11. SMART [Электронный ресурс] / Portale del Dipartimento di Matematica e Informatica. — 2017. / Режим доступа: https://www.dmi.unict.it/ ~faro/smart/. — Дата доступа: 28.03.2017.

12. Bit parallel length invariant matcher [Электронный ресурс] / Portale del Dipartimento di Matematica e Informatica. — 2017. / Режим доступа: https://www.dmi.unict.it/~faro/smart/algorithms.php?algorithm=BLIM. — Дата доступа: 28.03.2017.

13. Bitap algorithm [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Bitap_algorithm. — Дата доступа: 28.03.2017.

УДК 004.62

Студ. А.Н. Зайцев
Науч. рук.: ст. преп. Е.А. Блинова
(кафедра информационных систем и технологий, БГТУ)

АЛГОРИТМЫ ПРЕПРОЦЕССИНГА ИЗОБРАЖЕНИЙ ДЛЯ УЛУЧШЕНИЯ КАЧЕСТВА ОПТИЧЕСКОГО РАСПОЗНАВАНИЯ СИМВОЛОВ

Оптическое распознавание символов (англ. optical character recognition, OCR) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующихся для представления символов в компьютере (например, в текстовом редакторе)[1]. На данный момент лидером в области оптического распознавания является компания Abbyy. Они за долгие годы разработали крайне быстрый и эффективный механизм распознавания, недостатками которого являются только его цена и закрытость исходного кода.

В ходе работы были рассмотрены различные методы улучшения входного изображения для улучшения качества распознавания и их влияние на конечный результат. Определение качества распознавания — процент совпадения распознанного текста с текстом, который на самом деле находится на исходном изображении.

Рассмотрим методы улучшения качества распознавания.

Бинаризация. Для распознавания символов нам не нужна информация о цвете изображения, поэтому перед началом распознавания следует бинаризовать изображение, то есть привести его к чёрно-белому виду. Самым подходящим для большинства

изображений и эффективным по соотношению качество/скорость работы на данный момент является метод бинаризации Оцу[5].

Конвертация в серые тона. Для многих алгоритмов обработки изображений (например, детектор рёбер Кэнни[6]) информации из бинаризованного изображения мало, а информация о цвете лишняя. Тогда стоит применить конвертацию изображения в тона серого (обычно выбирают 8 битные изображения). Тут, также как и с алгоритмами бинаризации существует масса методов, но в основном применяется просто формула, где каждый канал входного изображения умножается на коэффициент [15].

Выравнивание. Крайне важно распознавать изображение, которое расположено ровно, иначе программа для распознавания не сможет корректно определить линии текста и результаты распознавания сильно ухудшаются. Для автоматического определения линий хорошо показывают себя алгоритм Хафа [8] и вероятностный алгоритм Хафа[9] для детектирования прямых. Также есть хорошо зарекомендовавший метод определения угла страницы по корешку книжного разворота: с помощью алгоритма Хафа можно определить местоположение корешка на фотографии, после этого определить угол поворота и повернуть страницу в противоположную сторону.

Устранение трапециевидных искажений. Если сделать фото страницы не под прямым углом к ней, то в результате у вас получится не прямоугольник страницы, а трапеция. В данном случае искажения будут такие, что, например, верхняя часть страницы будет отдалена и будет намного менее разборчива. Это также влияет на качество распознавания.

Устранение искажений «книжный разворот». Если фотографировать книжный разворот, то по центру книги, возле разворота, будут искажения. Также книжный разворот может давать неблагоприятную тень, которая крайне отрицательно сказывается на алгоритмах бинаризации (даже на алгоритме локальной бинаризации Оцу).

Устранение шумов и размытия. Чем больше на изображении шумов, нечётких краёв символов, тем меньше шансов на корректное распознавание символов. Следовательно, общее качество распознавания ухудшится. Для устранения шумов можно применять различные фильтры, такие как: билатеральный фильтр[12], фильтр Гаусса[11], медианный фильтр[10].

Гамма-коррекция, коррекция цвета, баланс белого. Изображение на входе может иметь цветовые искажения, что негативно скажется на

качестве бинаризации, а, следовательно, и на качестве распознавания. Существует алгоритмы автокоррекции, в том числе и реализованные в библиотеках с открытым исходным кодом (например, в библиотеке Ximgproc[13]).

Обрезание изображений. Также важной составляющей хорошего распознавания является то, какой процент входного изображения содержит текст. Например, если на вашей фотографии есть страница с текстом, но большая часть фотографии занята фоном, то это негативно скажется на качестве распознавания. Автоматизировать процесс обрезки очень хорошо получается с помощью алгоритма детектирования рёбер Canny. Также хорошо себя показывает алгоритм Хафа для детектирования линий.

Сегментация изображения. Входное изображение может быть сложным и содержать очень много лишней информации: изображения, текст со сложным форматированием (например, порядок колонок текста в журналах). Если мы сможем правильно сегментировать изображение, вырезать все области с текстом, сформировать их корректный порядок, то это значительно улучшит качество распознавания.

Удаление лишних деталей. Например, на вход у нас может подаваться таблица с данными. Линии таблицы могут пересекаться с буквами, что в свою очередь не позволит правильно распознать букву. В данном случае следует удалять такие линии с исходного изображения, но при этом сохранять информацию о форматировании для последующего форматирования выходного текста.

Обеспечение приемлемого DPI[7] (Dots Per Inch, точек на дюйм). При меньшем DPI качество распознавания падает. Чтобы обеспечить необходимый минимум DPI надо при необходимости уменьшать размер изображения. Например, Tesseract[4] рекомендует использовать 300 DPI.

Использование готовых наборов алгоритмов препроцессинга для различных видов изображений. Для каждого типа документов можно подобрать набор алгоритмов с нужными параметрами, которые значительно улучшают качество распознавания. Возможно, в данном случае здесь могут помочь нейронные сети, которые отлично справляются при должных мощностях с проблемами категоризации. Это очень действенный, но в то же самое время очень трудоёмкий метод. Получить такие наборы алгоритмов и константы крайне сложно. Данные, которые накапливают фирмы в годе долгих лет исследований и тестирования, являются коммерческой тайной.

Влияние методов на качество распознавания. Так как программы для распознавания символов работают с бинаризованными изображениями, то очевидно, что применение бинаризации обязательно, при этом в разных случаях следует применять разные алгоритмы бинаризации, например, для старых текстов очень хорошо себя показывает Sauvola фильтр [14]. Следующим по необходимости идет выравнивание: если текст имеет некоторый наклон, то результаты распознавания будут очень плохие. Обрезание позволяет удалить значительную часть мусора, который может быть ложно распознан как текст. Удаление шумов, цветокоррекция, устранение различных искажений — в целом улучшают качество распознавания, но в меньшей мере, чем предыдущие методы. Наилучшего качества распознавания можно добиться, только применяя к изображению все вышеперечисленные методики.

ЛИТЕРАТУРА

1. Optical character recognition [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Optical_character_recognition. — Дата доступа: 25.03.2017.

2. Tesseract (software) [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Tesseract_(software). — Дата доступа: 25.03.2017.

3. Image analysis [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Image_analysis. — Дата доступа: 25.03.2017.

4. Tesseract. Improve Quality [Электронный ресурс] / GitHub. — 2017. / Режим доступа: https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality. — Дата доступа: 25.03.2017.

5. Otsu binarization [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Otsu%27s_method. — Дата доступа: 25.03.2017.

6. Canny edge detector [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Canny_edge_detector. — Дата доступа: 25.03.2017.

7. Dots per inch [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Dots_per_inch. — Дата доступа: 25.03.2017.

8. Hough transform [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Hough_transform. — Дата доступа: 25.03.2017.

9. Probabilistic Hough transform [Электронный ресурс] / The university of Edinburgh. — 2017. / Режим доступа: http://homepages.inf.ed.ac.uk/rbf /CVonline/LOCAL_COPIES/AV1011/macdonald.pdf. — Дата доступа: 25.03.2017.

10. Median filter [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Median_filter. — Дата доступа: 25.03.2017.

11. Gaussian filter [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Gaussian_filter. — Дата доступа: 25.03.2017.

12. Билатеральный фильтр [Электронный ресурс] / Studopedia. — 2017. / Режим доступа: http://studopedia.info/3-111442.html. — Дата доступа: 25.03.2017.

13. Ximgproc [Электронный ресурс] / OpenCV. — 2017. / Режим доступа: http://docs.opencv.org/3.0-beta/modules/ximgproc/doc/ximgproc.html. — Дата доступа: 25.03.2017.

14. Sauvola binarization [Электронный ресурс] / Mathworks. — 2017. / Режим доступа: https://www.mathworks.com/matlabcentral/fileexchange/ 40266-sauvola-local-image-thresholding. — Дата доступа: 25.03.2017.

15. Grayscale [Электронный ресурс] / Wikipedia. — 2017. / Режим доступа: https://en.wikipedia.org/wiki/Grayscale. — Дата доступа: 25.03.2017.

УДК 004.451.9

студ. А. Ф. Булова, Е. И. Акшевская
Науч. рук. доц. Н. Н. Буснюк
(кафедра информационные системы и технологии, БГТУ)

BULL MILKMAN ИЛИ КАК СОЗДАТЬ ХОРОШИЙ ПРОТОТИП ИГРЫ ЗА 48 ЧАСОВ?

Главный вопрос: “Почему так быстро?!”. Весь прототип игры собирался в режиме хакатона, который проходил в Минске в марте 2017 г. Основной его задачей было собрать рабочую версию продукта за 48 часов. Так как большинство игр для мобильных платформ используют достаточно простую механику – за основу мы взяли Unity как игровой движок, и добавили качественной векторной графики.