УДК 004.352.242

УЛУЧШЕНИЕ КАЧЕСТВА МЕТОДА ОПТИЧЕСКОГО РАСПОЗНАВАНИЯ ТЕКСТОВ С ПОМОЩЬЮ СОВМЕСТНОГО ПРИМЕНЕНИЯ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЙ, КУРВЛЕТ-ПРЕОБРАЗОВАНИЙ И АЛГОРИТМОВ СЛОВАРНОГО ПОИСКА

Д.С. Григорьев, П.А. Хаустов, В.Г. Спицын

Томский политехнический университет
E-mail: _tryGX@sibmail.com, eXceibot@sibmail.com

Оптическое распознавание символов является комплексной задачей, для решения которой не существует определенного алгоритма. Существует множество подходов и методов для решения данной задачи. Предложенный ранее метод, основанный на совместном применении вейвлет-преобразования для сокращения размерности пространства признаков и вероятностной нейронной сети для классификации, показал приемлемые результаты. Однако предложенный метод может быть дополнен и улучшен алгоритмами предварительной и пост-обработки. Предложен метод предобработки отсканированных изображений на основе адаптивного порогового преобразования в алгоритмах дискретных вейвлет и курвлет-преобразований. Проведены численные эксперименты по выявлению наиболее результативного алгоритма для предобработки. В качестве алгоритма пост-обработки предложен метод улучшения качества распознавания текста на основе алгоритма словарного поиска с использованием динамического программирования.

Ключевые слова:
Пороговое преобразование, вейвлет-преобразование, курвлет-преобразование, динамическое программирование, префиксное дерево, словарный поиск.

Введение

Системы оптического распознавания текста состоят из следующих основных блоков, предполагающих аппаратную или программную реализацию:

- сегментации элементов текста;
- предобработки изображения;
- выделения признаков;
- распознавания символов;
- постобработки результатов распознавания.

После использования алгоритма оптического распознавания символов возникает необходимость в улучшении качества распознавания текста. Для того чтобы улучшить качество распознаваемого текста без изменения механизма решения задачи оптического распознавания символов, вводятся блоки предобработки изображений и постобработки результатов оптического распознавания.

Предобработка изображений

Как известно, вейвлет-преобразование широко применяется для анализа сигналов, а также зарекомендовало себя как эффективный инструмент для сжатия и предобработки изображений [1, 2]. Приемлемые результаты исследований в указанных работах обуславливают выбор метода дискретного вейвлет-преобразования (ДВП) для дальнейшего применения. Однако при вейвлет-преобразовании исходные данные претерпевают значительные потери при растяжении и вращении, также в преобразовании отсутствует пространственная ориентированность.

Курвлет-преобразование (curve – кривая, изгиб; curvelet – маленький изгиб) в данном случае является более подходящим инструментом для определения свойств ориентированности объекта на изображении, обеспечивая оптимальное представление о разреженности, предоставляя максимальную концентрацию энергии вдоль краев объекта [3]. Курвлет-преобразование является многомерным, многоуровневым и локализованным в окне масштаба, пропорционального следующему отношением длины и ширины: «*длина*^{2}≈*ширина*» [4]. Одной из целей данной работы является выявление наиболее подходящего алгоритма предобработки зашумленного изображения.

Курвлет-преобразование

Курвлеты – базовые элементы с высокой чувствительностью к ориентации и высокой анизотропностью [3, 4]. Дискретное курвлет-преобразование функции вариации яркости изображения $f(x,y)$ использует диадические последовательности масштабов и банков фильтров ($P_0f,\Delta_1f,\Delta_2f,\ldots$). Высокочастотные фильтры Ψ_{2S} взаимодействуют с частотами области $|\xi|\in[2^{2S},2^{2S+2}]$ и обладают рекурсивной конструкцией $\Psi_{2S}(x)=2^{4S}\Psi(2^{2S}x)$, а низкочастотный фильтр Φ_0 взаимодействует с частотами области $|\xi|\le 1$. Субполосное разложение выполняется при помощи операции свертки: $\Delta_Sf=\Psi_{2S}^{*}f$, $P_0f=\Phi_0^{*}f$. Схема алгоритма курвлет-преобразования приведена на рис. 1.

1. Субполосное разложение. Функция вариаций яркости изображения раскладывается в набор субполос: $f\rightarrow(P_0f,\Delta_1f,\Delta_2f,\ldots)$. Каждый набор Δ_Sf содержит детали различных частот: P_0 – фильтр нижних частот, $\Delta_1,\Delta_2,\ldots$ – фильтры высоких частот. Исходное изображение может быть восстановлено по формуле (1):

$$f = P_0(P_0f) + \sum_S \Delta_S(\Delta_Sf). \qquad (1)$$

При этом выражение (2) для сохранения энергии:

$$\| f \|_2^2 = \| P_0(P_0f) \|_2^2 + \sum_S \| \Delta_S(\Delta_Sf) \|_2^2. \qquad (2)$$

2. «Гладкое» разделение. Каждая субполоса локализуется в плавающем окне соответствующего масштаба, $\Delta_S\rightarrow(w_Q\Delta_Sf)_{Q\in Q_S}$. Здесь w_Q – это набор окон, локализованных вокруг диадических квадратов:

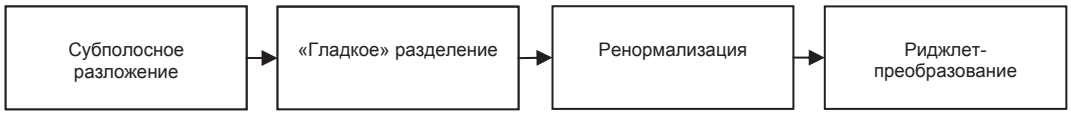

Рис. 1. *Основные этапы алгоритма курвлет-преобразования*

$$Q = [k_1 / 2^S, (k_1 + 1) / 2^S] \times [k_2 / 2^S, (k_2 + 1) / 2^S].$$

Умножая $\Delta_S f$ на w_Q, производим разделение функции на «квадраты» $h_Q = w_Q \cdot \Delta_S f$.

3. Ренормализация. Происходит центрирование каждого диадического квадрата к единичному: [0,1]×[0,1]. Для каждого Q оператор T_Q определен выражением (3):

$$(T_Q f)(x_1, x_2) = 2^S f(2^S x_1 - k, 2^S x_2 - k_2). \quad (3)$$

Каждый квадрат ренормализуется по формуле (4):

$$g_Q = T_Q^{-1} h_Q. \quad (4)$$

4. Риджлет-преобразование [5]. Разделение частотной области на диадическую «корону» определяется как $|\xi| \in [2^S, 2^{S+1}]$. Каждый элемент риджлет-преобразования в частотной области задается выражением (5):

$$\hat{\rho}_\lambda(\xi) =$$

$$= \tfrac{1}{2} |\xi|^{-\frac{1}{2}} (\widehat{\psi}_{j,k}(|\xi|) \cdot \omega_{i,l}(\theta) + \widehat{\psi}_{j,k}(-|\xi|) \cdot \omega_{i,l}(\theta + \pi)), \quad (5)$$

где $\omega_{i,l}$ – периодические вейвлеты, определенные на $[-\pi, \pi)$; i – угловой масштаб и $l \in [0, 2^{i-1} - 1]$ – расположение угла; $\psi_{j,k}$ – вейвлеты Мейера на $\Re$; j – масштаб риджлета и k – расположение риджлета. Каждый нормализованный квадрат подвергается анализу в риджлет-системе $\alpha_{(Q,\lambda)} = \langle g_Q, \rho_\lambda \rangle$, где каждый фрагмент обладает соотношением сторон $2^{-2S} \times 2^{-S}$. После ренормализации квадрат обладает частотой локализированной в полосе $|\xi| \in [2^S, 2^{S+1}]$ [3–5, 6].

Обратное курвлет-преобразование

Обратное курвлет-преобразование происходит в четыре основных этапа:

1. Риджлет-синтез (формула (6)):

$$g_Q = \sum_л б_{(Q,л)} \cdot c_л. \quad (6)$$

2. Ренормализация (формула (7)):

$$h_Q = T_Q g_Q. \quad (7)$$

3. «Гладкая» интеграция (формула (8)):

$$\Delta_s f = \sum_{Q \in \mathbf{Q}_s} w_Q h_Q. \quad (8)$$

4. Субполосная реконструкция (формула (9)):

$$f = P_0(P_0 f) + \sum_s \Delta_s(\Delta_s f). \quad (9)$$

Удаление шумов

Входное изображение представлено функцией вариации яркости двух переменных $f(x,y)$. Пусть зашумленное изображение $f_n(x,y) = f(x, y) + \sigma z_g(x,y)$, где σ – это стандартное отклонение шума, а $z_g(x,y)$ – значение белого шума с нулевым математическим ожиданием ($\mu_g = 0$) и единичной дисперсией $\sigma_g^2 = 1$. Ставится задача нахождения оптимальной конфигурации фильтра для очистки зашумленного изображения перед его последующей бинаризацией. Очищенное изображение на следующем этапе подвергается сегментации с целью выделения отдельных символов для распознавания. В данной работе представлены два метода для удаления шумов на изображениях.

В первом методе удаление шумов осуществляется на основе применения вейвлет-преобразования, а во втором методе – на основе применения курвлет-преобразования.

Метод вейвлет-преобразования

Метод предобработки, основанный на применении вейвлет-преобразования, представлен на рис. 2.

На представленной схеме обозначены основные блоки этапа предобработки изображения. На поступившее исходное изображение добавляется импульсные и Гауссовы шумы. Далее изображение подвергается двухуровневому дискретному вейвлет-преобразованию с базисной функцией Хаара для извлечения соответственно коэффициентов детализации и коэффициентов аппроксимации. Основным этапом шумоподавления является применение порога λ для набора коэффициентов деталей, который задается выражением (10):

$$\lambda_j = \sigma \sqrt{2 \log(N_j)}. \quad (10)$$

Формула порогового преобразования приведена в [2]. Здесь индекс j – уровень преобразования, а N_j – размер матрицы коэффициентов на соответствующем уровне преобразования. Значение σ вычисляется при помощи медианного абсолютного отклонения высокочастотных вейвлет-коэффициентов детализации (11):

$$\sigma = \frac{median(|\omega_k|)}{0,6745}. \quad (11)$$

На следующем этапе вычисляется обратное дискретное вейвлет-преобразование, и в результате на выходе получается очищенное изображение. Затем очищенное изображение подвергается бинаризации.

Метод курвлет-преобразования

Метод предобработки, основанный на применении курвлет-преобразования, представлен на рис. 3.

После добавления шума изображение подвергается дискретному курвлет-преобразованию. Затем извлекаются соответствующие зашумленному изображению курвлет-коэффициенты. Происходит вычисление стандартного отклонения значения шума, и производится оценка порогового преобразования для каждого масштаба аналогично (10). После оцен-

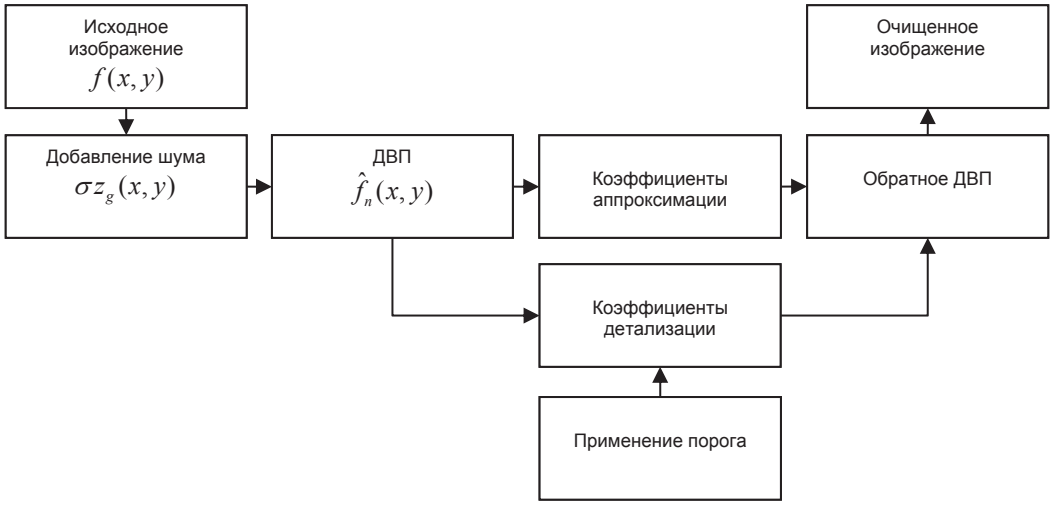

Рис. 2. *Схема метода предобработки при помощи вейвлет-преобразования*

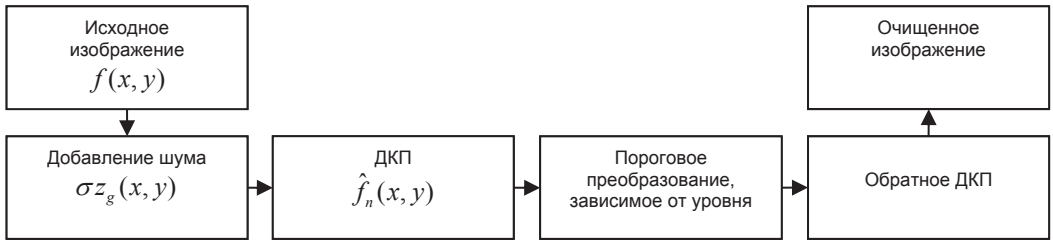

Рис. 3. *Схема метода предобработки при помощи курвлет-преобразования*

ки применяется пороговое преобразование для курвлет-коэффициентов [6]. На следующем этапе производится обратное курвлет-преобразование. Очищенное изображение подвергается бинаризации.

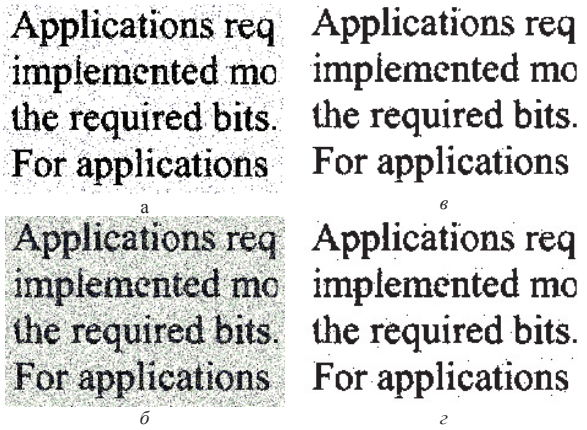

Рис. 4. *а) Исходное изображение с импульсным шумом; б) исходное изображение, зашумленное при помощи гауссовского и импульсного шумов; в) выходное бинаризированное изображение после обработки методом курвлет-преобразования; г) выходное бинаризированное изображение после обработки методом вейвлет-преобразования*

Сравнение результатов вейвлет и курвлет-преобразований

Результаты численных экспериментов по применению вейвлет и курвлет-преобразований для удаления гауссовского и импульсного шума на изображениях представлены на рис. 4. Обработке подвергалось изображение, представленное на рис. 4, *б*.

Полученные результаты показывают, что алгоритм, основанный на курвлет-преобразовании, позволяет получить изображение более высокого качества по сравнению с алгоритмом, основанным на вейвлет-преобразовании.

Словарный поиск

В случае распознавания не отдельных символов, а целых слов или даже текстов вероятностный смысл выходов PNN-сети рационально использовать для поиска наиболее вероятного совпадения текущего слова с некоторым словом в словаре [7]. Количество слов в словаре, как правило, достигает нескольких сотен тысяч, поэтому возникает необходимость в алгоритме, обладающем высоким быстродействием.

Тривиальным решением является использование взвешенного расстояния Левенштейна, где весами являлись бы вероятности с выходов PNN-сети, а также вероятности корректности распознавания и сегментации, полученные статистическими методами. Фактически при таком подходе будет определяться математическое ожидание расстояния Левенштейна [8] для всех возможных вариантов распознавания и сегментации слова. Для определения расстояния Левенштейна используется метод динамического программирования, который, в свою очередь, использу-

ет рекуррентное задание функции $F(c_1,c_2)$, заданной выражением (12) – значение функции расстояния Левенштейна для суффиксов двух строк, начинающихся с позиций c_1 и c_2 соответственно:

$$F(c_1,c_2)=\min(F(c_1+1,c_2)\cdot W_1,F(c_1,c_2+1)\times \\ \times W_2,F(c_1+1,c_2+1)\cdot W_3). \quad (12)$$

Существенным недостатком такого подхода является необходимость в нахождении расстояния Левенштейна до каждого из слов в словаре. Стоит учесть, что сложность вычисления расстояния Левенштейна линейно зависит от произведения длины распознаваемого слова и длины словарного слова. Учитывая, что размер словаря может быть достаточно большим, данный алгоритм обладает недостаточным быстродействием – его асимптотическая оценка $O(KL^2)$, где K – количество слов в словаре, L – среднестатистическая длина слова. Однако преимуществом этого алгоритма является то, что он рассматривает каждое слово словаря.

Для того чтобы улучшить быстродействие этого алгоритма, можно воспользоваться тем фактом, что достаточно большое количество слов в большинстве языков имеют общий префикс. Так, например, в английском языке слова «preposition», «predicate» и «present» имеют одинаковый префикс «pre». Очевидно, для подобных префиксов функцию расстояния Левенштейна считать более одного раза не имеет смысла. Следовательно, имеет смысл использовать следующее улучшение алгоритма. Для хранения словаря целесообразно применить префиксное дерево [9]. В префиксном дереве каждому ребру соответствует определенный символ. Каждое слово в этом дереве представлено путем от корня к некоторой вершине, которая является терминальной. Терминальными вершинами являются только вершины, в которых заканчивается путь некоторого слова. Теперь при поиске расстояния Левенштейна будет использоваться не позиция символа в конкретном слове, а вершина префиксного дерева. В таком случае для всех возможных префиксов значение функции расстояния Левенштейна будет посчитано ровно один раз. Фактически расстояние Левенштейна будет считаться параллельно для всех слов с одинаковым префиксом.

На рис. 5 можно увидеть префиксное дерево, построенное по словам: «car», «card», «carry», «cart», «cat», «cel», «celery», «close», «closely», «closet» и «clue». Терминальные вершины, соответствующие этим словам, выделены черным цветом.

При таком подходе функция расстояния Левенштейна принимает следующий вид: $F(c,v)$ задается выражением (13), где v – вершина в префиксном дереве, а c – текущий символ, для которого будет искаться соответствие при переходе по ребрам из вершины v:

$$F(c,v)=\min(F(c+1,v_i')\cdot W_i). \quad (13)$$

При переходе по ребрам дерева значение вероятности распознавания соответствующего символа является значением вероятности перехода в вершину, в которую ведет это ребро. Таким образом, наиболее вероятен переход по ребру, соответствующему символу, к которому отнесен соответствующий образ с наибольшей вероятностью. Фактически функция $F(v,c)$ позволяет найти наиболее вероятный путь в дереве, ведущий к терминальному состоянию.

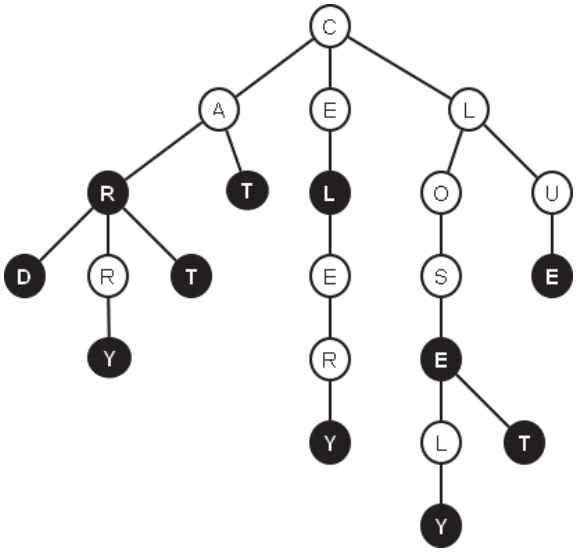

Рис. 5. *Пример префиксного дерева*

Очевидно, стоит учесть, что при распознавании могла быть допущена ошибка. Для этого необходимо задать некоторую величину вероятности P_E, с которой PNN-сеть допустила ошибку при распознавании. Значение вероятности P_E можно определить эмпирическим путем. Тогда с вероятностью (P_E/A) необходимо осуществлять переход по каждому из ребер, ведущих из вершины v, где A – количество ребер, выходящих из вершины v.

Операции удаления и добавления символа при поиске расстояния Левенштейна используются для учета ошибок сегментации. Вероятность ошибки сегментации P_S можно так же, как и вероятность ошибки при распознавании P_E, определить экспериментальным путем.

Можно дать асимптотическую оценку сложности работы полученного алгоритма. Пусть количество вершин в дереве префиксов равно V, а среднестатистическая длина слова в языке равна L, тогда асимптотическая оценка алгоритма равна $O(VL)$. Для реально существующих языков V намного меньше, чем KL (из-за большого количества слов с одинаковыми префиксами), отсюда можно сделать вывод, что использование префиксного словарного дерева увеличивает быстродействие алгоритма.

При поиске слова в словаре необходимо учитывать некоторые особенности этого слова. Так, например, необходимо посчитать вероятность того, что первая буква этого слова является заглавной. Если эта вероятность более 0,5, следует пересчитать вероятности принадлежности каждому из классов и при поиске искать то же самое слово, но без учета заглавной буквы. Можно также посчитать математическое ожидание количества заглавных букв, чтобы определять слова, являющиеся

аббревиатурами. Стоит отметить, что предыдущее правило не применимо для аббревиатур, что тоже необходимо учесть.

Для того чтобы не искать в словаре слова, которые содержат маленькое количество букв, можно заранее посчитать математическое ожидание количества букв в слове. Если это значение недостаточно велико, то словарный поиск осуществлять не имеет смысла. Так, например, нет смысла искать в словаре числа или какие-либо численно-буквенные обозначения. Также можно отбрасывать первый и последний символы, если они являются знаками препинания с достаточно большой вероятностью. Такое достаточно часто случается из-за того, что знаки препинания присоединяются к слову в результате сегментации или при наличии небольшого пиксельного шума в строке с этим словом.

Зачастую из-за ошибок сегментации некоторые слова склеиваются с использованием знаков препинания. Подобное возможно из-за пиксельного шума, который ошибочно воспринимается как точка, запятая, апостроф или двоеточие. При этом, казалось бы, поиск по словарю не имеет смысла, ведь даже информация о реальном количестве символов в слове является утерянной. В таком случае возникает необходимость в алгоритме, способном находить наиболее вероятное разбиение полученной лексемы на словарные слова.

Идея алгоритма, предложенного для решения такого рода задачи, также основывается на принципе динамического программирования. Будем считать некоторую последовательность символов словарно-представимой, если она состоит только из знаков препинания или образует слово, которое содержится в словаре. Тогда требуется найти наиболее вероятное разбиение имеющейся лексемы на словарно-представимые последовательности символов. Если применить идею динамического программирования, то для каждого суффикса полученной лексемы можно находить наиболее вероятное разбиение на словарно-представимые последовательности. Для того чтобы найти наиболее вероятное разбиение некоторого суффикса $P_{SUF}(i)$ достаточно перебрать все последовательности символов, начинающиеся с этой позиции. Затем для каждой из них необходимо найти вероятность $P_{SUBSTR}(i,j-1)$ того, что эта последовательность является словарно-представимой, умножить ее на вероятность наиболее вероятного разбиения оставшейся суффиксной части лексемы $P_{SUF}(j)$ и выбрать из всех этих значений максимум (14):

$$P_{SUF}(i) = \max(P_{SUF}(j) \cdot P_{SUBSTR}(i, j-1)). \qquad (14)$$

Таким образом, искомое разбиение для суффикса наибольшей длины и будет являться наиболее вероятным разбиением на словарно-представимые последовательности всей лексемы.

Результат применения алгоритма словарного поиска

Для апробации предложенного метода был использован электронный словарь «ewords», все слова которого хранятся в абстрактном типе данных – префиксном дереве (trie-дерево). Целесообразность такого способа хранения была обусловлена ускоренным способом подсчета функции Левенштейна для всех слов словаря. Однако для оценки эффективности такого представления данных следует также оценить различия в быстродействии и в потребляемой оперативной памяти.

Для того чтобы сравнить объем памяти, потребляемой при тривиальном способе хранения, с объемом памяти, который требуется для хранения префиксного дерева, достаточно воспользоваться нативными средствами языка C++ и оператором sizeof.

Для словаря «ewords» были определены следующие значения объема потребляемой памяти. При тривиальном способе хранения задействуется 5419672 байт оперативной памяти. При способе хранения с использованием префиксного дерева – 4520710 байт. Как можно заметить, объемы потребляемой памяти для двух описанных способов хранения паритетны. Небольшое преимущество способа хранения с использованием префиксного дерева объясняется существенной экономией потребляемой памяти из-за большого количества словоформ и других слов с одинаковыми префиксами.

Для сравнения быстродействия был выбран один из текстов ранее обработанного набора данных с большим количеством пиксельного шума. Для всех слов этого текста поиск по словарю занял 1136,788 секунд при тривиальном проходе по всем словам и нахождении функции Левенштейна для каждого из них независимо. При использовании предложенного алгоритма, который позволяет осуществлять поиск функции Левенштейна параллельно сразу для нескольких слов словаря с одинаковым префиксом, время работы существенно ниже – 415,445 секунд.

Несложно объяснить такое преимущество времени обработки, если еще раз обратить внимание на асимптотические оценки алгоритмов. Тривиальный подход имеет сложность $O(L_T L_A)$, где L_T – суммарная длина всех слов в словаре, а L_A – средняя длина слова. При подходе с использованием префиксного дерева асимптотическая сложность алгоритма – $O(VL_A)$, где V – количество вершин в префиксном дереве. Очевидно, зависимость отношения количества времени в первом случае к количеству времени во втором будет оцениваться примерно, как отношение L_T к V. И действительно, для проведенного эксперимента количество вершин в полученном префиксном дереве равно 502302, суммарная длина всех слов равна 1354918. Отношение L_T к V примерно равно 2,7. Это соответствует экспериментально полученным данным.

Заключение

1. Апробированы два метода предобработки изображений – вейвлет- и курвлет-преобразования для удаления пиксельного шума с отсканированных изображений.
2. Установлено, что наиболее подходящим алгоритмом для удаления пиксельного шума с от-

сканированных изображений является курвлет-преобразование.
3. Предложен оригинальный метод словарного поиска с использованием префиксного дерева и динамического программирования.
4. Экспериментально установлено, что хранение словаря с использованием префиксного дерева позволяет улучшить быстродействие, что подтверждено асимптотическими оценками.
5. В дальнейшем планируется применение предложенных методов в системе оптического распознавания текстов. При этом предполагается использовать не только пороговые преобразования, но и морфологические для предобработки изображений и удаления пиксельного шума.

Работа выполнена при финансовой поддержке гранта РФФИ № 12-08-00296а.

СПИСОК ЛИТЕРАТУРЫ

1. Misiti M., Misiti Y., Oppenheim G., Poggi J. Wavelets and their applications. – London: ISTE, 2007. – 352 p.
2. Gnanadurai D., Sadasivam V. An efficient adaptive thresholding technique for wavelet based image denoising // World Academy of Science, Engineering and Technology. – 2006. – V. 1 (2). – P. 114–119.
3. Donoho D.L., Duncan M.R. Digital curvelet transform: strategy, implementation and experiments // Proc. Aerosense2000, Wavelet Applications VII, SPIE. – Stanford, California, 2000. – V. 4056. – P. 12–29.
4. Donoho D.L. De-noising by soft thresholding // IEEE Transaction on Information Theory. – Stanford, California: IEEE, 1995. – V. 41. – P. 613–627.
5. Cande`s E.J. Ridgelets: theory and applications: Ph.D. Thesis. – Stanford, 1998. – 13 p.
6. Starck J., Candes E.J., Donoho D.L., The curvelet transform for image denoising // IEEE transactions on image processing. – 2002. – V. 11. – № 6. – P. 61–66.
7. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2000. – 224 с.
8. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады Академий Наук СССР. – М.: Проспект, 2009. – С. 56–59.
9. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – М.: Изд. дом «Вильямс», 2011. – 1293 с.

Поступила 03.09.2013 г.

UDC 004.352.242

IMPROVING THE QUALITY OF OPTICAL CHARACTER RECOGNITION METHOD BY JOINT APPLICATION OF WAVELET-AND CURVELET-TRANSFORMS AND VOCABULARY SEARCH ALGORITHM

D.S. Grigoryev, P.A. Khaustov, V.G. Spitsyn

Tomsk Polytechnic University

Optical character recognition is a complex problem, which has no definite solution. There are a lot of approaches and methods to solve this problem. The proposed approach, based on aggregate usage of wavelet-transformation for reducing the feature space and probabilistic neural network for classification, has shown a good quality of recognition. However the proposed approach can be improved with preprocessing and postprocessing algorithms. The algorithm of preprocessing based on adaptive thresholding for curvelet and wavelet transformations is proposed. The numerical experiments are held to determine the most efficient algorithm of preprocessing. The approach based on vocabulary search and dynamic programming is proposed for postprocessing.

Key words:
Thresholding, wavelet-transform, curvelet-transform, dynamic programming, trie-tree, vocabulary search.

REFERENCES

1. Misiti M., Misiti Y., Oppenheim G., Poggi J. *Wavelets and their applications*. London, ISTE, 2007. 352 p.
2. Gnanadurai D., Sadasivam V. An efficient adaptive thresholding technique for wavelet based image denoising. *World Academy of Science, Engineering and Technology*, 2006, vol. 1 (2), pp. 114–119.
3. Donoho D.L., Duncan M.R. Digital curvelet transform: strategy, implementation and experiments. *Proc. Aerosense2000, Wavelet Applications VII, SPIE*. Stanford, California, 2000, vol. 4056, pp. 12–29.
4. Donoho D.L. De-noising by soft thresholding. *IEEE Transaction on Information Theory*. Stanford, California, IEEE, 1995, vol. 41, pp. 613–627.
5. Cande`s E.J. *Ridgelets: theory and applications*. Ph.D. thesis. Stanford, 1998. 13 p.
6. Starck J., Candes E.J., Donoho D.L. The curvelet transform for image denoising. *IEEE transactions on image processing*, 2002, vol. 11, no. 6, pp. 61–66.
7. Kruglov V.V., Dli M.I., Golunov R.Yu. *Nechetkaya logika i iskusstvennye neyronnye seti* [Fuzzy Logic and Artificial Neural Networks]. Moscow, Fizmatlit Publ., 2000. 224 p.
8. Levenshteyn V.I. Dvoichnye kody s ispravleniem vypadeniy, vstavok i zameshcheniy simvolov [Binary codes with corrections of removed, inserted and replaced characters]. *Doklady Akademij Nauk SSSR* [Reports of USSR scientific academy]. Moscow, Prospekt Publ., 2009. pp. 56–59.
9. Kormen T., Leyzerson Ch., Rivest R., Shtayn K. *Algoritmy: postroenie i analiz* [Introduction to Algorithms]. Moscow, Williams Publ., 2011. 1293 p.