Оригинальная статья / Original article

УДК 004.93

https://doi.org/10.21869/2223-1560-2024-28-2-148-165

Интеллектуализация процесса детектирования форм геометрических объектов

Н.А. Милостная [1] ✉

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: nat_mil@mail.ru

Резюме

Цель исследования. *Разработка программного кода на языке программирования C#, реализующего алгоритм распознавания формы геометрических объектов на входном изображении при обеспечении надежности этого вычислительного процесса.*

Методы. *Алгоритм распознавания геометрических форм объектов основан на сочетании традиционных методов обработки изображений и интеллектуальных правил, определяющих вид геометрической формы объекта в зависимости от характеристик контуров, таких как моменты, количество их сторон и т.д. Для реализации данного метода в статье предлагается последовательность математических операций, включающая следующие этапы. Во-первых, метод включает операции размытие, преобразование исходного изображения в градации серого и инвертирование. На втором этапе осуществляется детектирование контуров и определение их характеристик, таких как моменты, периметр и др. На финальном этапе в зависимости от числа сторон, входящих в структуру контура, на основе интеллектуальных правил осуществляется сопоставление каждого найденного контура определенной геометрической фигуры.*

Результаты. *Разработаны алгоритм и инструкции создания программного кода для программной реализации процесса распознавания геометрической формы объектов. Определено, что предложенный алгоритм имеет высокую надежность, составляющую порядка 97%.*

Заключение. *Традиционные методы обработки изображений, такие как размытие и преобразование в градации серого, могут успешно сочетаться с методами выделения контуров и определения их геометрических характеристик. Подобная синергия методов обработки изображений позволяет создать алгоритм распознавания геометрических форм. Важно учитывать, что надежность и эффективность подобного алгоритма зависит от настройки пороговых значений, используемых в функциях обработки изображений, и дальнейшее исследование их характеристик может привести к улучшению результатов рассмотренного в статье алгоритма.*

Ключевые слова: *распознавание; выделение контуров; обработка изображения; OpenCV; EMGU.*

Конфликт интересов: *Автор декларирует отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Финансирование*: Работа выполнена в рамках гранта РНФ №24-21-00055 – «Разработка интеллектуальной высокопроизводительной нейро-нечеткой системы технического зрения».*

Для цитирования: Милостная Н.А. Интеллектуализация процесса детектирования форм геометрических объектов // Известия Юго-Западного государственного университета. 2024. Т. 28, №2. С. 148-165. https://doi.org/10.21869/2223-1560-2024-28-2-148-165.

Поступила в редакцию 29.03.2024 *Подписана в печать 06.05.2024* *Опубликована 25.06.2024*

Intellectualization of the process of detecting shapes of geometric objects

Natalia A. Milostnaya [1] ✉

[1] Southwest State University
50 Let Oktyabrya str. 94, Kursk 305040, Russian Federation

✉ e-mail: nat_mil@mail.ru

Abstract

Purpose of research. *Development of program code in the C# programming language that implements an algorithm for recognizing the shape of geometric objects in the input image while ensuring the reliability of this computational process is the main goal of the article.*

Methods. *The algorithm for recognizing the geometric shapes of objects is based on a combination of traditional image processing methods and intelligent rules that determine the type of geometric shape of an object depending on the characteristics of the contours, such as moments, the number of their sides, etc. To implement this method, the following sequence of mathematical operations which includes the following stages is proposed in the article. Firstly, this method includes the following operations: blur, convert the original image to grayscale, and invert. Detection of contours and determination of their characteristics, such as moments, perimeter, etc., is carried out at the second stage of the method. And at the final stage, the comparison of each found contour of a certain geometric figure is carried out depending on the number of sides included in the structure of the contour.*

Results. *An algorithm and instructions for creating program code have been developed for the software implementation of the process of recognizing the geometric shape of objects. It was determined that the proposed algorithm has a high reliability approximately equal 97%.*

Conclusion. *Traditional image processing methods such as blurring and grayscale conversion can be successfully combined with methods for identifying contours and determining their geometric characteristics. This synergy of image processing methods makes it possible to create an algorithm for recognizing geometric shapes. It is important to consider that the reliability and efficiency of such an algorithm depends on the settings of the threshold values used in the image processing functions and further study of their characteristics can lead to improved results of the algorithm presented in the article.*

Keywords: *recognition; contour extraction; image processing; OpenCV; EMGU.*

Conflict of interest. *The author declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

Funding. *The work was carried out within the framework of the RNF grant No. 24-21-00055 – "Development of an intelligent high-performance neuro-fuzzy vision system".*

For citation: Milostnaya N. A. Intellectualization of the process of detecting shapes of geometric objects. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2024; 28(2): 148-165 (In Russ.). https://doi.org/10.21869/2223-1560-2024-28-2-148-165.

Received 29.03.2024 *Accepted 06.05.2024* *Published 25.06.2024*

***

Введение

Распознавание форм геометрических объектов является одной из фундаментальных задач в области компьютерного зрения и обработки изображений. Процесс детектирования форм геометрических объектов, как правило, заключается в определении максимальных перепадов значений интенсивности на искомом изображении. Найденные перепады интенсивности определяют признаки контуров искомых форм геометрических объектов.

В рамках теории распознавания образов выделяются несколько методов для решения подобных задач, к которым относятся:

1. Методы на основе выделения признаков контуров. Эти методы анализируют геометрические характеристики распознаваемых объектов, например, количество контуров, их площадь и/или периметр, другие параметры и реализуются на основе различных дескрипторов, таких как Фурье [1] и другие. Например, в статье [2] авторы использовали алгоритм группировки геометрических объектов при автоматическом раскрое листового материала с использованием локальных геометрических характеристик формы исследуемых деталей.

2. Методы на основе анализа геометрических преобразований топологии анализа данных. Эти методы исследуют зависимости, связанные с поворотом, масштабированием исходного изображения по отношению к эталонному изображению, объемом, цветом и другими характеристиками геометрических объектов. В качестве примеров подобных фильтров используются следующие технологии: Гамильтонова механика точечных ориентиров изображения, комплекс Вьеториса – Рипса или с помощью исследования расстояния Вассерштейна [3, 4].

3. Методы на основе моделей выделения контуров геометрических объектов. Одной из основных моделей данной группы является оператор Хафа, использующийся для нахождения различных форм объектов (круг, ромб, линия и др.) [5]. Другая группа методов этой группы включает операторы нахождения и поиска границ изображения, например, с помощью оператора Собеля [6, 7], детектора Кэнни [8-10].

4. Методы на основе машинного обучения. Эти методы используют обученные модели для классификации геометрических форм объектов, учитывая свойства пикселей и текстурные характеристики изображения. К данным моделям относятся методы, основанные на сверточных нейронных сетях [11, 12], алгоритме случайного леса[1] и методе опорных векторов [13, 14].

[1] Свидетельство о государственной регистрации программы для ЭВМ № 2019612230 Российская Федерация. Классификатор рукописных цифр на основе алгоритма случайного леса: № 2018661129: заявл. 12.10.2018: опубл. 13.02.2019 / А. В. Хинензон.

Методы на основе анализа нечетких признаков изображения. Основными операциями, реализуемыми в рамках этих моделей, является бинаризация изображения с дальнейшим выделением геометрических признаков исследуемых объектов на основе нечетко-логических операций [15-20].

Материалы и методы

Рассмотрим процесс детектировать геометрические фигуры на изображении с помощью библиотеки Emgu.CV, которая является кросс-платформенной оберткой для библиотеки обработки изображений OpenCV (Open Source Computer Vision Library), предназначенной для языков программирования .NET. Она позволяет вызывать функции OpenCV из .NET-совместимых языков, таких как C#, VB.NET и C++.

Алгоритм детектирования геометрических фигур включает следующие этапы.

1. Загрузка изображения.

Для реализации этого этапа необходимо воспользоваться методов openFileDialog (см. Листинг 1)

```
DialogResult res = openFileDialog.ShowDialog();
if(res == DialogResult.OK)
{
    input_image = new Image<Bgr, byte>(openFileDialog.FileName);
    pictureBox1.Image = input_image.Bitmap;
}
else
{
    MessageBox.Show("Изображение не Выбрано!",
    "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error );
}
```

Листинг 1. Загрузка изображения

Listing 1. Uploading an image

2. Размытие, преобразование в градации серого и инвертирование изображения.

Для реализации этой процедуры используем доступные функции библиотеки emgu.CV, такие как SmoothGaussian(), Convert<Gray, byte>() и ThresholdBinaryInv (new Gray(230), new Gray(255)). Вначале процедура SmoothGaussian() размывает входное изображение, затем функция Convert<Gray, byte>() преобразует полученное изображение в градации серого, и на последнем шаге процедура ThresholdBinaryInv(new Gray(230), new Gray(255)) инвертирует полученное изображение с учетом пороговых уровней, заданных в ней (см. Листинг 2).

```
Image<Gray, byte> grayImage = input_image.SmoothGaussian(5).Convert<Gray, byte>().ThresholdBinaryInv(new Gray(230), new Gray(255));
pictureBox2.Image = grayImage.Bitmap;
```

Листинг 2. Размытие, преобразование в оттенки серого и инвертирование

Listing 2. Blur, grayscale, and invert

Визуальное представление этих операций изображено на рис. 1.

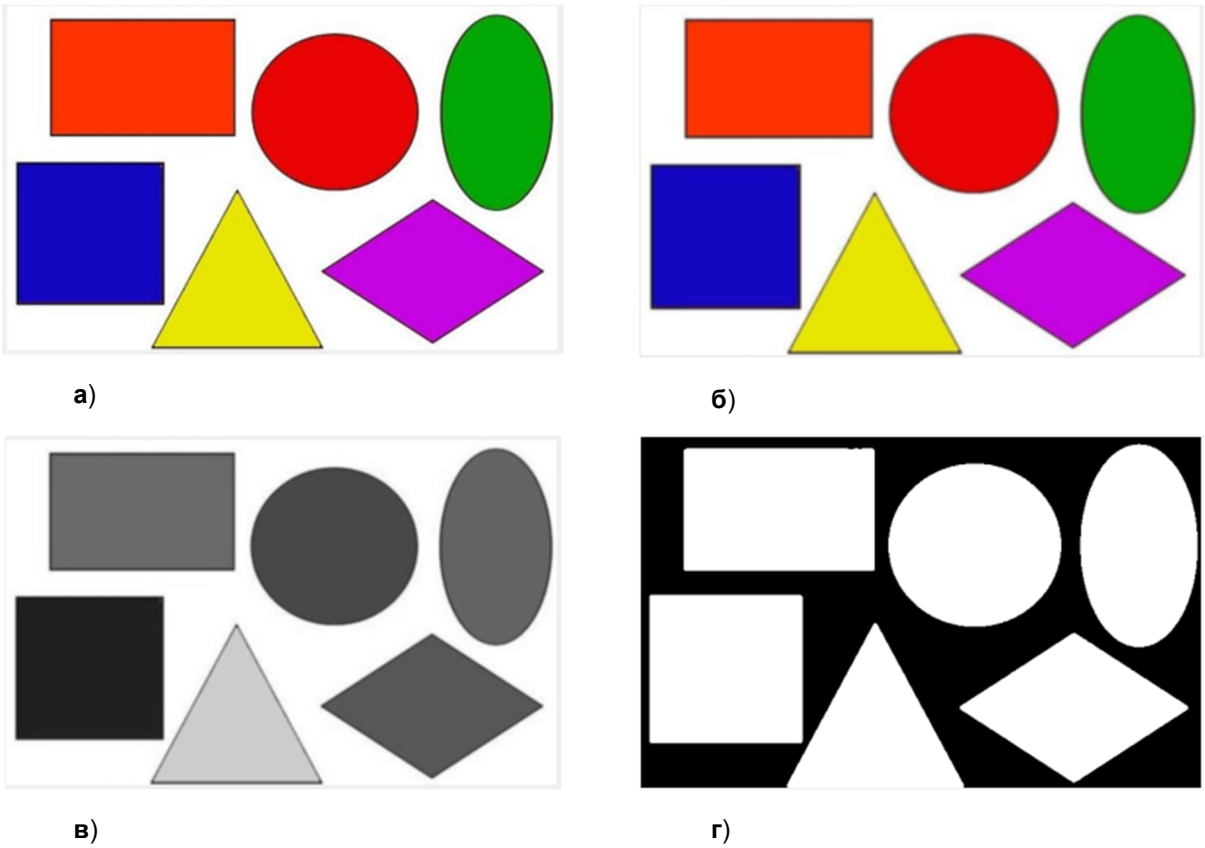


Рис. 1. Пошаговое выполнение операций: **а** – исходное изображение; **б** – размытие с помощью фильтра Гаусса; **в** – преобразование в градации серого; **г** – инверсия изображения

Fig. 1. Step-by-step execution of operations: **а** – original image; **б** – blur using a Gaussian filter; **в** – conversion to grayscale; **г** – image inversion

3. Создание переменной для хранения контуров.

Данная процедура в библиотеке EmguCV создается с помощью класса VectorOfVectorOfPoint [21]. Объявление переменной в C# для создания вектора контуров осуществляется с помощью команды:

```
VectorOfVectorOfPoint contours =
new VectorOfVectorOfPoint();
// contours – это переменная, ко-
торая хранить Вектор контуров
// Добавление контуров В Вектор
contours.Push(new VectorOfPoint(new
Point[] { new Point(10, 20),
          new Point(30, 40) }));
contours.Push(new VectorOfPoint(new
Point[] { new Point(50, 60),
          new Point(70, 80) }));
// Получение контуров из Вектора
VectorOfPoint contour1 = contours[0];
VectorOfPoint contour2 = contours[1];
```

Листинг 3. Построение контуров объекта

Listing 3. Building object contours

4. Создание многомерной матрицы для хранения массива контурных векторов.

Создание многомерной матрицы для хранения массива полученных векторов на предыдущем шаге использует

класс Mat(). В данном случае он представляет собой базовый контейнер для хранения изображений и матриц. Создание подобной матрицы осуществляется с помощью команды:

```
Mat matrix = new Mat();
```

где matrix – это переменная, которая хранит объект класса Mat().

5. Создание процедуры поиска контуров объектов.

Процедура поиска контуров объектов осуществляется с помощью класса CvInvoke.FindContours [22]. Данный метод позволяет извлечь контура из бинарного изображения полученного на втором шаге алгоритма. Процедура извлечения контуров записывается:

```
CvInvoke.FindContours(invert_grayImage,
contours, matrix_contours,
Emgu.CV.CvEnum.RetrType.External,
Emgu.CV.CvEnum.ChainApproxMethod.
ChainApproxSimple);
```

где invert_grayImage – изображение, из которого извлекаются контура (изображение, хранимое в переменной invert_grayImage должно быть в оттенках серого и бинарным);

contours – выходной параметр, в котором будут храниться обнаруженные контуры, представляющие собой вектор точек (координат пикселей на изображении);

matrix_contours – многомерная матрица, в которой будут храниться извлеченные контура с помощью метода CvInvoke.FindContours;

Emgu.CV.CvEnum.RetrType.External – указание режима извлечения контуров, в данном случае метод External означает, что извлекаться будут только внешние контуры, то есть те контуры, которые не содержаться внутри других контуров).

Emgu.CV.CvEnum.ChainApproxMethod.ChainApproxSimple – указание метода аппроксимации контуров, в данном случае используется простая аппроксимация ChainApproxSimple, которая используется для сокращения точек, описывающих извлеченный контур из бинарного изображения.

Отметим, что контурная аппроксимация – это процесс редукции количества точек в извлеченном контуре из бинарного изображения, при сохранении общей формы полученного контура. Например, для хранения прямоугольника достаточно хранить координаты (x, y) верхней крайней левой его точки и нижней правой точки. Данный метод аппроксимации основан на алгоритме Рамера-Дугласа-Пекера [23]. Он аппроксимирует контур, заменяя длинные последовательности близких точек одной прямой линией между начальной и конечной точками. Например, для описания прямой линии, состоящей из 100 точек, достаточно хранить только 2 точки – начальную и конечную.

6. Распознанных фигур по найденным контурам.

Данный этап включается несколько операций. Причем данная процедура выполняется в цикле и для всех найденных контуров на предыдущем этапе.

6.1. Нахождение периметра распознанного контура.

Данная процедура осуществляется с помощью метода CvInvoke.ArcLength, при этом в вещественную переменную perimeter записывается найденное значение периметра контура или его длина:

```
double perimetr =
CvInvoke.ArcLength(contours[i], true);
```

где CvInvoke.ArcLength – метод для нахождения периметра контура; contours[i] – переменная, которая указывает контур, хранящийся в массиве matrix_contours, для которого осуществляет расчет периметра; *i* – номер итерации или номер контура, для которого осуществляется расчет периметра; true – логическая переменная, если установлена в значение Истина, то метод рассматривает кривую как замкнутую, в противном случае, кривая рассматривается как незамкнутая.

6.2. Создание массива вектора точек.

Данная процедура осуществляется с помощью класса VectorOfPoint, который в C# библиотеке EmguCV используется для хранения и обработки наборов точек. Создание нового экземпляра класса VectorOfPoint осуществляется с помощью следующего конструктора:

```
VectorOfPoint approx =
new VectorOfPoint();
```

где approx – переменная, в которую записывается вектор аппроксимированных точек контура.

6.3. Аппроксимация найденного контура.

Данная процедура осуществляется с помощью метода CvInvoke.ApproxPolyDP, который используется для создания нового контура найденной геометрической фигуры с меньшим числом вершин на основе заданной точности редукции числа точек:

```
CvInvoke.ApproxPolyDP(contours[i],
approx, 0.04*perimetr, true);
```

где approx – переменная, в которой хранится результат аппроксимации или редукции числа точек; 0.04*perimetr – переменная, определяющая точность аппроксимации, то есть максимальное расстояние между исходной кривой и ее аппроксимацией; true – логическая переменная, указывающая, в случае Истина, что аппроксимированная кривая замкнута, то есть её первая и последняя вершины соединены, в противном случае она не замкнута.

6.4. Обрисовка контуров, найденных геометрических фигур.

Данная процедура осуществляется с помощью метода CvInvoke.DrawContours, который в библиотеке EmguCV используется для обрисовки контуров распознанных геометрических фигур на указанном изображении:

```
CvInvoke.DrawContours(input_image,
contours, i,
        new MCvScalar(0, 33, 255), 3)
```

где input_image – это входное изображение, на котором обрисовываются контуры; contours – массив контуров, который представлен в виде вектора то-

чек; i – индекс контура, который необходимо нарисовать, в случае указания отрицательного значения будут обрисованы все контуры; new MCvScalar(0, 33, 255) – цвет обрисовки контуров (оранжевый); 3 – толщина линий обрисовки контуров.

Программный код для реализации данной процедуры представлен в Листинге 4.

```
for (int i = 0; i < contours.Size; i++)
{
    double perimetr =
    CvInvoke.ArcLength(contours[i], true);
    VectorOfPoint approx =
    new VectorOfPoint();
    CvInvoke.ApproxPolyDP(contours[i],
approx, 0.04*perimetr, true);
    CvInvoke.DrawContours(input_image,
contours, i, new MCvScalar(0, 33, 255), 3);
    pictureBox2.Image =
    input_image.Bitmap;
}
```

Листинг 4. Процедура для детектирования контуров всех найденных геометрических объектов.

Listing 4. Procedure for detecting the contours of all found geometric objects

Результат работы программного кода, указанного в Листинге 3, приведен на рис. 2.

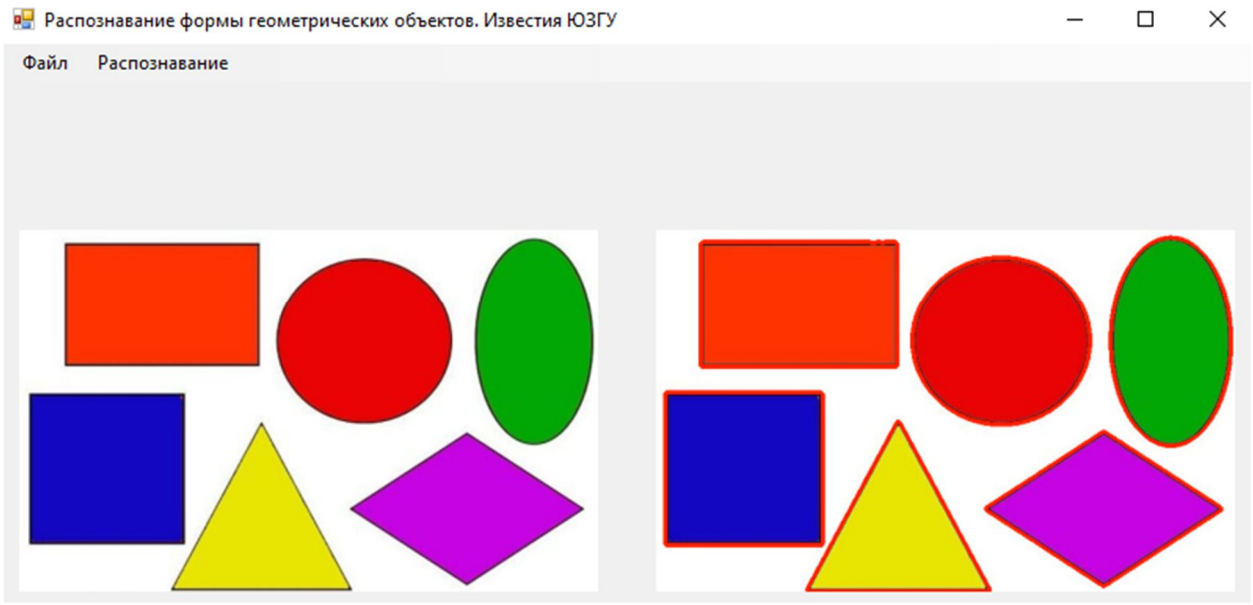


Рис. 2. Обрисовка распознанных контуров геометрических фигур

Fig. 2. Drawing recognized contours of geometric shapes

7. Распознавание формы найденных геометрических фигур.

Данный этап также состоит из нескольких вычислительных процедур, реализованных в одном цикле.

7.1. Нахождение центра масс распознанного контура.

С помощью этой процедуры в центре распознанного объекта будет размещаться надписать, соответствующая

форме найденной геометрической фигуры. Осуществляется с помощью созданного нового класса Moments, который используется для вычисления моментов (пространственных или центральных) до третьего порядка. С помощью моментов первого порядка вычисляются координаты центра тяжести (цента масс) распознанных геометрических фигур или их площадь. Для реализации этих вычислений рекомендуется пользоваться функциями: moments.M10 и moments.M01 – это моменты первого порядка по осям X и Y соответственно. Например, moments.M10 вычисляется как отношение суммы координат точек пикселя по оси абсцисс на их количество. Нулевой момент moments.M00 определяет площадь найденного контура геометрической фигуры. В нашем методе используем моменты только первого порядка:

```
Moments moments = CvIn-
voke.Moments(contours[i]);
int x = (int)(moments.M10
/ moments.M00);
int y = (int)(moments.M01
/ moments.M00);
```

Следует отметить, что моменты второго порядка: moments.M20, moments.M11 и moments.M02 используются для определения ориентации геометрического объекта. В то время как моменты третьего порядка moments.M30, moments.M21, moments.M12 и moments.M03 применяются для анализа формы объекта.

7.2. Анализ формы детектируемого объекта.

В данной процедуре анализируется количество контуров в распознанном объекте. Данная величина хранится в переменной, созданной на 6.2 этапе рассматриваемого алгоритма. В зависимости от значения этой переменной определяется форма геометрической фигуры.

В случае, если переменная approx=3 равняется трем, это означает, что распознанная фигура имеет форму треугольника. Тогда для добавления текста в точку центра распознанного треугольника используется метод CvInvoke.PutText библиотеки EmguCV. Он позволяет отображать на экране монитора текст с заданными параметрами шрифта и цвета:

```
CvInvoke.PutText(input_image,
"Triangle", new Point(x – 55, y),
   Em-
gu.CV.CvEnum.FontFace.HersheySimplex, 1,
   new MCvScalar(0, 0, 0), 2);
```

где input_image – указывается изображение и будет отображаться текст; "Triangle" – текст, который будет напечатан и обозначает форму распознанной фигуры (в данном случае треугольник); new Point(x – 55, y) – в эту переменную указываются координаты левого нижнего угла первой буквы текста смещенной на 55 пикселей по оси абсцисс; Emgu.CV.CvEnum.FontFace.HersheySimplex – указывается тип шрифта, в данном случае Hershey Simplex; 1 – в данной переменной указан масштаб шрифта;

new MCvScalar(0, 0, 0) – переменная в которой указан цвет шрифта, в данном случае черный; 2 – указывается толщина шрифта.

В случае, если переменная approx=4 равняется четырем, это означает, что распознанная фигура имеет прямоугольную форму. Данную форму могут иметь несколько типов фигур, например, квадрат, ромб и непосредственно прямоугольник.

Для распознавания квадрата необходимо вычислить соотношение сторон (ширины и высоты) распознанной геометрической фигуры rectRatio и проверить насколько близко это отношение к единице.

```
double rectRatio = CvIn-
voke.ContourArea(contours[i]) /
  (contours[i].Size * contours[i].Size);
if (rectRatio >= 0.8 && rectRatio <= 1.2)
      CvInvoke.PutText(input_image,
    "Square", new Point(x - 55, y),
  Emgu.CV.CvEnum.FontFace.HersheySimplex, 1,
   new MCvScalar(0, 0, 0), 2);
```

В противном случае форма распознанной фигуры будет прямоугольником.

Для проверки распознавания ромба необходимо вычислить показатель rhRatio, который определяет соотношение площади геометрической фигуры к квадрату его периметра. В случае, если разница модуля этой величины и единицы меньше порогового значения (по умолчанию 1.2), то это означает, что распознанная геометрическая фигура является ромбом.

```
double rhRatio = CvIn-
voke.ContourArea(contours[i]) /
(contours[i].Size * contours[i].Size);
      if (Math.Abs(rhRatio - 1.0)<1.2)
   CvInvoke.PutText(input_image,
"Rhombus", new Point(x - 55, y),
   Emgu.CV.CvEnum.FontFace.HersheySimplex, 1,
   new MCvScalar(0, 0, 0), 2);
```

В случае, если переменная approx=5 равняется пяти, это означает, что распознанная фигура имеет форму пентагона.

В случае, если переменная approx>5 равняется пяти, это означает, что распознанная фигура имеет закругленную форму. В этом случае необходимо распознать, является ли геометрическая фигура кругом или овалом. Для решения этой задачи необходимо ввести показатель crRatio, который также показывает отношение площади распознанной геометрической фигуры к его периметру в квадрате. В случае, если абсолютная разность этого параметра с единицей не превышает заданное пороговое значение (по умолчанию 0.15), то это означает, что найденная геометрическая фигура является кругом, в противном случае – это овал.

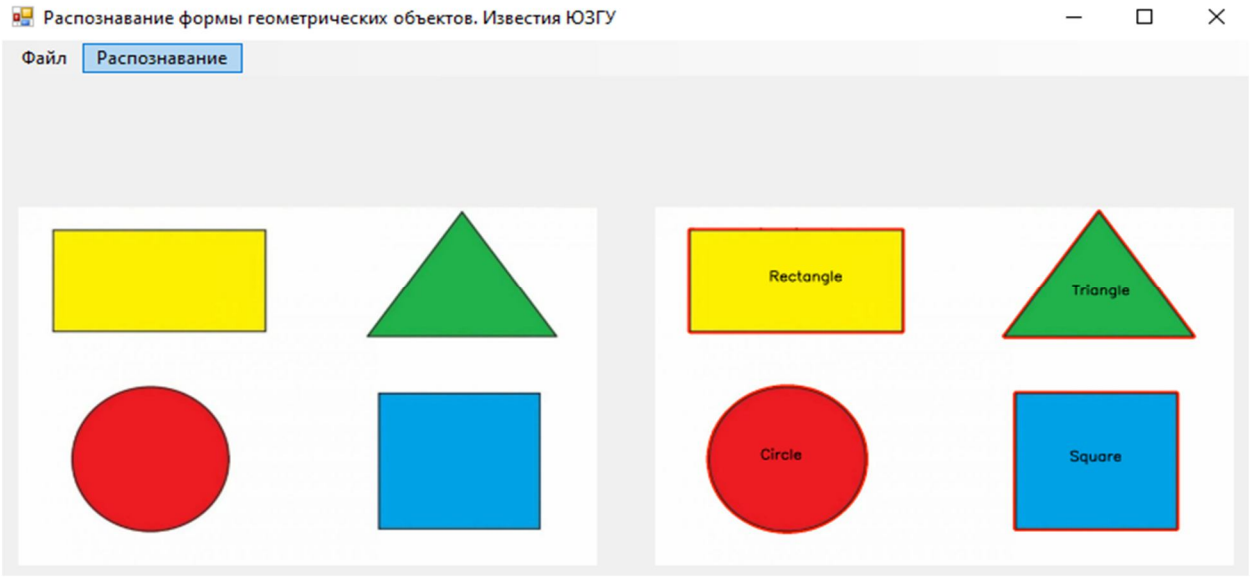
```
double crRatio = 4 * Math.PI *
moments.M00 /
   (CvInvoke.ArcLength(contours[i], true) *
   CvInvoke.ArcLength(contours[i], true));
if (Math.Abs(crRatio - 1.0) < 0.15)
     CvInvoke.PutText(input_image,
"Circle", new Point(x - 55, y),
    Emgu.CV.CvEnum.FontFace.HersheySimplex, 1,
        new MCvScalar(0, 0, 0), 2);
```

Представленные в этом разделе семь этапов позволяют реализовать программный код для детектирования формы геометрических фигур.

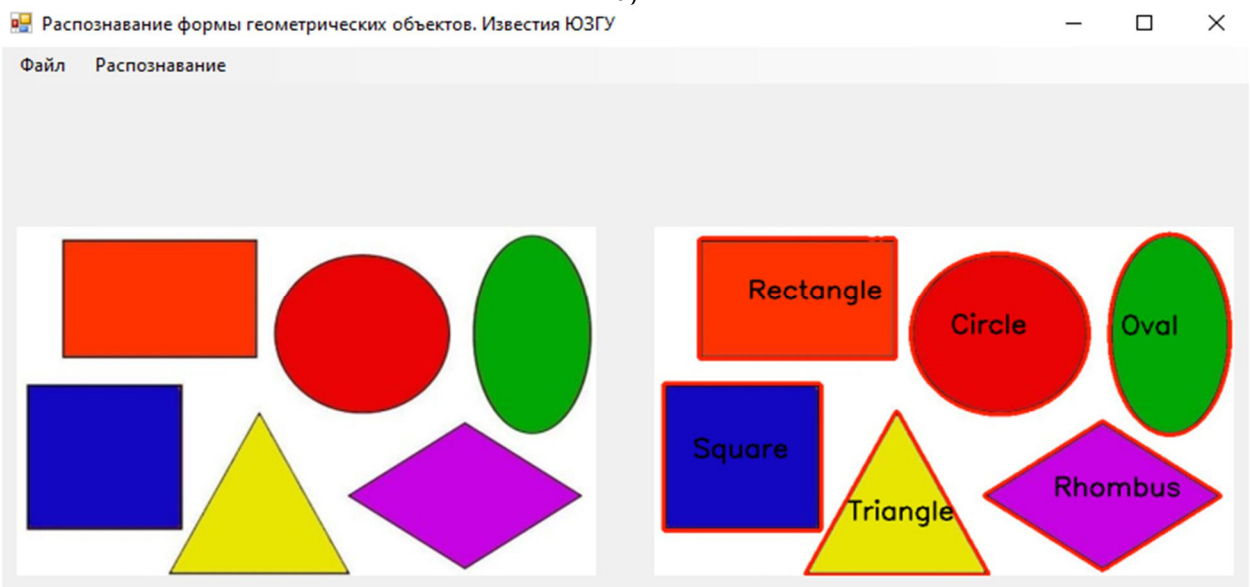
Результаты и их обсуждение

Алгоритм распознавание формы геометрических объектов был реализован в виде специализированного программного обеспечения, разработанного на языке C# в среде Microsoft Visual Studio 2022. При этом использовался ПК IntelCore i5-8600K (3,60 ГГц), ОЗУ 16 ГБ, Win10. Эксперименты повторялись 100 раз для каждого исходного изображения, по которому необходимо было распознать геометрическую форму объекта.

Визуализация процесса детектирования формы геометрических объектов представлена на рис. 3.



а)



б)

Рис. 3. Распознавание формы геометрических объектов: **а** – Эксперимент 1; **б –** Эксперимент 2

Fig. 3. Recognition of the shape of geometric objects: **а** – Experiment 1; **б –** Experiment 2

Для обеспечения надежности работоспособности разработанной программы распознавания геометрических фигур был выполнен тест для распознавания форм фигур, используя тестовый набор фигур, расположенных на ресурсе EMGU.COM[1] [25]. Результаты теста приведены на рис. 4.

Результаты вычислительного эксперимента распознавания формы геометрических фигур с использованием библиотеки Emgu CV сведены в табл. 1.

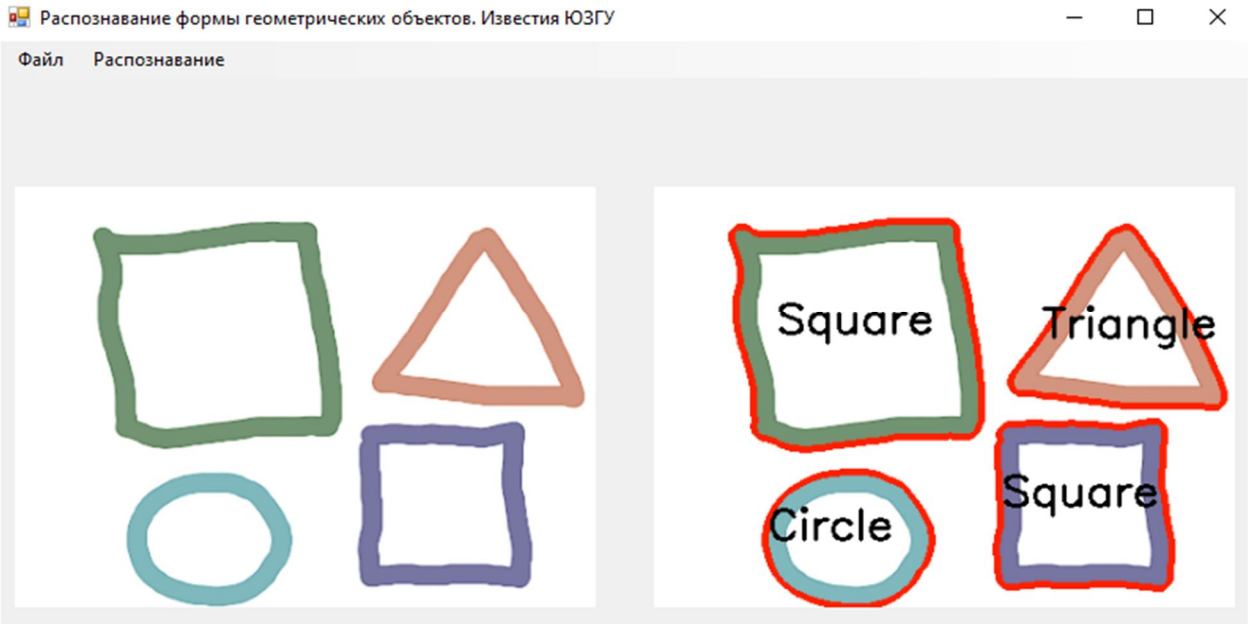


Рис. 4. Распознавание формы геометрических объектов для тестовой выборки ресурса emgu.com

Fig. 4. Shape recognition of geometric objects for a test sample of the emgu.com resource

Таблица 1. Вычислительный эксперимент распознавания геометрических фигур

Table 1. Computational experiment of recognition of geometric shapes

Фигура / Figure	Количество обнаруженных объектов / Number of detected objects		
	Эксперимент 1 / Experiment 1	Эксперимент 2 / Experiment 2	Эксперимент 3 / Experiment 3
Треугольник	1	1	1
Квадрат	1	1	2
Ромб		1	
Прямоугольник	1	1	
Круг	1	1	1
Овал		1	

[1] Адрес тестового ресурса emgu.com https://www.emgu.com/wiki/index.php/Shape_%28Triangle,_Rectangle,_Circle,_Line%29_Detection_in_CSharp (Доступ свободный).

Как видно из табл. 1, все фигуры для всех изображений были распознаны правильно.

Для оценки надежности алгоритма распознавания форм геометрических объектов использовалась следующая методика.

1. В процессе исследования использовались 100 различных изображений с разными геометрическими фигурами, и для каждого из них применялся предложенный во втором разделе статьи алгоритм.

2. Для каждой из фигур, сведенных в табл. 1, сравнивались результаты распознавания геометрических фигур при работе алгоритма с реальными объектами на изображениях.

3. Данные сводились в сводную таблицу, аналогичную табл. 1.

4. Затем анализировались результаты эксперимента.

4.1. Если алгоритм правильно обнаруживал объект, то считалось как True Positive (TP).

4.2. Если алгоритм ошибочно обнаруживал объект, то считалось как False Positive (FP).

4.3. Если алгоритм не обнаруживал объект, то считалось как False Negative (FN).

Данные анализа работы алгоритма сведены в табл. 2.

Таблица 2. Анализ тестирования 100 изображений

Table 2. Analysis of testing 100 images

Фигура / Figure	Количество обнаруженных объектов / Number of detected objects		
	Правильно обнаруженные объекты / Correctly detected objects	Ложно обнаруженные объекты / Falsely detected objects	Пропущенные объекты / Missed objects
Треугольник	95		
Квадрат	102	1	1
Ромб	43	2	2
Прямоугольник	73		
Круг	54	2	
Овал	32	3	1
	TP	FP	FN
Сумма	399	8	4

5. Надежность эксперимента вычислялась с помощью следующей формулы:

$$N = \frac{TP}{TP + FP + FN},$$

где TP – количество верно обнаруженных объектов; FP – количество ложно обнаруженных объектов; FN – количество пропущенных объектов.

Учитывая данные табл. 2, надежность определилась как

$$N = \frac{399}{399 + 8 + 4} = 0,97$$

и составила 97%, что говорит о высокой надежности рассмотренного алгоритма.

Выводы

В статье представлен алгоритм распознавания геометрической формы объектов и его программная реализация на основе библиотеки EmguCV на языке программирования C#. Вычислительный процесс детектирования геометрической формы объектов хорошо проиллюстрирован. Проведены экспериментальные исследования, в результате которых было установлено, что предложенная модель обладает высокой надежностью и составляет 97%. Улучшить надежность возможно за счет подбора пороговых значений при детектировании таких форм, как овал и ромб в условиях, задающих их определение.

Список литературы

1. Гнеушев А. Н., Мурынин А. Б. Адаптивный градиентный метод выделения контурных признаков объектов на изображениях реальных сцен // Известия Российской академии наук. Теория и системы управления. 2003. № 6. С. 153-160.

2. Алгоритм группировки геометрических объектов при автоматическом раскрое листового материала с использованием локальных характеристик формы / М. А. Чертов, Г. Е. Руденский, С. Г. Псахье, А. В. Скворцов // Вычислительные технологии. 2006. Т. 11, № 2. С. 93-107.

3. Чуканов С. Н. Сравнение изображений объектов методами вычислительной топологии // Труды СПИИРАН. 2019. Т. 18, № 5. С. 1043-1065. https://doi.org/10.15622/sp.2019.18.5.1043-1065.

4. Чуканов С. Н. Сравнение диффеоморфных изображений на основе формирования персистентных гомологий // Моделирование и анализ информационных систем. – 2019. Т. 26, № 3. С. 450-468. https://doi.org/10.18255/1818-1015-2019-3-450-468.

5. Шабалин А. С., Рахматуллов И. И., Полянин Н. А. Сравнение двух методов распознавания образов геометрических фигур // Ученые записки УлГУ. Серия: Математика и информационные технологии. 2021. № 2. С. 82-89.

6. Воронова Л. В., Панишева Е. В. К вопросу выбора метода определения границ и распознавания графического объекта применительно к задаче идентификации номера автомобиля // Технологии и качество. 2022. № 2(56). С. 46-50. https://doi.org/10.34216/2587-6147-2022-2-56-46-50.

7. Бобырь М.В., Емельянов С.Г., Милостная Н.А. *Алгоритм построения 3d сцен распознанных объектов по картам глубин* // Известия Юго-Западного государственного университета. 2023. Т. 27, №2. С. 90-104. https://doi.org/10.21869/2223-1560-2023-27-2-90-104.

8. A method for creating a depth map based on a three-level fuzzy model / M. Bobyr, A. Arkhipov, S. Emelyanov, N. Milostnaya // Engineering Applications of Artificial Intelligence. 2023. Vol.117, no. Part B. P. 105629. https://doi.org/10.1016/ j.engappai.2022.105629.

9. Построение карты глубины с использованием модернизированного фильтра Канни. Часть 1 / М. В. Бобырь, А. Е. Архипов, А. С. Якушев, Ц. Цао // Промышленные АСУ и контроллеры. 2021. № 4. С. 12-20. https://doi.org/10.25791/asu.4.2021.1271.

10. Построение карты глубины с использованием модернизированного фильтра Канни. Часть 2 / М. В. Бобырь, А. Е. Архипов, А. С. Якушев, С. Бхаттачарья // Промышленные АСУ и контроллеры. 2021. № 5. С. 3-15. https://doi.org/10.25791/ asu.5.2021.1277.

11. Годунов А. И., Баланян С. Т., Егоров П. С. Сегментация изображений и распознавание объектов на основе технологии сверточных нейронных сетей // Надежность и качество сложных систем. 2021. № 3(35). С. 62-73. https://doi.org/10.21685/2307-4205-2021-3-8.

12. Лобанов М. Г., Шоломов Д. Л. Об ускорении архитектуры сверточной нейронной сети на базе ResNet в задаче распознавания объектов дорожной сцены // Информационные технологии и вычислительные системы. 2019. № 3. С. 57-65. https://doi.org/ 10.14357/20718632190305.

13. Кадуков Е. П. Распознавание объектов контроля на радиолокационных изображениях с использованием метода опорных векторов // Вопросы оборонной техники. Серия 16: Технические средства противодействия терроризму. 2022. № 9-10(171-172). С. 96-105.

14. А. Ю. Поливанов, Ю. В. Иванов Распознавание человека в системе технического зрения мобильного робота на основе метода опорных векторов // Вестник МГТУ "Станкин". 2023. № 3(66). С. 17-27.

15. Пугин Е. В., Жизняков А. Л. Алгоритмы обработки изображений для обнаружения объектов с использованием нечётких признаков // Радиотехнические и телекоммуникационные системы. 2020. № 2(38). С. 59-65.

16. Нечетко-логические методы в задаче детектирования границ объектов / М. В. Бобырь, А. Е. Архипов, С. В. Горбачев, Ц. Цао, С. Бхаттачарья // Информатика и автоматизация. 2022. Т.21, №2. С. 376-404. https://doi.org/10.15622/ia.21.2.6

17. Прикладные нейро-нечеткие вычислительные системы и устройства / М. В. Бобырь, С. Г. Емельянов, А. Е. Архипов, Н. А. Милостная. М.: Издательский Дом "Инфра-М", 2023. 263 с. https://doi.org/ 10.12737/1900641.

18. Bobyr M. V., Milostnaya N. A., Bulatnikov V. A. The fuzzy filter based on the method of areas' ratio // Applied Soft Computing. 2022. Vol. 117. P. 108449. https://doi.org/ 10.1016/j.asoc.2022.108449.

19. Бобырь М. В., Милостная Н. А. Анализ использования мягких арифметических операций в структуре нечетко-логического вывода // Вестник компьютерных и информационных технологий. 2015. № 7(133). С. 7-15. https://doi.org/10.14489/vkit.2015.07.pp.007-015.

20. Бобырь М. В., Архипов А. Е., Якушев А. С. Распознавание оттенка цветовой метки на основе нечёткой кластеризации // Информатика и автоматизация. 2021. Т. 20, № 2. С. 407-434. https://doi.org/10.15622/ia.2021.20.2.6.

21. Emgu CV Library Documentation. VectorOfVectorOfPoint Class. URL: www.emgu.com/wiki/files/3.0.0/document/html/b29e1d11-e75c-6bbe-4b3b-7d8e6395a733.htm (Доступ свободный).

22. Emgu CV Library Documentation. CvInvoke.FindContours Method. URL: www.emgu.com/wiki/files/4.5.5/document/html/M_Emgu_CV_CvInvoke_FindContours.htm (Доступ свободный).

23. Балун В. Н., Дроздов Н. А., Дорошенко Е. В. Программная реализация алгоритма Дугласа-Пекера для уменьшения числа точек полилинии // Современные тенденции развития и перспективы внедрения инновационных технологий в машиностроении, образовании и экономике. 2019. Т. 5, № 1. С. 157-163.

References

1. Gneushev A. N., Murynin A. B. Adaptive gradient method for selecting contour features of objects in images of real scenes. *Izvestiya Rossiiskoi akademii nauk. Teoriya i sistemy upravleniya = Izvestia of the Russian Academy of Sciences. Theory and Control Systems*. 2003; (6): 153-160. (In Russ.).

2. Chertov M. A., Rudensky G. E., Psakhye S. G., Skvortsov A.V. An algorithm for grouping geometric objects in automatic cutting of sheet material using local shape characteristics. *Vychislitel'nye tekhnologii = Computational Technologies*. 2006; 11 (2): 93-107. (In Russ.).

3. Chukanov S. N. Comparison of images of objects by methods of computational topology. *Trudy SPIIRAN = Proceedings of SPIIRAN*. 2019; 18(5): 1043-1065. (In Russ.). https://doi.org/10.15622/ sp.2019.18.5.1043-1065.

4. Chukanov S. N. Comparison of diffeomorphic images based on the formation of persistent homologies. *Modelirovanie i analiz informatsionnykh sistem = Modeling and Analysis of Information Systems*. 2019; 26 (3): 450-468. (In Russ.). https://doi.org/10.18255/1818-1015-2019-3-450-468.

5. Shabalin A. S., Rakhmatullov I. I., Polyanin N. A. Comparison of two methods of pattern recognition of geometric shapes. *Uchenye zapiski UlGU. Seriya: Ma-tematika i informatsionnye tekhnologii = Scientific Notes of UlSU. Series: Mathematics and Information Technology.* 2021: (2): 82-89. (In Russ.).

6. Voronova L. V., Panisheva E. V. On the issue of choosing a method for determining boundaries and recognizing a graphic object in relation to the problem of identifying a car number. *Tekhnologii i kachestvo = Technology and Quality*. 2022; (2): 46-50. (In Russ.). https://doi.org/10.34216/2587-6147-2022-2-56-46-50.

7. Bobyr M. V., Emelyanov S. G., Milostnaya N. A. Algorithm for Creating 3d Scenes of Recognized Objects from Depth Maps. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2023; 27(2): 90-104 (In Russ.). https://doi.org/10.21869/2223-1560-2023-27-2-90-104.

8. Bobyr M., Arkhipov A., Emelyanov S., Milostnaya N. A method for creating a depth map based on a three-level fuzzy model. *Engineering Applications of Artificial Intelligence*. 2023; 117 (Part B): 105629. https://doi.org/10.1016/j.engappai.2022.105629.

9. Bobyr M. V., Arkhipov A. E., Yakushev A. S., Cao Ts. Building a depth map using an upgraded Canny filter. Pt. 1. *Promyshlennye ASU i kontrollery = Industrial automated control systems and controllers*. 2021; (4): 12-20. (In Russ.). https://doi.org/10.25791/asu.4.2021.1271.

10. Bobyr M. V., Arkhipov A. E., Yakushev A. S., Bhattacharya S. Building a depth map using an upgraded Canny filter. Pt. 2. *Promyshlennye ASU i kontrollery = Industrial automated control systems and controllers*, 2021; (5): 3-15. (In Russ.). https://doi.org/10.25791/asu.5.2021.1277.

11. Godunov A. I., Balanyan S. T., Egorov P. S. Image segmentation and object recognition based on convolutional neural network technology. *Nadezhnost' i kachestvo slozhnykh sistem = Reliability and quality of complex systems*. 2021; (3): 62-73. (In Russ.). https://doi.org/10.21685/2307-4205-2021-3-8.

12. Lobanov M. G., Sholomov D. L. On accelerating the architecture of a convolutional neural network based on ResNet in the task of recognizing objects of the road scene. *Informatsionnye tekhnologii i vychislitel'nye sistemy = Information Technologies and Computing Systems*. 2019; (3): 57-65. (In Russ.).https://doi.org/10.14357/ 20718632190305.

13. Kadukov E. P. Recognition of control objects on radar images using the method of reference vectors. *Voprosy oboronnoi tekhniki. Seriya 16: Tekhnicheskie sredstva protivodeistviya terrorizmu = Questions of defense technology. Series 16: Technical means of countering terrorism*. 2022; (9-10): 96-105. (In Russ.).

14. Polivanov A. Yu., Ivanov Yu. V. Human recognition in the technical vision system of a mobile robot based on the method of support vectors. *Vestnik MGTU "Stankin" = Bulletin of the Moscow State Technical University "Stankin"*. 2023; (3): 17-27. (In Russ.).

15. Pugin E. V., Zhiznyakov A. L. Image processing algorithms for detecting objects using fuzzy features. *Radiotekhnicheskie i telekommunikatsionnye sistemy = Radio Engineering and Telecommunication Systems*. 2020; (2): 59-65. (In Russ.).

16. Bobyr M. V., Arkhipov A. E., Gorbachev S. V., Cao Ts., Bhattacharya S. Fuzzy logic methods in the problem of detecting object boundaries. *Informatika i avtomatizatsiya = Computer Science and Automation*. 2022; (21): 376-404. (In Russ.). https://doi.org/10.15622/ia .21.2.6

17. Bobyr M. V., Yemelyanov S. G., Arkhipov A. E., Milostnaya N. A. Applied neuro-fuzzy computing systems and devices. Moscow: Infra-M; 2023. 263 p. (In Russ.). https://doi.org/10.12737/1900641.

18. Bobyr M. V., Milostnaya N. A., Bulatnikov V. A. The fuzzy filter based on the method of areas' ratio. *Applied Soft Computing*. 2022; 117: 108449. https://doi.org/ 10.1016/j.asoc.2022.108449.

19. Bobyr M. V., Milostnaya N. A. Analysis of the use of soft arithmetic operations in the structure of fuzzy logical inference. *Vestnik komp'yuternykh i informatsionnykh tekhnologii = Bulletin of Computer and Information Technologies*. 2015; (7): 7-15. (In Russ.). https://doi.org/10.14489/vkit.2015.07.pp.007-015.

20. Bobyr M. V., Arkhipov A. E., Yakushev A. S. [Recognition of the shade of a color label based on fuzzy clustering]. *Informatika i avtomatizatsiya = Informatics and Automation*, 2021; 20 (2): 407-434. (In Russ.). https://doi.org/10.15622/ia.2021.20.2.6.

21. Emgu CV Library Documentation. VectorOfVectorOfPoint Class. Available at: www.emgu.com/wiki/files/3.0.0/document/html/b29e1d11-e75c-6bbe-4b3b-7d8e6395a733.htm (Доступ свободный).

22. Emgu CV Library Documentation. CvInvoke.FindContours Method. Available at: www.emgu.com/wiki/files/4.5.5/document/html/M_Emgu_CV_CvInvoke_FindContours.htm (Доступ свободный).

23. Balun V. N., Drozdov N. A., Doroshenko E. V. Software implementation of the Douglas-Pecker algorithm for reducing the number of polyline points. *Sovremennye tendentsii razvitiya i perspektivy vnedreniya innovatsionnykh tekhnologii v mashinostroenii, obrazovanii i ekonomike = Modern development trends and prospects for the introduction of innovative technologies in mechanical engineering, education and economics*. 2019; 5 (1): 157-163. (In Russ.).

Информация об авторе / Information about the Author

Милостная Наталья Анатольевна,
доктор технических наук, доцент,
Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: nat_mil@mail.ru,
ORCID: https://orcid.org/0000-0002-3779-9165

Natalya A. Milostnaya, Dr. of Sci. (Engineering), Associate Professor, Southwest State University, Kursk, Russian Federation,
e-mail: nat_mil@mail.ru,
ORCID: https://orcid.org/0000-0002-3779-9165