//-----------------------------------【头文件包含部分】---------------------------------------

// 描述：包含程序所依赖的头文件

//----------------------------------------------------------------------------------------------

#include <opencv2/opencv.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include<iostream>

#include<algorithm>

//-----------------------------------【命名空间声明部分】---------------------------------------

// 描述：包含程序所使用的命名空间

//-----------------------------------------------------------------------------------------------

using namespace cv;

using namespace std;

//-----------------------------------【main( )函数】--------------------------------------------

// 描述：控制台应用程序的入口函数，我们的程序从这里开始

//-----------------------------------------------------------------------------------------------

int main()

{

VideoCapture cap(0); //capture the video from web cam

if (!cap.isOpened()) // if not success, exit program

{

cout << "Cannot open the web cam" << endl;

return -1;

}

vector<Vec3f> circles;

int mark[24];

int a = 0;

while ( a<400 || circles.size() != 24) //第一个循环，扫描棋盘，检测到24个点且出现棋子时跳出此循环（这个程序一般会在跳出此循环后出问题）

{

Mat srcImage;

bool bSuccess = cap.read(srcImage); // read a new frame from video

if (!bSuccess) //if not success, break loop

{

cout << "Cannot read a frame from video stream" << endl;

break;

}

Mat midImage, dstImage;//临时变量和目标图的定义

int iLowH1 = 155;

int iHighH1 = 179;

int iLowS1 = 90;

int iHighS1 = 255;

int iLowV1 = 10;

int iHighV1 = 255;

int iLowH2 = 75;

int iHighH2 = 105;

int iLowS2 = 90;

int iHighS2 = 255;

int iLowV2 = 0;

int iHighV2 = 255;

Mat imgHSV;

vector<Mat> hsvSplit;

cvtColor(srcImage, imgHSV, COLOR\_BGR2HSV); //Convert the captured frame from BGR to HSV

//在HSV空间做直方图均衡化

split(imgHSV, hsvSplit);

equalizeHist(hsvSplit[2], hsvSplit[2]);

merge(hsvSplit, imgHSV);

Mat imgThresholded1;

inRange(imgHSV, Scalar(iLowH1, iLowS1, iLowV1), Scalar(iHighH1, iHighS1, iHighV1), imgThresholded1); //Threshold the image

Mat imgThresholded2;

inRange(imgHSV, Scalar(iLowH2, iLowS2, iLowV2), Scalar(iHighH2, iHighS2, iHighV2), imgThresholded2); //Threshold the image

//开操作 (去除一些噪点)

Mat element = getStructuringElement(MORPH\_RECT, Size(5, 5));

morphologyEx(imgThresholded1, imgThresholded1, MORPH\_OPEN, element);

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_OPEN, element);

//闭操作 (连接一些连通域)

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_CLOSE, element);

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_CLOSE, element);

imshow("Original", srcImage); //show the original image

int count = 0;

for (int i = 0; i< imgThresholded1.rows; ++i)

{

uchar \* p = imgThresholded1.ptr<uchar>(i);

for (int j = 0; j< imgThresholded1.cols; ++j)

{

if (p[j] == 255)

{

count++;

}

}

}

for (int i = 0; i< imgThresholded2.rows; ++i)

{

uchar \* p = imgThresholded2.ptr<uchar>(i);

for (int j = 0; j< imgThresholded2.cols; ++j)

{

if (p[j] == 255)

{

count++;

}

}

}

a = count;

//【3】转为灰度图，进行图像平滑

cvtColor(srcImage, midImage, CV\_BGR2GRAY);//转化边缘检测后的图为灰度图

GaussianBlur(midImage, midImage, Size(9, 9), 2, 2);

//【4】进行霍夫圆变换

HoughCircles(midImage, circles, CV\_HOUGH\_GRADIENT, 1.5, 10, 200, 15, 8, 13);

//【5】依次在图中绘制出圆

for (size\_t i = 0; i < circles.size(); i++)

{

Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));

int radius = cvRound(circles[i][2]);

//绘制圆心

circle(srcImage, center, 3, Scalar(0, 255, 0), -1, 8, 0);

//绘制圆轮廓

circle(srcImage, center, radius, Scalar(155, 50, 255), 3, 8, 0);

}

//【6】显示效果图

imshow("【效果图】", srcImage);

cout << circles.size() << endl;

if (circles.size() == 24) //将扫描到的24个点对应在circles中的标号

//按照棋盘1-24的顺序放入数组mark中

{

int line1[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 3; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

line1[count] = j;

count++;

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line1[j]][0]))

{

mark[count] = line1[j];

count++;

}

int line2[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 6; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 4)

{

line2[count - 4] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line2[j]][0]))

{

mark[count + 3] = line2[j];

count++;

}

int line3[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 9; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 7)

{

line3[count - 7] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line3[j]][0]))

{

mark[count + 6] = line3[j];

count++;

}

int line4[6];

count = 0;

for (int i = 0; i < midImage.rows&&count < 15; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 10)

{

line4[count - 10] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 6; j++)

if (i == cvRound(circles[line4[j]][0]))

{

mark[count + 9] = line4[j];

count++;

}

int line5[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 18; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 16)

{

line5[count - 16] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line5[j]][0]))

{

mark[count + 15] = line5[j];

count++;

}

int line6[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 21; i++)

{

for (int j = 0; j <= 23; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 19)

{

line6[count - 19] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line6[j]][0]))

{

mark[count + 18] = line6[j];

count++;

}

int line7[3];

count = 0;

for (int i = 0; i < midImage.rows&&count < 24; i++)

{

for (int j = 0; j < 24; j++)

if (i == cvRound(circles[j][1]))

{

count++;

if (count >= 22)

{

line7[count - 22] = j;

}

}

}

count = 0;

for (int i = 0; i < midImage.cols; i++)

for (int j = 0; j < 3; j++)

if (i == cvRound(circles[line7[j]][0]))

{

mark[count + 21] = line7[j];

count++;

}

}

char key = (char)waitKey(60);

}

if (!cap.isOpened()) // if not success, exit program

{

cout << "Cannot open the web cam" << endl;

return -1;

}

while (true) //第二个循环，获取棋子位置

{

Mat srcImage;

bool bSuccess = cap.read(srcImage); // read a new frame from video

if (!bSuccess) //if not success, break loop

{

cout << "Cannot read a frame from video stream" << endl;

break;

}

Mat midImage, dstImage;//临时变量和目标图的定义

int iLowH1 = 155;

int iHighH1 = 179;

int iLowS1 = 90;

int iHighS1 = 255;

int iLowV1 = 10;

int iHighV1 = 255;

int iLowH2 = 80;

int iHighH2 = 110;

int iLowS2 = 90;

int iHighS2 = 255;

int iLowV2 = 0;

int iHighV2 = 255;

Mat imgHSV;

vector<Mat> hsvSplit;

cvtColor(srcImage, imgHSV, COLOR\_BGR2HSV); //Convert the captured frame from BGR to HSV

//在HSV空间做直方图均衡化

split(imgHSV, hsvSplit);

equalizeHist(hsvSplit[2], hsvSplit[2]);

merge(hsvSplit, imgHSV);

Mat imgThresholded1;

inRange(imgHSV, Scalar(iLowH1, iLowS1, iLowV1), Scalar(iHighH1, iHighS1, iHighV1), imgThresholded1); //Threshold the image

Mat imgThresholded2;

inRange(imgHSV, Scalar(iLowH2, iLowS2, iLowV2), Scalar(iHighH2, iHighS2, iHighV2), imgThresholded2); //Threshold the image

//开操作 (去除一些噪点)

Mat element = getStructuringElement(MORPH\_RECT, Size(5, 5));

morphologyEx(imgThresholded1, imgThresholded1, MORPH\_OPEN, element);

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_OPEN, element);

//闭操作 (连接一些连通域)

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_CLOSE, element);

morphologyEx(imgThresholded2, imgThresholded2, MORPH\_CLOSE, element);

imshow("Thresholded Image1", imgThresholded1); //show the thresholded image1

imshow("Thresholded Image2", imgThresholded2); //show the thresholded image2

imshow("Original", srcImage); //show the original image

//check

for (int i = 0; i < 24; i++)

cout << mark[i] << ",";

cout << endl;

cout << cvRound(circles[mark[7]][0]) << endl;

//

int array1[24];

int array2[24];

int array[24];

for (int i = 0; i < 24; i++)

{

if (imgThresholded1.at<uchar>(cvRound(circles[mark[i]][1]), cvRound(circles[mark[i]][0])) == 255)

{

array1[i] = 1;

}

else

{

array1[i] = 0;

}

if (i == 23)

break;

}

for (int i = 0; i < 24; i++)

{

if (imgThresholded2.at<uchar>(cvRound(circles[mark[i]][1]), cvRound(circles[mark[i]][0])) == 255)

{

array2[i] = -1;

}

else

{

array2[i] = 0;

}

if (i == 23)

break;

}

for (int i = 0; i < 24; i++)

{

array[i] = array1[i] + array2[i];

cout << array[i] << ",";

if (i == 23)

break;

}

cout << endl;

char key = (char)waitKey(60);

if (key == 27)

break;

}

return 0;

}