УДК 004

Создание настольного приложения для распознавания регистрационных знаков транспортных средств на базе библиотеки EmguCV с помощью языка программирования C#

Эрдман Александр Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрен процесс создания настольного приложения с функцией распознавания государственных регистрационных номеров транспортных средств. Приложение написано на языке программирования C# с помощью библиотек WinForms и EmguCV. Результатом исследования будет являться настольное приложения и описание его работы.
Ключевые слова: C#, WinForms, EmguCV, компьютерное зрение

Creating a desktop application for vehicle registration plate recognition based on the EmguCV library using the C programming language#

Erdman Alexander Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article describes the process of creating a desktop application with the function of recognizing state registration numbers of vehicles. The application is written in the C# programming language using the WinForms and EmguCV libraries. The result of the study will be a desktop application and a description of its operation.
Keywords: C#, WinForms, EmguCV, computer vision

1 Введение

1.1 Актуальность

Компьютерное зрение является фундаментальной областью исследований и разработок, обретающей все большую актуальность в современном мире. С ростом доступности высокотехнологичных камер, датчиков и сенсоров, а также с прогрессом в области машинного обучения, компьютерное зрение открывает уникальные возможности для автоматизации процессов, оптимизации бизнес-процессов и улучшения жизни людей. Эта технология находит применение в различных областях, включая медицину, автомобильную промышленность, безопасность, розничную торговлю и многое другое. Системы компьютерного зрения способны не только распознавать объекты и лица, но и анализировать контекст, делая их незаменимыми инструментами для решения сложных задач.

В данной статье была выбрана область применения компьютерного зрения в сфере транспорта. Распознавание государственных регистрационных номеров (ГРЗ) с использованием компьютерного зрения представляет собой актуальную и стратегически значимую область разработки в современном мире. С ростом числа автомобилей и увеличением потока транспорта на дорогах, эффективные системы распознавания номерных знаков становятся неотъемлемой частью обеспечения безопасности и контроля на дорогах. Технологии компьютерного зрения позволяют автоматически и точно распознавать номера автомобилей, что не только улучшает процессы управления транспортной инфраструктурой, но и играет ключевую роль в борьбе с преступностью и обеспечении безопасности общества. В связи с актуальностью темы в исследование будет описан процесс разработки настольного приложения для распознавания ГРЗ автомобилей.

1.2 Обзор исследований

К.С. Евстраткин, А.Р. Султанова и А.В. Ерпелев рассмотрели основные возможности библиотеки OpenCV и варианты её использования [1]. В.Э. Юзиев в своей работе использовал библиотеку EmguCV для решения задачи распознавания контуров пространственных объектов на растровых картах [2]. Е.Г. Шапович и А.В. Шах исследовали библиотеку EmguCV и её возможности для реализации модуля обнаружения лиц на изображении [3]. В.В. Скрыпкин описал систему контроля посещаемости занятий студентами при помощи технологии машинного зрения EmguCV [4]. В.В. Кравченков разработал систему распознавания лиц на базе методов Виолы-Джонса и SURF [5].

1.3 Цель исследования

Целью исследования является создание настольного приложения для распознавания ГРЗ автомобилей на фото с помощью языка программирования C#.

2 Материалы и методы

Для создания приложения используется язык программирования C#, библиотеки WinForms и EmguCV. В качестве IDE используется Visual Studio 2022.

3 Результаты и обсуждения

Для реализации программы нужно создать графический интерфейс, то есть настроить форму, написать главный класс с методами, а также реализовать обработчики событий на кнопки в интерфейсе.

Перед выполнением основных действий создаётся проект Windows Forms. В проект для дальнейшей работы с библиотекой EmguCV загружается соответствующий модуль с помощью инструмента NuGet, затем начинается работа над интерфейсом.

Настройка графического интерфейса заключается в размещение на форме элементов – menuStrip, в которой реализуются вкладки «Файл» и её

дочерняя вкладка «Открыть», с помощью которых можно будет открывать изображения для работы; элемент toolStrip используется для размещения кнопки, которая будет вызывать методы для распознавания ГРЗ автомобилей на изображении; элемент TableLoyoutPanel разделяет окно приложения на две области, в которых дополнительно размещаются Panel, содержащие исходное изображение и распознанные номера. Графическая составляющая приложения реализована (рис. 1).

Рисунок 1. Внешний вид приложения

После завершения работы над графической составляющей создаётся класс NumberPlateRecognizer, который будет содержать все необходимые методы для обработки изображения.

В начале класс подключается модули EmguCV. Далее создаётся приватное поле Tesseract и конструктор для класса. Конструктор содержит два параметра типа string – первый параметр отвечает за путь к папке с языковыми моделями, а второй параметр определят язык. В конструкторе создаётся новый экземпляр класса Tesseract и присвоение ссылки на новый класс. Параметры наследуются от конструктора, а также объявляется уникальный OcrEngineMode.TesseractLstmCombined -метод распознавания (рис. 2).

```
using Emgu;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.OCR;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using Emgu.Util;

namespace GRZDetector
{
    Ссылок: 3
    internal class NumberPlateRecognizer : DisposableObject
    {
        private Tesseract ocr;

        Ссылок: 1
        public NumberPlateRecognizer(string tessdataPath, string lang)
        {
            ocr = new Tesseract(tessdataPath, lang, OcrEngineMode.TesseractLstmCombined);
        }
```

Рисунок 2. Объвление модулей и реализация конструктора NumberPlateRecognizer

Затем начинается программирование главного публичного метода, который будет возвращать распознанные ГРЗ в виде текста, а также картинок с вырезанными с общего изображения с ГРЗ.

Перед реализацией класса создаются приватные методы, которые будут представлять основной инструментарий класса.

Первым методом прописывается GetNumberOfChildren (рис. 3). Данный метод используется для обнаружения в предполагаемом регионе изображения с номером символов, свидетельствующих о том, что на изображение находится ГРЗ. В случае же их отсутствия, то попытка на распознание регистрационных знаков автомобиля будет прекращена.

```
private int GetNumberOfChildren(int[,] hierarchy, int index)
{
    index = hierarchy[index, 2];

    if (index < 0)
        return 0;

    int count = 1;

    while (hierarchy[index, 0] > 0)
    {
        count++;
        index = hierarchy[index, 0];
    }
    return count;
}
```

Рисунок 3. Метод GetNumberOfChildren

Метод принимает двумерный массив hierarchy и индекс элемента в этом массиве. В начале метода значение index изменяется на значение, хранящееся в ячейке hierarchy по индексу и второму столбцу (hierarchy[index, 2]). Затем

выполняется цикл while, в котором увеличивается счетчик count, и индекс обновляется значением из первого столбца (hierarchy[index, 0]) до тех пор, пока это значение больше нуля. Метод возвращает общее количество потомков для указанного элемента в массиве hierarchy.

Следующим объявляемым методом является FilterPlate, который на входе получает изображение с ГРЗ автомобиля, а на выходе возвращает изображение без шума (рис. 4). Данное действие необходимо для повышения точности распознавания шрифта знака.

```
private static UMat FilterPlate(UMat plate)
{
    UMat thresh = new UMat();

    CvInvoke.Threshold(plate, thresh, 120, 255, ThresholdType.BinaryInv);

    Size plateSize = plate.Size;

    using (Mat plateMask = new Mat(plateSize.Height, plateSize.Width, DepthType.Cv8U, 1))
    {
        using (Mat plateCanny = new Mat())
        {
            using (VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint())
            {
                plateMask.SetTo(new MCvScalar(255.0));

                CvInvoke.Canny(plate, plateCanny, 100, 50);

                CvInvoke.FindContours(plateCanny, contours, null, RetrType.External, ChainApproxMethod.ChainApproxSimple);
                int count = contours.Size;
                for (int i = 0; i < count; i++)
                {
                    using (VectorOfPoint contour = contours[i])
                    {
                        Rectangle rect = CvInvoke.BoundingRectangle(contour);
                        if (rect.Height > (plateSize.Height >> 1))
                        {
                            rect.X -= 1;
                            rect.Y -= 1;
                            rect.Width += 2;
                            rect.Height += 2;

                            Rectangle roi = new Rectangle(Point.Empty, plate.Size);

                            rect.Intersect(roi);

                            CvInvoke.Rectangle(plateMask, rect, new MCvScalar(), -1);
                        }
                    }
                }
                thresh.SetTo(new MCvScalar(), plateMask);
            }
        }
    }

    CvInvoke.Erode(thresh, thresh, null, new Point(-1, -1), 1, BorderType.Constant, CvInvoke.MorphologyDefaultBorderValue);
    CvInvoke.Dilate(thresh, thresh, null, new Point(-1, -1), 1, BorderType.Constant, CvInvoke.MorphologyDefaultBorderValue);

    return thresh;
}
```

Рисунок 4. Метод FilterPlate

Как описывалось выше, данный метод принимает изображение типа UMat. Данный тип данных предоставляет унифицированный интерфейс для работы с матрицами изображений. В начале метода создается новый UMat с именем thresh. Затем используется функция CvInvoke.Threshold для бинарной инвертированной пороговой обработки входного изображения plate с пороговым значением 120. Далее создаются матрицы plateMask и plateCanny, а также вектор contours для хранения контуров. Заполняется plateMask белым цветом. Применяется операция Canny для обнаружения границ на изображении plate, и затем находятся контуры с использованием функции FindContours. Затем происходит цикл по контурам, где для каждого контура

вычисляется ограничивающий прямоугольник (bounding rectangle). Если высота прямоугольника больше половины высоты изображения plate, то прямоугольник модифицируется (увеличивается и сдвигается), чтобы применить его в виде маски для обнуления соответствующих областей в thresh.

В конце метода применяются морфологические операции эрозии и дилатации для улучшения полученной маски. В результате возвращается полученная и обработанная маска thresh.

Для очищения ресурсов программы, которые были использованы в ходе работы программы, создаётся метод DisposeObject. Для корректной реализации метода необходимо сделать наследование класса NumberPlateRecognizer, а затем осуществить переопределение метода с помощью операции ocr.Dispose.

После написания вспомогательных команд осуществляется программирование главной функции, в которой происходит обнаружение всех имеющихся на изображении областей с ГРЗ и их непосредственное распознавание (рис. 5, 6).

```
private void FindLicensePlate(VectorOfVectorOfPoint contours,
    int[,] hierarchy,
    int index,
    IInputArray gray,
    IInputArray canny,
    List<IInputOutputArray> licensePlateImageList,
    List<IInputOutputArray> filteredLicensePlateImageList,
    List<RotatedRect> detectedLicensePlateRegionList,
    List<string> licenses)
{
    for (; index >= 0; index = hierarchy[index, 0])
    {
        int numberOfChildren = GetNumberOfChildren(hierarchy, index);

        if (numberOfChildren == 0)
            continue;
        using (VectorOfPoint contour = contours[index])
        {
            if (CvInvoke.ContourArea(contour) > 400)
            {
                if (numberOfChildren < 3)
                {
                    FindLicensePlate(contours, hierarchy, hierarchy[index, 2], gray, canny,
                    licensePlateImageList, filteredLicensePlateImageList, detectedLicensePlateRegionList,
                    licenses);

                    continue;
                }

                RotatedRect box = CvInvoke.MinAreaRect(contour);

                if (box.Angle < -45.0)
                {
                    float tmp = box.Size.Width;
                    box.Size.Width = box.Size.Height;
                    box.Size.Height = tmp;
                    box.Angle += 90.0f;
                }
                else if (box.Angle > 45.0)
                {
                    float tmp = box.Size.Width;

                    box.Size.Width = box.Size.Height;
                    box.Size.Height = tmp;
                    box.Angle -= 90.0f;
                }
                double whRatio = (double)box.Size.Width / box.Size.Height;
```

Рисунок 5. Код функции FindLicensePlate

В начале функции FindLicensePlate принимает параметры, такие как вектор контуров (contours), иерархия контуров (hierarchy), индекс текущего контура (index), изображение в оттенках серого (gray), изображение после операции Canny (canny), списки для хранения изображений номерных знаков (licensePlateImageList и filteredLicensePlateImageList), список для хранения обнаруженных повернутых прямоугольных областей (detectedLicensePlateRegionList) и список для хранения распознанных регистрационных номеров автомобиля (licenses).

Основной цикл for проходит по иерархии контуров, начиная с заданного индекса и двигаясь вверх по иерархии (hierarchy[index, 0]). Внутри цикла проверяется количество потомков текущего контура с использованием метода GetNumberOfChildren. Если количество потомков равно 0, цикл продолжается со следующего контура. Затем проверяется площадь текущего контура, и, если она больше 400 и количество потомков меньше 3, вызывается рекурсивный вызов метода FindLicensePlate для обработки потомка текущего контура.

Далее, для каждого контура, вычисляется ограничивающий повернутый прямоугольник (box) с использованием метода MinAreaRect. В зависимости от угла поворота прямоугольника, его размеры и угол могут быть скорректированы. Затем вычисляется отношение ширины к высоте (whRatio) для ограничивающего прямоугольника.

После проверки отношения ширины к высоте ограничивающего прямоугольника, выполняется дополнительная проверка. Если отношение не попадает в заданный диапазон ($3.0 < whRatio < 10.0$), и у текущего контура есть потомок ($hierarchy[index, 2] > 0$), то выполняется рекурсивный вызов метода FindLicensePlate для обработки потомка. Это помогает улучшить обнаружение, пропуская некоторые некорректные контуры. Затем создаются временные UMat (tmp1 и tmp2), которые будут использоваться для преобразования и изменения размера изображения. Вычисляются углы исходного и целевого прямоугольников; с использованием метода GetAffineTransform создается матрица преобразования rot. Исходное изображение gray преобразуется с помощью WarpAffine с использованием этой матрицы, что позволяет выровнять регистрационный номер в горизонтальное положение. Далее определяется размер (approxSize), к которому будет масштабировано изображение. Вычисляется коэффициент масштабирования, а затем изменяется размер изображения с использованием метода Resize. Применяется фильтрация номерного знака с использованием метода FilterPlate, который был описан выше.

После фильтрации выполняется распознавание текста с использованием OCR (оптического распознавания символов). Результат распознавания добавляется в список licenses. Также сохраняются изображение ГРЗ, отфильтрованное изображение, и информация о прямоугольной области в соответствующих списках (licensePlateImageList, filteredLicensePlateImageList, detectedLicensePlateRegionList).

```
if (!(3.0 < whRatio && whRatio < 10.0))
{
    if (hierarchy[index, 2] > 0)
    {
        FindLicensePlate(contours, hierarchy, hierarchy[index, 2], gray, canny,
        licensePlateImageList, filteredLicensePlateImageList, detectedLicensePlateRegionList,
        licenses);

        continue;
    }
}
using (UMat tmp1 = new UMat())
{
    using (UMat tmp2 = new UMat())
    {
        PointF[] srcCorners = box.GetVertices();
        PointF[] destCorners = new PointF[]
        {
            new PointF(0, box.Size.Height - 1),
            new PointF(0, 0),
            new PointF(box.Size.Width - 1, 0),
            new PointF(box.Size.Width - 1, box.Size.Height - 1)
        };
        using (Mat rot = CvInvoke.GetAffineTransform(srcCorners, destCorners))
        {
            CvInvoke.WarpAffine((IInputArray)gray, tmp1, rot, Size.Round(box.Size));
        }

        Size approxSize = new Size(240, 180);

        double scale = Math.Min(approxSize.Width / box.Size.Width,
            approxSize.Height / box.Size.Height);

        Size newSize = new Size((int)Math.Round(box.Size.Width * scale),
            (int)Math.Round(box.Size.Height * scale));

        CvInvoke.Resize(tmp1, tmp2, newSize, 0, 0, Inter.Cubic);

        int edgePixelSize = 3;
        Rectangle newRoi = new Rectangle(new Point(edgePixelSize, edgePixelSize),
            tmp2.Size - new Size(2 * edgePixelSize, 2 * edgePixelSize));

        UMat plate = new UMat(tmp2, newRoi);

        UMat filteredPlate = FilterPlate(plate);

        StringBuilder stringBuilder = new StringBuilder();

        using (UMat tmp = filteredPlate.Clone())
        {
            ocr.SetImage(tmp);

            ocr.Recognize();

            stringBuilder.Append(ocr.GetUTF8Text());

        }
        licenses.Add(stringBuilder.ToString());
        licensePlateImageList. Add(plate);
        filteredLicensePlateImageList.Add(filteredPlate);
        detectedLicensePlateRegionList.Add(box);
```

Рисунок 6. Код функции FindLicensePlate (продолжение)

Последний метод позволяет взаимодействовать и получать возвращаемые данные от класса NumberPlateRecognizer после своей работы, а также этот метод будет осуществлять предподготовку данных перед вызовом FindLicensePlate (рис. 7)

```
public List<string> DetectLicensePlates(IInputArray image,
    List<IInputOutputArray> licensePlateImageList,
    List<IInputOutputArray> filteredLicensePlateImageList,
    List<RotatedRect> detectedLicensePlateRegionList)
{

    List<string> licenses = new List<string>();

    using (Mat gray = new Mat())
    {
        using (Mat canny = new Mat())
        {
            using (VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint())
            {
                CvInvoke.CvtColor(image, gray, ColorConversion.Bgr2Gray);
                CvInvoke.Canny(gray, canny, 100, 50, 3, false);
                int[,] hierarchy = CvInvoke.FindContourTree(canny, contours, ChainApproxMethod.ChainApproxSimple);

                FindLicensePlate(contours, hierarchy, 0, gray, canny,
                    licensePlateImageList, filteredLicensePlateImageList, detectedLicensePlateRegionList,
                    licenses);
            }
        }
    }

    return licenses;
}
```

Рисунок 7. Метод DetectLicensePlates

После вызова метода происходит получение в качестве возвращаемого значения список с текстовыми представлениями ГРЗ. Также с помощью параметра detectedLicensePlateRegionList, который представляет из себя список, обращение к которому позволяет получить все прямоугольники, в которых находятся автомобильные номера для дальнейшей обводки на изображении.

Главный функциональный класс реализован и описан. В заключительном этапе разработки необходимо запрограммировать кнопки интерфейса, то есть создать обработчики событий.

В начале файле Form1.cs происходит подключение модулей EmguCV. Затем объявляются приватные поля: NumberPlateRecognizer, подключающий методы из класса с таким же названием; Point, представляющее точку, с которой будет начинаться процесс создания лейблов и pictureBox, которые в свою очередь будут содержать найденные номера автомобилей в левой панели интерфейса; Mat – входное изображение.

Первым обработчиком события является загрузка формы, в которой инициализируется класс plateRecognizer, содержащий путь к файлу с моделями библиотеки EmguCV, а также язык распознавания, в данном случае используется русский для распознавания российских ГРЗ автомобилей.

Вторым обработчиком является нажатие на кнопку «Открыть», после чего предоставляется выбор файла с помощью проводника.

Для того, чтобы программа добавляла лейбл и изображение в левой панели, используется метод AddLabelAndImage, позволяющий динамически генерировать интерфейс (рис. 8). В качестве параметров принимаются текст лейбла и изображение. В методе создается новый экземпляр Label, устанавливаются его текст, ширина, высота и расположение в соответствии с текущей точкой startPoint. После этого startPoint смещается по вертикали на высоту метки. Метка добавляется в коллекцию элементов контейнера panel1.Controls. Затем создается экземпляр PictureBox, в который загружается

изображение из входного массива image. Размер PictureBox устанавливается в размер матрицы изображения, а расположение устанавливается в текущую точку startPoint. После этого startPoint снова смещается по вертикали на высоту PictureBox 10 единиц для создания отступа. PictureBox также добавляется в коллекцию элементов контейнера panel1.Controls.

```
private void AddLabelAndImage(string labelText, IInputArray image)
{
    Label label = new Label();

    label.Text = labelText;
    label.Width = 100;
    label.Height = 30;
    label.Location = startPoint;

    startPoint.Y += label.Height;

    panel1.Controls.Add(label);

    PictureBox box = new PictureBox();

    Mat m = image.GetInputArray().GetMat();

    box.ClientSize = m.Size;
    box.Image = m.Bitmap;
    box.Location = startPoint;
    startPoint.Y += box.Height + 10;

    panel1.Controls.Add(box);
}
```

Рисунок 8. Метод размещения AddLabelAndImage

Последним методом класса является ProcessImage, который будет вызывать при нажатии на кнопку «Распознать ГРЗ». В данном методе происходит создание всех необходимых списков, которые будут передаваться в экземпляры методы класса NumberPlateRecognizer (риис. 9).

В начале метода создаются четыре списка: licensePlateImageList, filteredLicensePlatImageList, licenseBoxList и recognizedPlates. Затем происходит вызов метода DetectLicensePlates из объекта plateRecognizer, передавая ему изображение и созданные списки в качестве параметров. Метод DetectLicensePlates выполняет распознавание номерных знаков на изображении и возвращает список распознанных номеров. Далее происходит очистка панели panel1 и установка startPoint в начальную точку (10, 10).

В цикле перебираются все распознанные номера и для каждого номера создается новый объект типа Mat. Затем два изображения из licensePlateImageList и filteredLicensePlatImageList объединяются вертикально и помещаются в созданный объект Mat. После вызывается метод AddLabelAndImage, который добавляет метку с номером и объединенное изображение на панель panel1. Создается новый объект outputImage типа Image<Bgr, byte> и копируется входное изображение в него.

Во втором в цикле перебираются все объекты типа RotatedRect из списка licenseBoxList. Для каждого объекта получаются вершины прямоугольника и рисуется контур прямоугольника на изображении outputImage. В конце метода изображение outputImage устанавливается в качестве изображения для pictureBox1.

```
private void ProcessImage(IInputOutputArray image)
{
    List<IInputOutputArray> licensePlateImageList = new List<IInputOutputArray>();
    List<IInputOutputArray> filteredLicensePlatImageList = new List<IInputOutputArray>();
    List<RotatedRect> licenseBoxList = new List<RotatedRect>();

    List<string> recognizedPlates = plateRecognizer.DetectLicensePlates(image,
        licensePlateImageList, filteredLicensePlatImageList,
        licenseBoxList);

    panel1.Controls.Clear();

    startPoint = new Point(10, 10);

    for (int i = 0; i < recognizedPlates.Count; i++)
    {
        Mat dest = new Mat();

        CvInvoke.VConcat(licensePlateImageList[i], filteredLicensePlatImageList[i], dest);

        AddLabelAndImage($"Номер: {recognizedPlates[i]}", dest);
    }

    Image<Bgr, byte> outputImage = inputImage.ToImage<Bgr, byte>();

    foreach(RotatedRect rect in licenseBoxList)
    {
        PointF[] v = rect.GetVertices();

        PointF prevPoint = v[0];
        PointF firstPoint = prevPoint;
        PointF nextpoint = prevPoint;
        PointF lastpoint = nextpoint;

        for (int i = 1; i < v.Length; i++)
        {
            nextpoint = v[i];

            CvInvoke.Line(outputImage, Point.Round(prevPoint), Point.Round(nextpoint), new MCvScalar(0, 0, 255), 5,
                LineType.EightConnected, 0);
            prevPoint = nextpoint;
            lastpoint = prevPoint;
        }

        CvInvoke.Line(outputImage, Point.Round(lastpoint), Point.Round(firstPoint), new MCvScalar(0, 0, 255), 5,
            LineType.EightConnected, 0);
    }
    pictureBox1.Image = outputImage.Bitmap;
}
```

Рисунок 9. Метод ProcessImage

Функционал программы реализован. Для проверки приложения происходит тестирование на практике (рис. 10).

Из результатов видно, что приложение справилось с поставленной задачей, но имеет некоторые неточности – программа лучше распознаёт номера на изображениях, где присутствует минимум лишних объектов, то есть близко. На изображениях, где автомобиль стоит во дворе и имеет множество сторонних объектов, приложение справляется, но вместе с номерами автомобилей обнаруживаются зоны, которые подходят заданным условиям, но при этом не содержат номеров.

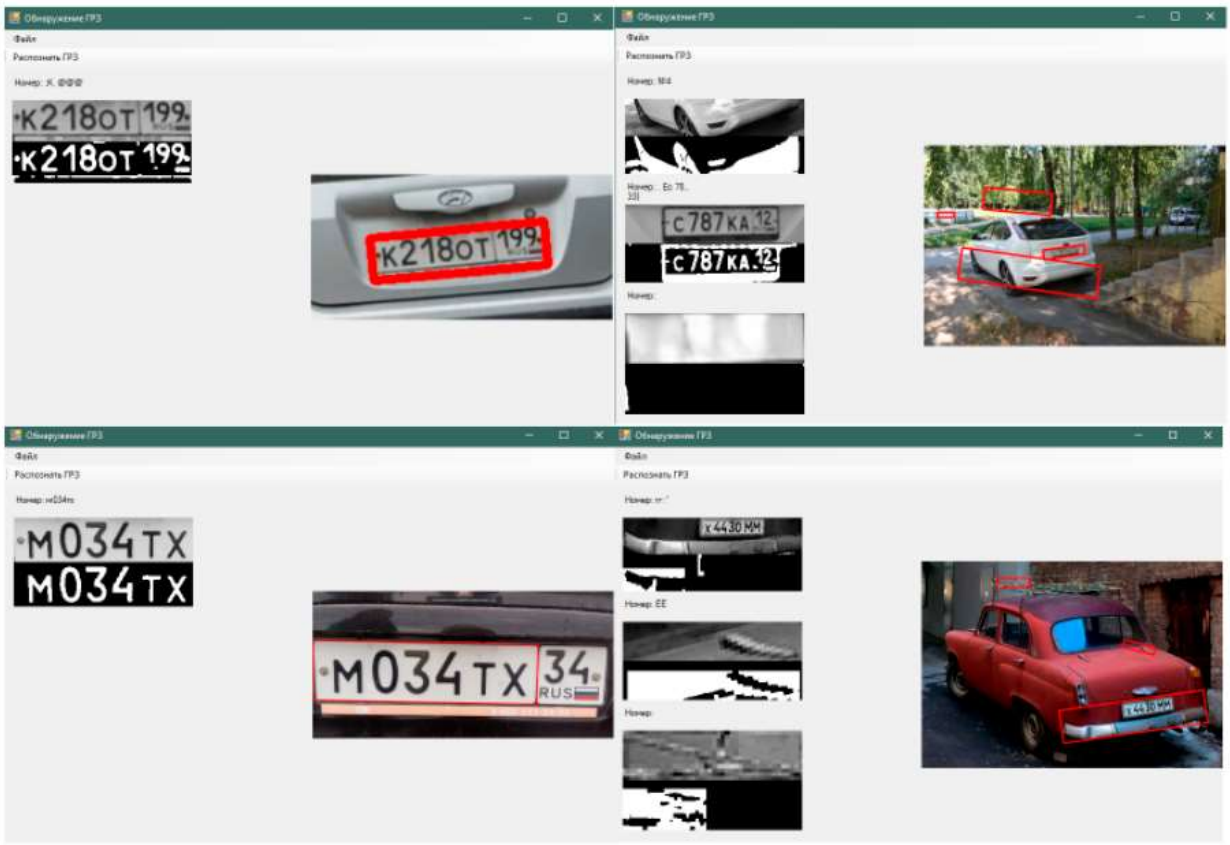

Рисунок 10. Результатты работы программы

Таким образом в результате исследования было реализовано и описано настольное приложение на языке программирования C# с функционалом поиска ГРЗ автомобилей на изображениях. Графическая составляющая была реализована на базе модуля WinForms. Функционал приложения был реализован с помощью модуля EmguCV. Также было проведено тестирование программы, в ходе которого были обнаружены номера автомобилей на изображениях. Было замечено, что приложение лучше определяло номера на изображениях, где было минимальное количество посторонних объектов.

Библиографический список

1. Евстраткин К.С., Султанова А.Р., Ерпелев А.В. OpenCV: варианты использования компьютерного зрения // В сборнике: Цифровые технологии: наука, образование, инновации. Материалы III Международного научного Форума профессорско-преподавательского состава и молодых ученых. Под редакцией А.В. Олейник, А.А. Зеленского. Москва, 2021. С. 28-31.
2. Юзиев В.Э. Применение библиотеки EmguCM для задач распознавания контуров пространственных объектов на растровых картах // Студенческий вестник. 2021. № 16-5 (161). С. 73-76.
3. Шапович Е.Г., Шах А.В. Модуль обнаружения лиц на изображении с использованием библиотеки EmguCV // Актуальные направления научных

исследований XXI века: теория и практика. 2019. Т. 7. № 1 (44). С. 480-483.
4. Скрыпкин В.В. Контроль посещаемости занятий студентами при помощи технологии машинного зрения EmguCV // В сборнике: Информационные технологии в процессе подготовки современного специалиста. Межвузовский сборник научных трудов. Липецк, 2020. С. 152-159.
5. Кравченков В.В. Распознавание лиц на основе применения методов Виолы-Джонса и SURF // Системы компьютерной математики и их приложения. 2017. № 18. С. 87-91.