Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

УДК 004.932.72 https://doi.org/10.33619/2414-2948/49/29

# РАСПОЗНАВАНИЕ РУКОПИСНОГО И ПЕЧАТНОГО ТЕКСТА ПРИ РАЗРАБОТКЕ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

***©Казнин А. А.,*** *ORCID: 0000-0002-8833-2522, SPIN-код: 8861-3315, канд. техн. наук, Северный (Арктический) федеральный университет им. М.В. Ломоносова, г. Архангельск, Россия, a.kaznin@narfu.ru*

# RECOGNITION HANDWRITING AND PRINTED TEXT FOR SOFTWARE REQUIREMENTS ENGINEERING

***©Kaznin A.,*** *ORCID: 0000-0002-8833-2522, SPIN-code: 8861-3315, Northern (Arctic) Federal University named after M.V. Lomonosov, Arkhangelsk, Russia, a.kaznin@narfu.ru*

*Аннотация.* В данной статье рассмотрены проблемы сбора требований к программному обеспечению. Проанализированы существующие технологии компьютерного зрения и обоснован выбор технологии для распознавания рукописного и печатного текста. Описаны входные данные для экспериментов и представлены результаты распознавания символов для каждой категории изображений. Разработан метод предварительной обработки изображения и распознавания символов текста на мобильных устройствах с применением параллельных вычислений. На основе предложенного метода разработан прототип мобильного приложения для сбора и цифровизации данных получаемых при разработке требований к программному обеспечению.

*Abstract.* This article discusses the problems of collecting software requirements. The existing computer vision technologies are analyzed and the choice of technology for recognizing handwritten and printed text is justified. The input data for the experiments are described and the results of character recognition for each image category are presented. A method of image preprocessing and recognition of text characters on mobile devices using parallel computing has been developed. On the basis of the proposed method, a prototype mobile application for collecting and digitalizing data obtained during the requirements engineering has been developed.

*Ключевые слова:* распознавание текста, предварительная обработка изображений, мобильные устройства.

*Keywords:* text recognition, image preprocessing, mobile devices.

## *Введение*

Разработка требований к программному облесению – один из этапов разработки программного обеспечения. От того, на сколько хорошо проработан данный этап, зависит вся разработка программного продукта и его жизненный цикл в целом. Существует несколько стандартов, методологий к проектированию, в том числе разработке требований (инженерии требований) к программному обеспечению [1]. Например, международный стандарт ISO/IEC/IEEE 29148-2018 [2] содержит описание о процессах и продуктах, связанных с разработкой требований к системам, программным продуктам и сервисам на протяжении всего жизненного цикла. Он определяет структуру требований, атрибуты и характеристики требований, а также описывает итеративное и рекурсивное применение процессов требований в течение всего жизненного цикла. Широко применяется гибкая методология

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

разработки Agile, основанная на ряде подходов, предполагающих непрерывное сотрудничество между заинтересованными лицами и выпуск версий минимально жизнеспособного продукта, и принципов, описанных в Agile–манифесте [3]. Однако, прежде чем приступить к формализации требований, необходимо собрать информацию о планируемом функционале программном обеспечении, составить глоссарий проекта. Существует несколько методов выявления требований: семинар, интервью с заказчиком, фокус–группы, наблюдение, опросные листы, анализ документов [1], где чаще всего применяется бумага, презентации. Обычно эту информацию разработчики программного обеспечения фотографируют, а потом пытаются вручную переводить в редактируемый вид или же используют стационарный сканер и компьютер для цифровизации данных, что влияет на время проектирования программного обеспечения. Эта проблема до сих пор полностью не решена.

С широким распространением мобильных устройств с камерами, технологий придачи данных на высокой скорости появляется возможность автоматизировать процесс сбора данных. Идея разработки мобильного приложения была представлена на научном семинаре IEEE 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems в 2018 году [4] и опубликована в сборнике трудов [5]. Суть идеи состоит в том, чтобы фотографировать все исходные данные (записи на бумаге, на маркерной доске, раздаточный материал и т. п.) и преобразовывать их в редактируемые данные для дальнейшей работы с ними (составление глоссария, построение моделей проекта и т. д.).

*Материал и методы исследования*

Для преобразования данных с фотографий в редактируемый вид необходимо использовать технологии распознавания (компьютерного зрения). Сегодня существует ряд таких технологий: Google Cloud Vision API [6], Google Mobile Vision API [7], Microsoft vision [8], ABBYY OCR [9], OpenCV [10]. Поскольку мобильное приложение должно работать на платформах iOS и Android в качестве среды разработки использовалась Microsoft Visual Studio [11] с платформой для разработки мобильных приложений Xamarin [12] (позволяет разрабатывать нативные приложения Android, iOS, tvOS, watchOS, macOS, Windows приложения с помощью .NET и языка программирования c#.), была выбрана технология той же организации Microsoft Vision с целью минимизировать возможные проблемы применения технологии распознавания при разработке. Технология Microsoft Vision позволяет распознавать печатный и рукописный текст. Для этого необходимо преобразовать изображение в массив байтов, после этого создается HTTP-запрос или ComputerVisionClient, в который помещаются исходные данные и указываются переменные окружения – сервер и ключ. Переменные окружения предоставляются по подписке Microsoft Azure, для проекта использовалась бесплатная версия подписки. После распознавания данные возвращаются в формате JSON [13] (в случае HTTP-запроса), формат ответа с распознанным текстом представлен на Рисунке 1.

Благодаря тому, что все распознанные данные имеют координаты четырехугольника, их описывающего, возможно (при необходимости) их поместить в то место, где они были изначально на изображении.

В качестве входных данных для экспериментов использовались изображения с количеством точек на дюйм 72 по горизонтали и вертикали. Все изображения — печатный текст и рукописный текст. Фотографии с печатным текстом были сделаны с листа бумаги и экрана монитора, фотографии с рукописным текстом — с листа бумаги. В исходной базе

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

данных изображений для проведения экспериментов все фотографии были разделены на 8 категории (Таблица 1).

Массив строк:

Координаты четырехугольника, содержащего первую строку текста

Текст первой строки

Массив слов первой строки:

Координаты четырехугольника, содержащего первое слово в первой строке

Первое слово первой строки

Координаты четырехугольника, содержащего второе слово в первой строке

Второе слово первой строки

...

Координаты четырехугольника, содержащего n-слово в первой строке

n-слово первой строки

...

Координаты четырехугольника, содержащего m-строку текста

Текст m-строки

Массив слов m-строки:

Координаты четырехугольника, содержащего n-слово m-строки

n-слово m-строки

Рисунок 1. Формат возвращаемых данных.

Таблица 1.

КАТЕГОРИИ ИЗОБРАЖЕНИЙ

| *Номер категории* | *Источник* | *Вид текста* | *Неравномерность освещения* | *Шумы* | *Блики* | *Иные графические элементы* |
|---|---|---|---|---|---|---|
| 1 | Лист бумаги | Печатный | Да | Нет | Нет | Нет |
| 2 | Лист бумаги | Печатный | Да | Да | Нет | Нет |
| 3 | Экран монитора | Печатный | Нет | Да | Нет | Нет |
| 4 | Экран монитора | Печатный | Да | Да | Да | Нет |
| 5 | Лист бумаги | Рукописный | Нет | Нет | Нет | Листок «в линейку», графические фигуры |
| 6 | Лист бумаги | Рукописный | Нет | Нет | Нет | Листок «в клеточку», графические фигуры |
| 7 | Лист бумаги | Рукописный | Да | Да | Нет | Нет |
| 8 | Лист бумаги | Рукописный | Нет | Нет | Нет | Графические фигуры |

На Рисунках 2–3 представлены примеры каждой из категорий, описанных в Таблице 1.

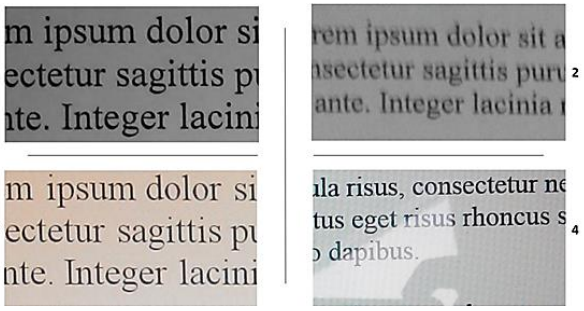


Рисунок 2. Фрагменты печатного текста (номер изображения соответствует номеру категории в Таблице 1).

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

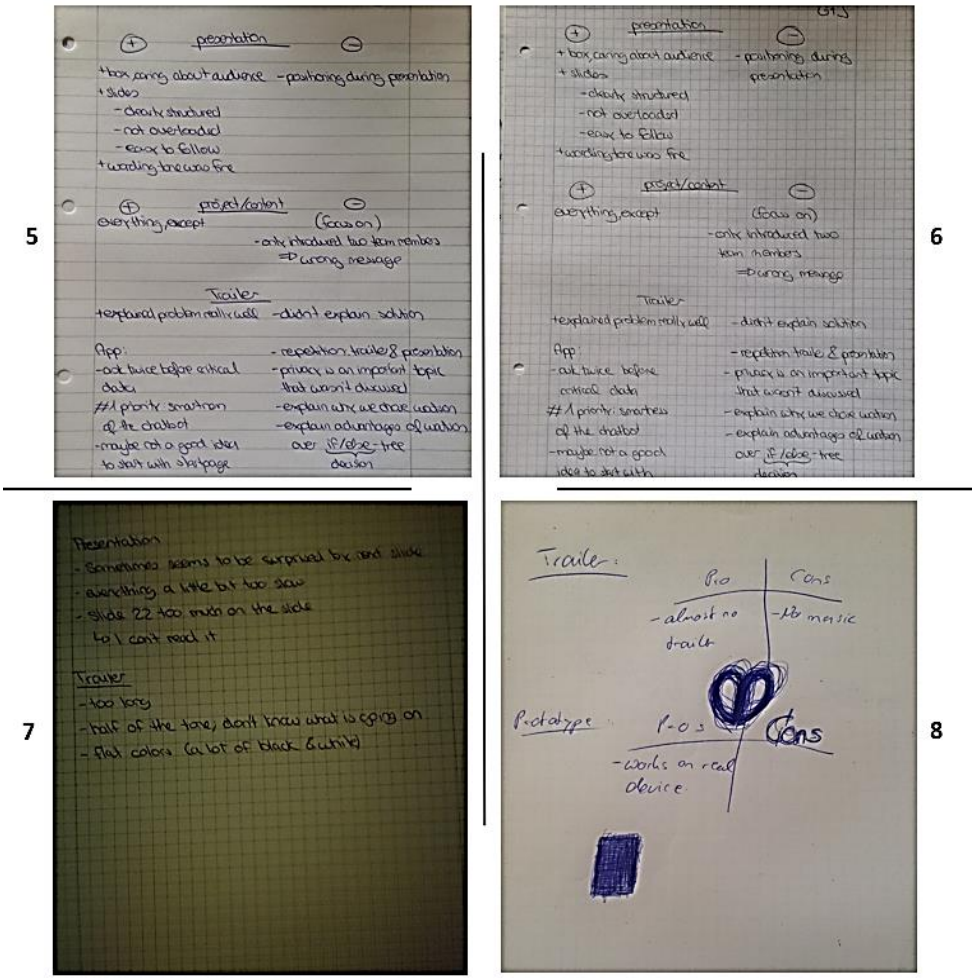


Рисунок 3. Фрагменты рукописного текста (номер изображения соответствует номеру категории в Таблице 1).

*Результаты и обсуждение*

После проведения нескольких экспериментов по распознаванию текста было отмечено, что без ошибок распознается текст на изображениях с хорошим качеством. Ошибки в распознавании текста связаны с шумами, размытыми границами символов и бликами. Также ошибки распознавания рукописного текста связаны особенностями рукописного написания символов и присутствием графических изображений (лист «в клеточку», «в линейку» и других изображений). В реальных условиях беседы с потенциальным заказчиком чаще используются рукописные записи, особенно записи с графическим сопровождением — символы многократно обведены, исправления в тексте, рисунки. Применение в таком виде технологии компьютерного зрения становится необоснованным с точки зрения дальнейших временных затрат — корректировки текста поле распознавания.

Одним из наиболее подходящих решений проблемы плохого распознавания символов могла бы стать предварительная обработка изображений. Для предварительной обработки применены алгоритм преобразование изображения в оттенки серого [14] и алгоритм ранговой обработки сигнала [15], данный алгоритм позволил улучшить результаты распознавания текста с помощью технологии Google Mobile Vision API [16].

Алгоритм ранговой обработки сигнала «проходит» каждый пиксель изображения окном размером $h \times h$ пикселей с центром в анализируемой точке (пикселе), $h$ — всегда нечетное число (чем больше его значение, тем больше область учитываемых значений цвета соседних пикселей, влияющих на итоговое значение цвета анализируемого пикселя). Из

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

значений цветов пикселей, которые находятся в скользящем окне, формируется мужество значений всех цветов пикселей — вариационный ряд. Цвет центрального пикселя заменяется $p(k)$ или $p(h^2-k+1)$ элементом. Выбор элемента осуществляется по признаку наиболее близкого значения цвета пикселя к значению цвета центрального пикселя:

$$T(I)'_{x,y} = \begin{cases} p(k), & |p(k)-T(I)_{x,y}| < |p(h^2-k+1)-T(I)_{x,y}|; \\ p(h^2-k+1), & \text{в противном случае.} \end{cases} \quad (1)$$

где $T(I)_{x,y}$ — исходное значение цвета пикселя,
$T(I)'_{x,y}$ — конечное значение цвета пикселя,
$p(k), k = (1..h^2)$ — множество значений всех цветов пикселей в скользящем окне.

Перед применением алгоритма ранговой обработки сигнала цветное изображение необходимо преобразовать в оттенки серого цвета (градации серого). Для этого могут использоваться преобразования с разными коэффициентами для составляющих цвета: стандартное преобразование (2) и преобразование, учитывающее особенности восприятия изображения в градациях серого человеческим глазом (3).

$$C'_{x,y} = 0.299 * R_{x,y} + 0.587 * G_{x,y} + 0.114 * B_{x,y} \quad (2)$$

где $C'_{x,y}$ — конечное значение цвета пикселя в градациях серого,
$R_{x,y}$ — составляющая красного цвета исходного цвета пикселя,
$G_{x,y}$ — составляющая зеленого цвета исходного цвета пикселя,
$B_{x,y}$ — составляющая синего цвета исходного цвета пикселя,
$x, y$ — координаты пикселя.

$$C'_{x,y} = 0.2126 * R_{x,y} + 0.7152 * G_{x,y} + 0.0722 * B_{x,y} \quad (3)$$

Алгоритм ранговой обработки сигнала имеет входные параметры и алгоритм преобразования изображения в оттенки серого имеет разные коэффициенты для каждого цвета. Для поиска наиболее подходящего сочетания (наилучший результат распознания) были проведены эксперименты для 8 категории изображений с разными входными параметрами и коэффициентами, результаты представлены в Таблице 2.

Для некоторых категории изображений применение алгоритма ранговой обработки сигнала (центральный пиксель выбирается после построения вариационного рядя) ухудшило результаты распознавания. Это объясняется тем, что выбор цвета центрального пикселя происходит после построения вариационного ряда, т. е. фактически происходит сравнение не с текущим значением цвета пикселя, а со значением цвета пикселя, находящегося в середине вариационного ряда. Если вычисляется цвет пикселя в размытой области изображения или на границе символа и фона, или на изображении, где толщина линий символа измеряется 1–4 пикселями, то вероятность того, что текущий цвет пикселя окажется дальше по значению от цвета центрального пикселя, после построения вариационного ряда велика. Это приводит к потере информации на изображении, цвету пикселя неправильно назначается значение нового цвета. Чтобы избежать потери информации при обработке изображения с помощью алгоритма ранговой обработки сигнала необходимо проводить выбор цвета центрального пикселя до построения вариационного ряда.

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

Таблица 2.

РЕЗУЛЬТАТЫ РАСПОЗНАВАНИЯ

| *Номер категории* | Алгоритм ранговой обработки сигнала (центральный пиксель выбирается после построения вариационного ряда) | Алгоритм ранговой обработки сигнала (центральный пиксель выбирается до построения вариационного ряда) | Преобразование изображения в оттенки серого (стандартное преобразование), | Преобразование изображения в оттенки серого (учитывающее особенности восприятия изображения человеческим глазом) | Средний процент нераспознанных символов, % |
|---|---|---|---|---|---|
| 1 | — | — | — | — | 0 |
| | + | — | — | + | 0 |
| | — | + | — | + | 0 |
| | + | — | + | — | 0 |
| | — | + | + | — | 0 |
| 2 | — | — | — | — | 1,2 |
| | + | — | — | + | 1,3 |
| | — | + | — | + | 0,7 |
| | + | — | + | — | 1,4 |
| | — | + | + | — | 0,8 |
| 3 | — | — | — | — | 0 |
| | + | — | — | + | 0 |
| | — | + | — | + | 0 |
| | + | — | + | — | 0 |
| | — | + | + | — | 0 |
| 4 | — | — | — | — | 2,2 |
| | + | — | — | + | 2,2 |
| | — | + | — | + | 1,8 |
| | + | — | + | — | 2,2 |
| | — | + | + | — | 1,9 |
| 5 | — | — | — | — | 2,5 |
| | + | — | — | + | 2,5 |
| | — | + | — | + | 0,4 |
| | + | — | + | — | 1,9 |
| | — | + | + | — | 1,7 |
| 6 | — | — | — | — | 3,3 |
| | + | — | — | + | 6,2 |
| | — | + | — | + | 2,5 |
| | + | — | + | — | 7,7 |
| | — | + | + | — | 2,5 |
| 7 | — | — | — | — | 3,7 |
| | + | — | — | + | 4,5 |
| | — | + | — | + | 3,7 |
| | + | - | + | — | 4,8 |
| | — | + | + | — | 3,7 |
| 8 | — | — | — | — | 20 |
| | + | — | — | + | 14 |
| | — | + | — | + | 9,8 |
| | + | — | + | — | 14,1 |
| | — | + | + | — | 9,9 |

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

Исходя из полученных результатов, можно сделать вывод, что наиболее подходящий вариант из множества проведенных экспериментов — предварительная обработка с использованием преобразования изображения в оттенки серого (3) с последующим применением алгоритма ранговой обработки сигнала (выбор цвета центрального пикселя до построения вариационного ряда). На Рисунке 4 показаны результаты предварительной обработки изображения.

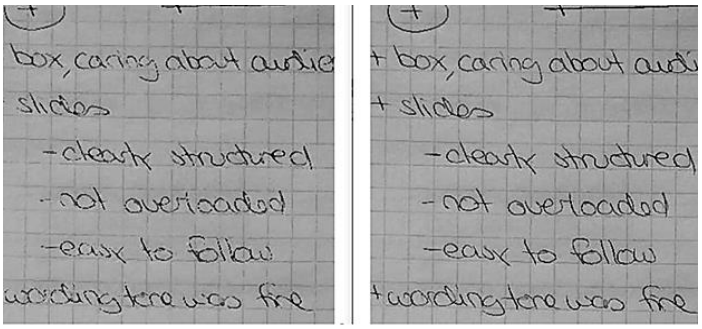

Рисунок 4. Сравнение исходного изображения (слева) и обработанного (справа) изображения.

Метод предварительной обработки и распознания текста требует перевода изображения в массив байтов. Для обращения к отдельным пикселям битовых карт (Bitmap) изображения существуют методы GetPixel и SetPixel [17]. Метод GetPixel позволяет получить цвет пикселя по заданным координатам, SetPixel — задать новое значение цвета пикселя по заданным координатам. Однако, данные методы целесообразно применять только если необходимо работать с небольшим количеством пикселей, так как эти методы медленно работают. Для быстрого получения массива байтов рекомендуется использовать метод Marshal.Copy [18], который копирует указанную область данных из памяти в массив байтов. Применение метода требует заблокировать биты в памяти на время работы с ними, это требует использования пространства имен System.Drawing.Bitmap, которое не поддерживается в iOS и Android. Для работы с битовой картой был использован пакет API SkiaSharp [19], который включает инструменты для работы с 2D-изображениями.

Современные мобильные устройства позволяют получать фотографии с большим разрешением [20], например, смартфон HUAWEI P30 Pro имеет основную камеру 40 мегапикселей [21], смартфон Xiaomi Redmi Note 7 [22] имеет основную камеру 48 мегапикселей. Смартфон с меньшими характеристиками камеры, например, смартфон ZTE V9 Blade [23] имеет основную камеру 16 мегапикселей и озволяет делать фотографии с разрешением 4608×3456. Это означает, что для такого изображения будет выполняться минимум 16 миллионов операций.

Сложность алгоритма ранговой обработки сигнала будет вычисляться следующим образом:

$$O(n^2*W*H), \quad (4)$$

где n — размер «скользящего окна» вокруг обрабатываемого пикселя, $W$ — ширина изображения, $H$ — высота изображения.

Если рассматривать изображение объемом16 мегапикселей, то только для каждого пикселя надо совершить количество проходов (циклов) $H*W = 15925248$, кроме того, для каждого прохода необходимо построить вариационный ряд из «скользящего окна» размером $n \times n$.

Сложность алгоритма преобразования изображения в градации серого:

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

$$O(W*H). \quad (5)$$

Сложности алгоритмов 3 и 4 могли быть представлены в стандартном виде — со степенями, вышеизложенное представление сделано для наглядности.

Даже если не использовать медленные методы для работы с пикселями изображения GetPixel и SetPixel и предварительно уменьшить размер изображения, процесс выполнения задачи предварительной обработки изображения достаточно ресурсоемкий. Большинство современных мобильных устройств имеет 4, 6, 8-ядерный процессор [24]. В этом случае решать задачу предварительной обработки изображения можно с распараллеливанием вычислений, это позволит более эффективно использовать ресурсы мобильного устройства и уменьшить время на предварительную обработку.

Если рассматривать битовую карту изображения как одномерный массив байтов (один элемент массива соответствует цвету в градациях серого пикселя из битовой карты), а высчитываемое значение цвета пикселя зависит от исходных значений соседних пикселей и не зависит от полученных значений после вычислений, то можно вычислить новое значение цвета пикселей отдельно, параллельно. В Таблице 3 приведены результаты тестирования мобильного приложения с использованием 1, 4 потоков параллельной обработки информации. Данные результаты были получены на мобильном устройстве ZTE V9 Blade, которое имеет процессор Qualcomm Snapdragon 450.

Таблица 3.

ВРЕМЯ ОБРАБОТКИ ИЗОБРАЖЕНИЯ

| *Исходный размер изображения* | *Время предварительной обработки изображения без уменьшения размера исходных данных и выполнением в 1 поток* | *Время предварительной обработки изображения* | |
|---|---|---|---|
| | | *1 поток (без параллельной обработки)* | *4 потока* |
| 16 мегапикселей (4608×3456), 9,98 мегабайт | 71,93 с. | 19,71 с. | 6,78 с. |

В Таблице 3 не представлены результаты распараллеливания задачи на 8 потоков, так как результат получился хуже, чем с применением 4 потоков. Это связано с тем, что системные процессы выполняется на некоторых ядрах процессора и не все процессоры имеют одинаковые характеристики производительности для каждого ядра.

Предложенный метод позволил повысить эффективность — увеличить количество корректно распознанных символов, снизить время загрузки изображения на сервер и снизить интернет-трафик при использовании технологии Технология Microsoft Computer Vision API.

На основе предложенного метода разработан прототип мобильного приложения, который протестирован на телефоне ZTE V9 Blade (Рисунок 5).

Приложение позволяет выбрать исходное изображение или PDF файл, распознавать изображение в редактируемый текст или в формат JSON, редактировать распознанные данные, сохранять или делиться результатами.

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

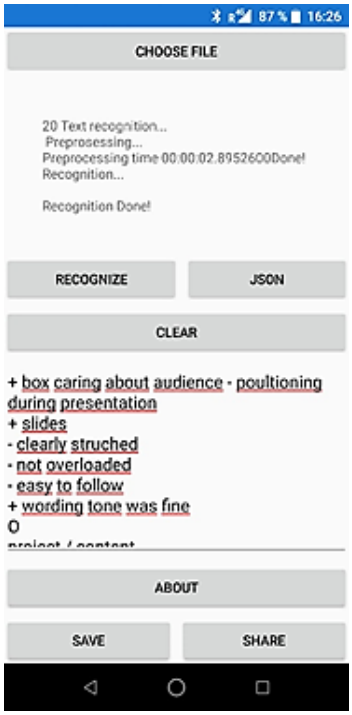


Рисунок 5. Прототип мобильного приложения

*Заключение*

В результате проведенного исследования предложен метод распознавания образов (символов текста) на мобильных устройствах путем применения технологии компьютерного зрения с предварительной обработкой данных, включающей определение входных параметров алгоритмов, последовательности выполнения алгоритмов и их модернизации. Также была проведена оптимизация времени выполнения предложенного метода с помощью параллельных вычислений. На основе предложенного метода разработан прототип мобильного приложения для сбора и цифровизации данных получаемых при разработке требований к программному обеспечению.

*Автор выражает признательность исследователю в Университете Гамбурга Clara Marie Lüders за предоставленные исходные данные для исследования — изображения с рукописным текстом.*

*Автор благодарит за предоставленную возможность проведения исследования совместную программу Михаил Ломоносов Министерства науки и высшего образования Российской Федерации (проект 2.13435.2019/13.2) и Германской службы академических обменов DAAD.*

*Список литературы:*

1. Вигерс К., Битти Д. Разработка требований к программному обеспечению. СПб.: БХВ-Петербург, 2019. 736 с.
2. IEEE 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Requirements engineering. https://clck.ru/L49gk
3. Manifesto for Agile Software Development. https://agilemanifesto.org
4. RESACS. https://resacs2018.wordpress.com
5. Mannov N., Lüders C. M., Kaznin A. ReqVision: Digitising Your Analog Notes into Readable and Editable Data // 2018 4th International Workshop on Requirements Engineering for

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS). IEEE, 2018. P. 20-23. https://doi.org/10.1109/RESACS.2018.00009

6. Vision AI. https://cloud.google.com/vision
7. Mobile Vision. https://clck.ru/L49hK
8. Computer Vision - Microsoft Azure. https://clck.ru/L49hh
9. Abbyy. https://www.abbyy.com/
10. OpenCV. https://opencv.org/
11. Microsoft Visual Studio. https://visualstudio.microsoft.com/ru/
12. Xamarin. https://clck.ru/L49hw
13. JSON. https://www.json.org/
14. Рекомендация «Значения параметров стандартов ТВЧ для производства программ и международного обмена программами» от 06.2015 №МСЭ-R BT.709-6
15. Маслов А. М., Сергеев В. В. Идентификация линейной искажающей системы с использованием ранговой обработки сигналов // Компьютерная оптика. 1990. №6. С. 97-102.
16. Kaznin A. A., Sushko O. P., Babkin A. V. Developing the algorithm allowing business-dedicated mobile applications to read texts // 2017 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS). IEEE, 2017. P. 207-214. https://doi.org/10.1109/ITMQIS.2017.8085798
17. Bitmap.GetPixel. https://clck.ru/L49jR
18. Marshal. Copy Method. https://clck.ru/L49jo
19. SkiaSharp. https://clck.ru/L49ju
20. Пушкарев Г. Смартфоны с лучшей камерой 2019 // Комсомольская правда. 2019.
21. HUAWEI P30 Pro. https://clck.ru/L49k4
22. Redmi Note 7. https://clck.ru/L49kC
23. ZTE Blade V9. https://clck.ru/L49kN
24. Процессоры для смартфонов и планшетов // Мобильные компьютеры. https://clck.ru/L49kc

*References:*

1. Vigers, K., & Bitti, D. (2019). Razrabotka trebovanii k programmnomu obespecheniyu. St. Petersburg. (in Russian).
2. IEEE 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes - Requirements engineering. https://clck.ru/L49gk
3. Manifesto for Agile Software Development. https://agilemanifesto.org
4. RESACS. https://resacs2018.wordpress.com
5. Mannov, N., Lüders, C. M., & Kaznin, A. (2018). ReqVision: Digitising Your Analog Notes into Readable and Editable Data. *In: 2018 4th International Workshop on Requirements Engineering for Self-Adaptive, Collaborative, and Cyber Physical Systems (RESACS), IEEE,* 20-23. https://doi.org/10.1109/RESACS.2018.00009
6. Vision AI. https://cloud.google.com/vision
7. Mobile Vision. https://clck.ru/L49hK
8. Computer Vision - Microsoft Azure. https://clck.ru/L49hh
9. Abbyy. https://www.abbyy.com/
10. OpenCV. https://opencv.org/
11. Microsoft Visual Studio. https://visualstudio.microsoft.com/ru/
12. Xamarin. https://clck.ru/L49hw
13. JSON. https://www.json.org/

Т. 5. №12. 2019
DOI: 10.33619/2414-2948/49

14. Rekomendatsiya Znacheniya parametrov standartov TVCh dlya proizvodstva programm i mezhdunarodnogo obmena programmami, 06.2015, no. МСЭ-R BT.709-6.

15. Maslov, A. M., & Sergeev, V. V. (1989). Identifikatsiya lineinoi iskazhayushchei sistemy s ispol'zovaniem rangovoi obrabotki signalov. *Komp'yuternaya optika,* (06), 97-102.

16. Kaznin, A. A., Sushko, O. P., & Babkin, A. V. (2017). Developing the algorithm allowing business-dedicated mobile applications to read texts. *In: 2017 International Conference Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS),* 207-214. IEEE. https://doi.org/10.1109/ITMQIS.2017.8085798

17. Bitmap.GetPixel. https://clck.ru/L49jR

18. Marshal. Copy Method. https://clck.ru/L49jo

19. SkiaSharp. https://clck.ru/L49ju

20. Pushkarev, G. (2019). Smartfony s luchshei kameroi 2019. Komsomol'skaya pravda. (in Russian).

21. HUAWEI P30 Pro. https://clck.ru/L49k4

22. Redmi Note 7. https://clck.ru/L49kC

23. ZTE Blade V9. https://clck.ru/L49kN

24. Protsessory dlya smartfonov i planshetov. Mobil'nye komp'yutery. https://clck.ru/L49kc

*Работа поступила*
*в редакцию 07.11.2019 г.*

*Принята к публикации*
*12.11.2019 г.*

---

*Ссылка для цитирования:*

Казнин А. А. Распознавание рукописного и печатного текста при разработке требований к программному обеспечению // Бюллетень науки и практики. 2019. Т. 5. №12. С. 246-256. https://doi.org/10.33619/2414-2948/49/29

*Cite as (APA):*

Kaznin, A. (2019). Recognition Handwriting and Printed Text for Software Requirements Engineering. *Bulletin of Science and Practice, 5*(12), 246-256. https://doi.org/10.33619/2414-2948/49/29 (in Russian).