МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Кафедра информационных систем и технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

По курсу «Методы распознавания образов и анализа изображений»

на тему:

«Программа для распознания цветных фигур с использованием библиотеки OpenCV»

по направлению подготовки 09.04.01 Автоматизированные системы обработки информации и управления (уровень магистратуры)

направленность (профиль) «Информационные системы»

Студент группы № 6222-090401D\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Гуреев

Преподаватель, к.т.н., доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Бибиков

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2020

**Цель работы.**

Целью данной лабораторной работы является разработка программы для распознания цветных круглых фигур на изображении с камеры в режиме реального времени, а также трассировка найденного объекта при движении.

**Алгоритм.**

1. Перевод в RGB
2. Поворот, флип и ресаиз изображения
3. Размытие
4. Перевод фото в HSV формат
5. Установка цветового диапазона
6. Поиск цвета из заданного диапазона на HSV изображении
7. Отрисовка контура / Поиск и отрисовка круга с цветным индикатором по центру для трассировки.

**Описание приложения и скриншоты**

Экран программы для поиска круглых фигур на изображении представлен на рисунке 1. Имеется возможность откорректировать цветовой диапазон для фигуры, передвинув по три верхних ползунка слева и справа. Параметры интервала задаются в формате HSV. Корректировка параметров поиска круглого объекта осуществляется передвижением трёх нижних ползунков. Два предустановленных цвета для поиска: зелёный и синий. Доступны два режима работы: поиск контуров (рисунок 2 и 3) и выделение круглого объекта на изображении (рисунок 4 и 5) заданного цвета.

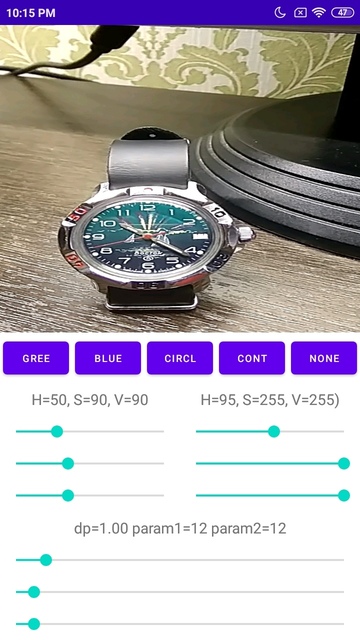


Рисунок 1 – Экран программы

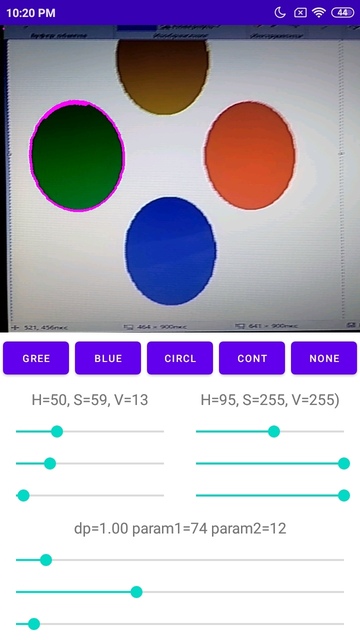


Рисунок 2 – Выделение контуров зелёного цвета

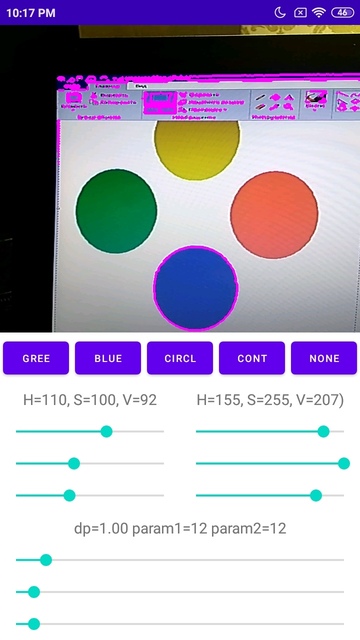


Рисунок 3 – Выделение контуров синего цвета

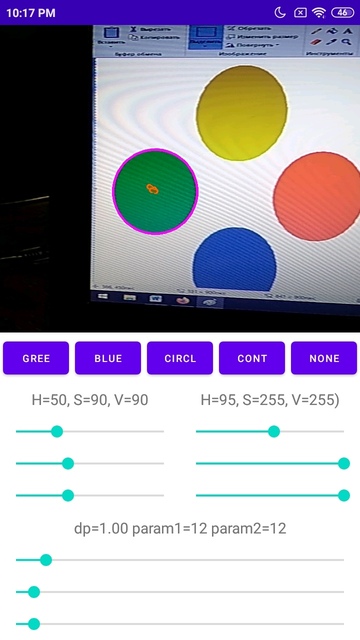


Рисунок 4 – Выделение круглой фигуры зелёного цвета и трассировка

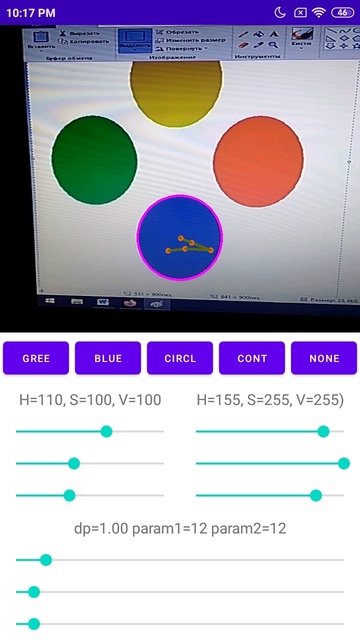


Рисунок 5 – Выделение круглой фигуры синего цвета и трассировка

**Выводы**

С помощью библиотеки OpenCV была разработана программа для распознания и трассировки при движении цветных круглых фигур для мобильной платформы Android.

**Код программы**

*//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by Fernflower decompiler)  
//*package ru.gureev.opencvapplication;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.view.View.OnClickListener;  
import android.widget.Button;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import java.util.ArrayList;  
import java.util.List;  
import org.opencv.android.CameraBridgeViewBase;  
import org.opencv.android.OpenCVLoader;  
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;  
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;  
import org.opencv.core.Core;  
import org.opencv.core.Mat;  
import org.opencv.core.MatOfPoint;  
import org.opencv.core.Point;  
import org.opencv.core.Scalar;  
import org.opencv.core.Size;  
import org.opencv.imgproc.Imgproc;  
  
public class MainActivity extends AppCompatActivity implements CvCameraViewListener2 {  
 private static final String TAG = "MainActivity";  
 private static final int REQUEST\_CODE\_PERMISSION\_CAMERA = 7777;  
 private CameraBridgeViewBase mOpenCvCameraView;  
 Mat mRGBA;  
 Mat mRGBAT;  
 Mat dst;  
 Mat blurredImage;  
 Mat hsvImage;  
 Mat mask;  
 Mat morphOutput;  
 Mat circles;  
 Mat masked;  
 Mat thresh;  
 Mat hierarchy;  
 Scalar minValues;  
 Scalar maxValues;  
 Boolean detectCircle = false;  
 Boolean detectContour = false;  
 String minText;  
 String maxText;  
 String circleConfigText;  
 List<Point> tracking = new ArrayList();  
 private TextView textMinHSV;  
 private TextView textMaxHSV;  
 private TextView textHoughCircles;  
 private SeekBar seekBarMinH;  
 private SeekBar seekBarMinS;  
 private SeekBar seekBarMinV;  
 private SeekBar seekBarMaxH;  
 private SeekBar seekBarMaxS;  
 private SeekBar seekBarMaxV;  
 private SeekBar seekBarDp;  
 private SeekBar seekBarParam1;  
 private SeekBar seekBarParam2;  
 private Button buttonRed;  
 private Button buttonGreen;  
 private Button buttonBlue;  
 private Button buttonCircle;  
 private Button buttonContour;  
 private Button buttonNone;  
 OnClickListener onClickListener = new OnClickListener() {  
 public void onClick(View v) {  
 switch(v.getId()) {  
 case 2131230809:  
 MainActivity.this.seekBarMinH.setProgress(110);  
 MainActivity.this.seekBarMinS.setProgress(100);  
 MainActivity.this.seekBarMinV.setProgress(100);  
 MainActivity.this.seekBarMaxH.setProgress(155);  
 MainActivity.this.seekBarMaxS.setProgress(255);  
 MainActivity.this.seekBarMaxV.setProgress(255);  
 break;  
 case 2131230810:  
 MainActivity.this.detectCircle = true;  
 MainActivity.this.detectContour = false;  
 break;  
 case 2131230811:  
 MainActivity.this.detectCircle = false;  
 MainActivity.this.detectContour = true;  
 break;  
 case 2131230812:  
 MainActivity.this.seekBarMinH.setProgress(50);  
 MainActivity.this.seekBarMinS.setProgress(90);  
 MainActivity.this.seekBarMinV.setProgress(90);  
 MainActivity.this.seekBarMaxH.setProgress(95);  
 MainActivity.this.seekBarMaxS.setProgress(255);  
 MainActivity.this.seekBarMaxV.setProgress(255);  
 break;  
 case 2131230813:  
 MainActivity.this.detectCircle = false;  
 MainActivity.this.detectContour = false;  
 case 2131230814:  
 default:  
 break;  
 case 2131230815:  
 MainActivity.this.seekBarMinH.setProgress(0);  
 MainActivity.this.seekBarMinS.setProgress(110);  
 MainActivity.this.seekBarMinV.setProgress(110);  
 MainActivity.this.seekBarMaxH.setProgress(15);  
 MainActivity.this.seekBarMaxS.setProgress(255);  
 MainActivity.this.seekBarMaxV.setProgress(255);  
 }  
  
 }  
 };  
  
 public MainActivity() {  
 }  
  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 this.setContentView(2131427356);  
 int permissionStatus = ContextCompat.checkSelfPermission(this, "android.permission.CAMERA");  
 if (permissionStatus == 0) {  
 Toast.makeText(this, "CAMERA\_PERMISSION\_GRANTED", 0).show();  
 } else {  
 ActivityCompat.requestPermissions(this, new String[]{"android.permission.CAMERA"}, 7777);  
 Toast.makeText(this, "CAMERA\_PERMISSION\_NOT\_GRANTED", 0).show();  
 System.exit(-1);  
 }  
  
 this.getWindow().addFlags(128);  
 this.mOpenCvCameraView = (CameraBridgeViewBase)this.findViewById(2131230912);  
 this.mOpenCvCameraView.setCvCameraViewListener(this);  
 this.seekBarMinH = (SeekBar)this.findViewById(2131231017);  
 this.seekBarMinS = (SeekBar)this.findViewById(2131231018);  
 this.seekBarMinV = (SeekBar)this.findViewById(2131231019);  
 this.seekBarMaxH = (SeekBar)this.findViewById(2131231014);  
 this.seekBarMaxS = (SeekBar)this.findViewById(2131231015);  
 this.seekBarMaxV = (SeekBar)this.findViewById(2131231016);  
 this.seekBarDp = (SeekBar)this.findViewById(2131231013);  
 this.seekBarParam1 = (SeekBar)this.findViewById(2131231020);  
 this.seekBarParam2 = (SeekBar)this.findViewById(2131231021);  
 this.textMinHSV = (TextView)this.findViewById(2131231076);  
 this.textMaxHSV = (TextView)this.findViewById(2131231075);  
 this.textHoughCircles = (TextView)this.findViewById(2131231074);  
 this.buttonRed = (Button)this.findViewById(2131230815);  
 this.buttonGreen = (Button)this.findViewById(2131230812);  
 this.buttonBlue = (Button)this.findViewById(2131230809);  
 this.buttonCircle = (Button)this.findViewById(2131230810);  
 this.buttonContour = (Button)this.findViewById(2131230811);  
 this.buttonNone = (Button)this.findViewById(2131230813);  
 this.buttonRed.setOnClickListener(this.onClickListener);  
 this.buttonGreen.setOnClickListener(this.onClickListener);  
 this.buttonBlue.setOnClickListener(this.onClickListener);  
 this.buttonCircle.setOnClickListener(this.onClickListener);  
 this.buttonContour.setOnClickListener(this.onClickListener);  
 this.buttonNone.setOnClickListener(this.onClickListener);  
 }  
  
 public void onCameraViewStarted(int width, int height) {  
 this.mRGBAT = new Mat();  
 this.dst = new Mat();  
 this.blurredImage = new Mat();  
 this.hsvImage = new Mat();  
 this.mask = new Mat();  
 this.morphOutput = new Mat();  
 this.circles = new Mat();  
 this.masked = new Mat();  
 this.thresh = new Mat();  
 this.hierarchy = new Mat();  
 this.minValues = new Scalar((double)this.seekBarMinH.getProgress(), (double)this.seekBarMinS.getProgress(), (double)this.seekBarMinV.getProgress());  
 this.maxValues = new Scalar((double)this.seekBarMaxH.getProgress(), (double)this.seekBarMaxS.getProgress(), (double)this.seekBarMaxV.getProgress());  
 }  
  
 public void onCameraViewStopped() {  
 }  
  
 public Mat onCameraFrame(CvCameraViewFrame inputFrame) {  
 this.mRGBA = inputFrame.rgba();  
 Core.transpose(this.mRGBA, this.mRGBAT);  
 Core.flip(this.mRGBAT, this.mRGBAT, 1);  
 Imgproc.resize(this.mRGBAT, this.dst, this.mRGBA.size());  
 this.mRGBA.release();  
 this.mRGBAT.release();  
 return this.configOpenCVCamera();  
 }  
  
 protected void onPause() {  
 super.onPause();  
 if (this.mOpenCvCameraView != null) {  
 this.mOpenCvCameraView.disableView();  
 }  
  
 }  
  
 protected void onResume() {  
 super.onResume();  
 OpenCVLoader.initDebug();  
 this.mOpenCvCameraView.enableView();  
 }  
  
 protected void onDestroy() {  
 super.onDestroy();  
 if (this.mOpenCvCameraView != null) {  
 this.mOpenCvCameraView.disableView();  
 }  
  
 }  
  
 Mat configOpenCVCamera() {  
 Imgproc.blur(this.dst, this.blurredImage, new Size(1.0D, 1.0D));  
 Imgproc.cvtColor(this.blurredImage, this.hsvImage, 40);  
 this.minValues.set(new double[]{(double)this.seekBarMinH.getProgress(), (double)this.seekBarMinS.getProgress(), (double)this.seekBarMinV.getProgress()});  
 this.maxValues.set(new double[]{(double)this.seekBarMaxH.getProgress(), (double)this.seekBarMaxS.getProgress(), (double)this.seekBarMaxV.getProgress()});  
 this.runOnUiThread(() -> {  
 this.minText = String.format("H=%d, S=%d, V=%d", this.seekBarMinH.getProgress(), this.seekBarMinS.getProgress(), this.seekBarMinV.getProgress());  
 this.maxText = String.format("H=%d, S=%d, V=%d)", this.seekBarMaxH.getProgress(), this.seekBarMaxS.getProgress(), this.seekBarMaxV.getProgress());  
 this.circleConfigText = String.format("dp=%.2f param1=%d param2=%d", 0.1D \* (double)this.seekBarDp.getProgress(), this.seekBarParam1.getProgress(), this.seekBarParam2.getProgress());  
 this.textMinHSV.setText(this.minText);  
 this.textMaxHSV.setText(this.maxText);  
 this.textHoughCircles.setText(this.circleConfigText);  
 });  
 Core.inRange(this.hsvImage, this.minValues, this.maxValues, this.mask);  
 if (this.detectCircle) {  
 Imgproc.HoughCircles(this.mask, this.circles, 3, 0.1D \* (double)this.seekBarDp.getProgress(), (double)(this.mask.rows() / 2), (double)this.seekBarParam1.getProgress(), (double)this.seekBarParam2.getProgress(), 0, 200);  
 Log.d("MainActivity", "configOpenCVCamera: mask.rows() = " + this.mask.rows());  
 Log.d("MainActivity", "configOpenCVCamera: mask.cols() = " + this.mask.cols());  
 Log.d("MainActivity", "configOpenCVCamera: circles.rows() = " + this.circles.rows());  
 Log.d("MainActivity", "configOpenCVCamera: circles.cols() = " + this.circles.cols());  
  
 for(int x = 0; x < this.circles.cols(); ++x) {  
 double[] c = this.circles.get(0, x);  
 Point center = new Point((double)Math.round(c[0]), (double)Math.round(c[1]));  
 int radius = (int)Math.round(c[2]);  
 Log.d("MainActivity", "configOpenCVCamera: radius = " + radius);  
 Imgproc.circle(this.dst, center, radius, new Scalar(255.0D, 0.0D, 255.0D), 3, 8, 0);  
 this.addNewTrackPoint(center);  
  
 for(int i = 0; i < this.tracking.size() - 1; ++i) {  
 Imgproc.circle(this.dst, (Point)this.tracking.get(i), 3, new Scalar(0.0D, 100.0D, 255.0D), 3, 8, 0);  
 Imgproc.line(this.dst, (Point)this.tracking.get(i), (Point)this.tracking.get(i + 1), new Scalar(0.0D, 100.0D, 100.0D), 3, 8, 0);  
 }  
  
 if (x == 0) {  
 break;  
 }  
 }  
  
 if (this.circles.cols() == 0) {  
 this.tracking.removeAll(this.tracking);  
 }  
 } else if (this.detectContour) {  
 this.dst.copyTo(this.masked, this.mask);  
 Imgproc.threshold(this.mask, this.thresh, 1.0D, 255.0D, 0);  
 List<MatOfPoint> contours = new ArrayList();  
 Imgproc.findContours(this.thresh, contours, this.hierarchy, 0, 2);  
 if (this.hierarchy.size().height > 0.0D && this.hierarchy.size().width > 0.0D) {  
 for(int idx = 0; idx >= 0; idx = (int)this.hierarchy.get(0, idx)[0]) {  
 Imgproc.drawContours(this.dst, contours, idx, new Scalar(255.0D, 0.0D, 255.0D), 3, 8);  
 }  
 }  
 }  
  
 Imgproc.cvtColor(this.dst, this.dst, 3);  
 this.masked.release();  
 this.hsvImage.release();  
 this.thresh.release();  
 this.hsvImage.release();  
 this.mask.release();  
 this.blurredImage.release();  
 this.circles.release();  
 return this.dst;  
 }  
  
 void addNewTrackPoint(Point point) {  
 if (this.tracking.size() <= 5) {  
 this.tracking.add(point);  
 } else {  
 this.tracking.remove(0);  
 this.tracking.add(point);  
 }  
  
 }  
  
 public void startActivity(Intent intent) {  
 super.startActivity(intent);  
 }  
  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 switch(requestCode) {  
 case 7777:  
 if (grantResults.length > 0 && grantResults[0] == 0) {  
 Toast.makeText(this, "CAMERA\_PERMISSION\_GRANTED", 0).show();  
 } else {  
 Toast.makeText(this, "CAMERA\_PERMISSION\_NOT\_GRANTED", 0).show();  
 System.exit(-1);  
 }  
  
 return;  
 default:  
 }  
 }  
}