREA : 제본표 보간법 - 짝수인 픽셀만 이용  Mat Mask = Origin\_frame + HAND\_MASK; //촬영한 이미지에 mask를 적용한다.  Mat tmp = getHandMaskWithBGR(Mask); //위에서 정의한 함수를 이용하여 손을 검출한다.  Mat hand(Mask.size(), CV\_8UC3, Scalar(0)); //손을 추출한 이미지를 저장하기 위한 변수를 선언한다.  add(Mask, Scalar(0), hand, tmp); //tmp에서 0이 아닌 픽셀만 hand 변수로 더해진다.  cvtColor(hand, hand, CV\_BGR2GRAY); //gray\_scale 이미지로 변환한다.    Mat thresh\_skin\_gray;  threshold(hand, thresh\_skin\_gray, 0, 255, THRESH\_BINARY || THRESH\_OTSU); //otsu 방법으로 thresholding을 한다.  Mat tild\_thresh\_skin\_gray = ~thresh\_skin\_gray; //물체부와 배경부가 표현된 값을 뒤바꾼다.  erode(tild\_thresh\_skin\_gray, tild\_thresh\_skin\_gray, Mat(3, 3, CV\_8U, Scalar(1)), Point(-1, -1), 2); //erode 연산으로 노이즈를 제거한다.  Mat Mask\_gray;  cvtColor(Mask, Mask\_gray, CV\_BGR2GRAY); //gray\_scale로 변환한다.  Mat Skin\_gray = Mask\_gray + (thresh\_skin\_gray); //Mask\_gray와 thresh\_skin\_gray을 더하여 손만 추출한다.  Mat dist;  distanceTransform(tild\_thresh\_skin\_gray, dist, CV\_DIST\_L2, 5); //거리변환 함수를 적용하여 손바닥 중심점을 찾게 된다.  normalize(dist, dist, 255, 0, NORM\_MINMAX, CV\_8UC1); //거리 변환된 값에 대하여 정규화를 수행한다.  int maxldx[2];  int minldx[2];  double radius(0);  minMaxIdx(dist, NULL, NULL, minldx, maxldx, dist); //dist에서 최대수치, 최저수치, 최저수치의 위치, 최대수치의 위치를 얻어낼 수 있다.  Point center(maxldx[1], maxldx[0]);  if (center.x < 0) center.x = 0;  if (center.y < 0) center.y = 0;  if (radius < 0) radius = 0;  radius = (maxldx[0] - minldx[0]) / 2; //최저수치의 위치, 최대수치의 위치를 이용하여 radius를 구한다. 이 radius 값은 가변적이다.  circle(Origin\_frame, center, (int)(radius + 0.5), Scalar(125, 255, 0), -1);  Mat Skin\_gray\_clone = Skin\_gray.clone(); //Skin\_gray를 clone한다.  circle(Skin\_gray\_clone, center, (int)(radius + 0.5), Scalar(0, 0, 0), -1);//circle 함수로 손바닥 중심점을 지정한다.  Mat Sub = hand - Skin\_gray\_clone; //subtract 연산으로 손금을 추출한 영역을 불러온다.  Mat CannyEdge;  Canny(Sub, CannyEdge, 40, 110); //hysteresis thresholding으로 edge를 찾는다.  circle(CannyEdge, center, (int)(radius + 0.5), Scalar(0, 0, 0), 4); //edge에서 원 경계선을 제거한다.  imshow("Result", CannyEdge); //최종적으로 손금을 출력한다.  waitKey();  }  } |