#include "stdafx.h"

#include<opencv2/core/core.hpp>

#include<opencv2/highgui/highgui.hpp>

#include<opencv2/imgproc/imgproc.hpp>

#include<sstream>

#include<iostream>

using namespace cv;

using namespace std;

//TODO: Get the screen split into even chunks

int main() {

VideoCapture capWebcam(1); //use webcam 1 if working with the usb webcam, cam 0 is laptop webcam

if (capWebcam.isOpened() == false) // To check if object was associated to webcam successfully

{

cout << "error: Webcam connect unsuccessful\n"; // if not then print error message

return(0); // and exit program

}

capWebcam.set(CV\_CAP\_PROP\_FRAME\_WIDTH, 900); //sets input capture to a set resolution of 1024x1024

capWebcam.set(CV\_CAP\_PROP\_FRAME\_HEIGHT, 600);

Mat imgOriginal; //original image capture

Mat hsvImg; //image to hold the hsv values

Mat threshImg; //hsv image we isolate the color from

ostringstream outputss; //for console clarity

bool coordsOK = false;

char charCheckForEscKey = 0; //esc key

int lowH = 149;//21; // Set Hue

int highH = 179;//30;

int lowS = 110;//200; // Set Saturation

int highS = 228;//255;

int lowV = 185;//102; // Set Value

int highV = 255;//225;

int iLastX = -1;

int iLastY = -1;

// HUE for YELLOW is 21-30.

//HSV reference table

//Yellow H: 21-30

//Pink H: 149-179 S: 110 - 228 V: 185,255

while (charCheckForEscKey != 27 && capWebcam.isOpened()) { // until the Esc is pressed or webcam connection is lost

bool blnFrameReadSuccessfully = capWebcam.read(imgOriginal); // get next frame

if (!blnFrameReadSuccessfully || imgOriginal.empty()) { // if frame read unsuccessfully

cout << "error: frame can't read \n"; // print error message

break; // jump out of loop

}

cvtColor(imgOriginal, hsvImg, CV\_BGR2HSV); // Convert Original Image to HSV Thresh Image

inRange(hsvImg, Scalar(lowH, lowS, lowV), Scalar(highH, highS, highV), threshImg);

GaussianBlur(threshImg, threshImg, Size(3, 3), 0); //Blur Effect

dilate(threshImg, threshImg, 0); // Dilate Filter Effect

erode(threshImg, threshImg, 0); // Erode Filter Effect

dilate(threshImg, threshImg, 0);

erode(threshImg, threshImg, 0);

Moments oMoments = moments(threshImg);

double dM01 = oMoments.m01;

double dM10 = oMoments.m10;

double dArea = oMoments.m00;

if (dArea > 10000) {

int posX = dM10 / dArea;

int posY = dM01 / dArea;

//Put the little circle on the center

if (iLastX >= 0 && iLastY >= 0 && posX >= 0 && posY >= 0) {

outputss << "X: " << posX << " Y: " << posY << endl;

cout << outputss.str() << endl;

circle(imgOriginal, cvPoint(posX, posY), 20, Scalar(0, 255, 0));

coordsOK = true;

}

//Draws lines to indicate the bounding boxes

//line(imgOriginal, Point(0,0), Point(0, 1000), Scalar(0, 255,0), 5);

//line(imgOriginal, Point(100, 0), Point(100, 1000), Scalar(0, 255, 0), 3);

//line(imgOriginal, Point(250, 0), Point(250, 1000), Scalar(0, 255, 0), 3);

//line(imgOriginal, Point(750, 0), Point(750, 1000), Scalar(0, 255, 0), 3);

//line(imgOriginal, Point(900, 0), Point(900, 1000), Scalar(0, 255, 0), 3);

//line(imgOriginal, Point(1000, 0), Point(1000, 1000), Scalar(0, 255, 0), 3);

iLastX = posX;

iLastY = posY;

}

// declare windows

namedWindow("imgOriginal", CV\_WINDOW\_AUTOSIZE);

namedWindow("threshImg", CV\_WINDOW\_AUTOSIZE);

/\* Create trackbars in "threshImg" window to adjust according to object and environment.\*/

/\*

createTrackbar("LowH", "threshImg", &lowH, 179); //Hue (0 - 179)

createTrackbar("HighH", "threshImg", &highH, 179);

createTrackbar("LowS", "threshImg", &lowS, 255); //Saturation (0 - 255)

createTrackbar("HighS", "threshImg", &highS, 255);

createTrackbar("LowV", "threshImg", &lowV, 255); //Value (0 - 255)

createTrackbar("HighV", "threshImg", &highV, 255);

\*/

imshow("imgOriginal", imgOriginal); // show windows

imshow("threshImg", threshImg);

charCheckForEscKey = waitKey(1); // delay and get key press

}

return(0);

}