МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет информатики
Кафедра программной инженерии

УДК 681.03

ДОПУСТИТЬ К ЗАЩИТЕ В ГАК
Зав. кафедрой, профессор, д.ф-м.н.
_____ О.А.Змеев
«___» _____ 2011 г.

Мухин Сергей Константинович

РАСПОЗАВАНИЕ СИМВОЛОВ МЕТОДОМ ПОБЛОЧНОГО СРАВНЕНИЯ

Выпускная квалификационная работа

Научный руководитель:
Ст. преп. ФИнф Матушевский В.В.

Исполнитель:
студ. гр. 1471 Мухин С.К.

Томск – 2011

Реферат

Выпускная квалификационная работа, 27 с., 17 рис., 11 источников.

СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЯ, КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЯ, УМЕНЬШЕНИЕ ШУМОВ ТЕКСТА.

Объект исследования – процесс распознавания текста на растровом графическом изображении.

Предмет исследования – алгоритмизация и моделирование процесса распознавания текста.

Цель работы – разработка эффективного алгоритма распознавания текста методом поблочного сравнения.

Сфера применения – распознание номеров, отсеивание брака, оцифровка текстов, создание электронных баз литературы в библиотеках

В рамках работы был разработан специализированный алгоритм, решающий поставленную задачу.

Алгоритм, разработанный в рамках предлагаемого подхода, был реализован на Borland Delphi.

Содержание

ВВЕДЕНИЕ

Несмотря на то, что в настоящее время большинство документов составляется на компьютерах, задача создания полностью электронного документооборота ещё далека до полной реализации. Как правило, существующие системы охватывают деятельность отдельных организаций, а обмен данными между организациями осуществляется с помощью традиционных бумажных документов. Задача перевода информации с бумажных на электронные носители актуальна не только в рамках потребностей, возникающих в системах документооборота. Современные информационные технологии позволяют нам существенно упростить доступ к информационным ресурсам, накопленным человечеством, при условии, что они будут переведены в электронный вид. Наиболее простым и быстрым является сканирование документов с помощью сканеров. Результат работы является цифровое изображение документа – графический файл. Более предпочтительным, по сравнению с графическим, является текстовое представление информации. Этот вариант позволяет существенно сократить затраты на хранение и передачу информации, а также позволяет реализовать все возможные сценарии использования и анализа электронных документов. Поэтому наибольший интерес с практической точки зрения представляет именно перевод бумажных носителей в текстовый электронный документ.

Особо следует отметить распознавание полноценных изображений. Область применения данного раздела многогранна. Например, на современных заводах контроль качества производимой продукции зачастую производят с использованием систем распознавания, которые отсеивают брак. Распознавание полноценных изображений применяется также на дорогах, для определения и распознавания номеров автомобилей, контроль их скорости. Обработка изображений актуальна и при анализе снимков из космоса и с самолётов. Таким образом, видно, что область применения распознавания изображений широка и многогранна и позволяет намного сократить и упростить рабочий процесс и вместе с тем повысить его качество. Однако, возможности интеллектуального анализа изображений с помощью компьютеров оставляют желать лучшего. Можно с уверенностью отметить лишь успехи в распознавании букв и цифр в документах и текстах, а также анализе изображений специального вида. Такая область как распознавание текстур, исследование в которой проводятся не одно десятилетие, пока не имеет универсальных методов.

Но возникает ряд трудностей и проблем. Чаще всего это связано с тем, что изображения предъявляются на сложном фоне или изображения эталона и входные изображения отличаются положением в поле зрения, или входные изображения не совпадают с эталонами за счет случайных помех.

МЕТОДЫ РАСПОЗНАВАНИЯ СИМВОЛОВ

В случае, когда речь идет о распознавании печатных символов следует упомянуть, что почти бесконечное разнообразие печатной продукции изготавливается при помощи ограниченного набора оригиналов символов, которые группируются по стилю (набору художественных решений), который отличает данную группу от других. Одна группа, включающая все алфавитные знаки, цифры и стандартный набор служебных символов, называется гарнитурой. Однако в широком кругу людей, имеющих дело с производством различного рода документации, утвердилось другое название гарнитуры - шрифт; этого термина мы и будем придерживаться в дальнейшем.

Итак любой печатный текст имеет первичное свойство - шрифты, которыми он напечатан. С этой точки зрения существуют два класса алгоритмов распознавания печатных символов: шрифтовой и безшрифтовый [9]. Шрифтовые или шрифтозависимые алгоритмы используют априорную информацию о шрифте, которым напечатаны буквы. Это означает, что программе ОРС должна быть предъявлена полноценная выборка текста, напечатанного данным шрифтом. Программа измеряет и анализирует различные характеристики шрифта и заносит их в свою базу эталонных характеристик. По окончании этого процесса шрифтовая программа оптического распознавания символов (ОРС) готова к распознаванию данного конкретного шрифта. Этот процесс условно можно назвать обучением программы. Далее обучение повторяется для некоторого множества шрифтов, которое зависит от области применения программы. К недостаткам данного подхода следует отнести следующие факторы:

- Алгоритм должен заранее знать шрифт, который ему представляют для распознавания, т.е. он должен хранить в базе различные характеристики этого шрифта. Качество распознавания текста, напечатанного произвольным шрифтом, будет прямо пропорционально корреляции характеристик этого шрифта со шрифтами, имеющимися в базе программы. При существующем богатстве печатной продукции в процессе обучения невозможно охватить все шрифты и их модификации. Другими словами, этот фактор ограничивает универсальность таких алгоритмов.
- Для работы программы распознавания необходим блок настройки на конкретный шрифт. Очевидно, что этот блок будет вносить свою долю ошибок в интегральную оценку качества распознавания, либо функцию установки шрифта придется возложить на пользователя.
- Программа, основанная на шрифтовом алгоритме распознавания символов, требует от пользователя специальных знаний о шрифтах вообще, об их группах и отличиях друг от друга, шрифтах, которыми напечатан документ, пользователя. Отметим, что в случае, если бумажный документ не создан самим пользователем, а пришел к нему извне, не существует регулярного способа узнать с использованием каких шрифтов этот документ был напечатан [3]. Фактор необходимости специальных знаний сужает круг потенциальных пользователей и сдвигает его в сторону организаций, имеющих в штате соответствующих специалистов.

С другой стороны, у шрифтового подхода имеется преимущество, благодаря которому его активно используют и, по-видимому, будут использовать в будущем. А именно, имея детальную априорную информацию о символах, можно построить весьма точные и надежные алгоритмы распознавания. Вообще, при построении шрифтового алгоритма распознавания (в отличие от безшрифтового, о чем будет сказано ниже), надежность распознавания символа является интуитивно ясной и математически точно выразимой

величиной. Эта величина определяется как расстояние в каком-либо метрическом пространстве от эталонного символа, предъявленного программе в процессе обучения, до символа, который программа пытается распознать.

Второй класс алгоритмов - безшрифтовые или шрифтонезависимые, т.е. алгоритмы, не имеющие априорных знаний о символах, поступающих к ним на вход. Эти алгоритмы измеряют и анализируют различные характеристики (признаки), присущие буквам как таковым безотносительно шрифта и абсолютного размера (кегля), которым они напечатаны [4]. В предельном случае для шрифтонезависимого алгоритма процесс обучения может отсутствовать. В этом случае характеристики символов измеряет, кодирует и помещает в базу программы сам человек. Однако на практике, случаи, когда такой путь исчерпывающе решает поставленную задачу, встречаются редко. Более общий путь создания базы характеристик заключается в обучении программы на выборке реальных символов. К недостаткам данного подхода можно отнести следующие факторы:

- Реально достижимое качество распознавания ниже, чем у шрифтовых алгоритмов. Это связано с тем, что уровень обобщения при измерениях характеристик символов гораздо более высокий, чем в случае шрифтозависимых алгоритмов. Фактически это означает, что различные допуски и огрубления при измерениях характеристик символов для работы безшрифтовых алгоритмов могут быть в 2-20 раз больше по сравнению с шрифтовыми.
- Следует считать большой удачей, если безшрифтовый алгоритм обладает адекватным и физически обоснованным, т.е. естественно проистекающим из основной процедуры алгоритма, коэффициентом надежности распознавания. Часто приходится мириться с тем, что оценка точности либо отсутствует, либо является искусственной. Под искусственной оценкой подразумевается то, что она существенно не совпадает с вероятностью правильного распознавания, которую обеспечивает данный алгоритм.

Достоинства этого подхода тесно связаны с его недостатками [5]. Основными достоинствами являются следующие:

- *Универсальность.* Это означает с одной стороны применимость этого подхода в случаях, когда потенциальное разнообразие символов, которые могут поступить на вход системы, велико. С другой стороны, за счет заложенной в них способности обобщать, такие алгоритмы могут экстраполировать накопленные знания за пределы обучающей выборки, т.е. устойчиво распознавать символы, по виду далекие от тех, которые присутствовали в обучающей выборке.
- *Технологичность.* Процесс обучения шрифтонезависимых алгоритмов обычно является более простым и интегрированным в том смысле, что обучающая выборка не фрагментирована на различные классы (по шрифтам, кеглям и т.д.). При этом отсутствует необходимость поддерживать в базе характеристик различные условия совместного существования этих классов (некоррелированность, не смешиваемость, систему уникального именования и т.п.). Проявлением технологичности является также тот факт, что часто удается создать почти полностью автоматизированные процедуры обучения.
- *Удобство в процессе использования программы.* В случае, если программа построена на шрифтонезависимых алгоритмах, пользователь не обязан знать что-либо о странице, которую он хочет ввести в компьютерную память и уведомлять об этих знаниях программу. Также упрощается пользовательский интерфейс программы за счет отсутствия набора опций и диалогов, обслуживающих обучение и управление базой характеристик. В этом случае процесс распознавания можно представлять пользователю как “черный ящик” (при этом пользователь полностью лишен

возможности управлять или каким-либо образом модифицировать ход процесса распознавания). В итоге это приводит к расширению круга потенциальных пользователей за счет включения в него людей обладающих минимальной компьютерной грамотностью.

Выше рассматривались особенности, достоинства и недостатки двух подходов к созданию алгоритмов ОРС. Из обзора следует, что достоинства и недостатки обоих подходов определяются одними и теми же свойствами алгоритмов: большей либо меньшей степенью универсальности, степенью достижимой точности распознавания и т.п. Сравнительные недостатки и достоинства обоих подходов сведены в таблицу.

Свойства	Шрифтовые алгоритмы	Безшрифтовые алгоритмы
Универсальность	Малая степень универсальности, обусловленная необходимостью предварительного обучения всему, что предъявляется для распознавания	Большая степень универсальности, обусловленная независимостью обучающей выборки от какой-либо системы априорной классификации символов
Точность распознавания	Высокая, обусловлена детальной классификацией символов в процессе обучения. А также тем, что материал распознавания находится строго в рамках классов, созданных в процессе обучения	Низкая (в сравнении с шрифтовыми алгоритмами), что обусловлено высокой степенью обобщения и огрубленными измерениями характеристик символов
Технологичность	Низкая (в сравнении с безшрифтовыми алгоритмами), обусловлена различными накладными расходами, связанными с поддержкой классификации символов	Высокая, обусловлена отсутствием какой-либо априорной системы классификации символов
Поддержка процесса распознавания со стороны пользователя	Необходима: - на этапе обучения для задания системы классификации; - на этапе распознавания для указания конкретных классов символов	Не требуется

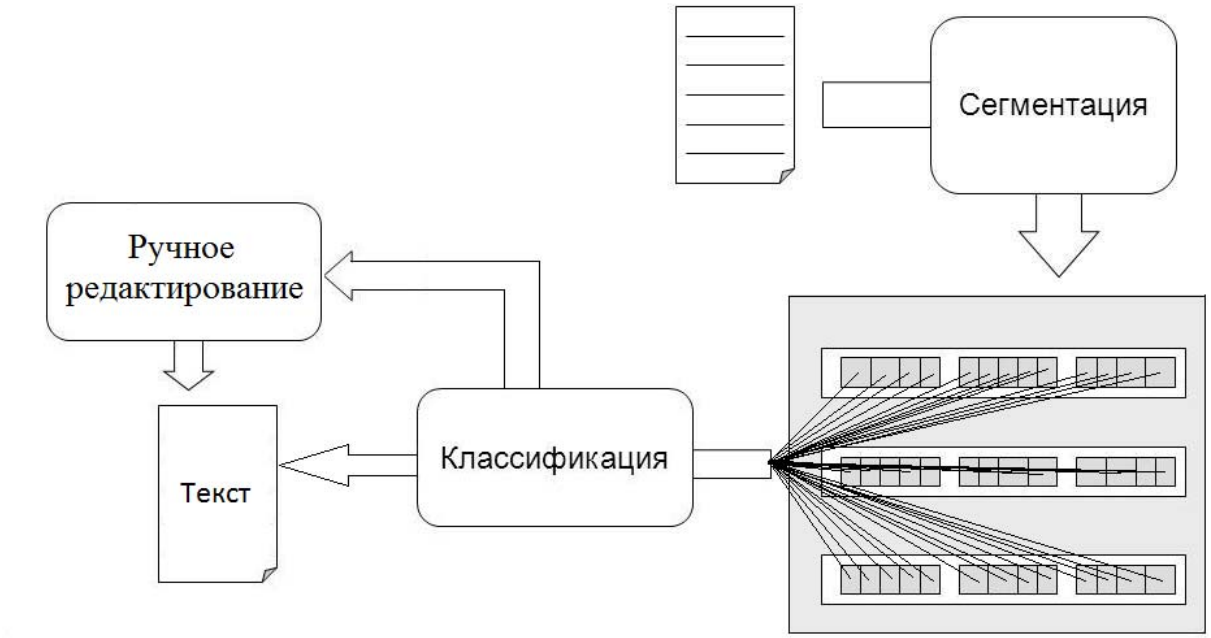
Рассмотрение обоих подходов в сравнении друг с другом приводит к целесообразности их объединения. Цель объединения очевидна - получить метод, совмещающий одновременно универсальность и технологичность безшрифтового подхода и высокую точность распознавания шрифтового. Предпосылками для исследования в этом направлении послужил следующий круг идей и фактов. Любой алгоритм распознавания символов становится применим на практике при качестве распознавания 94-99%. “Дожимание” последних процентов, т.е. окончательная доводка алгоритма всегда является трудоемкой и дорогостоящей работой. Внутри сферы распознавания символов любой алгоритм имеет свою специфичную область действия, для которой он разработан и в которой проявляет себя наилучшим образом. В целом, путь

увеличения качества распознавания лежит не в изобретении сверхинтеллектуального алгоритма, который заменит собой все остальные, а в комбинировании нескольких алгоритмов, каждый из которых сам по себе прост и обладает эффективной вычислительной процедурой. При комбинировании различных алгоритмов важно, чтобы они опирались на независимые источники информации о символах. В случае, если два алгоритма работают над сильно коррелированными между собой данными, то вместо увеличения качества распознавания будет увеличиваться суммарная ошибка. С другой стороны, знания о распознанных символах должны накапливаться и использоваться в последующих шагах процесса распознавания. Более того, как окончательный критерий можно использовать точный шрифтозависимый алгоритм, база характеристик которого построена прямо в процессе работы (“на лету”) по результатам предыдущих шагов распознавания. Метод, обладающий указанным выше свойством, называется адаптивным распознаванием, т.к. он использует динамическую настройку (адаптацию) на входные символы. Подобный механизм использует популярная программа Fine Reader.

ОБЩИЙ АЛГОРИТМ РАБОТЫ

В свою очередь шрифтозависимые и шрифтонезависимые алгоритмы реализуются различными методами. В просе изучения алгоритмов был разработан шрифтовой алгоритм поблочного сравнения.

На рисунке представлена общая схема программы. На первом этапе подготовленное растровое изображение сегментируется: выделяются строки, слова и буквы. На втором этапе каждая выделенная буква проходит классификацию, состоящая из масштабирования, а также механизма сравнения блоков и изучения эталонов. Результат выводится в выходной текстовый файл или переходит на этап ручного редактирования.



СЕГМЕНТАЦИЯ

Сегментация текста требует ровного расположения отсканированного документа, иначе вычисляется угол наклона страницы, который учитывается при распознании [2]. В данной работе рассматривается случай ровного положения страницы.

Сегментацию изображения текста будем проводить в три этапа:

1. выделение строк - исходное изображение текста необходимо "разрезать" на полосы-строки нужной ширины.
2. сегментация слов - в изображении текстовой строки выделяем изображения слов.
3. сегментация символов - в изображении слова проводим границы символов.

Будем рассматривать изображение текста в градациях серого. Исходное изображение можно представить как матрицу яркостей точек B .

$$B=\{b_{ij}\}$$

$$0\le b_{ij}\le b^{max}$$

$$i=1\ldots n\ ;\ j=1\ldots m$$

где n - ширина картинки, m - высота картинки

Для определённости будем считать, что максимальное значение яркости (b^{max}) соответствует чёрному цвету а минимальное (равное 0) - белому.

СЕГМЕНТАЦИЯ СТРОК

Задача выделения строк сводиться к нахождению верхних и нижних граней строк текста, изображённого на исходной картинке.

Алгоритм сегментации строк основывается на том, что средняя яркость в изображениях межстрочных промежутках существенно ниже средней яркости в изображениях текстовых строк.

1. Сначала для всех пиксельных строк исходного изображения находим их средние значения яркости

$$s_j = s_j(B) = \frac{1}{n} \cdot \sum_{i=1}^{n} b_{ij}$$

2. Затем определяем среднее значение яркости всего изображения

$$s(B) = \frac{1}{m} \cdot \sum_{j=1}^{m} s_j(B)$$

3. Средняя яркость в межстрочных промежутках текста должна быть невелика (в идеальном случае она равна нулю). Поэтому яркость верхней границы текстовой строки можно выразить через среднюю яркость изображения

$$s^t = k^t * s(B)$$

где $0<k^t<1$ - коэффициент

4. Аналогично яркость нижней границы текстовой строки, также может быть выражена через среднюю яркость всего изображения

$$s^b = k^b * s(B)$$

где $0<k^b<1$ - коэффициент

Работа алгоритма сегментации строк заключается в последовательном просмотре массива средних значений $(s_1,...,s_m)$ и выявлении множества пар индексов (s^t_i,s^b_i) пиксельных строк, соответствующих верхней s^t_i и нижней s^b_i граням изображения строки номер i, удовлетворяющих следующим условиям.

1. Условия верхней границы текстовой строки.

   Начало текстовой строки или области устойчивого повышения яркости фиксируется, если выполняется следующий комплекс условий:

   - яркость текущей пиксельной строки превышает границу s^t
   - яркость двух предыдущих пиксельных строк ниже этой границы
   - яркость трех последующих строк выше границы s^b

   т.е. в пиксельной строке с номером i начинается изображение текстовой строки если

$$(s_{i-2} < s^t) \wedge (s_{i-1} < s^t) \wedge (s_i > s^b) \wedge (s_{i+1} > s^b) \wedge (s_{i+2} > s^b) \wedge (s_{i+3} > s^b)$$

2. Условия нижней границы текстовой строки.

   Конец области устойчивого повышения яркости определяется, если выполняется следующие условия:

   - было зафиксировано начало области
   - яркость текущей пиксельной строки превышает границу s^t
   - яркость последующей пиксельной строки ниже границы s^b

   Или:

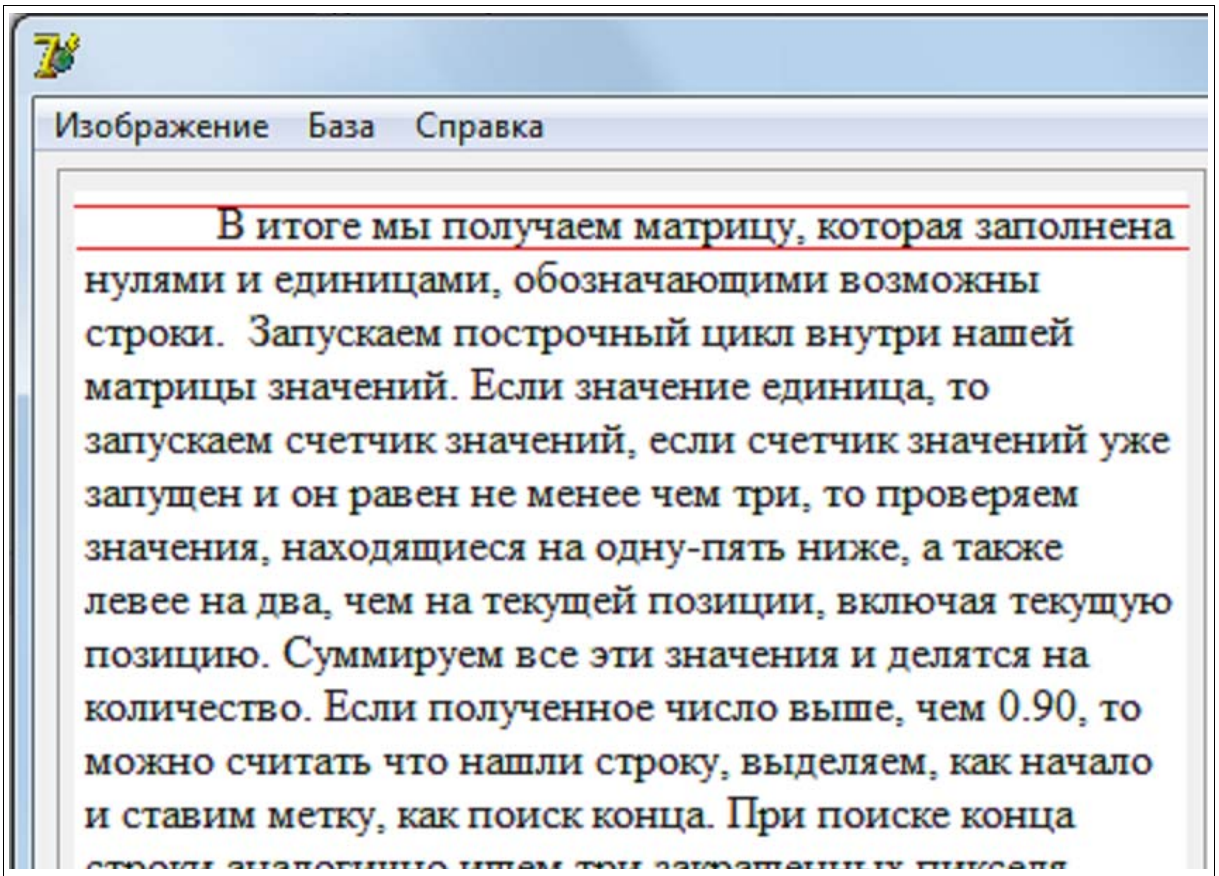
   - было зафиксировано начало области
   - яркость трех последующих строк ниже границы s^b

   т.е. в пиксельной строке с номером i заканчивается изображение текстовой строки, если ранее было определено, что строка началась, и выполняется условие

$$((s_i > s^t) \wedge (s_{i+1} < s^b)) \vee ((s_{i+1} < s^b) \wedge (s_{i+2} < s^b) \wedge (s_{i+3} < s^b))$$

В результате формируется множество пар индексов верхних и нижних граней строк. Разность между этими индексами дает высоты текстовых строк. Однако такой алгоритм находит среднюю высоту каждой текстовой строки и "срезает" символы, выступающие по высоте за эту среднюю высоту.

Чтобы избежать этого, необходимо расширить найденные границы. Можно предложить следующий алгоритм расширения границ. Среди найденных текстовых строк определяется строка с минимальной высотой H_{min} и, затем все границы с каждой стороны расширяются на величину $0.3 * H_{min}$. Это не приводит к слиянию строк, т.к. межстрочные интервалы текста, как правило, больше чем высота строки.

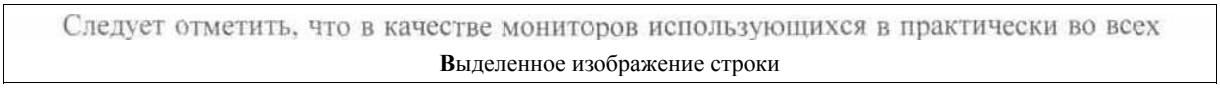


Изображение текста с выделенными строками

Таким образом, в результате работы алгоритма на исходном изображении отмечается положение всех текстовых строк.

СЕГМЕНТАЦИЯ СЛОВ

На втором этапе решения задачи сегментации изображения текста, из изображений строк. Входом для алгоритма сегментации слов служит изображение, какой либо одной текстовой строки, которое получается из исходного изображения документа после применения к нему алгоритма сегментации строк.



Выделенное изображение строки

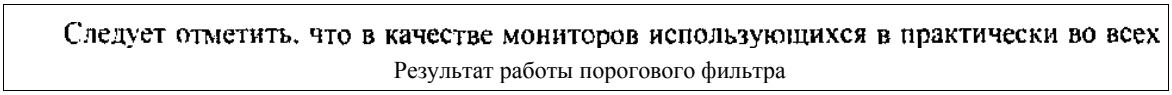
Для улучшения качества работы алгоритма выделения слов из строки вначале его работы выполняются два преобразования входного изображения.

1. Пороговый фильтр повышения контрастности

$$b_{ij} = \begin{cases} b_{max} & ; \ b_{ij} > b_0 \\ 0 & ; \ b_{ij} \le b_0 \end{cases}$$

где $i=1... n$; $j=1... m$; b_0 - порог яркости

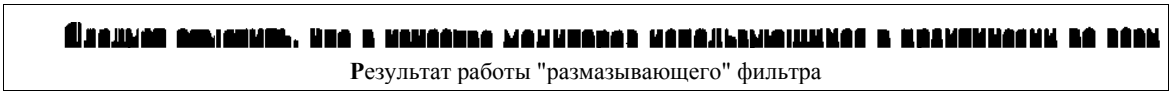
Такое преобразование, при правильно выбранном пороге b_0, помогает снизить уровень шума, т.е. убрать значительное количество лишних точек.



Результат работы порогового фильтра

2. "Размазывающий" фильтр - для каждой яркой (чёрной) точки исходного изображения закрашиваем соседние точки.

В результате такого преобразования близкие точки объединяются в непрерывную область и вместо множества маленьких точек получаем картинку состоящую из нескольких сплошных пятен с достаточно чёткой границей (рис.5).


Результат работы "размазывающего" фильтра

Далее выполняем собственно процедуру сегментации. Алгоритм сегментации слов основывается на том, что средняя яркость в межсловных интервалах существенно ниже средней яркости в изображениях слов. Он похож на алгоритм сегментации строк, только просмотр идет по пиксельным столбцам изображения строки.

1. Для всех пиксельных столбцов исходного изображения строки находим их средние значения яркости

$$c_i = c_i(B) = \frac{1}{m} \cdot \sum_{j=1}^{m} b_{ij}$$

где m, - высота текущей строки в точках

2. Затем определяем среднее значение яркости для данного изображения строки

$$c(B) = \frac{1}{n} \cdot \sum_{i=1}^{n} c_i(B)$$

где n, - ширина текущей строки в точках

3. Средняя яркость в межсловных интервалах должна быть невелика (в идеальном случае она равна нулю). Поэтому ее левую границу (начало слова) можно выразить через среднюю яркость изображения строки

$$c^l = k^l * c(B)$$

где $0<k^l<1$ - коэффициент

4. Аналогично яркость правой границы (конец слова), также может быть выражена через среднюю яркость всего изображения

$$c^r = k^r * c(B)$$

где $0<k^r<1$ - коэффициент

Работа алгоритма сегментации слов заключается в последовательном просмотре множества средних значений яркости столбцов $(c_1,...,c_n)$ и выявлении множества пар индексов (c^l_i,c^r_i) пиксельных строк, соответствующих левой c^l_i и правой c^r_i граням изображения слова номер i, удовлетворяющих следующим условиям.

1. Условия левой границы (начало слова).

   Начало слова или области устойчивого повышения яркости фиксируется, если выполняются следующие условия;

   - яркость текущего и последующего пиксельного столбца превышает левую границу яркости для слова c^l
   - яркость предыдущего пиксельного столбца ниже этой границы

   т.е. в пиксельном столбце с номером j начинается изображение слова если

$$(c_{j-1} < c^l) \wedge (c_j > c^l) \wedge (c_{j+1} > c^l)$$

2. Условия правой границы (конец слова).

   Конец области устойчивого повышения яркости определяется, если выполняются следующие условия;

   - было зафиксировано начало слова
   - яркость текущего и четырех последующих пиксельных столбцов ниже границы яркости межсловного интервала c^r
   - яркость двух предыдущих пиксельных столбцов выше этой границы

   т.е. слово заканчивается в пиксельном столбце с номером j , если ранее было определено, что слово началось, и выполняется условие

$$(c_{j-2} > c^r) \wedge (c_{j-1} > c^r) \wedge (c_j < c^r) \wedge (c_{j+1} < c^r) \wedge (c_{j+2} < c^r) \wedge (c_{j+3} < c^r) \wedge (c_{j+4} < c^r)$$

Изображение строки с выделенными словами

СЕГМЕНТАЦИЯ СИМВОЛОВ

В большинстве изображений слов символы расположены близко друг к другу и межсимвольные интервалы не так ярко выражены, как в случае межстрочных или

межсловных интервалов. Поэтому алгоритм сегментации символов сложнее и не так очевиден как рассмотренные ранее алгоритмы сегментации строк и слов.

Изображение слова

Входом для алгоритма сегментации символов служит изображение, какого либо слова, которое получается из изображения текстовой строки после применения к нему алгоритма сегментации слов.

Алгоритм сегментации символов основывается на том, что средняя яркость в межсимвольных интервалах, по крайней мере, ниже средней яркости в изображениях символов. Его (алгоритма сегментации) общая схема состоит из двух основных частей.

1. находим все индексы столбцов, соответствующие локальным минимумам средней яркости столбцов c^i.
2. выявляем и удаляем из этого списка индексов ложные границы символов

Конечная цель работы - найти индексы столбцов-границ между символами.

Поиск локальных минимумов яркости

Поиск локальных минимумов средней яркости столбцов c^i происходит на смежных интервалах изменения индекса столбца. Размер интервала выбирается исходя из высоты строки. Для большинства шрифтов отношение ширины символа к его высоте не превышает величину *0.3*. Поэтому размер интервала выбран

$$d_j = 0.3 * m$$

где m - высота слова в точках.

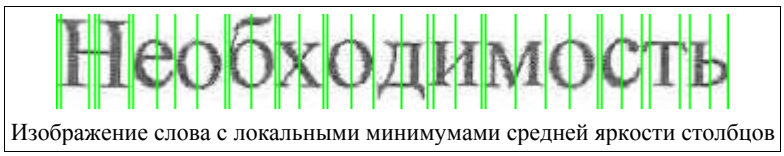
Поиск минимумов работает следующим образом.

1. Сначала для всех пиксельных столбцов исходного изображения находим их средние значения яркости

$$c_i = c_i(B) = \frac{1}{m} \cdot \sum_{j=1}^{m} b_{ij}$$

   где m - высота слова в точках.

2. Среди значений c_i первый минимум ищем на отрезке $i=1,...,d_j$.
3. Предположим, что он нашелся для индекса i^1_{min}.
4. Следующий минимум ищем на отрезке $i=(i^1_{min}+1),...,(i^1_{min}+1+d_j)$.
5. Процедура поиска повторяется, до достижения границы ($i=n$) изображения слова. Все значения индекса i^j_{min} , соответствующих локальным минимумам, сохраняются в списке W_0.



Изображение слова с локальными минимумами средней яркости столбцов

Удаление ложных границ

Удаление ложных межсимвольных границ будем проводить в несколько этапов.

1. Локальный минимум яркости в столбце номер i является "кандидатом" на принадлежность к межсимвольному интервалу, если значение средней яркости c_i в этом столбце меньше определённой границы яркости c^b и при этом значение средней яркости в столбцах отстоящих от данного локального минимума на 2 пикселя слева или справа больше границы яркости. Границу яркости можно определить через среднюю яркость картинки.

$$c^b = k^b \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} c_i(B)$$

где $0<k^b<1$. - коэффициент, n - ширина изображения слова в точках.

Первое условие межсимвольных границы можно записать в следующем виде.

$$(c_i < c^b) \wedge ((c_{i-2} > c^b) \vee (c_{i+2} > c^b))$$

В результате из списка индексов локальных минимумов W_0 удаляются индексы столбцов, средняя яркость которых не удовлетворяет этому условию, формируется второй список W_1 индексов-"кандидатов" в межсимвольные границы.

Изображение слова после удаления части ложных границ

2. Выявление связей между столбцами пикселей. На этом шаге алгоритма сегментации будем анализировать связность изображений символов и убирать из списка W_1 ложные границы, которые разрезают символ на части. Это может происходить с широкими слабосвязанными символами, например символы русского алфавита П, Н, Ц. Причём, символ может быть связан, либо в верхней (П), либо в средней (Н), либо в нижней части (Ц) пиксельных столбцов. Чтобы избежать неправильной классификации связности, разделим изображение на три уровня по вертикали и будем анализировать эти уровни отдельно друг от друга. Разделение изображения символа на части происходит в следующей пропорции: верхний уровень - 30% от высоты символа, средний уровень - 40% от высоты символа, нижний уровень - 30% от высоты символа.

Сформулируем условия связности двух соседних пиксельных столбцов k и $k+1$.

1. Для максимумов яркости трех уровней $b_{k\,h1}, b_{k\,m1}, b_{k\,l1}$ столбца k и максимумов яркости трех уровней $b_{(k+1)\,h2}, b_{(k+1)\,m2}, b_{(k+1)\,l2}$ столбца $k+1$ должно выполняться условие

$$(h_1 = h_2) \vee (m_1 = m_2) \vee (l_1 = l_2)$$

2. Средняя яркость столбца k должна быть меньше максимума яркости соседнего столбца $k+1$

$$c(k) < c_{max}(k+1)$$

3. максимум яркости в столбце k должен быть больше удвоенного абсолютного значения разности между значениями максимумов яркости столбца k и соседнего столбца $k+1$

$$c_{max}(k) > 2 * | c_{max}(k) - c_{max}(k+1) |$$

Если для данного столбца выполняются все условия связности с соседями слева и справа то граница удаляется как ложная, в противном случае выполняется ещё одна проверка. Расстояние до предыдущей (левой) границы d_k должно быть больше допустимого минимума d_{min}.

$$d_k > d_{min}.$$

$$d_{min} = 0.4 * n.$$

где n - высота изображения слова

В результате из списка индексов "кандидатов" W_1 удаляются индексы столбцов, которые имеют связь с соседями слева и справа, формируется конечный список индексов границ W_2 .

Изображение слова после обработки

КЛАССИФИКАЦИЯ

На этапе классификации рассматриваем отдельные изображения символов, полученные на этапе сегментации. Каждое такое изображение проходит три этапа: масштабирование до размеров эталона (16x16), вычисление коэффициента похожести (КП) и принятие решения.

МАСШТАБИРОВАНИЕ

Алгоритм распознания должен быть корректен для символов любого размера, поэтому мы должны привести любой символ к эталонному значению. Наиболее простой в реализации алгоритм масштабирования: Scale2x, являющийся усовершенствованием алгоритма EPX (Eric's Pixel eXpansion, разработанный Эриком Джонстаном, Lucas Art [10]). Алгоритм преобразовывает девятипикселовый квадрат в четырехпикселовый и наоборот. Поэтому нам нужно вычислять такие квадраты в каждом изображении символа. Но перед тем как преобразовывать такие квадраты, нужно рассмотреть текущий случай, а их может быть пять:

1. эталон в 2 или более раз больше
2. эталон в 1.5 раза больше
3. эталон сравним с изображением
4. эталон в 1.5 раза меньше
5. эталон в 2 раза меньше

Для второго и четвертого случая выполняется Scale2x, для первого и пятого случая делается кратное увеличение/уменьшение изображения. И наконец для третьего случая изображение адаптируется под эталон. Если после масштабирования изображение символа еще не совпало с размером эталона, выполняется еще одно масштабирование. Такие действия выполняется до тех пор, пока размер символа и эталона не совпадут.

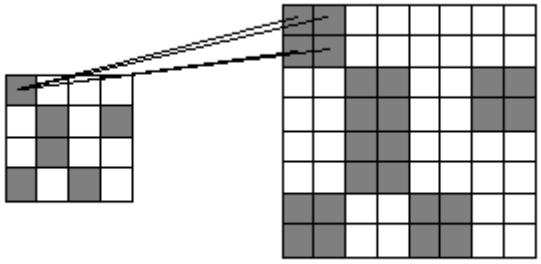
Scale2x. Пять элементов пиксельной матрицы 3x3 берутся как опорные. Все пиксели новой матрицы 2x2 получаются яркость центрального пикселя матрицы 3x3. Четыре крайних пикселя сравниваются между собой для определения значения яркости других пикселей матрицы 2x2. Записать эти действия можно так:

```
0 A 0  --\ 1 2
C P B  --/ 3 4
0 D 0
```

```
 1:=P; 2:=P; 3:=P; 4:=P;
if (C=A) and (C<>D) and (A<>B) then 1:=A;
if (B=A) and (C<>A) and (D<>B) then 2:=B;
if (B=D) and (C<>D) and (A<>B) then 3:=D;
if (C=D) and (B<>D) and (A<>C) then 4:=C;
```

Обратная операция аналогична, только в этом случае рассматриваются равные значения матрицы 2x2. Если три значения матрицы 2x2 равны, то элемент P будет кА равные элементы, иначе P получает средневзвешенную яркость четырех пикселей.

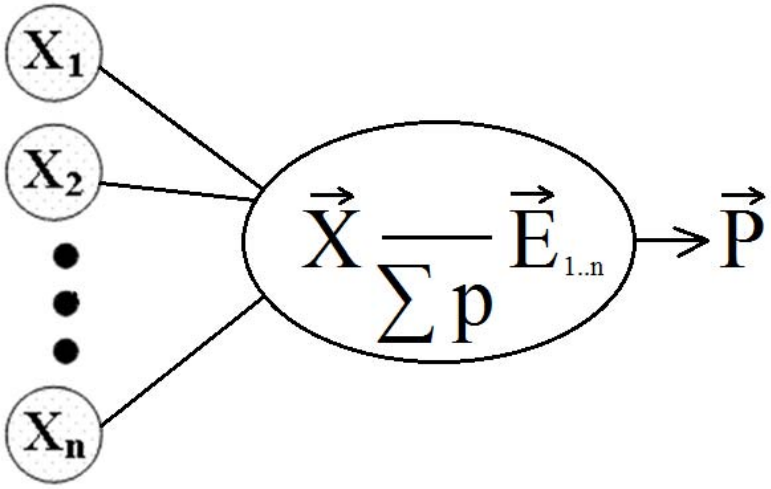
Кратное увеличение. Образец увеличивается в два раза от прежнего состояния. Каждому пикселю образца будут соответствовать три четыре пикселя нового образца с той же яркостью.



Адаптация изображения под матрицу эталонов происходит путем дублирования значений яркости границ. Если отличие нечетно, то берется первой правая или нижняя граница, если отличие четно, то берутся обе границы в одинаковом соотношении.

ОРГАНИЗАЦИЯ ПОБЛОЧНОГО СРАВНЕНИЯ

Рассмотрим работу программы методом поблочного сравнения. Для входных данных возьмем комбинированную структуру: четыре объединенных пикселя, образующих матрицу 2x2. Значение яркости для такого блока будет сумма всех значений яркости каждого пикселя блока. Так как значение яркости нельзя описать однозначно: есть или нет, то опишем значения яркости блока в промежутке от 0 до 1. Так максимальная яркость (четыре черных пикселя с максимальным значением) будет 255*4=1020, т.к. значение яркости пикселей варьируется от 0 (белый) до 255 (черный). В практическом вычислении яркости изображения, при условии, что оно имеет только оттенки серого, получаем значение одного из параметров RGB. Так как оттенки серого имеют одинаковые значения для всех трех параметров (RGB), возьмем за показатель значение R. Значению R 255 соответствует белый цвет, поэтому при вычислениях значение яркости будем принимать как 255-R.



Объединенные четырехпиксельные блоки поступают на вход алгоритма сравнения. Определим похожесть(p) как 1-((abs(X-E))/1024), где X – сумма яркости блока символа, E – сумма яркости блока эталона. На выходе получаем вектор похожести на все эталоны. Четыре объединенных пикселя позволяют ИНС нивелировать небольшим отличием изображения от эталона. Таким образом, вне зависимости от расположения пикселя внутри блока, A остается неизменным. Предполагаемым символом будет значение наибольшей похожести, среди значений вектора. Если уровень похожести не

ниже 0.85, можно считать, что распознавание успешное, значение эталона поступает в выходной файл. Если уровень похожести не ниже 0.65, то символ предположительно определился. Если пользователь задает возможность ручной проверки, то выводится предполагаемое значение, которое можно или принять или изменить и изучить изображения, оно добавляется в список эталонов. Изучение в первом случае не происходит. При ошибке ниже 0.65 распознавание считается неудачным, символ не определяется, на выход выводится вопросительный знак.

ДОБАВЛЕНИЕ ЭТАЛОНОВ

Добавление эталонов в представленной программе организуется двумя способами: вручную и автоматически. При добавлении вручную изображение вводится в область эталонов, каждому эталону присваиваются значения. Для одного значения может быть неограниченное число эталонов (обуславливается разнообразными шрифтами). Возможно подключение базы, содержащий определенный шрифт. Автоматическое добавлене происходит во время работы программы. Если уровень похожести выше 0.85, а также заранее было разрешение на автоматическое изучение, то наиболее похожий символ добавляется в список эталонов со значением элемента с максимальной похожестью. Также добавление возможно в полуавтоматическом режиме, при более низкой похожести.

ЗАКЛЮЧЕНИЕ

В данной работе были представлены вопросы, связанные с задачей распознавания символов методом поблочного сравнения.

Была составлена общая структура алгоритмов распознавания, подробно разобраны алгоритм сегментации простой страницы в оттенках серого, алгоритм масштабирования символа, алгоритм классификации. Представленные алгоритмы являются базовыми и могут быть расширены за счет уменьшения шума входных символов, настройки алгоритмов под разные случаи: поворот входной страницы, снижение контрастности фона относительно символов.

Помимо теоретических изысканий, в рамках выпускной квалификационной работы было разработано программное средство, представляющее возможность распознавания текста на отсканированных страницах. Программный продукт реализует базовый вариант алгоритма, но может быть легко расширен в дальнейших версиях.

Программный продукт написан на языке Pascal от компании Borland© в среде разработки Delphi.

В связи с высокой актуальностью проблемы, с учетом возможной доработки программного обеспечения, может использоваться для практической работы.

ЛИТЕРАТУРА

1. В. Л. Арлазаров, В.В. Троянкер, Н.В. Котович. Адаптивное распознание символов, 2000.
2. Борисов Е. Сегментация изображений текста, 2008, СпбГУ ПМ-ПУ, 20 с.
3. Сборник Классификация и кластер. М. : "Мир", 1980
4. Розанов Ю.А. Теория вероятностей, случайные процессы и математическая статистика. М. : “Наука”, 1989
5. Ян Д.Е., Анисимович К.В., Шамис А.Л. Новая технология распознавания символов. Теория, практическая реализация, перспективы. М. : Препринт, 1995
6. Промахина И.М., Коростелев А.П. Об одном классе вероятностных рекуррентных алгоритмов распознавания. М. : Препринт, 1984
7. Корн Г., Корн Т. Справочник по математике. М. : “Наука”, 1984
8. Y-H Pao Adaptive pattern recognition and neural network “Addison-Wesley” 1989
9. Журавлев Ю.И. Алгоритмы вычисления оценок и их применение Ташкент, “Фан” 1974
10. Jonstan Eric, Scale2x [Электронный ресурс]. – Режим доступа: http://www.socallinuxexpo.org/past/2003
11. Алгоритмы распознавания [Электронный ресурс]. – Режим доступа: http://super-profi.com

Приложение А. Руководство программиста: демонстрационное приложение «Распознаватель»

Исходный тест программы состоит из нескольких модулей и голоного файла проекта среды Delphi 7. Для компиляции необходимо запустить Delphi 7 и загрузить проектный файл diplom.dpr. Далее можно откомпилировать проект и работать в соответствии с руководством пользователя.

Далее перечисляются модули и даются описания используемым процедурам и функциям.

diplom.pas – основной модуль программы, в котором выполняются все основные действия.

text.pas – модуль редактирования выходного текста.

razb – процедура разбивает изображение в поле эталонов на блоки по четыре пикселя.

MaxRValue – функция, возвращающая значение максимального элемента в массиве реальных чисел.

Takestring – процедура разбивает изображение на строки, на выходе получается массив индексов строк.

Takeword – процедура разбивает строки на слова, на выходе получатся массив индексов слов

Takesymbol – процедура разбивает слова на символы, на выходе получается массив индексов символов

Resetnoise – функция возвращает матрицу пикселей строки со сниженными шумами

AllOne – функция возвращает матрицу пикселей строки с эффектом размытости

Mashtab – процедура масштабирования изображения символа в поле эталонов.

Scale – процедура масштабирования методом Scale2x

Mup – процедура масштабирования методом кратного увеличения

Adept – процедура масштабирования методом адаптации

Button9click – основная процедура. Выполняет сегментацию, классификацию, на выходе передает распознанный текст в поле редактирования или выходной файл.

Button6click – Скрывает или раскрывает поле просмотра результатов тестирования

PaintBox3MouseDown, PaintBox3MouseUp, PaintBox3MouseMove – процедуры, позволяющие закрашивать пиксели или рисовать символы в поле эталонов

N8Click – загружает шаблонный графический файл

N7Click – загружает выбранный графический файл расширения BMP

Button7click – в режиме тестирования сравнивает символ в поле эталонов с эталонами

Button8click – изображенный в поле эталонов символ добавляется в базу эталонов

Button5click – поле эталонов очищается

Также используется таблица базы Access, Tnew2 содержащая значения эталонов символов, а также яркость блоков каждого символа.

К программе приложено изображение text.bmp являющимся шаблоном для тестирования программы.

Приложение Б. Руководство пользователя.

Программа «Распознаватель» по умолчанию имеет в своей базе основной шрифт, поэтому готова к работе сразу. Для увеличения точности распознания необходимо обучать программу новым шрифтам.

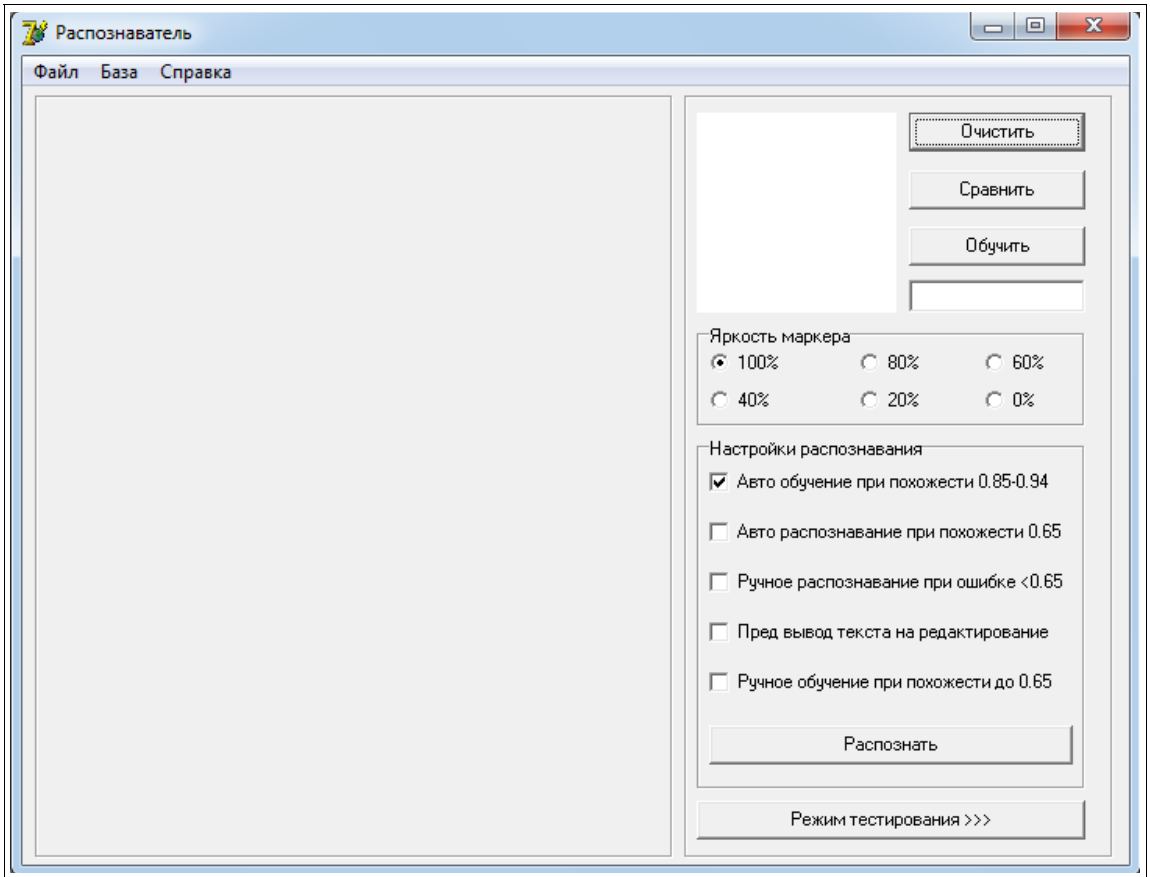


Рисунок 1. Основное окно программы

Для загрузки шаблонного изображения или открытия своего, необходимо в главном меню программы выбрать файл. «Открыть» позволяет выбрать любое bmp изображение, «Загрузить шаблон» открывает стандартный текст прилагающийся к программе test.bmp (рис. 2)

После нажатия кнопки «Распознать» в соответствии с выбранными параметрами будет произведено распознавание текста. Результат будет выведет в текстовый файл или поле редактирования.

Авто обучение при похожести 0.85-0.94: при выбранном параметре программа будет автоматически заносит новые эталоны в базу со значением наибольшей похожести среди текущих эталонов, но только в том случае, если максимальная среди всех похожестей не превышает 0.94 и не меньше 0.85.

Авто распознание при похожести 0.65: при выбранном параметре если при распознавании максимальная похожесть будет не ниже 0.65, то на выход будет поступать символ с этим значением похожести.

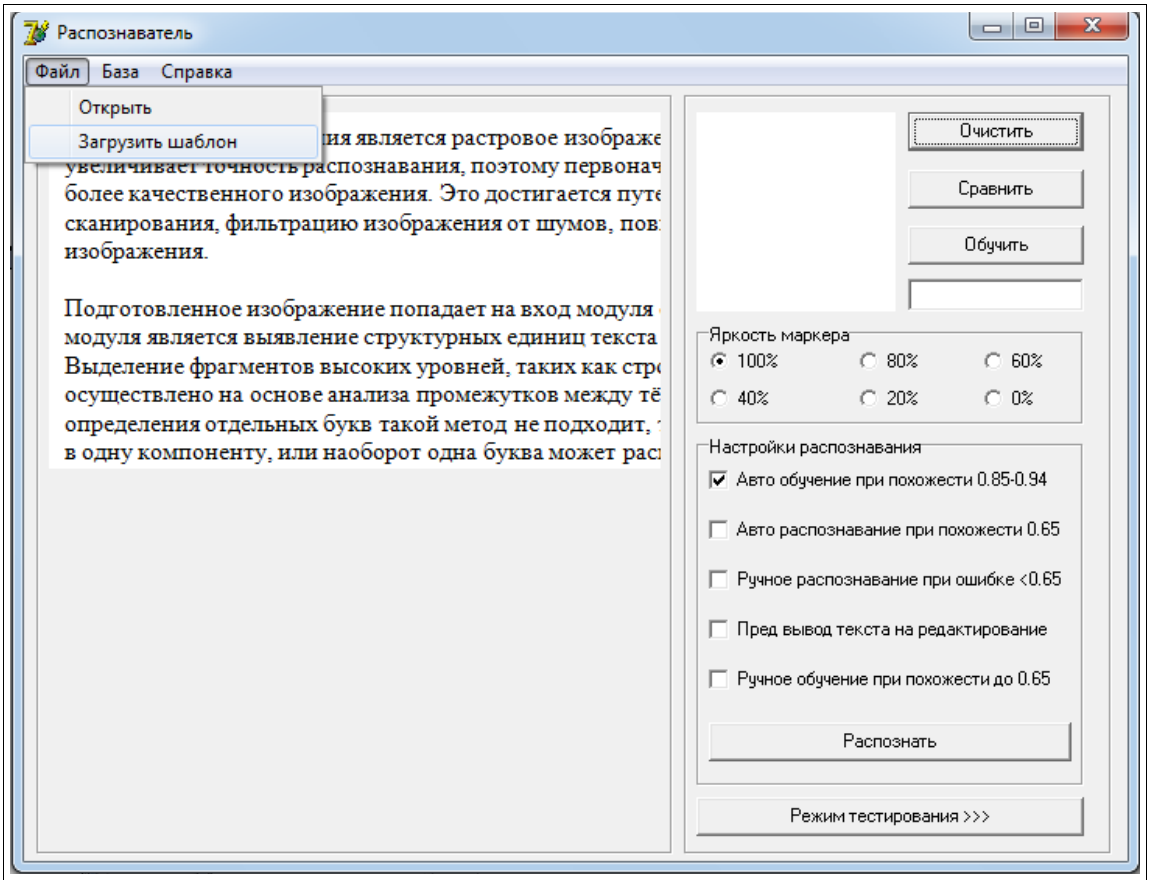


Рисунок 2. Загрузка стандартного шаблона

Ручное распознавание при ошибке <0.65: при выбранном параметре, если при распознавании максимальная похожесть будет ниже 0.65, то пользователю будет предложено значение символа с максимальной похожестью, а также возможность принять это значение или задать иное. Если параметр не выбран, на выход значение символа поступает как «?»

Пред вывод текста на редактирование: при выбранном параметре на выход текст поступает не в выходной файл, а в поле редактирования (рис. 3).

Ручное обучение при похожести до 0.65: при выбранном параметре, если при распознавании максимальная похожесть будет не ниже 0.65 и не выше 0.84, то пользователю будет предложено занести значение эталона в базу со значением символа с максимальной похожестью.

Поле эталонов позволяет вручную обучать программу новым эталонам. Для этого необходимо выбрать маркер с необходимой яркостью, нарисовать эталон в поле и задать ему значение справа от поля. После нажатия кнопки «обучить» символ добавляется в базу. Кнопка «очистить» позволяет полностью очистить поле эталонов. Кнопка «сравнить» позволяет сравнить нарисованный эталон со значениями в базе. Доступно только в режиме тестирования.

Режим тестирования позволяет проверить эффективность алгоритма классификации. Для этого необходимо нажать кнопку «Режим тестирования». Появится окно сравнения. После этого можно нарисовать изображение в поле эталонов, после нажатия кнопки «сравнить» в после сравнения будет выведены коэффициенты похожести на все изученные эталоны. Максимальный коэффициент выделен красным цветом (рис. 4).

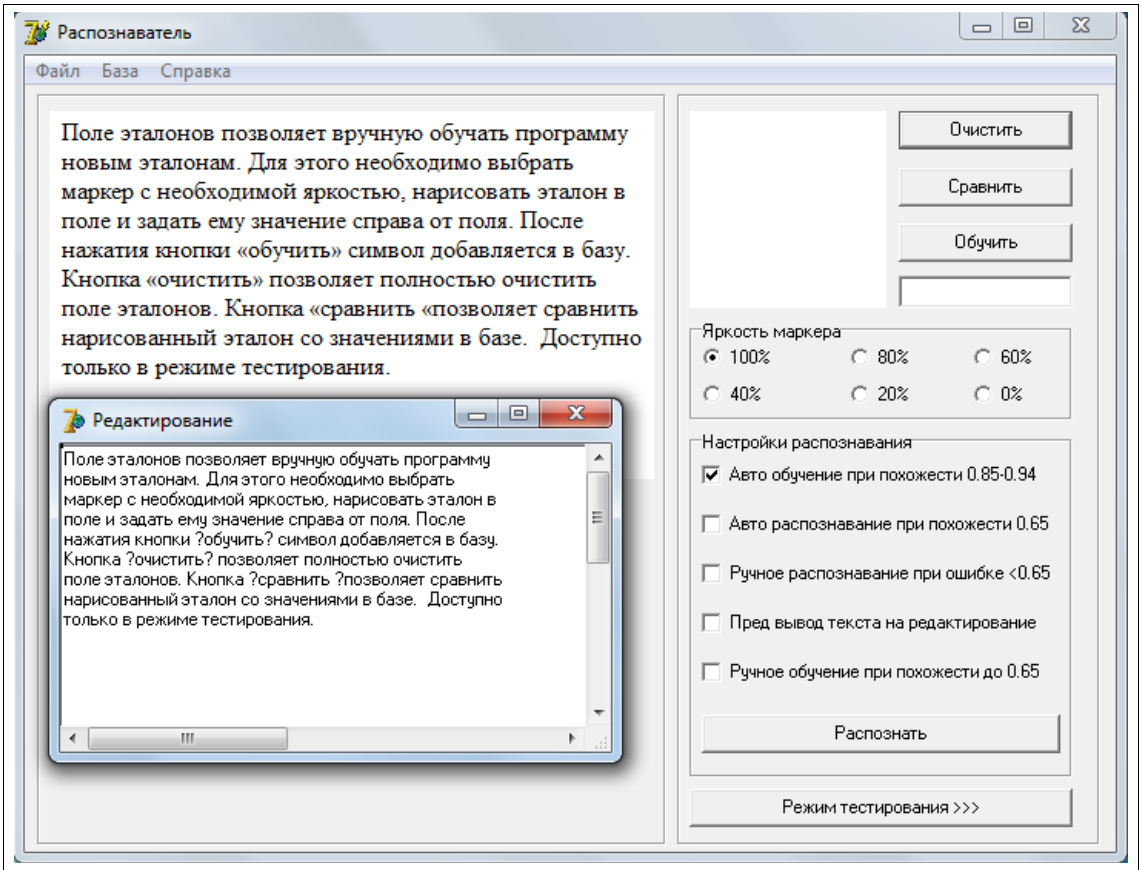


Рисунок 3. Вывод результата в поле редактирования

Нажатие кнопки «Режим распознания» снова уберет поле сравнений.

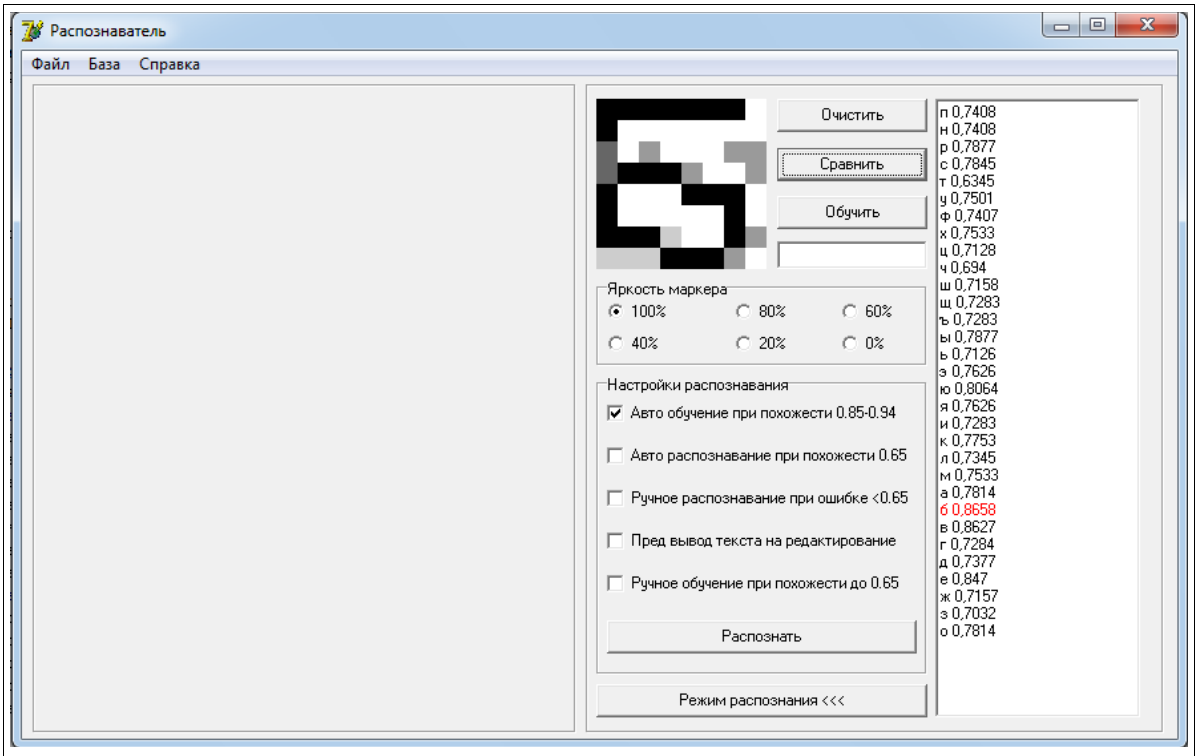


Рисунок 4. Режим тестирования