Система распознавания символов на изображениях со сложным фоном

Анна Балахонцева, Александр Годоба, Нгуен Тьен
Институт компьютерных систем
Одесский национальный политехнический университет, Одесса, Украина
a.c.balackhontseva@gmail.com

Аннотация

Предложена система сегментации и распознавания символов на изображениях со сложным фоном. Алгоритм сегментации использует метод кратчайшего пути по яркости пикселей для расчета разрезов между символами. Распознавание символов основывается на алгоритме масштабного пространства кривизны (curvature scale space - CSS). Предлагается использовать вейвлеты Гаусса для расчета кривизны кривых в алгоритме CSS для упрощения разрабатываемой системы.

Ключевые слова: *сегментация, распознавание, сложный фон, вейвлет Гаусса.*

1. ВВЕДЕНИЕ

Современные технологии распознавания текста работают вполне отлично для печатных документов с высоким разрешением. Распознавание же наложенного текста на изображениях со сложным фоном (когда на изображении кроме текста присутствуют и другие объекты – деревья, дома, люди и т.д.) все ещё является трудно решаемой задачей. Существует специализированные приложения OCR, которые доступны в качестве коммерческого продукта для распознавания отсканированных документов с высоким разрешением, системы распознавания номерных знаков и рукописных символов [6]. Но эти приложения, как правило, не доступны для большинства изображений окружающей среды. Это связано со следующими трудностями, возникающими при распознавании текста на изображении:

- низкое разрешение изображения,
- сложный и структурированный фон, такой, что достоверное различие между символами и фоном не возможно,
- различные текстовые шрифты и размеры.

2. ПОСТАНОВКА ЗАДАЧИ

Целью работы является разработка системы распознавания текста на изображениях со сложным фоном, которая использует в качестве алгоритма сегментации метод кратчайшего пути, основываясь на яркости пикселей, а в качестве алгоритма распознавания символов – алгоритм CSS, который разглаживает контур буквы с помощью Гауссовой функции ядра и отслеживает её точки перегиба [1-5]. Для более точной классификации будет представлен расширенный алгоритм CSS для генерации особых точек для вогнутых и выпуклых сегментов контура. Для расчета кривизны кривых будет использоваться вейвлет Гаусса. На вход для обработки подается текстовая область, а на выходе будет генерироваться распознанный текст.

3. СЕГМЕНТАЦИЯ СИМВОЛОВ

Шаг сегментации очень важен в данном подходе, т.к. методики классификации, основанные на контурном анализе, терпят неудачу без надёжной сегментации. Алгоритм сегментации будет основываться на модернизированном алгоритме нахождения кратчайшего пути [2] и состоять из следующих этапов:

1 Этап: Перед тем как начать разрез текста на отдельные символы, необходимо задаться начальными точками разрезов. Очевидно, что вертикальная гистограмма текстовой области (рис. 1а) покажет яркие пики в местах, где символ имеет чёткую вертикальную линию, менее яркие пики в местах, где символ имеет небольшие глифовые перемычки (например, горизонтальная перемычка буквы «н»), и низкие значения гистограммы в местах разделения символов (рис. 1б). Определив вогнутые участки гистограммы, методом поиска экстремумов, можно найти области, достаточно похожие на места разделения символов.

Рис 1: Сегментация слова на символы.

2 Этап: Определившись с начальными точками разреза, можно было бы опустить прямые из этих точек и получить сегментированные символы. К сожалению, из-за излишней линейности данного подхода и зашумленности исходного изображения, разрез не всегда может пройти между буквами (рис. 2а). Для этого ограничим путь движения разреза тремя направлениями: на один пиксель вниз, на один пиксель вниз и влево и на один пиксель вниз и вправо. В итоге алгоритм должен принять решение, в каком из трёх направлений двигаться, решение принимается на основе яркости пикселей допустимых путей: выигрывает пиксель с наименьшим значением яркости (рис. 2б).

3 Этап: Для того чтобы уменьшить количество бесполезных движений вправо или влево, связанное с шумами на изображении, пиксели смещающие движение разреза по горизонтали умножаются на коэффициент $\zeta > 1$ с целью наказания движения в сторону. Хороший результат показал коэффициент $\zeta = 1.4$. Все веса точек пути разреза, включая

умножение на ζ, аккумулируются в переменной S, названной стоимостью пути, для дальнейшего использования.

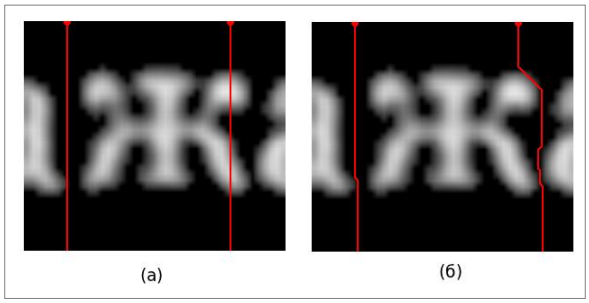

Рис 2: Пути разреза символов.

4 Этап: Несмотря на то, что описанный выше подход позволяет разрезать изображение на наиболее подходящие области, полученные разрезы всё ещё могут проходить через центры символов (рис. 1в) один или более раз. Очевидно, что стоимость пути разреза, прошедшего через начертание глифа, будет существенно большей стоимости разреза, прошедшего между символами. Воспользовавшись методом определения выбросов из математической статистики, приняв множество стоимостей разрезов как выборку, а ложные разрезы как выбросы, можно достаточно точно отфильтровать ложные разрезы. Хороший результат показал фильтр, основанный на межквартильном расстоянии [4] усечённом сверху.

Отфильтрованные таким образом разрезы имеют наиболее удовлетворительный результат (рис. 1г). Основным преимуществом этого подхода является тот факт, что не требуется порога для нахождения разделений между символами.

4. КЛАССИФИКАЦИЯ СИМВОЛОВ

Далее требуется выполнить анализ контура буквы и получить признаки для классификации для каждой из выделенных букв.

Для обучения мы сохраняем инверсное и обычное CSS изображение обучаемой выборки, которое затем сравниваем с получаемым на изображении, поданном на распознавание.

4.1 Стандартный CSS алгоритм

Он состоит из следующих этапов:

1 Этап: Для получения контура каждой из букв применяется алгоритм Кенни [3].

2 Этап: Символы могут оказаться размазанными, и контур, возможно, будет иметь разрывы в некоторых местах. Для устранения этих разрывов применяется морфологическое замыкание. Результат этого шага представлен на рис. 3.

Рис 3: Результат 1 и 2 этапа классификации.

3 Этап: Так как алгоритм CSS работает только с одним контуром, для букв с несколькими контурами оставляем только внешний (рис.4).

Рис 4: Результат 3 этапа классификации.

4 Этап: Имея закрашенные фигуры, нужно получить их контура, но уже не просто в виде картинок, а в виде функций от одной переменой. Для этого обходим букву по контуру, собирая его координаты, и нормируем по параметру длины дуги u (рис.5).

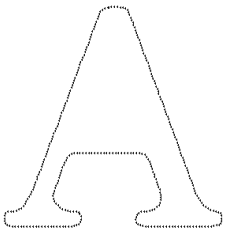

Рис 5: Результат 4 этапа классификации.

5 Этап: После предыдущего этапа у нас будет плоская кривая $\Gamma(u)$

$$\Gamma(u) = \{(x(u),y(u))|u \in [0,1]\}, \quad (1)$$

с нормированным параметром длины дуги u. Кривая сглаживается одноразмерной гауссианой с ядром $g(u,\sigma)$ с шириной σ. Деформация замкнутой плоской кривой представлена в виде

$$\Gamma(u,\sigma) = \{(X(u,\sigma),Y(u,\sigma))|u \in [0,1]\}, \quad (2)$$

где X(u,σ) и Y (u,σ) обозначают компоненты x(u) и y(u) после свертки с $g(u,\sigma)$. Вид кривой представлен на рис.6

Рис 6: Результат 5 этапа классификации.

6 Этап: Рассчитаем кривизну κ(u,σ) видоизменённой кривой. В стандартном методе предлагается использование частных производных полученной кривой $Xu(u,\sigma)$, $Xuu(u,\sigma)$, $Yu(u,\sigma)$, и $Yuu(u,\sigma)$. А после предыдущего этапа это функции свертки исходной кривой и функции Гаусса. Так как расчет производных от кривой, представленной дискретными значениями, сложен, и его результат не даст ожидаемой сглаженности, предлагается воспользоваться правилом дифференцирования [7] и вместо этого использовать в свертке первую и вторую производные функции Гаусса, т.е. WAVE-вейвлет и MHAT-вейвлет. И тогда по

$$k(u,\sigma) = \frac{X_u(u,\sigma) * Y_{u\ \ (u,\sigma}) - X_{u\ \ (u,\sigma})) * Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + \quad ,\sigma)^2)^{3/2}} \quad (3)$$

рассчитаем кривизну $k(u,\sigma)$ видоизменённой кривой.

7 Этап: Получим CSS изображение $I(u,\sigma)$ (рис. 7), определяющееся как

$$I(u,\sigma) = \{(u,\sigma)|\kappa(u,\sigma) = 0\}. \quad (4)$$

Рис 7: Стандартное CSS изображение символа «А».

Изображение CSS показывает нулевые пересечения с точки зрения их расположения на контуре и ширины ядра Гаусса.

Во время процесса деформации, нулевые пересекающие сливаются, так как переходы между сегментами контура различной кривизны уравниваются. Следовательно, после определённого количества итераций, точки перегиба исчезают, а форма замкнутой кривой становится выпуклой. Значительные свойства контура, которые видны во время большого числа итераций, выливаются в высокие пики на CSS изображении. Тем не менее, области с быстро меняющейся кривизной, вызванной шумом, выдают только небольшие локальные максимумы.

4.2 Расширенный CSS алгоритм

Основным недостатком стандартного подхода является неполноценное представление выпуклых сегментов контура. Так как CSS изображение представляет расположение точек перегиба, нужны сегменты вогнутого контура. А контур символов без вогнутых сегментов (например «О» рис. 8) не может быть выделен.

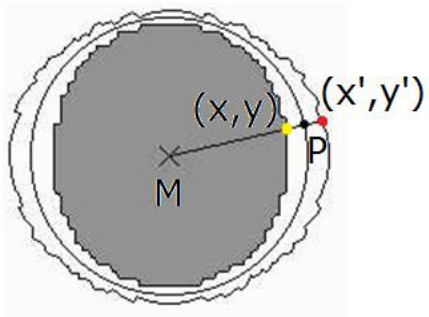

Рис 8: Стандартный и инверсный контур буквы «О».

Мы используем вначале стандартный CSS подход, чтобы получить собственные векторы функций, которые классифицируют вогнутые части контура очень хорошо. Общая идея состоит сейчас в том, чтобы создать второй контур, который обеспечит дополнительные возможности для выпуклых сегментов оригинального контура. Оригинальный контур переходит в новый контур с инвертированной кривизной. Сильновыпуклые сегменты исходного контура становятся вогнутыми сегментами отображенного контура. Значительная кривизна в исходном контуре также существует в отображенном контуре.

Чтобы создать преобразованный контур, мы ограничиваем контур символа окружностью радиуса R и определяем точку P окружности, ближайшую к каждому пикселю контура. Контур пикселей зеркально отображается относительно окружности в P. Сегменты контура, которые имеют сильновыпуклую кривизну, сопоставляются с вогнутыми сегментами. Степень искривления преобразованных контуров зависит от радиуса R и от кривизны оригинального контура.

Расчёт преобразованного контура достаточно быстр. Каждый пиксель контура в позиции u замкнутой плоской кривой $(x(u),y(u))$ отображается на кривой $(x`(u),y`(u))$. Центр окружности (Mx,My) с радиусом R рассчитывается как среднее положение контура пикселей. Результат создания инверсного CSS изображения показан на рис. 9.

Рис 9: CSS изображение символа «А»: а) – стандартное, б) - инверсное.

5. ТЕСТИРОВАНИЕ

Система обучалась на выборке букв русского алфавита на 69 изображениях. Все обучающие буквы были написаны шрифтом Calibri с размером 100. Тестирование системы проводилось на выборке из 115 изображений подобранных с из интернета фотографий, а так же сделанных самостоятельно. Диапазон качества изображений был испробован от 300 dpi до 100dpi. В качестве шрифтов для тестируемых изображений использовался не только Calibri, но и Times New Roman, Courier, Tahoma, Verdana, Arial Black. Так же на изображениях присутствовал фон различной сложности (рис. 10).

В качестве критерия оценки распознавания была выбрана ошибка первого рода. В роли гипотезы выступает правильное распознавание символа. Процент распознавания определялся как:

$$Rez = (x * 100\%)/y \quad (5)$$

где x - общее количество символов на изображении, а y – количество правильно распознанных. Общий результат вычислялся как среднее значение по всей выборке оттестированных изображений и составил 85%.

6. ЗАКЛЮЧЕНИЕ

В результате выполнения описанных алгоритмов на языке программирования Java 1.7 нами была получена система, которая считывает текст с изображения. На вход для обучения подается выборка букв русского алфавита. Распознавание же мы проводили на изображениях с текстом различного шрифта и с различным фоном (рис. 10).

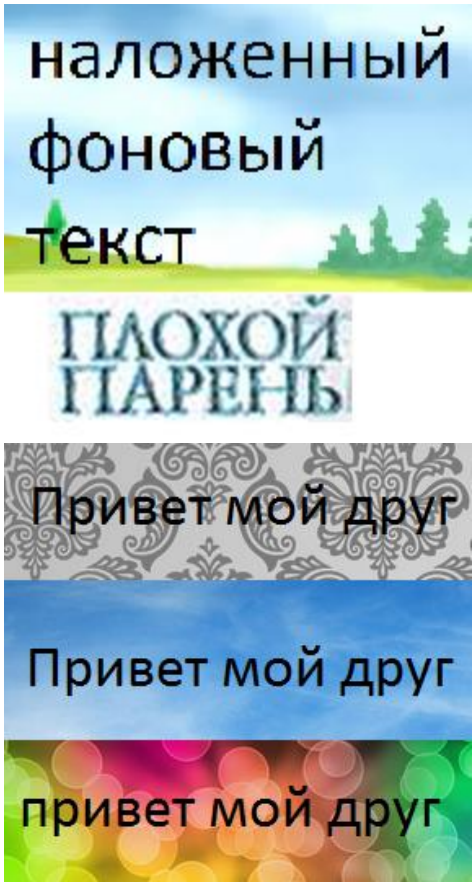

Рис 10: Текстовые изображения на сложном фоне

Результаты показали (рис. 11, 12), что для всех этих шрифтов осуществляется распознавание текста, даже не смотря на то, что обучение проводилось на другом шрифте. Также были проверены различные размеры шрифта в диапазоне от 14 до 30 с курсивным и жирным стилем. Распознавания дают правильный текст в 85% (рис.11, 12), что говорит о неплохой надёжности, но и о потенциальной дальнейшей работе в этом направлении.

```
[
нал0Женнь!й
ф0н0вь!й
текст
]

[
ПЛ0Х0й
ПАРЕНь
]
```

Рис 11: Результаты работы системы

```
[
привРт М0% друг
]
[
привет М0й друг
]
[
привет М0й друг
]
```

Рис 12: Результаты работы системы

7. ССЫЛКИ

[1] Kopf S., Haenselmann T., Effelsberg, W. *Enhancing curvature scale space features for robust shape classifi-cation. Multimedia and Expo, 2005. ICME 2005. IEEE International Conference, 2005.*

[2] Wang J., Jean J. *Segmentation of merged characters by neural networks and shortest-path, Pattern Recogni-tion, 1994, pp 649-658.*

[3] Билл Грин. *Алгоритм выделения контуров CANNY, Дрексельская лаборатория автоматизированных систем, 2002.*

[4] Дэйвид Г. *Порядковые статистики. Москва «Наука»: Главная редакция физико-математической литературы, 1979.*

[5] Кобзарь Г.А. *Модель межмасштабного пространства кривизны для представления формы геометрических объектов // Искусственный интеллект. – 2008. – с. 92-101.*

[6] http://www.cvisiontech.com/reference/general-information/ocr-applications.html

[7] ru.wikipedia.org/wiki/Свёртка_(математический_анализ).

Об авторах

Анна Балахонцева – студентка ОНПУ. Ее адрес: a.c.balackhontseva@gmail.com.

Александр Годоба – студент ОНПУ. Его адрес: hedgecrab@gmail.com.

Тьен Нгуен – аспирант ОНПУ. Ее адрес: ktien85@ukr.net.