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 II

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.*

Часть I

Системи обробки інформації.
2013. – Вип. 2(109). – С. 36-45.

Часть II

Дополнительные меры повышения качества классификации и удобочитаемости текста

Своей высокой эффективностью алгоритм ИЛЛ обязан не только удачно выбранной и легко вычисляемой мере отличия, но и в большей степени тому, что он использует процедуру *статистического усреднения*. Это означает, что после проведения классификации с помощью описанной выше меры отличия для каждого класса находится усредненное изображение. Процедура усреднения состоит в наложении друг на друга всех изображений класса, совмещающем их «центры тяжести», и вычисления среднего значения яркости для каждой точки с последующим округлением. На рис. 6 иллюстрируется процесс усреднения для трех классов изображений одного и того же символа «n». Черные точки означают, что среднее значение яркости в них равно 0, серые – что среднее значение меньше или равно ½, а светлые – больше ½.

При округлении черные и серые точки превращаются в черные, а светлые – в белые. Результат показан на рис. 7.

Совокупность полученных средних изображений в алгоритме ИЛЛ используется как промежуточный словарь, то есть все исходные изображения символов более не используются. Легко видеть (см. рис. 7), что благодаря процедуре усреднения искажения, вызванные шумами печати и сканирования, заметно меньше, чем в исходных изображениях (см. рис. 2,, часть I).

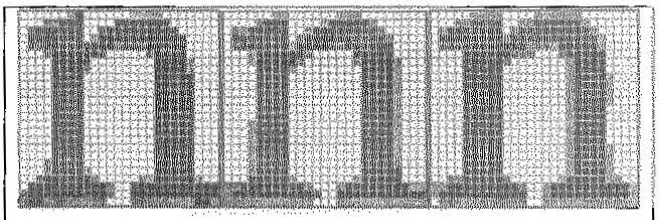


Рис. 6. Усреднение изображений в трех различных классах символа "n"

Однако в полученном промежуточном словаре усредненных изображений по-прежнему присутствуют разные изображения одних и тех же символов, хотя и в значительно меньшем количестве, чем до классификации. Их количество для тестовой страницы указано в третьем столбце табл. 1 (часть I).

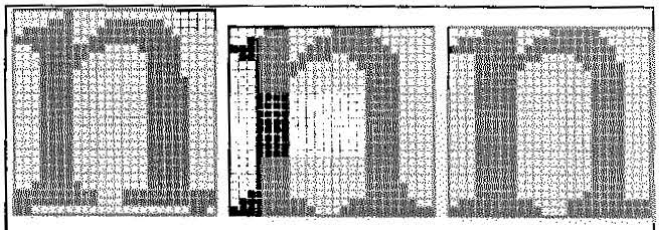


Рис. 7. Изображения средних в трех классах изображений символа "n"

Качество классификации можно улучшить, применив еще одну классификацию, призванную объе-

динить в один класс изображения одного и того же символа. Для этой классификации, как и для предыдущей, используется нечувствительная к контурным шумам мера отличия ε с тем же порогом ε_{opt}, но алгоритм классификации выбирается другим.

Дело в том, что теперь разные изображения одного того же символа очень близки друг к другу – случайная компонента изображений в значительной мере подавлена. Поэтому существенно меньше опасность спутать изображения двух разных символов. Это позволяет применить алгоритм «наращивания классов», близкий к используемому в алгоритме Межирова.

После получения новых классов, изображения в каждом из них снова усредняются, и получившийся набор изображений представляет собой окончательный словарь. На рис. 8 показано теперь единственное изображение символа «n», вошедшее в словарь. Эффективность повторной классификации можно проследить, сравнивая третий и четвертый столбцы табл. 1.

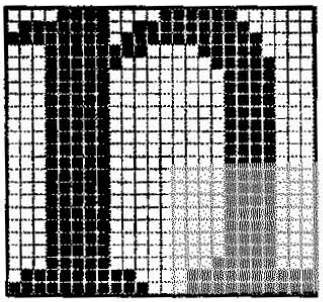


Рис. 8. Представитель класса изображений буквы «n», после повторных классификации и усреднения, который попадает в окончательный словарь

Сравнивая рис. 8 с рис. 2 (часть I), можно утверждать, что контурные шумы практически отсутствуют, и внешний вид символа значительно улучшился, по сравнению с вариантами его изображения в исходном тексте. Таким образом, качество восстановленного изображения текста в алгоритме ИЛЛ значительно выше исходного. В принципе так и должно быть, потому что значительная часть избыточной информация, которую устраняют алгоритмы сжатия изображений текста, состоит из случайных шумов печати и сканирования.

Методы ускорения классификации разделенных символов

Так как вычисление эффективной меры отличия для двух изображений символов всегда достаточно трудоемко и занимает много времени, то для ускорения кодирования очень полезна дополнительная простая быстро вычислимая мера отличия, которая позволяет для многих пар изображений быстро определить, что они представляют разные символы. Говоря формально, эта мера должна жестко удовлетворять условию 2 из пункта 3, для чего в качестве порога ε_{opt} выбирается порог ε_{max} (см. часть I раздел 4).

С помощью дополнительной меры отличия можно провести предварительную классификацию (алгоритм ИЛЛ), разделяющую всю совокупность изображений символов на «грубые» классы, а основной мерой отличия пользоваться только в пределах каждого такого класса. При этом количество сравнений существенно уменьшается.

Можно то же самое сделать неявным образом (алгоритм Межирова): перед вычислением основной меры отличия вычисляется дополнительная мера и в случае отрицательного ответа основная мера не вычисляется.

Рассмотрим дополнительную меру отличия, предложенную И. Межировым. Алгоритм генерирует для каждого выделенного изображения символа короткое слово – «подпись». Подпись может иметь любой наперед заданный размер. И. Межиров использует подпись длиной в 31 байт.

Подписи интерпретируются как точки в многомерном евклидовом пространстве, так что между ними определено расстояние. Это и есть предварительная мера различия: абсолютно одинаковые изображения попадают в одну точку, похожие изображения будут в той или иной степени близки, а если, что наиболее важно, расстояние слишком большое, то изображаемые символы можно с уверенностью считать различными.

Подписи каждого изображения символа вычисляются заранее один раз. Поэтому сравнение двух изображений сводится к вычислению расстояния между двумя векторами, что требует малых временных затрат.

Алгоритм построен по аналогии с двумерным вейвлет-преобразованием и работает следующим образом. Прямоугольник, в который вписано изображение символа, разрезается на две части по горизонтали так, чтобы число черных пикселей по обе стороны разреза было примерно равным. Затем каждый из двух получившихся прямоугольников разрезается надвое вертикальным разрезом по тому же правилу: число пикселей по разные стороны разреза должно быть примерно равным. Если какой-либо прямоугольник не содержит черных точек, то он делится посередине. Затем каждый из четырех прямоугольников разрезается по горизонтали и так далее, горизонтальные и вертикальные разрезы чередуются.

Положению каждого разреза соответствует число от 0 до 1: 0 соответствует крайнему левому или крайнему верхнему положению, 1 – крайнему правому или крайнему нижнему. Каждый разрез порождает два прямоугольника, которые тоже могут быть разрезаны. Таким образом, получается двоичное дерево, в каждой вершине которого стоит число

от 0 до 1. Для хранения в байте эти числа нормируются округляются.

В программе при подсчете их разности элементы подписи дополнительно умножаются на коэффициент, в геометрической прогрессии зависящий от уровня разреза. Подобранные экспериментально установленные пороги таковы: $\varepsilon_{max} = 170$ при показателе прогрессии $q = 0.9$, $\varepsilon_{max} = 250$ при $q = 1.15$ и $\varepsilon_{max} = 215$ при $q = 0$.

В алгоритме ИЛЛ предварительная классификация проводится методом «просеивания», уже описанном в разделе 4, также с помощью очень быстро вычисляемой дополнительной меры различия. Изображению каждого символа сопоставляется трехмерный вектор, и мера различия между двумя изображениями символов – расстояние между соответствующими векторами, нормированными так, чтобы порог не зависел от размера шрифта и разрешения сканирования.

Вектор параметров каждого изображения символа, на сравнении которых проводится классификация, состоит из следующих величин: H – высота символа, то есть высота прямоугольника, в который вписано изображение символа; W – аналогично определенная ширина символа; P – периметр изображения символа, подсчитанный в пикселях вдоль внешних и внутренних его границ.

При сравнении размеров двух символов вычисляется разность их физических размеров, например, для высоты – величина

$$\Delta H = |H_1 - H_2| \frac{100}{res(dpi)},$$

где H_1, H_2 – высоты первого и второго сравниваемых символов в пикселях, и ras(dpi) – разрешение сканирования, измеряемое в точках (пикселях) на дюйм.

Таким образом, разность высот выражается в сотых частях дюйма. Аналогично вычисляется разность ширин ΔW сравниваемых символов.

Переход к физическим линейным размерам оправдан тем, что на отклонения высоты и ширины изображений одного и того же символа в основном влияют шумы печати, а не сканирования. Последние могут изменить линейный размер не более, чем на 1 пиксель. Экспериментально установлено, что отклонение в линейных размерах, вызванное расплыванием краски, находится в пределах 0.01 дюйма. Поэтому линейные размеры символов вне зависимости от разрешения и размера шрифта можно считать значительно отличающимися, если $\Delta H > 1$ и $\Delta W > 1$.

Длина периметра, определяемая как число граничных пикселей в изображении символа, мало искажается шумами печати и сканирования. В основном она выражает индивидуальные особенности символов, то есть гарантированно различает, например, такие буквы, как «n» и «r». Поэтому для их сравнения вводится в рассмотрение безразмерная, то есть не зависящая ни от разрешения сканирования, ни от размера шрифта величина

$$\Delta P = \frac{|P_1 - P_2|}{\sqrt{P_1 P_2}} 100\%,$$

где P_1 и P_2 – периметры сравниваемых символов.

Периметры символов считаются далекими, если $\Delta P > 10\%$. Если этому сопутствует значительное различие в линейных размерах: $\Delta H > 1$ и $\Delta W > 1$, то можно с уверенностью утверждать, что сравниваемые изображения относятся к разным символам. Такая классификация, с одной стороны, имеет небольшое число классов, каждый из которых содержит изображения близких по начертанию символов. С другой стороны, выполнено условие 2 (см. раздел 4), то есть все изображения одного и того же символа попадают в один и тот же класс.

Дополнительное сжатие словаря и карты расположения классов

Размер графического файла, который сжат каким-либо алгоритмом, основанным на классификации, грубо говоря, состоит из размера словаря, и размера соответствующей карты расположения классов. Этот размер можно уменьшить, если дополнительно сжать каким-либо стандартным алгоритмом сжатия без потерь, например, алгоритмами RLE, Хаффмана, LZW и другими [12]. Или их комбинациями. И хотя при создании стандартного формата сжатия изображений текста на эти вопросы требуется обратить самое серьезное внимание, здесь мы на этом останавливаться не будем. Отметим только, что размер словаря во много раз превышает размер отвечающей ему карты (для одной страницы текста) и представляет собой набор изображений символов, для сжатия которых методы компрессии без потерь не являются самыми эффективными.

Рассмотрим предложенный в работе [13] алгоритм – будем называть его алгоритмом ИЛЛ2 – который сжимает словарь, рассматривая его не как последовательность бит, а как изображение, составленное из отдельных изображений входящих в него символов. Идея этого алгоритма состоит в следующем.

Если представить себе строку в изображении текста как прямоугольник с горизонтальным основанием, охватывающий строку, то *вертикальным элементом строки* называется пересечение этого прямоугольника с любой вертикальной линией шириной в один пиксель. На рис. 9 из работы [13] показано разбиение изображения буквы «е» на вертикальные элементы строки. Таким образом, все символы в строке, включая пробелы, можно представить как объединение ее вертикальных элементов одного и того же размера. Алгоритм ИЛЛ2 рассматривает словарь как изображение строки, составлен-

ной из изображений всех входящих в него символов. (Фрагмент такого словаря показан на рис. 10).

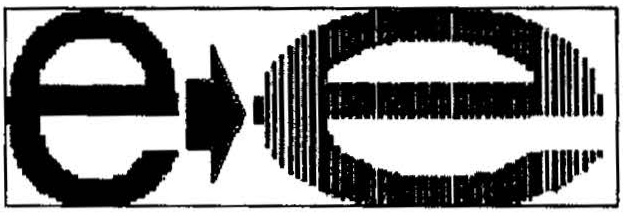


Рис. 9. Изображение буквы «е» и составляющие его вертикальные элементы строки

Рис. 10. Изображение фрагмента словаря для изображения текста с разрешением 300 dpi: а – словарь до сжатия; б – восстановленный словарь

Эта строка разбивается на вертикальные элементы, совокупность которых классифицируется способом, описанным ниже. В результате получается новый словарь, состоящий из вертикальных элементов строки, и карта размещения соответствующих ему классов. Смысл этих процедур заключается в том, что новый словарь занимает намного меньше места, чем старый, а дополнительная карта размещения классов невелика по размеру.

На рис. 10, б показан словарь символов, восстановленный с помощью словаря вертикальных элементов строки и соответствующей ему карты расположения классов. Некоторое незначительное ухудшение качества заметно только при большом увеличении, которое используется на этом рисунке.

Таким образом, алгоритм ИЛЛ2 работает в два этапа, как это схематически изображено на рис. 11.

Рис. 11. Схема двухэтапной обработки изображения текста алгоритмом ИЛЛ2

Фрагмент словаря вертикальных элементов строки для изображения текста с разрешением 300 dpi представлен на рис. 12 (серые полосы введены для наглядности как разделители между представителями отдельных классов вертикальных элементов строки).

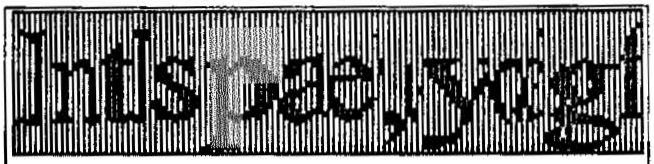


Рис. 12. Фрагмент словаря вертикальных элементов строки, полученный с помощью автоматической классификации

Перейдем к описанию алгоритма классификации вертикальных элементов строки, который базируется на автоматической классификации (алгоритме k-средних [10]). Вертикальные элементы строки, совокупность которых обозначим через X, рассматриваются как векторы с бинарными координатами (0 – черный цвет, 1 – белый). Единственным параметром классификации является k – число классов.

Некоторым образом из множества X выбираются элементов $e_1,...e_k$ – *центров класса* (в первом приближении). Все элементы изображения разбиваются на k классов $S_1,S_2,...S_k$ по правилу: для каждого элемента x находится ближайший в евклидовой метрике центр класса, после чего в один класс объединяются все элементы, имеющие один и тот же ближайший центр класса. Далее в каждом классе находятся новые центры класса (во втором приближении) путем усреднения элементов в каждом классе:

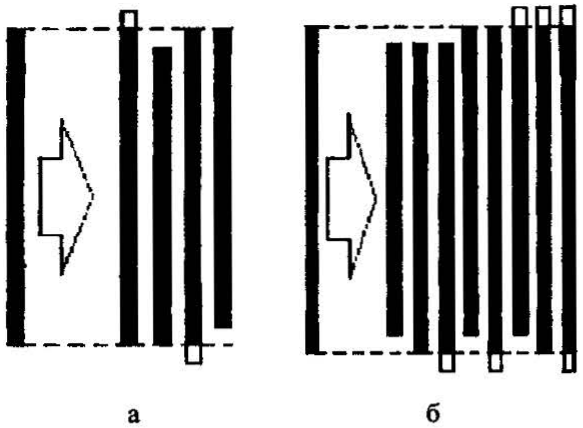
$$e_i = \frac{1}{N_i} \sum_{x \in S_i} x,$$

где N_i – число элементов в классе S_i, и $i = 1,2,...,k$, и последующим округлением до 0 или 1).

После этого процедура повторяется, исходя из новых центров классов. Алгоритм заканчивает работу, когда центры классов перестают изменяться. Их совокупность – словарь вертикальных элементов строки.

Автоматическая классификация – очень быстрый алгоритм, дающий классификацию высокого качества, если правильно выбраны число классов k и набор центров первого приближения. В алгоритме ИЛЛ2 и то, и другое определяется с помощью алгоритма просеивания, описанного в разделе 4, с мерой отличия, учитывающей контурный характер искажений, вносимых шумами печати и сканирования.

Эта мера отличия ε двух вертикальных элементов строки равна бесконечности во всех случаях кроме тех, когда каждой связной компоненте множества черных пикселей одного из них взаимно однозначно отвечает компонента другого так, что эти компоненты могут отличаться на один пиксель только на своих концах.



а б

Рис. 13. Возможные варианты близких связных компонент вертикальных элементов строки: а – $\varepsilon = 1$; б – $\varepsilon = 2$

Возможные варианты показаны на рис. 13. Во всех этих случая мера отличия ε равна максимальному по каждой компоненте числу несовпадений. Иначе говоря, $\varepsilon = 0$, если вертикальные элементы строки полностью совпадают, $\varepsilon = 1$, если все пары отвечающих друг другу компонент совпадают кроме некоторой, отличающихся на один пиксель на одном из концов, верхнем или нижнем, и $\varepsilon = 2$, если найдутся пары компонент, отличающихся на один пиксель и на нижнем, и на верхнем концах. Во всех остальных случаях $\varepsilon_{opt} = \infty$.

Выбор такой меры отличия объясняется тем, что контурные шумы печати и сканирования могут, как правило, изменить черную компоненту вертикального элемента строки в ту или иную сторону лишь на один приграничный пиксель.

После разбиения всей совокупности вертикальных элементов на классы методом просеивания с порогом $\varepsilon_{opt} = \varepsilon_{max} = 2$ в качестве центров нулевого приближения для алгоритма автоматической классификации выбираются средние в каждом из полученных классов за исключением тех классов, которые содержат 1 – 2 элемента.

Последнее объясняется тем, что шумы печати и сканирования порождают новые, как правило, многокомпонентные вертикальные элементы, смежные с вертикальными границами символов, а также сильно искажают соседние элементы, принадлежащие изображению символа (см., например, рис. 2.) Такие элементы при просеивании дают классы, состоящие из одного-двух элементов (для стандартной страницы текста) и не выбираются в качестве центров класса. При автоматической классификации благодаря проводимому на каждой итерации усреднению, они не оказывают никакого влияния на конечный результат. Это обстоятельство оправдывает выбор автоматической классификации.

Отметим также, что предварительное применение алгоритма просеивания значительно улучшает сходимость автоматической классификации по сравнению со случайным выбором центров первого приближения и, следовательно, сокращает вычислительное время. Это объясняется тем, что уже на первой итерации центры нулевого приближения хорошо аппроксимируют члены своего класса с точностью, не меньшей, чем $\sqrt{2n}$, где n – число черных компонент.

Размер файла, который сжат каким-либо алгоритмом, основанным на классификации, в основном состоит из размера словаря и размера карты размещения классов. Причем при классификации выделенных изображений символов размер словаря весьма значителен. Поэтому предложенный в рассматриваемой работе алгоритм автоматической классификации вертикальных элементов строки, уменьшающий размер словаря, обеспечивает более высокую степень компрессии исходного изображения текста.

В табл. 3 для тестовой страницы А4, шрифт Times New Roman 12 pt представлены объемы словаря символов, словаря вертикальных элементов строки, а также общего объема словаря вертикальных элементов строки и соответствующей ему карты расположения классов. Параллельно приведены те же характеристики после дополнительного сжатия без потерь с использованием алгоритма 7z, который является модификацией словарного метода компрессии LZ77– LZMA [12].

Таблица 3

Объем словаря символов

Разрешение сканирования (dpi)	200	300	400	500	600
Размер словаря символов/ то же после дополнительного сжатия(kb)	34,8/4,4	46,4/5,0	67,7/4,7	94,6/5,4	140,2/6,7
Размер словаря вертикальных элементов строки / то же после дополнительного сжатия (kb)	1,2/1,0	2,5/1,5	4,4/2,1	7,3/3,0	10,8/3,8
Словарь вертикальных элементов строки + карта размещения их классов / после дополнительного сжатия (kb)	7,3/3,0	8,7/3,4	10,1/4,2	13,7/5,2	18,8/6,6

Отметим, что полученный на втором этапе алгоритма ИЛЛ2 словарь вертикальных элементов строки в 20 – 30 раз меньше словаря символов, полученного на первом этапе. Однако после дополнительного сжа-

тия результат не такой впечатляющий – в 2 – 4 раза (хотя это тоже очень хороший результат). Это объясняется тем, что словарь вертикальных элементов практически не содержит избыточную информацию, так что «дожимать» по сравнению с исходным словарем символов фактически нечего.

Карта расположения классов вертикальных элементов строки, по сути, аналогична обычному тексту, скажем, в ASCII кодировке, и при дополнительном сжатии уменьшается всего в 2 – 3 раза. Это в значительной мере уменьшает эффективность второго этапа алгоритма ИЛЛ2.

Как показывает табл. 4, выигрыш в сжатии, полученный на втором этапе, существенен при разрешениях 200 и 300 dpi, что является неплохим результатом, так как эти разрешения наиболее популярны. При 400 dpi выигрыш не столь заметен и практически отсутствует при больших разрешениях. Это объясняется тем, что с повышением разрешения увеличивается число вертикальных элементов строки на каждый символ, и карта распюложения классов вертикальных элементов строки возрастает. Так как она сжимается слабо, то суммарный объем словаря вертикальных элементов строки и соответствующей ему карты с ростом разрешения сжимается все хуже и хуже.

В табл. 5 приведены значения коэффициентов сжатия изображения стандартной страницы текста алгоритмом ИЛЛ (или, что то же самое, после 1-го этапа алгоритма ИЛЛ2), алгоритмом ИЛЛ2 и алгоритмом JB, а также преимущество в сжатии алгоритма ИЛЛ2 над алгоритмами ИЛЛ и JB2 в процентном соотношении.

Таблица 4

Выигрыш в сжатии изображения текста
в результате компрессии словаря символов на втором этапе обработки

Разрешение изображения текста (dpi)	200	300	400	500	600
Коэффициент сжатия словаря символов (ИЛЛ = 1-й этап алгоритма ИЛЛ2)	7,9	9,28	14,4	17,51	20,92
Коэффициент сжатия словаря вертикальных элементов вместе с соответствующей ему картой размещения классов (II этап алгоритма ИЛЛ2)	11,6	13,64	16,11	18,19	21,24

Таблица 5

Выигрыш в степени сжатия
двухэтапного алгоритма в сравнении с алгоритмом JB2 (DjVu)

Разрешение изображения текста (dpi)	200	300	400	500	600
Алгоритм ИЛЛ (или 1-й этап алгоритма ИЛЛ2)	62,38	135	250,48	353,54	436,7
Алгоритм ИЛЛ2	74,3	166,18	267,18	361,76	445,34
Преимущество в сжатии алгоритма ИЛЛ2 над алгоритмом ИЛЛ в (%)	16%	19%	6%	3%	2%
Алгоритм JB2 (DjVu)	52,63	124,16	202,41	272,91	330,73
Преимущество в сжатии ИЛЛ2над алгоритмом JB2 в (%)	29%	25%	24%	24,5%	25,5%

Главный вывод, который можно сделать из этой таблицы, такой. Алгоритм ИЛЛ2 благодаря второму этапу повышает степень сжатия алгоритма ИЛЛ в основном при разрешениях сканирования в 200 и 300 dpi и, таким образом обеспечивает сжатие примерно на 25% лучшее, чем алгоритм JB2 при всех разрешениях сканирования в диапазоне 200 – 600 dpi.

Получение монохромных изображений

Алгоритм JB2 и другие рассмотрены выше предназначенные для сжатия только монохромных изображений (черно-белых, без промежуточных серых тонов). Файлы формата DjVu могут содержать и изображение в тонах серого, и цветные рисунки, но одинаково в таком файле отдельно сохраняется монохромный рисунок, а отдельно дополнительная информация о цветах. Поэтому, чтобы сжать в таком виде любое изображение, которое содержит текст, необходимо сначала выделить из него монохромную составляющею.

Самый очевидный способ для этого – объявить все пиксели, темнее чем половина белого цвета, черными, а другие – белыми. Например, под MS Windows эту операцию можно выполнить с помощью стандартной программы графического редактора MS Paint. Достаточно сохранить изображение как монохромное.

Этот способ хорошо работает только тогда, когда изображение уже напоминает монохромное. Однако, если в нем есть слишком бледные буквы или затемнены поля, то в результате одни контуры могут пропасть, а другие – появиться. Чтобы исправить ситуацию в таких случаях, нужен алгоритм, который обращает

большее внимания на резкость границ и правильность линий, чем на абсолютную величину цветов.

И. Межиров предложил алгоритм [9], который стремится выделить наиболее резкие контуры; при этом как побочный эффект удаляются черные поля вокруг отсканированного листа и между страницами книги.

Следовательно, сначала в изображении выделяются контуры – связные компоненты линий уровня яркости, в которых уровень кратен предварительно избранной величине. В алгоритме эта величина составляет одну четверть полного расстояния между белым и черным. При этом считается, что эти яркости не совпадают с яркостью пикселей, так что контур проходит между пикселями, а не через них. Топологически каждый контур является эквивалентным окружности, и у него есть бесконечная внешняя и законченная внутренняя часть. Контур может быть «что освещает» (внутри ярче) и «что затемняет» (внутри темнее).

Для каждого контура подсчитывается «резкость» – величина, которая равняется по определению модулю суммы перепадов яркости по всем ребрам контура (ребро – это часть контура, который проходит между двумя пикселями). Перепад яркости в ребре равняется разнице яркости того пикселя, что внутри контура, и того, который является внешним. При этом для контуров, которые проходят по краю всего изображения, пиксели, что лежат вне этого края, считаются абсолютно черными. При сканировании у краев рисунка обычно возникают черные поля, потому такое решение делает внешние контуры этих полей «что осветляет» и уменьшает их резкость.

Контуры могут быть вложены один в другой (точнее, один во внутренность другого). Искусственно добавляется фиктивный контур, который проходит по краю изображения, в который вложены все другие контуры. Тогда контуры образуют ориентированное дерево по вложенности, и фиктивный контур – его корень.

Контуры раскрашиваются в черный и белый цвета так, чтобы максимизировать описанную ниже функцию выигрыша. При этом каждый пиксель получит тот цвет, что имеет самый внутренний из контуров.

Функция выигрыша, которую нужно максимизировать, зависит от раскраски дерева контуров. При этом некоторые разрисовки запрещаются присвоением им значения «минус бесконечность».

Значение функции выигрыша равняется сумме по всем контурам следующей величины. Фиктивный корневой контур должен быть белым (этим и достигается удаление полей). Таким образом, черный корневой контур запрещен. Контур, который имеет те же цвета, что и «отец», дает нулевой взнос в выигрыш. В следующих случаях контуру запрещено иметь яркость, отличающуюся от яркости «отца»:

1. если отдельный описанный выше алгоритм для фильтрации ненужных контуров признал его «мусором»;
2. если контур «что осветляет», а «отец» белый;
3. если контур «что затемняет», а «отец» черный.

Если же «отец» контура, который затемняет, белый, а сам контур черный, или «отец» контура, который освещает, черный, а сам контур белый, то его взнос в выигрыш равняется резкости.

Для того, чтобы найти раскрашивание с максимальным выигрышем, используются два прохода по дереву. Во время первого прохода – от листьев к корню – для каждой вершины ищется максимально возможный выигрыш от ее поддерева (включая ее саму) при условии, что ее «отец» белый. И то же при условии, если ее «отец» черный. Чтобы найти каждую из этих величин, разбираются два случая: вершина белая или вершина черная. Наибольший выигрыш в этих двух случаях даст искомую величину. Если и цвет «отца», и цвета самой вершины известны, то максимальный выигрыш в ее поддереве равняется ее взносу в выигрыш плюс сумма по всем потомкам выигрыша на их поддеревьях при известном цвете их «отца».

Во время второго прохода – от корня к листьям – по найденным величинам для каждой из вершин назначаются окончательные цвета при известном цвете «отца» так, чтобы выигрыш от поддерева каждой вершины был наибольшим.

Алгоритм для устранения «мусора» – часть, которая требует наибольшей ручной настройки. Максимальный уровень яркости будем считать ровным 255. В программе принято следующее правило: контур считается подозрительным, если он «что затемняет» и его уровень (как линии уровня) менее 156 или если он «что освещает» и его уровень больше 100; подозрительный контур считается «мусором», если его резкость менее 10000 или отношение его резкости к его длине менее 100.

Время работы всего алгоритма пропорционально числу пикселей; время работы второй части, раскрашивания дерева, пропорционально числу контуров. На практике программа работает достаточно быстро.

Выводы

В настоящее время самыми мощными алгоритмами сжатия двуцветного изображения текста являются те, которые используют классификацию выделенных символов. При этом конечный результат – степень сжатия изображения – больше всего зависит от качества классификации, то есть количества полученных классов при непременном условии, что в каждый класс входят изображения только одного символа. Основным препятствием к достижению идеальной классификации, в которой количество классов ровно столько, сколько различных символов встречается в тексте, являются шумы печати и ска-

нирования, искажающие изображения символов. И хотя человек легко справляется с такой задачей (интересно, как мы это делаем?), пока не найдена мера отличия двух сравниваемых изображений символов, позволяющая сделать то же самое.

Еще один аспект при обсуждении любого метода сжатия – качество восстановленного изображения по сравнению с исходным сжимаемым оригиналом. Обычно, чем выше степень сжатия, тем это качество хуже. Методы сжатия сканированных изображений текста, основанные на классификации выделенных символов, позволяют получать восстановленные изображения символов более высокого качества, чем оригинальные. Причем, чем лучше проведена классификация, тем больше сжатие и тем лучше качество изображения символов. Этот парадоксальный факт объясняется очень просто. При классификации выделенных символов, если она обладает высоким качеством, получаются классы, состоящие из большого числа изображений одного и того же символа. Подходящая статистическая обработка этого класса позволяет избавиться от искажений, привнесенных при печати и сканировании и имеющих случайный характер.

Описанные выше меры отличия, учитывающие контурный характер шумов печати и сканирования, являются достаточно сложными. Но все же есть уверенность, что они будут существенно улучшены, так чтобы проводимая с их помощью классификация выделенных изображений была близка к идеальной.

Список литературы

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

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

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

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

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

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

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

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

9. Межиров И. Курсовая работа на тему «Алгоритмы сжатия данных»; механико-математический ф-т, научный руководитель А. Шень / И. Межиров.. – М.: МГУ им. Ломоносова, 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. – 607 с.

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

12. Ватолин Д. 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 26.04.2013

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.*