УДК 004:627

В.Г. Иванов, Ю.В. Ломоносов, М.Г. Любарский

Национальный университет
«Юридическая академия Украины имени Ярослава Мудрого», Харьков

КЛАССИФИКАЦИОННЫЕ МЕТОДЫ СЖАТИЯ ИЗОБРАЖЕНИЙ ОЦИФРОВАННОГО ТЕКСТА. ЧАСТЬ I

В работе рассматриваются методы классификации, применяемые при сжатии файла с битональным изображением текста, полученным сканированием или цифровым фотографированием. Особое внимание обращается на используемые при этом меры различия двух изображений символов, выделенных из изображения текста. Эти меры различия позволяют с той или иной степенью уверенности считать символы на сравниваемых изображениях или совпадающими, или различными. Для известных на сегодняшний день алгоритмов классификации, включая хорошо известный алгоритм JB2, приведены количественные характеристики классификации – число классов, получаемых этими алгоритмами для изображения стандартной страницы текста. Чем меньше это число, тем качество классификации считается выше, так как дает лучшее сжатие файла с изображением текста. Рассмотрены также методы ускорения алгоритмов, классифицирующих изображения символов, и повышения удобочитаемости восстановленного после сжатия изображения текста.

Ключевые слова: *изображение текста, методы классификации, сжатие данных.*

Вступление

Постановка проблемы и анализ литературы. Бесценные сокровища литературной, научной, философской мысли, которые накопило человечество за несколько тысяч лет своей истории, хранятся в многочисленных библиотеках в печатном или рукописном виде и нуждаются в переводе в электронную форму. Этому достаточно много причин. Например, физическое дряхление многих уникальных экземпляров, дороговизна содержания библиотек и библиотечных хранилищ, малая доступность редких книг, документов, периодики для читателя и многое-многое другое.

Процесс перевода бумажных книг в электронный (цифровой) вид *называется оцифровкой*. Электронные копии книг могут образовывать электронные библиотеки и распространяться в Интернете. Примером, может служить библиотека «Либрусек» (http://lib.rus.ec), насчитывающая более 100000 книг, и многие другие платные и бесплатные библиотеки. В результате оцифровки получаются *электронные книги* – то есть хранимый в файле текст, оформленный в виде привычной книги. Электронная книга обычно разделена на страницы, которые пронумерованы, имеют поля, иллюстрации и тому подобное.

Еще в недавнем прошлом создание электронной книги происходило только с помощью ручного набора текста, что является крайне трудоемкой и, следовательно, дорогой операцией. В настоящее время оцифровка печатных документов осуществляется с помощью сканера или цифрового фотоаппарата с последующей программной обработкой и сохранением в одном из форматов графических файлов. Этот этап обязателен. На втором, необязательном этапе производится оптическое распознавание текста (технология OCR), превращающая изображение текста в собственно текст, с последующим сохранением в одном из форматов электронных книг.

Таким образом, важно различать *сканированные* и *вёрстанные* электронные книги.

Вёрстанные книги – это либо материал, подготовленный авторами в каком-либо редакторе, например, во всем доступном MS Word, либо распознанная и вручную вычитанная и отформатированная печатная книга. Конечным результатом является электронная книга в формате PDF (Adobe Portable Document Format), e Book (Electronic Publication), FB2 (Fiction Book) и многих других. Такие файлы обычно содержат векторные шрифты и иллюстрации высокого качества, поэтому они пригодны для печати в любом разрешении, для просмотра на экране и для поиска по тексту книги, включая возможность выделять и копировать куски текста и иллюстрации. Файлы этого вида обычно называют *векторными*. Типичные векторные PDF-файлы имеют размеры около 10—15 килобайт на страницу, в зависимости от числа формул и иллюстраций. В этом случае становится возможен полнотекстовый поиск по книге и индексация больших массивов электронных книг, однако затрудняется воспроизведение оригинальной вёрстки, изображений, схем и формул, практически неизбежными становятся ошибки распознавания. Нынешнее состояние программ оптического распознавания заставляет форматировать всё это вручную и исправлять многочисленные ошибки в распознанном тексте. Поэтому для большинства пе-

чатных книг гораздо легче делать *растровые*, а не векторные электронные версии.

Растровая версия печатной книги представляет собой набор изображений каждой ее страницы. Даже в чисто текстовых книгах – без иллюстраций, таблиц или формул –оптическое распознавание порой даёт трудно выявляемые ошибки. А в растровых книгах полностью сохраняется оригинальная вёрстка и исключаются какие-либо ошибки. Изготовление растровой электронной книги очень дешево, так как основные трудозатраты приходятся на сканирование страниц исходной бумажной книги. Однако невозможен контекстный поиск или извлечение фрагментов текста, например, для цитирования. Еще один недостаток – без специального сжатия растровая книга занимает очень много места. Поэтому в последнее время усиленно ищутся специальные алгоритмы сжатия изображений страниц, которые в основном содержат текст, но могут включать иллюстрации, схемы, формулы. В этом направлении уже достигнуты серьезные результаты. Например, средний размер растровых книг в формате DjVu [1] – 13 КБ на страницу, т.е. примерно столько же, сколько и в векторном варианте.

Есть промежуточный путь. Некоторые программы позволяют делать файлы формата PDF[2], в которых весь плохо распознанный материал содержится в растровом виде, а остальная часть – в векторном. Такие PDF файлы однако сильно проигрывают чисто растровым книгам и по внешнему виду (нестыковка векторных шрифтов и фрагментов изображения страницы), и по размеру файлов. Так что истина не всегда посередине.

Из сказанного можно сделать вывод, что массовый перевод печатной продукции прошлых лет в векторную электронную форму – это слишком дорогой путь, по крайней мере, пока программы оптического распознавания не будут существенно усовершенствованы. Однако ждать этого так же перспективно, как и появления хороших электронных переводчиков с одного языка на другой. Остается единственный путь – улучшение сжатия растровых изображений текста.

В этом направлении сделаны существенные шаги, начиная от уже показавших свою практическую ценность форматов PDF и DjVu и заканчивая алгоритмами [8, 9, 13], находящимися еще в стадии разработки.

Цель настоящей статьи – дать обзор идей и методов, на которых основаны эти алгоритмы, и провести сравнение достигнутых с их помощью результатов по сжатию изображений текста.

Форматы сжатия изображений. Форматы DJVU и PDF

Создано очень много хороших форматов сжатия изображений, каждый из которых обладает своими достоинствами и недостатками. Наиболее популярны форматы JPEG [3] и JPEG2000 [4] в случае, если требуется хорошее сжатие без заметной потери качества изображения. Недаром большинство цифровых фотокамер сохраняют сделанные снимки (в том числе) в одном из этих форматов. Однако, как и все форматы сжатия изображений, основанные на алгоритмах ортогональных преобразований, они абсолютно непригодны для сжатия изображений текста. Дело в том, что избыточность информации, которую устраняют подобного рода алгоритмы, основана на малом градиенте яркости в большинстве точек растрового изображения. При этом гармоники (косинус преобразования в JPEG или вейвлет-гармоники в JPEG2000) быстро убывают с номером, и большое число пикселей исходного изображения после ортогонального преобразования можно, практически не теряя информации, заменить малым числом гармоник. Поэтому хорошо сжимаются размытые цветные или полутоновые изображения, такие как портреты, пейзажи и тому подобное. Но, чем больше мелких и контрастных деталей имеется на изображении, тем хуже сжатие. Бинарное (двуцветное) изображение текста представляет собой крайнюю степень контрастного изображения, изобилующего мелкими деталями (буквами, цифрами, знаками препинания). Поэтому сжатие такого изображения всеми алгоритмами, использующими ортогональные преобразования, является неудовлетворительным.

Однако книжная страница может содержать в качестве иллюстрации размытое изображение, и применение к нему, скажем, JPEG2000 более чем оправдано. Рассмотрим, как это делается в лучшем из имеющихся в настоящее время форматов DjVu [1], который сжимает текст одним алгоритмом, а иллюстрации – другим.

DjVu (с французского— «уже виденное») – технология сжатия изображений с потерями, разработанная специально для хранения сканированных документов: книг, журналов, рукописей и прочего. Формат очень эффективен, если необходимо передать все нюансы оформления книги, например, исторических документов, когда важно не только содержание, но и цвет, фактура бумаги, ее дефекты, следы, оставленные другими предметами и так далее.

DjVu стал основой для многих научных библиотек. Огромное количество книг в этом формате доступно в файлообменных сетях. Имеется несколько общедоступных программ для чтения файлов этого формата.

Для сжатия изображений в DjVu применяется специальная технология, разделяющая исходное изображение на три слоя: передний план, задний план и чёрно-белую (однобитовую) маску. Маска сохраняется с разрешением исходного файла. Именно она содержит изображение текста и прочие чёткие детали. Разрешение заднего плана, в котором остаются иллю-

страции и текстура страницы, по умолчанию понижается для увеличения сжатия. Передний план содержит цветовую информацию о маске, его разрешение обычно понижается ещё сильнее. Затем задний и передний планы сжимаются с помощью вейвлет-преобразования, а маска – алгоритмом JB2 [5, 6].

Этот алгоритм и подобные ему алгоритмы сжатия бинарных изображений текста и являются предметом настоящей работы. Важные, но по сути второстепенные задачи разделения страницы на слои, содержащие текст и изображения, мы оставляем в стороне, так как степень сжатия, если, конечно, книга не состоит из одних иллюстраций, определяется именно применяемым алгоритмом сжатия бинарного (обычно черно-белого) текста.

В формате PDF с этой целью используется алгоритм JBIG2 [5,7].Сравнивая форматы DjVu и PDF по уровню сжатия изображений книжных страниц, в основном нужно сравнивать алгоритмы сжатия JB2 и JBIG2 соответственно.

Изначально для сжатия чёрно-белых изображений существовали форматы Huffman, RLE, CCIT FAX G3 и CCIT FAX G4. В основном они предназначались для увеличение скорости передачи документов по факсу. Потом появился формат JBIG (1993), который был позже переименован в JBIG1. Но он не получил широкого распространения. Вместо него стал использоваться формат JBIG2, предложенный Группой Экспертов в Сжатии Бинарных Изображений (JointBi-level Image Experts Group), который в 2000 году был опубликован как международный стандарт ITU T.88 и в 2001 году как стандарт ISO/IEC 14492.

Начиная с 1996 года, появился формат DjVu, в котором был реализован алгоритм JB2.Этот алгоритм сжатия разработала фирма AT&T (автор JB2 – Пауль Говард) в ответ на создание формата JBIG2.

Как и JBIG2, так и JB2, существуют в двух вариантах: с потерями и без потерь. Нас интересует в основном варианты с потерями, так как они дают значительно большее сжатие при, практически, том же качестве изображения. Варианты без потерь используются сравнительно редко, в специальных целях (астрономических, медицинских, криминалистических и т. п.).

На рис. 1, взятом из работы [5], показаны результаты сравнительного тестирования алгоритмов JB2 и JBIG2.

Дадим некоторые пояснения:

a. DjVu Bitonal – это другое название JB2.

b. Цифры в круглых скобках непосредственно над сплошными и перфорированными столбцами – это степень сжатия.

c. "CCIT F4", "CCIT F7" и "CCIT F10" – это названия стандартных тестовых изображений для сравнения сжатия разных кодировщиков.

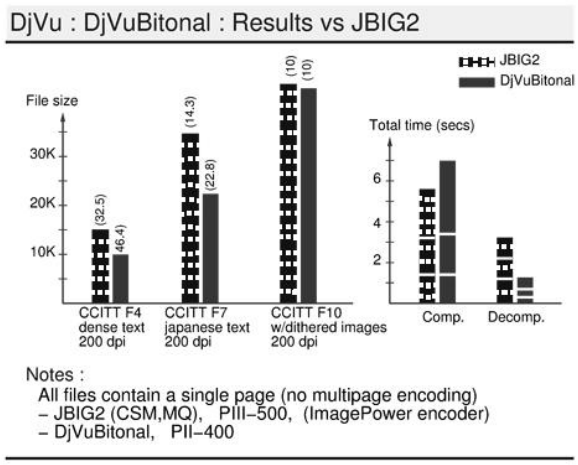

Рис. 1. Результаты сравнительного тестирования алгоритмов JB2 и JBIG2

Тесты показали, что файлы, закодированные алгоритмом JB2, значительно меньше, чем файлы, закодированные JBIG2, на большинстве текстовых изображений, и примерно одинаковы на простых и растрированных чёрно-белых изображениях. Кроме того, кодирование/декодирование у алгоритма JB2 быстрее, чем у алгоритма JBIG2.

Оба алгоритма схожи в следующем: они используют посимвольную сегментацию и словарь разделенных символов. Это означает, что страница текста делится на строки, а те, в свою очередь, на символы (буквы, цифры, знаки препинания и тому подобные). Разделенные символы сохраняются в словаре.

Однако есть существенные различия:

a. JB2 использует алгоритм "soft pattern matching" (см. далее), а JBIG2 – просто "pattern matching". JBIG2 значительно сильнее искажает символы, нежели JB2.

b. JB2 перед кодированием делает сглаживание символов, что даёт выигрыш в сжатии до 10%. Кроме того он удаляет шумовую составляющую изображения (Denoise) для режима максимального сжатия. JBIG2 (вроде бы, точно не известно) ничего этого не делает.

c. JB2 использует словарь разделённых символов, охватывающий группу смежных страниц, а JBIG2 (вроде бы, тоже точно не известно) составляет этот словарь в пределах одной страницы. (Кстати, характеристики сравниваемых алгоритмов, приведенные на рис. 1, относятся к сжатию одной страницы, так что это преимущество JB2 в тестах не отражено).

d. JB2 уступает алгоритму JBIG2 в кодировании полутоновых изображений. Но в формате DjVu такие изображения сжимаются с помощью вейвлет-преобразования, что значительно эффективнее, чем использование JBIG2.

Учитывая сказанное, можно только удивляться следующему. Алгоритм JBIG2 значительно сложнее: его спецификация занимает почти 200 страниц, в то

время как спецификация алгоритма JB2 умещается на 19 страницах вместе с описанием арифметического кодера. Возможно, это вызвано тем, что каждый член Группы Экспертов в Сжатии Бинарных Изображений хотел добавить что-либо своё в JBIG2. Считается, что JBIG2 настолько сложен, что мало какие JBIG2-кодировщики из уже реализованных используют все возможности, указанные в его спецификации.

В то же время алгоритм JB2 прост, изящен и более эффективен. Поэтому новые алгоритмы сжатия бинарных изображений текста имеет смысл сравнивать именно с ним.

Сжатие изображения текста на основе выделения символов и их классификации

Высокие результаты, демонстрируемые алгоритмом JB2, объясняются тем, что он использует классификацию символов. Вообще идея сжатия информации с помощью классификации очень проста и идеально подходит для сжатия изображений текста.

Пусть необходимо сжать некую информацию, которую можно разбить каким-то образом на элементы. Если эти элементы информации объединить в классы так, чтобы в каждом классе находились тождественные (pattern matching) или почти тождественные (soft pattern matching) элементы, то нет нужды хранить все элементы информации – достаточно хранить только по одному элементу из каждого класса. Совокупность таких элементов – представителей классов – называется *словарем*. Кроме того для восстановления информации нужно еще иметь таблицу, называемую «картой размещения классов», которая для каждого класса указывает, в каком месте исходной информации находятся его элементы.

Ясно, что степень сжатия информации с помощью классификации тем выше, чем меньше классов образуется при классификации и чем больше элементов в каждом классе.

В случае сжатия изображения бинарного (далее черно-белого) текста естественным элементом информации является изображение отдельного символа (буквы, цифры, знака препинания и т.п.). Выделение символов не представляет собой особо трудную задачу. Во всех известных алгоритмах, включая алгоритм JB2, символы выделяются как связные области, состоящие из черных точек.

Следует заметить, что при этом некоторые грамматические символы распадаются на части (например, буква «ё» дает три символа), а некоторые (например, сочетания вида “fh”) объединяются в один. Кроме того метод непригоден для текстов с псевдо рукописным шрифтом. Сжатие таких текстов алгоритмом JB2 и другими катастрофически низкое.

Однако не это представляет собой главную трудность при классификации уже разделенных символов.

На рис. 2, взятом из работы [8], представлены три случайно выбранные изображения буквы «n» из различных 257, входящих в изображение страницы текста формата A4, при разрешении сканирования 300 dpi.

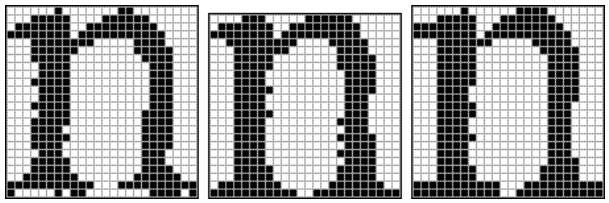

Рис. 2. Влияние шумов печати и сканирования на изображения символа «n»

Легко верится, и это действительно так, что на странице не найдется ни одной пары символов «n», полностью совпадающих друг с другом. То же, за редким исключением, относится и к другим символам, даже точкам. Причиной этого явления являются шумы (то есть случайные искажения), возникающие при печати страницы и ее последующем сканировании. Шумы печати в основном вызваны диффузией краски, жидкой или твердой, вдоль хаотически расположенных капилляров бумаги. Шумы сканирования – несовпадением контуров символа с матрицей сканера, подобно тому, как прямая наклонная линия на экране монитора отображается «ступеньками».

Человеку легко заметить, что все три изображения, приведенные на рис. 2, представляют собой букву «n». Однако пока не существует алгоритма, который мог бы установить тождественность этих символов с той же достоверностью, что и человек. Это и есть главная трудность, не позволяющая разбить изображения символов на классы, так чтобы одновременно выполнялись два условия:

Условие 1. В каждом классе находятся изображения только одного и того же символа;

Условие 2. Все изображения какого-либо символа находятся в одном классе.

Все алгоритмы классификации являются тем или иным компромиссом между этими условиями, причем условие 1 должно выполняться достаточно жестко, иначе в восстановленном тексте будут перепутаны символы, чем иногда грешит алгоритм JB2. Например, иногда путает между собой буквы «b» и «h».

Соблюдение условия 1 влечет за собой ужесточение алгоритма сравнения изображений символов, так что условие 2, практически, невыполнимо. Это приводит к появлению значительно большего числа классов, чем количество символов, изображенных на странице, так как практически все символы дают по нескольку классов своих изображений. Чем больше при классификации образуется классов, тем больше словарь и (логарифмически) больше карта расположения классов. Как следствие, понижается степень сжатия. И хотя алгоритм JB2 и другие используют те

или иные методы дополнительного сжатия словаря и карты, эффективность алгоритма в целом определяется *качеством классификации*, то есть количеством получившихся классов, которое в идеале (условие 2) должно совпадать с количеством символов, присутствующих в тексте, чье изображение сжимается.

В табл. 1 из работы [8] для различных разрешений сканирования показано количество классов, полученных предлагаемым в этой работе алгоритмом классификации (предпоследний столбец) и алгоритмом JB2 (последний столбец). Первый столбец показывает разрешение, использованное при сканировании одной и той же страницы формата A4 с черно-белым текстом, набранным шрифтом Times New Roman, 12 pt. Второй столбец – количество классов при тождественной классификации, то есть классов, состоящих из полностью совпадающих изображений символов (содержание третьего столбца будет обсуждаться позднее). Из таблицы следует несомненное превосходство алгоритма ИЛЛ – будем для краткости называть так алгоритм, предложенный в упомянутой работе [8], – над алгоритмом JB2. Словарь ИЛЛ получается почти в три раза короче, чем словарь JB2. Отсюда вытекает и превосходство в степени сжатия той же страницы, что показывает табл. 2, относящаяся к той же странице, что и табл. 1.

Таблица 1

Количество классов при рассматриваемых методах классификации

Разрешение сканирования (dpi)	Количество классов в исходном изображении	Количество классов после основной классификации $\varepsilon_{opt} = 6\%$	Количество классов после второй классификации $\varepsilon_{opt} = 6\%$	Количество классов после классификации алгоритмом JB2
600 dpi	3558	197	72	314
500 dpi	3557	137	72	259
400 dpi	3557	130	71	199
300 dpi	3545	122	95	235
200 dpi	3890	237	148	451

Таблица 2

Сравнительная степень сжатия изображения текста рассматриваемыми методами

Разрешение сканирования (dpi)	200	300	400	500	600
Исходный размер файла (kb)	505,3	1080,2	2003,9	3111,2	4498,0
Методы	**Размер файла после сжатия (kb) / Коэффициент сжатия**				
JPEG 2000	132,8/3,8	288,6/3,74	532,4/3,76	830,0/3,75	1200,3/3,75
JBIG2	61,4/8,2	96,1/11,2	119,6/16,7	148,9/20,9	178,9/25,1
JB2	9,6/52,6	8,7/124,1	9,9/202,4	11,4/272,9	13,6/330,7
ИЛЛ	8,1/62,3	8,0/135,0	8,0/250,4	8,8/353,5	10,3/436,7

Не слишком существенное различие между коэффициентами сжатия, продемонстрированными алгоритмами ИЛЛ и JB2, объясняется тем, что авторы алгоритма ИЛЛ интересовались только классификацией выделенных изображений символов и не оптимизировали алгоритм дополнительного сжатия словаря и карты размещения классов (использовался универсальный алгоритм без потерь 7z).

Кроме того таблица 2 показывает, что применение лучшего для сжатия размытых изображений алгоритма JPEG 2000 мало что дает при сжатии изображения черно-белого текста без иллюстраций.

Ниже будет рассматриваться еще один очень интересный алгоритм, предложенный И. Межировым [9], несмотря на то, что полученные им результаты более чем скромны: качество классификации в среднем составляет 5–10 изображений символов на класс, а файл сжимается на 50-60%%. Но нужно помнить, что в 2003 году, когда писалась эта работа, была доступна только демонстрационная версия JB2 из пакета DjVu Libre, которая давала, как указано в работе, сжатие на 25-35%%. И хотя можно предполагать, что сжималась изображение стандартной страницы текста, явно это в работе не указано. И, скорее всего, не использовалось дополнительное сжатие словаря и карты расположения классов.

Алгоритмы классификации разделенных символов

Чтобы разделить имеющуюся совокупность изображений символов на классы нужно:

1) выбрать какую-либо *меру отличий* изображений двух символов, позволяющую, если она достаточно мала, утверждать, что изображения представляют собой один и тот же символ(soft matching), и их можно отнести к одному и тому же классу;

2) выбрать какой-нибудь алгоритм разбиения на классы.

Второй пункт не очень существенен. В алгоритме ИЛЛ, например, используется известный алгоритм «просеивания» [10]. Он состоит в том, что к произвольному еще не классифицированному элементу присоединяются те также неклассифицированные элементы, которые согласно мере отличия из пункта 1 достаточно близки к нему. Первый элемент и все к нему присоединенные образуют класс. Алгоритм заканчивает работу, когда все элементы оказываются классифицированными.

Алгоритм Межирова использует процедуру близкую к известному алгоритму «выращивания областей» [10]: новый неклассифицированный элемент присоединяется к классу, если согласно мере отличий из первого пункта он достаточно близок хотя бы к одному из элементов этого класса. Классификация заканчивается, когда не остается ни одного неклассифицированного элемента.

Так как сравнение двух изображений сравнительно трудоемкий процесс, то для сокращения времени работы последнего алгоритма, он дополняется следующим положением. Если при проверке принадлежности элемента к классу встречается элемент класса достаточно отличающийся от проверяемого, то проверяемый элемент сразу же отвергается.

Перейдем к первому пункту – описанию мер отличия. Это – наиболее важная часть классификации, определяющая ее качество.

Мера отличия – это какая-либо функция, определенная для каждой пары элементов S_1 и S_2 классифицируемого множества. Для адекватно подобранной меры отличия ε существует два порога – ε_{min} и ε_{max}, такие что при $\varepsilon(S_1,S_2) < \varepsilon_{min}$ элементы S_1 и S_2 совпадают, а при $\varepsilon(S_1,S_2).\varepsilon_{max}$ – различны. Когда значения меры различия ε лежат между ε_{min} и ε_{max} в так называемом *интервале неопределенности*, то, возможно, элементы совпадают, а, возможно, – нет.

Для принятия решения, совпадают или не совпадают два элемента, выбирают (как правило, экспериментально) значение ε_{opt} из интервала неопределенности: $\varepsilon_{min} \le \varepsilon_{opt} \le \varepsilon_{max}$, и полагают, что при $\varepsilon(S_1,S_2) < \varepsilon_{opt}$ элементы S_1 и S_2 совпадают, а в противном случае – нет.

Если положить $\varepsilon_{opt} = \varepsilon_{min}$, то получим классификацию, удовлетворяющую условию 1 из предыдущего раздела. При $\varepsilon_{opt} = \varepsilon_{max}$, классификация будет соответственно удовлетворять условию 2. Таким образом мера различия, для которой $\varepsilon_{min} = \varepsilon_{max}$, дает идеальную классификацию, одновременно удовлетворяющую обоим условиям 1 и 2. Но такие меры различия пока не найдены. Так что выбор порога ε_{opt} из интервала неопределенности является нахождением разумного компромисса между желаниями удовлетворить и условию 1, и условию 2.

По имеющимся данным [9] (возможно, эти сведения устарели) алгоритм JB2 в качестве меры отличия использует отношение несовпадающих пикселей двух изображений символов к их общему числу (в одном изображении) после наиболее удачного наложения друг на друга, осуществляемого горизонтальными и вертикальными смещениями. Экспериментально подобранный порог $\varepsilon_{opt} = 6\%$.

Эта мера отличия очень хороша, если искажения распределены более или менее случайно по всей площади изображения символа. Однако, как обратил внимание И. Межиров [9],искажения, вызванные шумами печати и сканирования, являются *контурными*, то есть возникают только на границе между черными и белыми областями. Это хорошо видно, например, на рис. 2. Поэтому при сравнении двух изображений символов несовпадение точек на их границах несущественно. Вопрос состоит в том, как выяснить, является ли данная точка граничной. В алгоритме Межирова для этого используется один из вариантов «метода скелетизации» [11] изображения символов. Он состоит в следующем.

Определяется некоторое преобразование черно-белого изображения — «очистка». В ходе очистки белые пиксели остаются белыми, черные в некоторых случаях заменяются белыми. После нескольких последовательных очисток изображение перестает меняться. Черные пиксели, пережившие все очистки (то есть образующие «скелет»), объявляются наиболее важными. Важность остальных черных пикселей уменьшается в геометрической прогрессии по мере того, на сколько очисток меньше они выдержали.

Каждая очистка выполняется в два прохода. При каждом проходе последовательно просматриваются все пиксели изображения. Во время первого прохода некоторые черные пиксели объявляются кандидатами на удаление. Во время второго прохода некоторые кандидаты могут быть удалены, то есть перекрашены в белый.

Во время первого прохода пиксель объявляется кандидатом на удаление, если он и его восемь соседей *не* раскрашены ни одним из следующих пяти способов (с точностью до поворотов и отражений), показанных на рис. 3.

Знаки вопроса означают, что пиксель может быть любого цвета. Таким образом, картинка с четырьмя знаками вопроса эквивалентна шестнадцати отдельным раскраскам. В случае, если пиксель лежит на границе изображения, его соседи, лежащие за пределами изображения, считаются белыми.

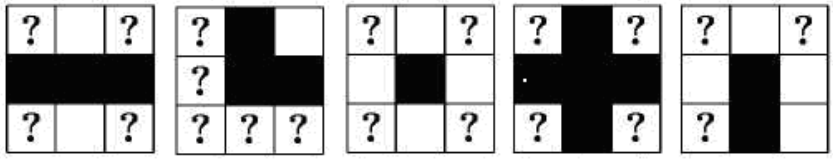

Рис. 3. Определение кандидата на удаление при скелетизации по И. Межирову

Первые два правила, представленных графически на рис. 4, означают, что связность буквы относительно переходов от пикселя к одному из его четырех соседей не должна нарушаться. Третье правило запрещает удаление одиночных пикселей. Четвертое правило означает, что только пиксели с границы черного и белого могут быть удалены. Пятое правило помогает сохранить форму буквы при очистке, запрещая удалять «выступающие» пиксели.

Во время второго прохода кандидаты на удаление перекрашиваются в белый цвет, если эти пиксели и их восемь соседей не раскрашены одним из первых четырех способов, показанных выше Правило сохранения формы теряет смысл, если часть изображения уже очищена, а часть – нет.

И. Межиров предложил следующее семейство меротличия, основанных на этом алгоритме. Изображения символов налагаются друг на друга, после чего мера отличия подсчитывается как сумма штрафных очков, начисляемых каждой точке отличия с учетом ее важности. Максимальный уровень важности оценивается в 1, а все следующие – в геометрической прогрессии с показателем $q<1$. Для обезразмеривания полученная сумма делится на площадь изображения (считая и белые, и черные пиксели). Численные значения порогов для них были подобраны экспериментально.

В работе [9] использовались две меры из этого семейства.

В первой мере отличия показатель q геометрической прогрессии, по которой убывает уровень важности черных пикселей, равен 0, то есть учитываются только самые важные пиксели, составляющие «скелет» символа. Экспериментально найдены пороги $\varepsilon_{min}=2.1\%$, ниже которого изображения признаются одинаковыми, и $\varepsilon_{max}=5\%$, выше которого –заведомо разными.

Во второй мере отличия показатель q геометрической прогрессии равен 0,85. Соответственно найдены пороги $\varepsilon_{min}=3.1\%$ и $\varepsilon_{max}=7.8\%$.

Алгоритм ИЛЛ также учитывает контурный характер шумов печати и сканирования, искажающих изображение символа. Мера отличия вычисляется следующим образом после наложения изображений символов друг на друга так, чтобы их «центры тяжести», подсчитанные по черным точкам, совпадали.

Пусть S_1 и S_2 – сравниваемые изображения. Подсчитываются две величины: $R(S_1,S_2)$ – количество «существенных отличий», и $D(S_1,S_2)$ – количество общих черных точек.

Первая величина – это количество несовпадающих точек, которые не являются смежными для совокупности общих черных точек. Таким образом, количество существенных отличий $R(S_1,S_2)$ игнорирует несовпадения в тех точках, которые лежат на периметрах изображений и, как правило, представляют собою шумы печати и сканирования. Например, при сравнении изображений букв «c» и «e» точки существенных отличий составляют горизонтальный отрезок, который имеется в букве «e» и отсутствует в букве «c».

Мера отличия ε изображений S_1 и S_2 определяется как отношение

$$\varepsilon(S_1,S_2)=\frac{R(S_1,S_2)}{D(S_1,S_2)}100\%,$$

где знаменатель нужен для обезразмеривания, чтобы значение порогов не менялось при изменении размера шрифта и разрешения сканирования.

На рис. 4 из работы [8] показаны, какие совокупности точек рассматриваются при сравнении двух трудно различимых букв «b» и «h». Справа показаны

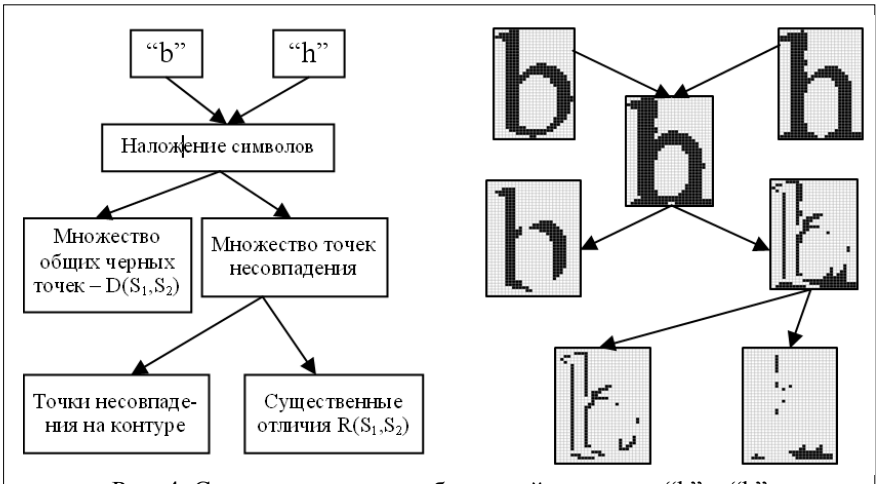

Рис. 4. Схема сравнения изображений символов “b” и “h”

их геометрические расположения, а слева – поясняющая схема.

Из рис. 4 видно, что среди точек существенного отличия имеются одиночные точки и малые группы точек, которые носят случайный характер, вызванный контурными шумами печати и сканирования. Действительные отличия в начертании букв «b» и «h» демонстрируют большие группы точек. Поэтому, вместо определенной выше функции $R(S_1,S_2)$, подсчитывающей количество «существенных отличий», в алгоритме ИЛЛ на самом деле используется ее модификация, которая подсчитывает это число с учетом веса каждой точки. Весовой коэффициент точки в $R(S_1,S_2)$ тем больше, чем больше у данной точки таких же смежных точек. Эта функция, будем по-прежнему обозначать ее через $R(S_1,S_2)$, дает классификацию с меньшим числом классов, чем первоначальный ее вариант без учета весов.

Таким образом, описанная выше мера отличия ε, мало чувствительна к шумам печати и сканирования. Она основана на функциях $R(S_1,S_2)$ и $D(S_1,S_2)$, которые почти не зависят от контурных шумов сравниваемых символов. Пороговое значение $\varepsilon_{opt} = 6\%$, до которого изображения считается изображениями одного и того же символа подобрано экспериментально.

Список литературы

1. Википедия. DjVu [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/DjVu.

2. Википедия [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Portable_Document_Format.

3. Википедия. JPEG [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/JPEG.

4. Википедия. JPEG_2000 [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/JPEG_2000.

5. Описание DjVu [Электронный ресурс]. – Режим доступа: http://www.djvu-soft.narod.ru/scan/bookscan_pdf.htm;

6. Спецификация DjVu [Электронный ресурс]. – Режим доступа: http://djvu.cvs.sourceforge.net/djvu/djvulibre-3.5/doc/djvu2spec.djvu;

7. Википедия. JBIG2 [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/JBIG2.

8. Иванов В.Г. Сжатие изображения текста на основе выделения символов и их классификации / В.Г. Иванов, М.Г. Любарский, Ю.В. Ломоносов // К.: Международный научно-технический журнал «Проблемы управления и информатики», 2010, №6. С. 111 – 122.

9. Межиров И. Курсовая работа на тему «Алгоритмы сжатия данных» / И. Межиров. – М., МГУ им. Ломоносова, механико-математический ф-т, научный руководитель А. Шень, 2004.

10. Прикладная статистика: Классификация и снижение размерности: [Справочник] / С.А. Айвазян, В.М. Бухштабер, И.С. Енюков и др.; Под ред. С.А. Айвазяна. – М.: Финансы и статистика, 1989. – 607с.;

11. Автоматический анализ сложных изображений [Сборник переводов] / Под ред. Э.М. Бравермана – М.: Издательство Мир, 1969. – 308 с.;

12. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В.Юкин. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с.

13. Иванов В.Г.. Сжатие изображения текста на основе формирования и классификации вертикальных элементов строки в графическом словаре символьных данных / В.Г. Иванов, М.Г. Любарский, Ю.В. Ломоносов // Проблемы управления и информатики. – К., 2011. – № 5. – С. 98-109.

Поступила в редколлегию 24.01.2013

Рецензент: д-р техн. наук, проф. А.С. Куценко, Национальный технический университет "ХПИ", Харьков.

КЛАСИФІКАЦІЙНІ МЕТОДИ СТИСНЕННЯ ЗОБРАЖЕНЬ ОЦИФРОВАНОГО ТЕКСТУ

В.Г. Іванов, Ю.В. Ломоносов, М.Г. Любарський

У роботі розглядаються методи класифікації, вживані при стисненні файлу з бітональним зображенням тексту, отриманим скануванням або цифровим фотографуванням. Особлива увага звертається на використовувані при цьому міри відмінності двох зображень символів, виділених із зображення тексту. Ці міри відмінності дозволяють з тим або іншим ступенем упевненості вважати символи на порівнюваних зображеннях або співпадаючими, або різними. Для відомих на сьогоднішній день алгоритмів класифікації, включаючи добре відомий алгоритм JB2, приведені кількісні характеристики класифікації – число класів, що отримуються цими алгоритмами для зображення стандартної сторінки тексту. Чим менше це число, тим якість класифікації вважається вище, оскільки дає краще стиснення файлу із зображенням тексту. Розглянуті також методи прискорення алгоритмів, що класифікують зображення символів, і підвищення легкості для читання відновленого після стиснення зображення тексту.

Ключові слова: *зображення тексту, методи класифікації, стиснення даних.*

CLASSIFICATION METHODS OF COMPRESSION OF IMAGES OF THE DIGITISED TEXT

V.G. Ivanov, Y.V. Lomonosov, M.G. Lyubarskiy

Methods are in-process examined classifications, applied at the compression of file with the bitonality image of phototypograph got scan-out or digital photographing. The special attention applies on the in-use here measures of distinction of two images of characters, abstracted from the image of text. These measures of distinction allow with one or another degree of confidence to count characters on the compared images or consilient, or different. For the algorithms of classification known to date, including the known algorithm of JB2 well, quantitative descriptions of classification – number of classes, got these algorithms for the image of standard page of text are resulted. What less than it is a number, quality of classification is considered that higher, because gives the best compression of file with the image of text. The methods of acceleration of algorithms, classifying the images of characters, and increases of easy-to-readness of the text recovered after the compression of image are considered also.

Keywords: *image of text, methods of classification, compression of data.*