УДК 004.855

ИСПОЛЬЗОВАНИЕ OPENCV В РАМКАХ ЗАДАЧИ ОФФЛАЙН РАСПОЗНАВАНИЯ РУКОПИСНОГО ТЕКСТА

Басанько А.С.[1], Рыбкин С.В.[1]

[1]Калужский филиал ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Калуга, e-mail: fn1-kf@mail.ru

Данная статья посвящена использованию библиотеки OpenCV в рамках решения задачи оффлайн распознавания рукописного текста. Данная задача обычно состоит из множества этапов. Этапы предварительной обработка изображения и сегментации являются одними из них. Библиотека OpenCV является достаточно мощным инструментом и вполне позволяет решить задачи этих двух этапов с хорошим результатом. OpenCV содержит реализации множества алгоритмов различных типов: распознавание объектов, устранение искажений, выявление сходства и формы объектов, отслеживание перемещения объекта, распознавание движений и жестов. Для предобработки изображений имеется множество методов, в том числе реализации фильтра Гаусса и медианного фильтра, метод преобразования изображения в градации серого, а также метод пороговой бинаризации. Также OpenCV имеет поддержку морфологических операций, посредством применения которых можно выделить слова на изображении. Таким образом, в данной статье будет подробно описано, каким образом можно объединить несколько методов, реализуемых в библиотеке OpenCV, для решения задач предобработки изображения и выделения на нём слов в рамках задачи оффлайн распознавания рукописного текста, а также каких результатов с этим можно достигнуть.

Ключевые слова: распознавание рукописного текста, OpenCV, выделение рукописных слов, обработка изображений

USING OF OPENCV WITHIN A TASK OF OFFLINE HANDWRITING RECOGNITION

Basanko A.S.[1], Rybkin S.V.[1]

[1]Калужский филиал ФГБОУ ВО «Московский государственный технический университет им. Н.Э. Баумана (национальный исследовательский университет)», Калуга, e-mail: fn1-kf@mail.ru

This article is devoted to the use of OpenCV library in solving the problem of offline handwriting recognition. This task usually consists of many stages. The image preprocessing and segmentation stages are among them. OpenCV library is quite a powerful tool and allows you to solve the problems of these two stages with a good result. OpenCV contains implementation of many algorithms of different types: object recognition, deskewing, identification of similarities and shape of the objects, tracking the moving object, motion detection and gestures. There are many methods available for preprocessing images, including the implementation of the Gauss filter and the median filter, the grayscale image conversion method, and the threshold binarization method. OpenCV also has support for morphological operations, through the use of which you can select words in the image. Thus, this article will describe in detail how to combine several methods implemented in the OpenCV library to solve the problems of image preprocessing and selection of words on it in task of offline handwriting recognition, as well as what results can be achieved with this.

Keywords: handwriting recognition, OpenCV, handwritten words highlighting, image processing

Введение. Задача оффлайн распознавания рукописного текста, как правило, состоит из нескольких этапов: предварительная обработка, сегментация, извлечение признаков, классификация и обработка результатов.

В рамках данной статьи сосредоточимся на первых двух этапах. На этапе предварительной обработки происходит обработка изображения с целью повышения его качества и приведения его к виду, удобному для сегментации. Этап сегментации подразумевает выделение текста на изображении и его разделение на составные части. Обычно текст обрабатывается иерархически: сначала выделяются отдельные строки, затем отдельные слова, затем символы или части символов.

Для реализации двух первых этапов оффлайн распознавания рукописного текста можно воспользоваться таким инструментом, как OpenCV.

OpenCV представляет собой библиотеку компьютерного зрения с открытым исходным кодом, которая разработана компанией Intel на языке программирования C/C++. Также существуют версии этой библиотеки и для других языков, таких как Python, Java, C#, MatLab и многих других. В данной библиотеке реализовано множество алгоритмов следующих типов [1, 2]:

- Распознавание объектов;
- Распознавание печатного текста;
- Устранение искажений;
- Выявление сходства и формы объектов;
- Отслеживание перемещение объекта;
- Распознавание движений, жестов.

Предварительная обработка изображения. Этап предварительной обработки изображения обычно состоит из следующих задач [3, 4]:

- Преобразование изображения в градации серого;
- Удаление дефектов.

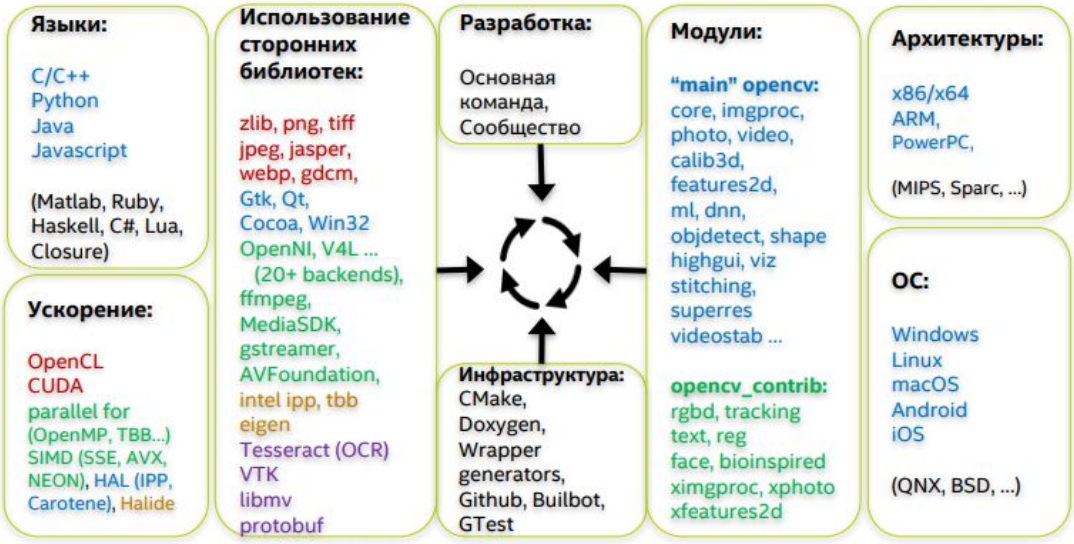

Рис. 1. Карта проекта OpenCV.

Изображение в OpenCV представляется с помощью класса Image [5]. При создании объекта этого класса необходимо указать путь к изображению. Для решения задачи преобразования изображения в градации серого достаточно вызвать метод Convert и указать конечный цвет изображения, в данном случае цвет Gray. Результат обработки можно увидеть на Рис.2.

Пример исходного кода, написанного на языке C#, с помощью библиотеки Emgu.CV, которая является оберткой под .NET платформу над оригинальной библиотекой OpenCV:

```
var img = new Image<Bgr, byte>(filePath);
var grayImage = img.Convert<Gray, byte>();
```

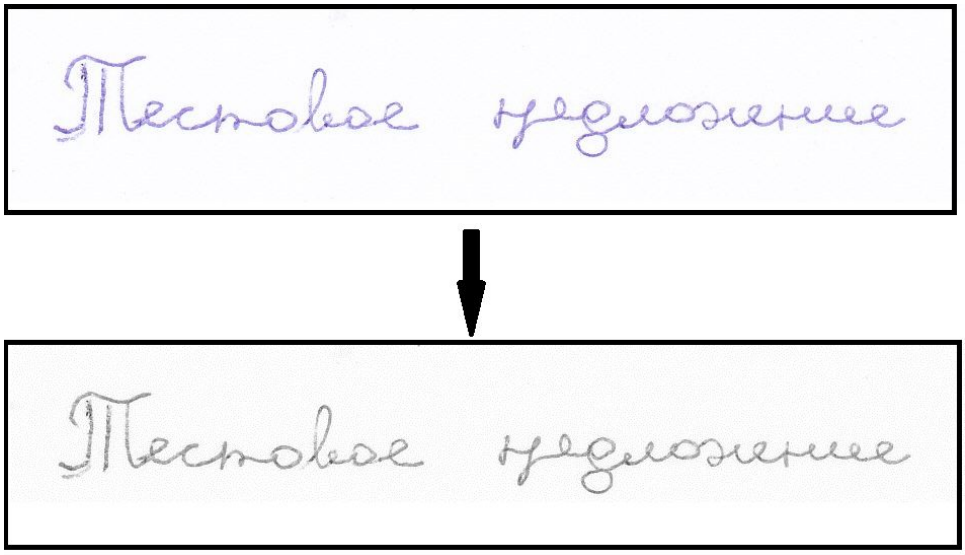

Рис. 2. Преобразование изображения в градации серого.

Удаление дефектов подразумевает собой применение различных фильтров для удаления шумов. Наиболее часто используемыми фильтрами являются фильтр Гаусса для подавления высокочастотного шума и медианный фильтр для удаления шума «соль и перец». Реализации этих фильтров также есть в OpenCV.

Для использования фильтра Гаусса достаточно воспользоваться функцией GaussianBlur поместив в качестве параметров входное и выходное изображения, размер ядра Гаусса, величины отклонения ядра Гаусса по осям X и Y, а также метод экстраполяции пикселей [6]. Под ядром Гаусса понимается матрица свертки, которая помогает для каждого пикселя изображения рассчитать средневзвешенное значение соседних пикселей. В результате чего все зашумленные пиксели (яркость которых сильно отличается от яркости соседних пикселей) примут усредненное значение. Ширина и высота ядра могут отличаться, но они оба должны быть положительными и нечетными. Также они могут быть равны 0, что будет означать, что размер ядра будет рассчитываться исходя из величины отклонения ядра Гаусса.

В свою очередь для использования медианного фильтра необходимо воспользоваться функцией MedianBlur поместив в качестве параметров входное и выходное изображения, размер окна (апертуры), которое должно быть нечётным. Результат применения медианного фильтра можно увидеть на Рис.3.

Пример исходного кода:

```
    CvInvoke.MedianBlur(grayImage, grayImage, 5);
    CvInvoke.GaussianBlur(grayImage, grayImage, new Size(5, 5),
1.5, 0, BorderType.Replicate);
```

Выделение слов на изображении. Как уже было сказано выше, следующим этапом в оффлайн распознавании рукописного текста является этап сегментации. Основная задача этого этапа – выделение текста на изображении. Если пренебречь тем, что рукописный текст может быть написан под различным наклоном, то с помощью OpenCV можно сразу выделять строки и слова, находящиеся в каждой из строк.

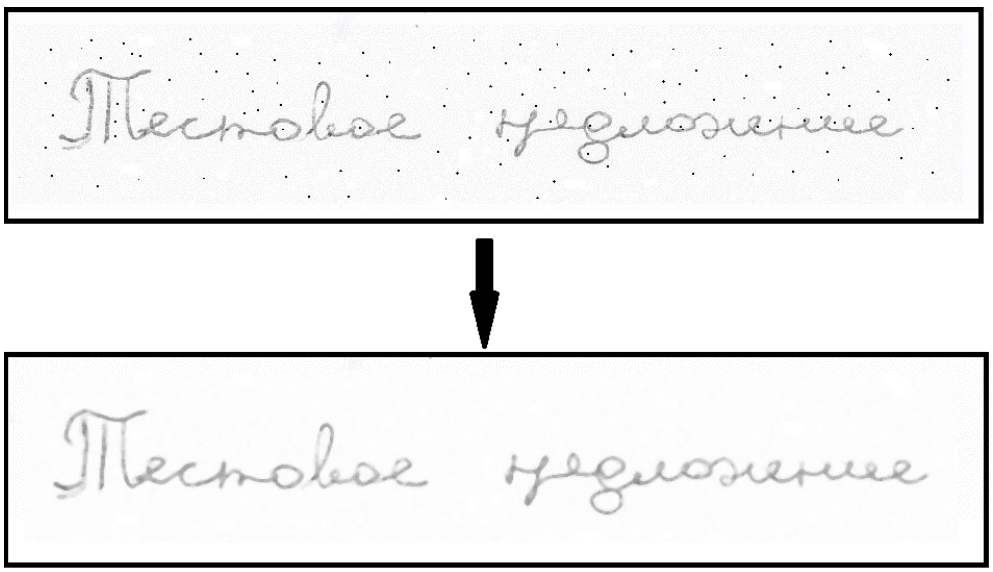

Рис. 3. Результат применения медианного фильтра.

Процесс выделения слов при помощи OpenCV можно разделить на следующие этапы:

- Пороговая бинаризация изображения;
- Применение операции морфологического замыкания;
- Выделение границ слов.

Метод пороговой бинаризации осуществляется при помощи функции Threshold, которая принимает на вход изображение в градациях серого, выходное изображение, величину порога и алгоритм вычисления порога. Наиболее известным алгоритмом вычисления порога является метод Оцу [7]. В результате бинаризации получаем изображения белого текста на черном фоне (Рис. 4).

Пример исходного кода:

```
    var thresholdImage = new Image<Gray, byte>(grayImage.Size);
    CvInvoke.Threshold(grayImage, thresholdImage, 0, 255,
ThresholdType.Otsu);
```

Для того чтобы на последнем этапе выделить слова с изображения, необходимо воспользоваться морфологической операцией замыкания [8]. Для этого служит функция

MorphologyEx, принимающая на вход изображение, полученное на предыдущем этапе, выходное изображение, тип морфологической операции (в данном случае операция замыкания) и структурный элемент, с которым будет осуществляться операция. В качестве структурного элемента обычно используют прямоугольник. Результат применения морфологической операции замыкания можно увидеть на Рис. 5.

Пример исходного кода:

```
var element = CvInvoke.GetStructuringElement(
ElementShape.Rectangle, new Size(5, 5), new Point(0, 0));
CvInvoke.MorphologyEx(thresholdImage, thresholdImage,
MorphOp.Close, element, new Point(-1, -1), 12, BorderType.Default,
new MCvScalar());
```

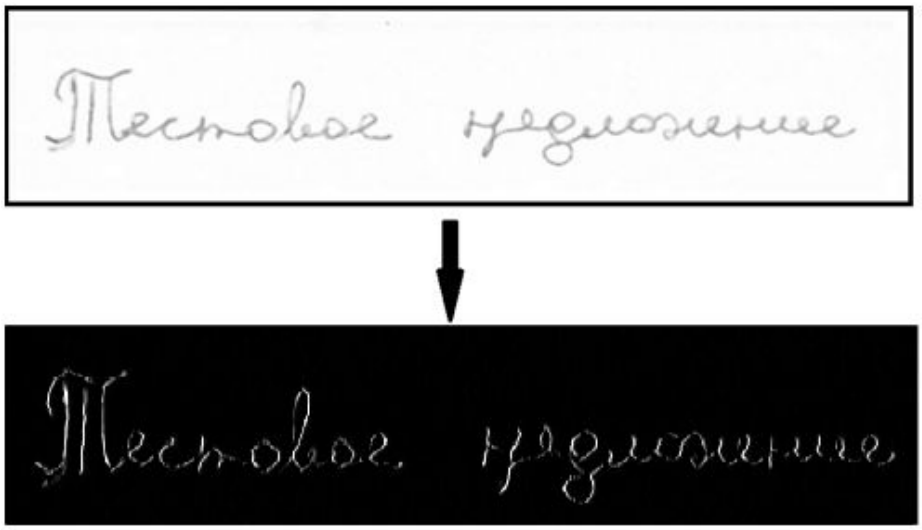

Рис. 4. Результат пороговой бинаризации.

Наконец, на последнем этапе выделяются слова на изображении. Для этого используются функции FindContours, которая находит все контуры слов на бинарном изображении и функция BoundingRectangle, которая возвращает ограничивающий прямоугольник для каждого слова на основе контура. В результате, с помощью этих прямоугольников мы можем выделить слова на конечном изображении (Рис. 6).

Пример исходного кода:

```
var contours = new VectorOfVectorOfPoint();
var rects = new List<Rectangle>();
CvInvoke.FindContours(thresholdImage, contours, null,
RetrType.External, ChainApproxMethod.ChainApproxSimple);
for (var i = 0; i < contours.Size; i++)
{
      var approxContour = new VectorOfPoint();
```

```
            CvInvoke.ApproxPolyDP(contours[i], approxContour, 0.005,
true);
            if (CvInvoke.ContourArea(approxContour) > 100)
            {
                var boundingRectangle = CvInvoke.BoundingRectangle
(approxContour);
                rects.Add(boundingRectangle);
            }
    }
```

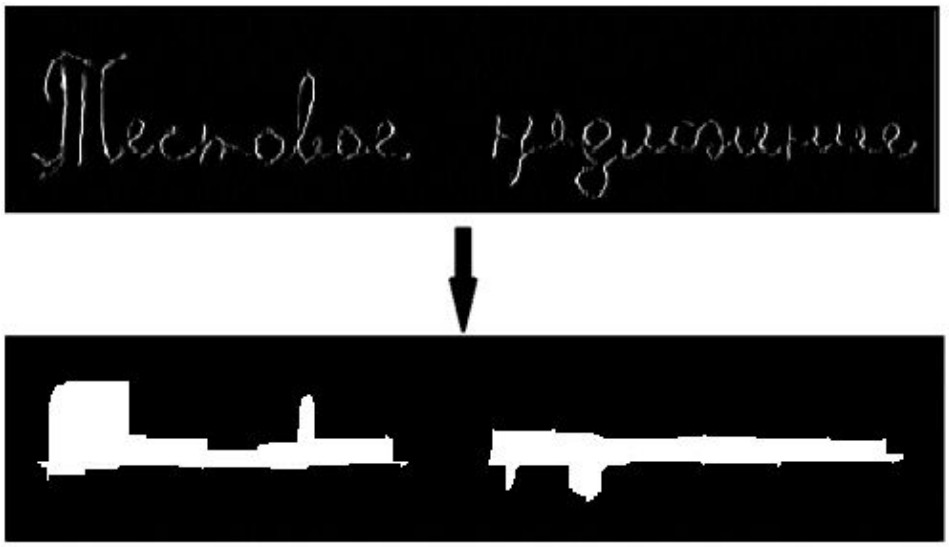

Рис. 5. Результат морфологической операции замыкания.

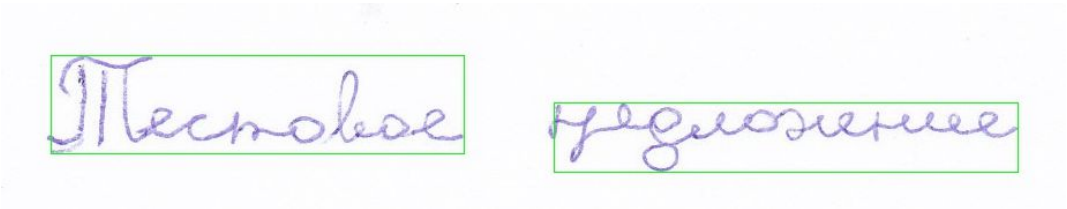

Рис. 6. Выделенные слова на изображении.

Заключение. Таким образом, в данной статье было рассмотрено использование открытой библиотеки OpenCV для предобработки изображения и выделения слов на изображении в рамках задачи оффлайн распознавания рукописного текста. Данная библиотека справляется с этой задачей достаточно качественно и быстро.

Список использованной литературы:

1. Разин И.Б., Миронов В.П., Муртазина А.Р., Оганезов К.В. Применение библиотеки OpenCV в системах автоматизированного проектирования // Дизайн, технологии и инновации в текстильной и легкой промышленности. Сборник материалов международной

научно-технической конференции. Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Московский государственный университет дизайна и технологии». 2015. С. 244-246. [Электронный ресурс] URL: https://elibrary.ru/download/elibrary_24392430_45225961.pdf (дата обращения: 21.11.2018)

2. Медведев А.А., Пакулич Д.В. Изучение возможностей библотеки OpenCV для обработки изображений // Математика. Информатика. Компетентностный подход к обучению в вузе и школе. Сборник материалов всероссийской научно-практической конференции. 2015. С. 103-104. [Электронный ресурс] URL: https://elibrary.ru/download/elibrary_29267849_47839333.pdf (дата обращения: 14.11.2018)

3. Басанько А.С., Белов Ю.С. Основные методы обработки изображений при оффлайн распознавании рукописного текста // Научное обозрение. Технические науки – 2018. - №3 - с. 5-8. [Электронный ресурс] URL: https://science-engineering.ru/ru/article/view?id=1184 (дата обращения: 11.11.2018)

4. Басанько А.С., Рыбкин С.В. Существующие проблемы распознавания рукописного текста // В сборнике: Научные исследования в области технических и технологических систем. Сборник статей Международной научно-практической конференции. 2018. С. 25-28.

5. Yue Yaru, Zhu Jialin Algorithm of fingerprint extraction and implementation based on OpenCV // 2nd International Conference on Image, Vision and Computing (ICIVC). 2017. P.163-167. [Электронный ресурс] URL: https://ieeexplore.ieee.org/document/7984539 (дата обращения: 05.11.2018)

6. Нанавова Т.А. Алгоритм извлечения текста из видео с использованием библиотеки компьютерного зрения OpenCV // Ростовский научный журнал. 2016. № 7. С. 21-40. [Электронный ресурс] URL: https://elibrary.ru/download/elibrary_26376289_32767465.pdf (дата обращения: 11.11.2018)

7. Ширабоков И.А., Щелканов А.В., Чумаченко И.М. Использование пороговых фильтров в OpenCV // Россия молодая: передовые технологии – в промышленность!. 2015. № 2. С. 86-89. [Электронный ресурс] URL: https://elibrary.ru/download/elibrary_25004826_76179940.pdf (дата обращения: 25.10.2018)

8. Гришанов К.М., Белов Ю.С. Морфологические операции для уменьшения шума на изображении // Электронный журнал: наука, техника и образование. 2016. № 2 (6). С. 90-95. [Электронный ресурс] URL: https://elibrary.ru/download/elibrary_26847525_60281350.pdf (дата обращения: 11.11.2018)