**IMAGE PROCESSING SYSTEM**

#include <opencv2/imgcodecs.hpp>

#include <opencv2/highgui.hpp>

#include <opencv2/imgproc.hpp>

#include <iostream>

using namespace cv;

using namespace std;

/////////////// Color Detection //////////////////////

//void main() {

//

// string path = "Resources/lambo.png";

// Mat img = imread(path);

// Mat imgHSV, mask;

// int hmin = 0, smin = 110, vmin = 153;

// int hmax = 19, smax = 240, vmax = 255;

//

// cvtColor(img, imgHSV, COLOR\_BGR2HSV);

//

// namedWindow("Trackbars", (640, 200));

// createTrackbar("Hue Min", "Trackbars", &hmin, 179);

// createTrackbar("Hue Max", "Trackbars", &hmax, 179);

// createTrackbar("Sat Min", "Trackbars", &smin, 255);

// createTrackbar("Sat Max", "Trackbars", &smax, 255);

// createTrackbar("Val Min", "Trackbars", &vmin, 255);

// createTrackbar("Val Max", "Trackbars", &vmax, 255);

//

// while (true) {

//

// Scalar lower(hmin, smin, vmin);

// Scalar upper(hmax, smax, vmax);

// inRange(imgHSV, lower, upper, mask);

//

// imshow("Image", img);

// imshow("Image HSV", imgHSV);

// imshow("Image Mask", mask);

// waitKey(1);

// }

//}

#include <opencv2/imgcodecs.hpp>

#include <opencv2/highgui.hpp>

#include <opencv2/imgproc.hpp>

#include <iostream>

using namespace cv;

using namespace std;

/////////////// Color Detection //////////////////////

void getContours(Mat imgDil, Mat img) {

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(imgDil, contours, hierarchy, RETR\_EXTERNAL, CHAIN\_APPROX\_SIMPLE);

//drawContours(img, contours, -1, Scalar(255, 0, 255), 2);

vector<vector<Point>> conPoly(contours.size());

vector<Rect> boundRect(contours.size());

for (int i = 0; i < contours.size(); i++)

{

int area = contourArea(contours[i]);

cout << area << endl;

string objectType;

if (area > 1000)

{

float peri = arcLength(contours[i], true);

approxPolyDP(contours[i], conPoly[i], 0.02 \* peri, true);

cout << conPoly[i].size() << endl;

boundRect[i] = boundingRect(conPoly[i]);

int objCor = (int)conPoly[i].size();

if (objCor == 3) { objectType = "Tri"; }

else if (objCor == 4)

{

float aspRatio = (float)boundRect[i].width / (float)boundRect[i].height;

cout << aspRatio << endl;

if (aspRatio > 0.95 && aspRatio < 1.05) { objectType = "Square"; }

else { objectType = "Rect"; }

}

else if (objCor > 4) { objectType = "Circle"; }

drawContours(img, conPoly, i, Scalar(255, 0, 255), 2);

rectangle(img, boundRect[i].tl(), boundRect[i].br(), Scalar(0, 255, 0), 5);

putText(img, objectType, { boundRect[i].x,boundRect[i].y - 5 }, FONT\_HERSHEY\_PLAIN, 1, Scalar(0, 69, 255), 2);

}

}

}

void main() {

string path = "Resources/shapes.png";

Mat img = imread(path);

Mat imgGray, imgBlur, imgCanny, imgDil, imgErode;

// Preprocessing

cvtColor(img, imgGray, COLOR\_BGR2GRAY);

GaussianBlur(imgGray, imgBlur, Size(3, 3), 3, 0);

Canny(imgBlur, imgCanny, 25, 75);

Mat kernel = getStructuringElement(MORPH\_RECT, Size(3, 3));

dilate(imgCanny, imgDil, kernel);

getContours(imgDil, img);

imshow("Image", img);

//imshow("Image Gray", imgGray);

//imshow("Image Blur", imgBlur);

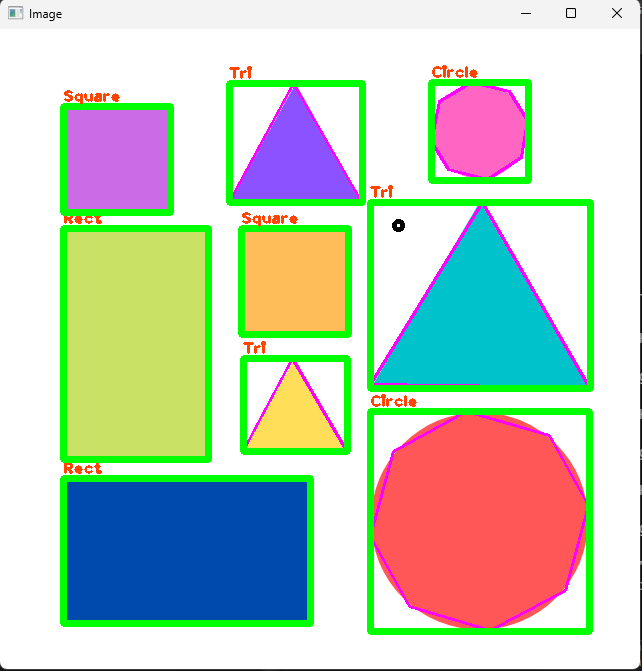
//imshow("Image Canny", imgCanny);

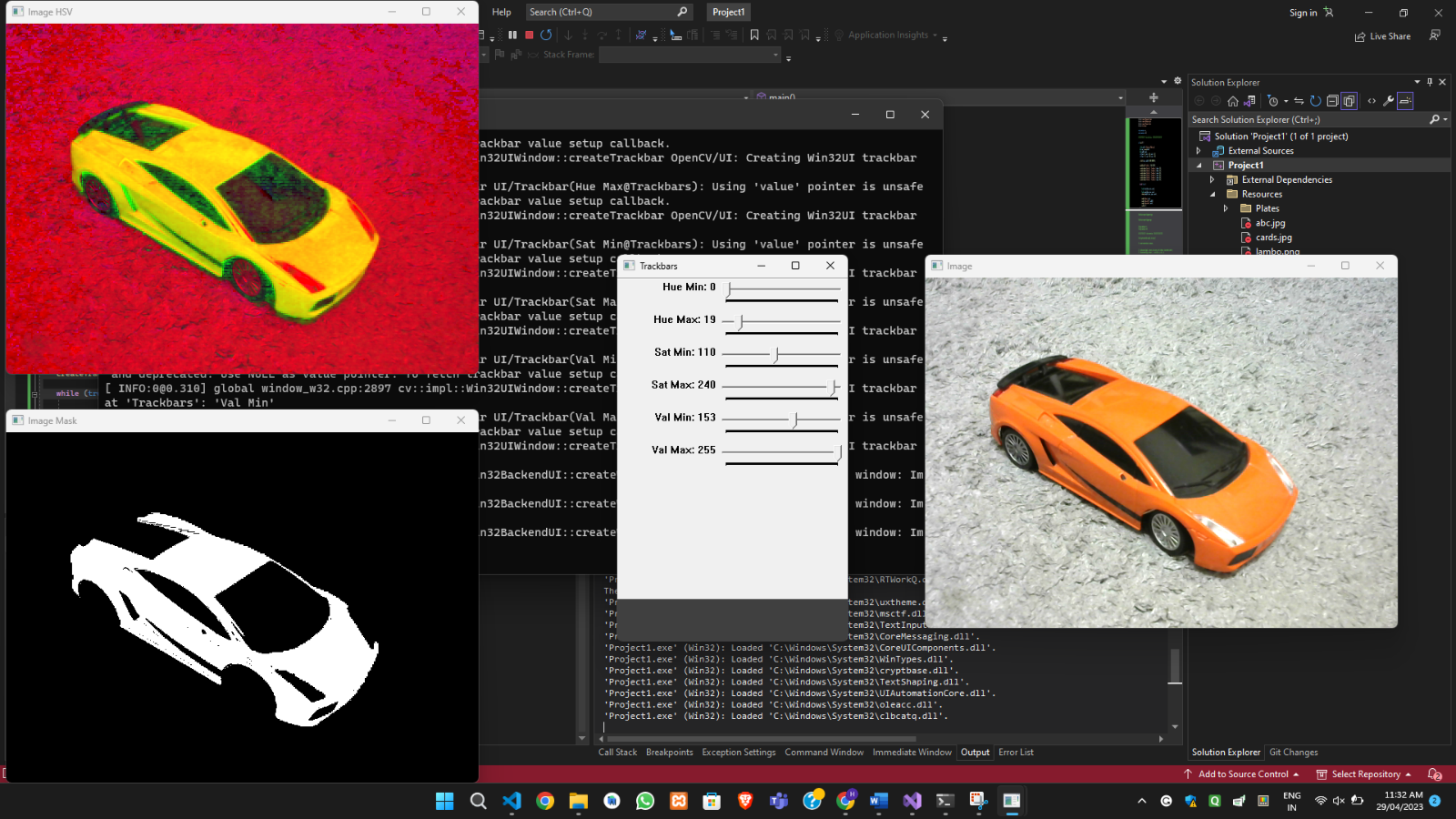
//imshow("Image Dil", imgDil);

waitKey(0);

}

* Shape Detection:



* Color Detection