#include "ros/ros.h" // ROS'header file

#include <opencv2/opencv.hpp>

#include <iostream>

#include <opencv2/core.hpp>

#include <opencv2/imgproc.hpp>

#include <opencv2/videoio.hpp>

#include <opencv2/highgui.hpp>

#include "ros\_brand/Message1.h"// MsgTutorial

using namespace cv;

using namespace std;

int main(int argc, char \*\*argv)

{

ros::init(argc, argv, "publisher\_node");

ros::NodeHandle nh;

ros::Publisher ros\_brand\_pub = nh.advertise<ros\_brand::Message1>("ros\_brand\_msg", 100);

//ros::Rate loop\_rate(10);

ros\_brand::Message1 msg;

const int max\_value\_H = 360 / 2;

const int max\_value = 255;

int thresh = 100;

int low\_H = 0, low\_S = 0, low\_V = 0;

int high\_H = max\_value\_H, high\_S = max\_value, high\_V = max\_value;

Vec3b lower\_blue1, upper\_blue1, lower\_blue2, upper\_blue2, lower\_blue3, upper\_blue3;

Mat img\_color;

Scalar red;

Mat bgr\_color = Mat(1, 1, CV\_8UC3, red);

Mat hsv\_color;

cvtColor(bgr\_color, hsv\_color, COLOR\_BGR2HSV);

int hue;

namedWindow("img\_color");

//create trackbar

createTrackbar("Low H", "img\_color", &low\_H, max\_value\_H);

setTrackbarPos("Low H", "img\_color", low\_H);

createTrackbar("high\_H", "img\_color", &high\_H, max\_value\_H);

setTrackbarPos("high\_H", "img\_color", high\_H);

createTrackbar("low\_S", "img\_color", &low\_S, max\_value);

setTrackbarPos("low\_S", "img\_color", low\_S);

createTrackbar("high\_S", "img\_color", &high\_S, max\_value);

setTrackbarPos("high\_S", "img\_color", high\_S);

createTrackbar("low\_V", "img\_color", &low\_V, max\_value);

setTrackbarPos("low\_V", "img\_color", low\_V);

createTrackbar("high\_V", "img\_color", &high\_V, max\_value);

setTrackbarPos("high\_V", "img\_color", high\_V);

Mat img\_hsv;

VideoCapture cap(2); //find camera number

if (!cap.isOpened()) {

cout << "Can't find camera." << endl;

return -1;

}

cap.set(CAP\_PROP\_FRAME\_WIDTH, 640); //320

cap.set(CAP\_PROP\_FRAME\_HEIGHT, 480); //240

while (ros::ok())

{

//Get present time

//double start = GetTickCount();

cap.read(img\_color);

cvtColor(img\_color, img\_hsv, COLOR\_BGR2HSV);

//Morphology

Mat img\_mask;

inRange(img\_color, Scalar(low\_H, low\_S, low\_V), Scalar(high\_H, high\_S, high\_V), img\_mask);

int morph\_size = 3;

Mat element = getStructuringElement(MORPH\_RECT, Size(2 \* morph\_size, 2 \* morph\_size),

Point(morph\_size, morph\_size));

morphologyEx(img\_mask, img\_mask, MORPH\_OPEN, element);

morphologyEx(img\_mask, img\_mask, MORPH\_CLOSE, element);

// find moments of the image

Moments m = moments(img\_mask, true);

Point p(m.m10 / m.m00, m.m01 / m.m00);

circle(img\_color, p, 2, 2, 2);

// coordinates of centroid

cout << "\tCenter coordinates : " << p<<endl;

msg.x\_coordinate = p.x;

msg.y\_coordinate = p.y;

imshow("img\_color", img\_color);

imshow("img\_mask", img\_mask);

ros\_brand\_pub.publish(msg);

//loop\_rate.sleep();

if (waitKey(1) > 0)

break;

}

return 0;

}