РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 648 638**(13) **C2**

(51) МПК
G06K 9/18 (2006.01)
G06K 9/62 (2006.01)
G06F 17/21 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06K 9/18 (2006.01); G06K 9/62 (2006.01); G06F 17/21 (2006.01); G06K 9/00456 (2006.01); G06K 9/6282 (2006.01)

(21)(22) Заявка: **2014103156, 30.01.2014**

(24) Дата начала отсчета срока действия патента:
30.01.2014

Дата регистрации:
26.03.2018

Приоритет(ы):
(22) Дата подачи заявки: **30.01.2014**

(43) Дата публикации заявки: **10.08.2015** Бюл. № 22

(45) Опубликовано: **26.03.2018** Бюл. № **9**

Адрес для переписки:
127273, Москва, а/я 56, ООО "Аби Девелопмент", Марей Сергей Владимирович

(72) Автор(ы):
Чулинин Юрий Георгиевич (RU)

(73) Патентообладатель(и):
Общество с ограниченной ответственностью "Аби Девелопмент" (RU)

(56) Список документов, цитированных в отчете о поиске: **RU 2260208 C2, 10.09.2005. RU 2251737 C2, 10.05.2005. RU 2439700 C1, 10.01.2012. US 2013/0121608 A1, 16.05.2013. GB 2500823 A, 02.10.2013.**

(54) Способы и системы эффективного автоматического распознавания символов, использующие множество кластеров эталонов символов

(57) Реферат:

Изобретение относится к области обработки изображений отсканированных документов и других изображений, содержащих текст. Технический результат заключается в повышении эффективности распознавания оптических символов. Технический результат достигается за счет идентификации изображений символов в содержащем текст отсканированном изображении документа; для каждой страницы документа, для каждого изображения символа на странице, идентификации каждой графемы из набора графем, которая соответствует нормированному изображению символа относительно эталона символа из набора эталонов символов, сортировки идентифицированных графем по частоте, с которой идентифицированные графемы соответствуют нормированному изображению символа относительно эталонов символа в наборе эталонов символов, и использования отсортированных идентифицированных графем для выбора кода символа, который представляет нормированное изображение символа; и подготовки обработанного документа, содержащего коды символов, которые представляют нормированные изображения символов из отсканированного изображения документа, и сохранения обработанного документа на одном или более из одного или более запоминающих устройств и модулей памяти. 3 н. и 17 з.п. ф-лы, 52 ил.

RU 2648638 C2

RU 2648638 C2

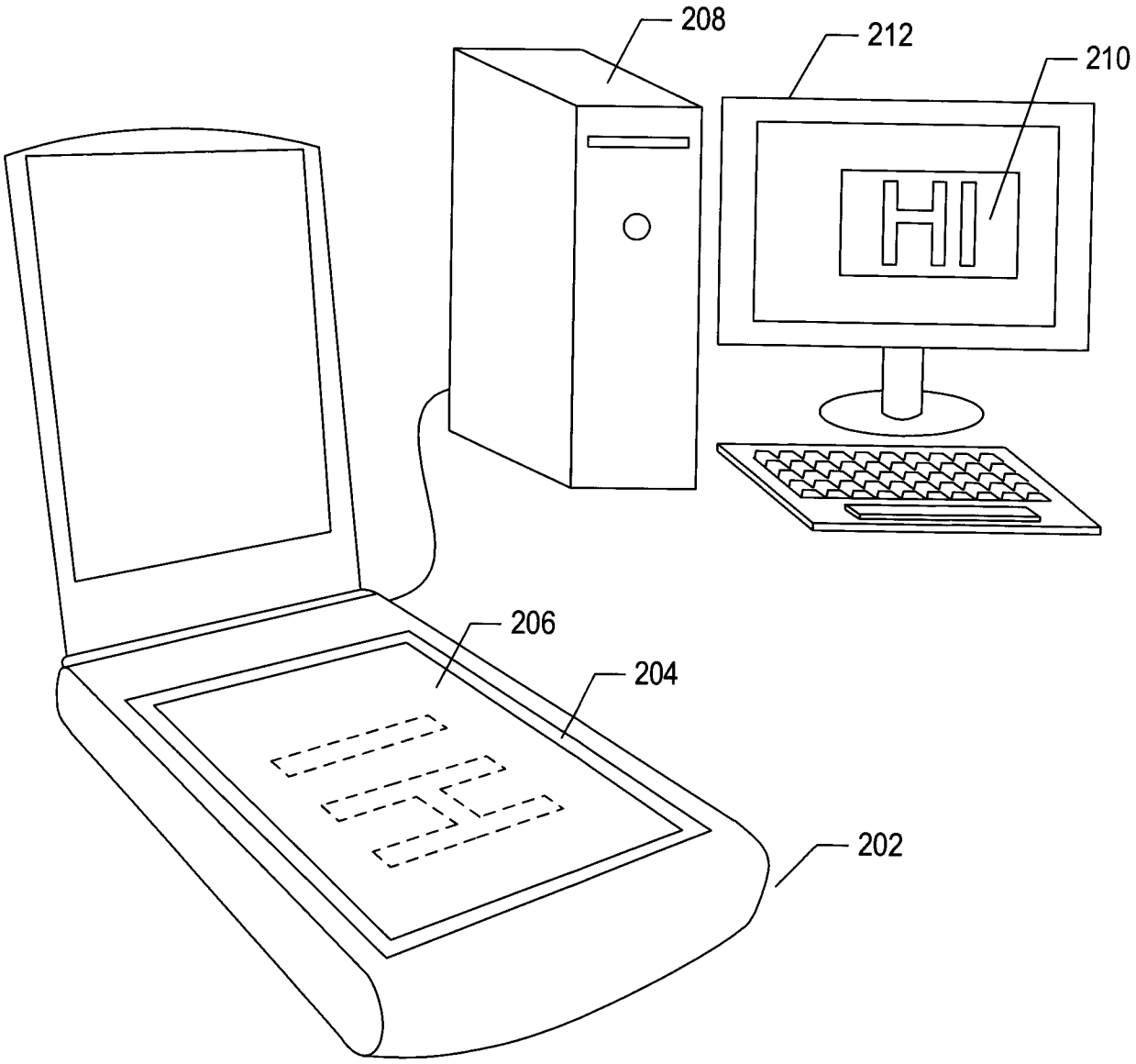


Рис. 2

RU 2648638 C2

RU 2648638 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 648 638**(13) **C2**

(51) Int. Cl.
G06K 9/18 (2006.01)
G06K 9/62 (2006.01)
G06F 17/21 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06K 9/18 (2006.01); G06K 9/62 (2006.01); G06F 17/21 (2006.01); G06K 9/00456 (2006.01); G06K 9/6282 (2006.01)

(21)(22) Application: **2014103156, 30.01.2014**

(24) Effective date for property rights:
30.01.2014

Registration date:
26.03.2018

Priority:
(22) Date of filing: **30.01.2014**

(43) Application published: **10.08.2015** Bull. № **22**

(45) Date of publication: **26.03.2018** Bull. № **9**

Mail address:
127273, Moskva, a/ya 56, OOO "Abi Development", Marej Sergej Vladimirovich

(72) Inventor(s):
Chulinin Yurij Georgievich (RU)

(73) Proprietor(s):
Obshchestvo s ogranichennoj otvetstvennostyu "Abi Development" (RU)

RU 2648638 C2

(54) METHODS AND SYSTEMS OF EFFECTIVE AUTOMATIC RECOGNITION OF SYMBOLS USING A MULTIPLE CLUSTERS OF SYMBOL STANDARDS

(57) Abstract:

FIELD: image processing means.

SUBSTANCE: invention relates to processing images of scanned documents and other images containing text. Technical result is achieved by identification of image symbols in the scanned document image containing text; for each page of the document, for each image of the symbol on the page, for identifying each grapheme from the set of graphemes, which corresponds to the normalized image of the symbol relative to the symbol standard from the set of symbol standards, sorting the identified graphemes by the frequency, with which the identified graphemes correspond to the normalized image of the symbol relative to the symbol standards in the set of symbol standards, and using the sorted identified graphemes to select the symbol code that represents the normalized image of the symbol; and preparing a processed document comprising character codes that represent normalized symbol images from the scanned image of the document, and storing the processed document on one or more than one from one or more than one memory devices and memory modules.

EFFECT: technical result is to increase the efficiency of recognition of optical symbols.

20 cl, 52 dwg

RU 2648638 C2

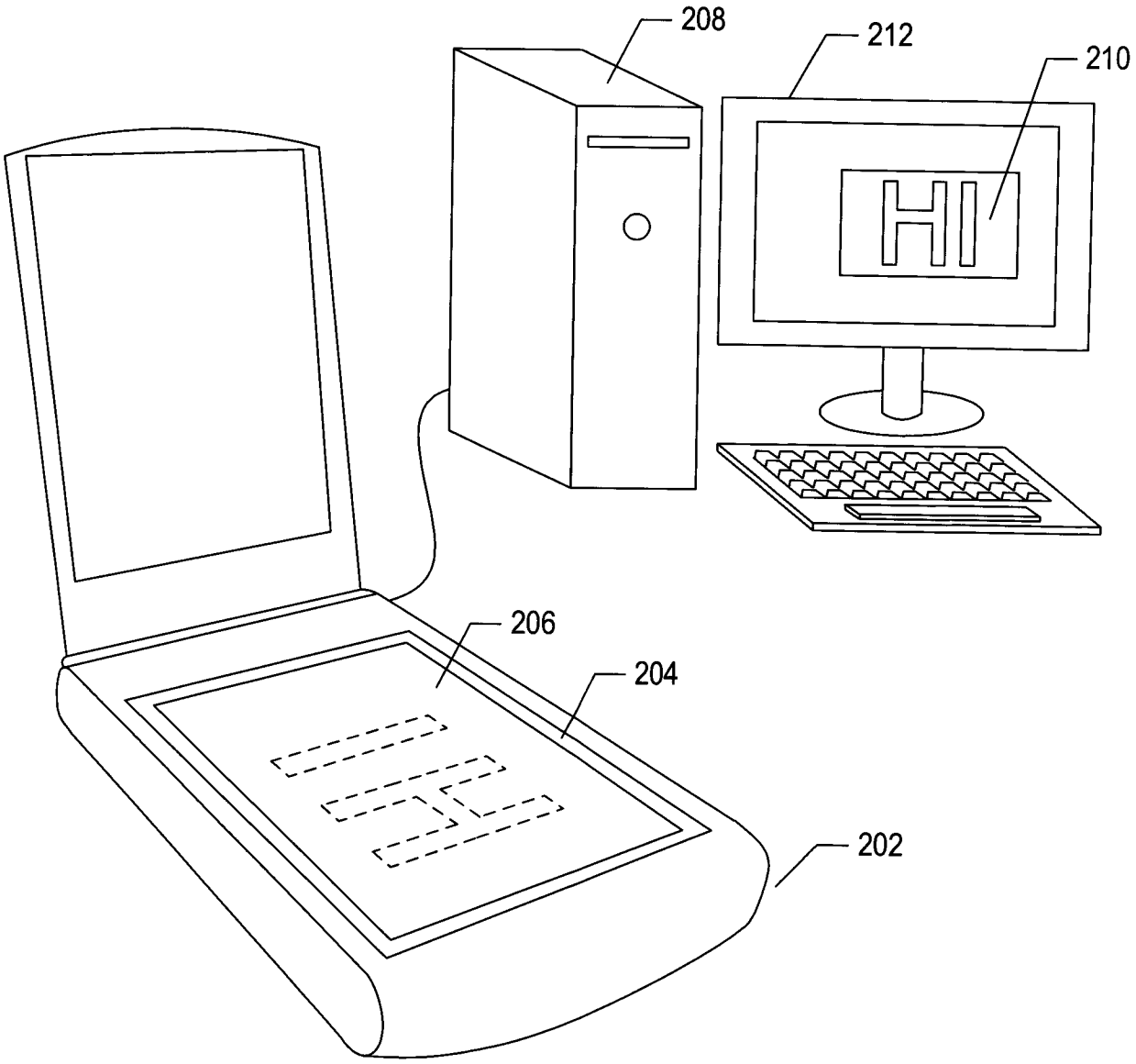


Рис. 2

RU 2648638 C2

RU 2648638 C2

ОБЛАСТЬ ТЕХНИКИ

Настоящая заявка относится к автоматической обработке изображений отсканированных документов и других изображений, содержащих текст, и, в частности, к способам и системам эффективного преобразования изображений символов, полученных из отсканированных документов, в кодовые комбинации соответствующих символов с использованием множества кластеров эталонов символов.

ПРЕДПОСЫЛКИ СОЗДАНИЯ ИЗОБРЕТЕНИЯ

Печатные, машинописные и рукописные документы на протяжении долгого времени используются для записи и хранения информации. Несмотря на современные тенденции отказа от бумажного делопроизводства, печатные документы продолжают широко использоваться в коммерческих организациях, учреждениях и домах. С развитием современных компьютерных систем создание, хранение, поиск и передача электронных документов превратились, наряду с непрекращающимся применением печатных документов, в чрезвычайно эффективный и экономически выгодный альтернативный способ записи и хранения информации. Из-за подавляющего преимущества в эффективности и экономической выгоде, обеспечиваемого современными средствами хранения и передачи электронных документов, печатные документы легко преобразуются в электронные с помощью различных способов и систем, включающих преобразование печатных документов в цифровые изображения отсканированных документов с использованием электронных оптико-механических сканирующих устройств, цифровых камер, а также других устройств и систем, и последующую автоматическую обработку изображений отсканированных документов для получения электронных документов, закодированных в соответствии с одним или более различными стандартами кодирования электронных документов. Например, в настоящее время можно использовать настольный сканер и современные программы оптического распознавания символов (OCR), позволяющие с помощью персонального компьютера преобразовывать печатный документ в соответствующий электронный документ, который можно просматривать и редактировать с помощью текстового редактора.

Хотя современные системы OCR развились до такой степени, что позволяют автоматически преобразовывать в электронные документы сложные печатные документы, включающие в себя изображения, рамки, линии границ и другие нетекстовые элементы, а также текстовые символы множества распространенных алфавитных языков, остается нерешенной проблема преобразования печатных документов, содержащих китайские и японские иероглифы или корейские морфо-слоговые блоки.

КРАТКОЕ ОПИСАНИЕ ИЗОБРЕТЕНИЯ

Настоящее изобретение относится к способам и системам распознавания символов, соответствующих изображениям символов, полученных из изображения отсканированного документа или другого изображения, содержащего текст, включая символы, соответствующие китайским или японским иероглифам или корейским морфо-слоговым блокам, а также символам других языков, в которых применяется большое количество знаков для записи и печати. В одном варианте осуществления способы и системы, описанные в настоящем изобретении, осуществляют стадию начальной обработки одного или более отсканированных изображений с целью идентификации набора графем для каждого изображения символа в отсканированном документе, которые с высокой частотой соответствуют эталонам символа, соответствующим, в свою очередь, изображению символа. Набор графем, определенных для изображения символа, связывается с изображением символа как набор графем-кандидатов для изображения символа. Набор графем-кандидатов затем используется в одной или более

последующих стадиях для того, чтобы связать каждое изображение символа с наиболее соответствующим кодом символа.

КРАТКОЕ ОПИСАНИЕ РИСУНКОВ

На рисунках 1A-B показан печатный документ.

На рисунке 2 показаны обычный настольный сканер и персональный компьютер, которые используются вместе для преобразования печатных документов в электронные, которые можно хранить на запоминающих устройствах и/или в электронной памяти.

На рисунке 3 показана работа оптических компонентов настольного сканера, изображенного на рисунке 2.

На рисунке 4 представлена общая архитектурная схема разных типов компьютеров и других устройств, управляемых процессором.

На рисунке 5 показано цифровое представление отсканированного документа.

На рисунке 6 показан гипотетический набор символов.

На рисунках 7A-B показаны различные аспекты наборов символов для естественных языков.

На рисунках 8A, B показаны параметры и значения параметров, рассчитанные для изображений символов.

На рисунке 9 показана таблица значений параметров, рассчитанных для всех символов из набора, изображенного в качестве примера на рисунке 6.

На рисунке 10 показан трехмерный график для символов из набора, изображенного в качестве примера на рисунке 6, на котором каждое из измерений представляет значения одного из трех разных параметров.

На рисунках 11A, B показаны символы, содержащиеся в каждом из кластеров, представленных точками трехмерного пространства, изображенного на рисунке 10.

На рисунке 12A показан отдельный параметр, который можно использовать в комбинации с тремя параметрами, соответствующими каждому из измерений трехмерного пространства параметров, изображенного на рисунке 10, для полного распознавания каждого из символов в кластере 8 (позиции 1212 и 1214 - фрагменты псевдокода компьютерной программы не переводятся на русский язык).

На рисунке 12B показано значение дополнительного параметра для каждого символа из кластера 8, которое следует рассматривать со ссылкой на рисунок 12A.

На рисунке 13 показано небольшое изображение, содержащее текст, которое было изначально обработано системой OCR для получения сетки окон символов 1300, в каждом из которых содержится изображение символа.

На рисунке 14 показан общий подход к обработке сетки окон символов, показанной на рисунке 13 (1404 на рисунке 14 компьютерная программа не переводится на русский язык).

На рисунке 15 показан первый подход к реализации подпрограммы «process» (компьютерная программа не переводится на русский язык).

На рисунках 16A, B показан второй вариант осуществления подпрограммы «process» (компьютерная программа не переводится на русский язык).

На рисунке 17 показан третий вариант осуществления подпрограммы «process», рассмотренной в предыдущем подразделе, с использованием тех же иллюстраций и условных обозначений в псевдокоде, которые использовались в предыдущем подразделе. (компьютерная программа не переводится на русский язык).

На рисунке 18 показаны структуры данных, обеспечивающие кластеризацию и предварительную обработку в одном варианте осуществления системы OCR, включающей в себя третий вариант осуществления подпрограммы «process», описанный

выше.

На рисунках 19А-З показана предварительная обработка изображения символа с использованием структур данных, рассмотренных выше со ссылкой на рисунок 18.

На рисунках 20А-П показана мультикластерная OCR-обработка документа, содержащего изображения символов.

На рисунках 21А-Г с помощью блок-схемы показан один из вариантов осуществления способа мультикластерной OCR-обработки документа.

ПОДРОБНОЕ ОПИСАНИЕ ИЗОБРЕТЕНИЯ

Настоящее изобретение относится к способам и системам распознавания символов, соответствующих изображениям символов, полученных из изображения отсканированного документа. В одном варианте осуществления в способах и системах, рассматриваемых в настоящем изобретении, на стадии начальной обработки одного или более отсканированных изображений определяется частота, с которой каждая графема из набора графем связывается с эталонами символов, соответствующими изображениям символов отсканированного документа или изображений. Для каждого эталона символа подсчитывается число, характеризующее частоту связей с графемами, на которые ссылаются эталоны символов, затем эталоны символов в каждом кластере эталонов сортируются по данным числам. Упорядочивание эталонов приводит к тому, что на следующей стадии оптического распознавания символов, в которой изображения символов связываются с одной или более графемами или кодировками символов, наиболее похожие эталоны символов встречаются в первую очередь.

Изображения отсканированных документов и электронные документы

На рисунках 1А-В показан печатный документ. На рисунке 1А показан исходный документ с текстом на японском языке. Печатный документ 100 включает в себя фотографию 102 и пять разных содержащих текст областей (104-108), включающих в себя японские иероглифы. Этот документ будет использоваться в качестве примера при рассмотрении способа и систем определения смысла, к которым относится настоящая заявка. Текст на японском языке может писаться слева направо, построчно, как пишется текст на английском языке, но альтернативно может использоваться способ написания сверху вниз в вертикальных столбцах. Например, область 107 явно содержит вертикально написанный текст, в то время как текстовый блок 108 содержит текст, написанный горизонтально. На рисунке 1В печатный документ, изображенный на рисунке 1А, показан переведенным на русский язык.

Печатные документы могут быть преобразованы в цифровые изображения отсканированных документов с помощью различных средств, включающих электронные оптико-механические сканирующие устройства и цифровые камеры. На рисунке 2 показаны обычный настольный сканер и персональный компьютер, которые используются вместе для преобразования печатных документов в электронные, которые можно хранить на запоминающих устройствах и/или в электронной памяти. Настольное сканирующее устройство 202 включает в себя прозрачное стекло 204, на которое лицевой стороной вниз помещается документ 206. Запуск сканирования приводит к получению оцифрованного изображения отсканированного документа, которое можно передать на персональный компьютер (ПК) 208 для хранения на запоминающем устройстве. Программа, предназначенная для отображения отсканированного документа, может вывести оцифрованное изображение отсканированного документа на экран 210 устройства отображения ПК 212.

На рисунке 3 показана работа оптических компонентов настольного сканера, изображенного на рисунке 2. Оптические компоненты этого CCD-сканера расположены

под прозрачным стеклом 204. Фронтально перемещаемый источник яркого света 302 освещает часть сканируемого документа 304, свет от которой отражается вниз. Этот свет отражается от фронтально перемещаемого зеркала 306 на неподвижное зеркало 308, которое отражает излучаемый свет на массив CCD-элементов 310, формирующих электрические сигналы пропорционально интенсивности света, поступающего на каждый из них. Цветные сканеры могут включать в себя три отдельные строки или массива CCD-элементов с красным, зеленым и синим фильтрами. Фронтально перемещаемые источник яркого света и зеркало двигаются вместе вдоль документа для получения изображения сканируемого документа. Другой тип сканера, использующего контактный датчик изображения, называется CIS-сканером. В CIS-сканере подсветка документа осуществляется перемещаемыми цветными светодиодами (LED), при этом отраженный свет светодиодов улавливается массивом фотодиодов, который перемещается вместе с цветными светодиодами.

На рисунке 4 представлена общая архитектурная схема разных типов компьютеров и других устройств, управляемых процессором. Архитектурная схема высокого уровня позволяет описать современную компьютерную систему (например, ПК, изображенный на рисунке 2), в которой программы отображения отсканированного документа и программы оптического распознавания символов хранятся на запоминающих устройствах для передачи в электронную память и выполнения одним или более процессорами, что позволяет преобразовать компьютерную систему в специализированную систему оптического распознавания символов. Компьютерная система содержит один или множество центральных процессоров (ЦП) 402-405, один или более модулей электронной памяти 408, соединенных с ЦП при помощи шины подсистемы ЦП/память 410 или множества шин, первый мост 412, который соединяет шину подсистемы ЦП/память 410 с дополнительными шинами 414 и 416 или другими средствами высокоскоростного взаимодействия, включающими в себя множество высокоскоростных последовательных линий. Эти шины или последовательные линии, в свою очередь, соединяют ЦП и память со специализированными процессорами, такими как графический процессор 418, а также с одним или более дополнительными мостами 420, взаимодействующими с высокоскоростными последовательными линиями или с множеством контроллеров 422-427, например с контроллером 427, которые предоставляют доступ к различным типам запоминающих устройств 428, электронным дисплеям, устройствам ввода и другим подобным компонентам, подкомпонентам и вычислительным ресурсам.

На рисунке 5 показано цифровое представление отсканированного документа. На рисунке 5 небольшой круглый фрагмент изображения 502 печатного документа 504, используемого в качестве примера, показан в увеличенном виде 506. Соответствующий фрагмент оцифрованного изображения отсканированного документа 508 также представлен на рисунке 5. Оцифрованный отсканированный документ включает в себя данные, которые представляют собой двухмерный массив значений пикселей. В представлении 508 каждая ячейка сетки под символами (например, ячейка 509) представляет собой квадратную матрицу пикселей. Небольшой фрагмент 510 сетки показан с еще большим увеличением (512 на рисунке 5), при котором отдельные пиксели представлены в виде элементов матрицы (например, элемента матрицы 514). При таком уровне увеличения края символов выглядят зазубренными, поскольку пиксель является наименьшим элементом детализации, который можно использовать для излучения света заданной яркости. В файле оцифрованного отсканированного документа каждый пиксель представлен фиксированным числом битов, при этом кодирование пикселей

осуществляется последовательно. Заголовок файла содержит информацию о типе кодировки пикселей, размерах отсканированного изображения и другую информацию, позволяющую программе отображения оцифрованного отсканированного документа получать данные кодирования пикселей и передавать команды устройству отображения или принтеру с целью воспроизведения двухмерного изображения исходного документа по этим кодировкам. Для представления оцифрованного отсканированного документа в виде монохромных изображений с оттенками серого обычно используют 8-разрядное или 16-разрядное кодирование пикселей, в то время как при представлении цветного отсканированного изображения может выделяться 24 или более бит для кодирования каждого пикселя, в зависимости от стандарта кодирования цвета. Например, в широко применяемом стандарте RGB для представления интенсивности красного, зеленого и синего цветов используются три 8-разрядных значения, закодированных с помощью 24-разрядного значения. Таким образом, оцифрованное отсканированное изображение, по существу, представляет собой документ в той же степени, в какой цифровые фотографии представляют визуальные образы. Каждый закодированный пиксель содержит информацию о яркости света в определенных крошечных областях изображения, а для цветных изображений в нем также содержится информация о цвете. В оцифрованном изображении отсканированного документа отсутствует какая-либо информация о значении закодированных пикселей, например информация, что небольшая двухмерная зона соседних пикселей представляет собой текстовый символ. Фрагменты изображения, соответствующие изображениям символов, могут обрабатываться для получения битов изображения символа, в котором биты со значением «1» соответствуют изображению символа, а биты со значением «0» соответствуют фону. Растровое отображение удобно для представления как полученных изображений символов, так и эталонов, используемых системой OCR для распознавания конкретных символов.

В отличие от этого обычный электронный документ, созданный с помощью текстового редактора, содержит различные типы команд рисования линий, ссылки на представления изображений, таких как оцифрованные фотографии, а также текстовые символы, закодированные в цифровом виде. Одним из наиболее часто используемых стандартов для кодирования текстовых символов является стандарт Юникод. В стандарте Юникод обычно применяется 8-разрядный байт для кодирования символов ASCII и 16-разрядные слова для кодирования символов и знаков множества языков, включая японский, китайский и другие неалфавитные текстовые языки. Большая часть вычислительной работы, которую выполняет программа OCR, связана с распознаванием изображений текстовых символов, полученных из оцифрованного изображения отсканированного документа, и с преобразованием изображений символов в соответствующие кодовые комбинации стандарта Юникод. Очевидно, что для хранения текстовых символов стандарта Юникод будет требоваться гораздо меньше места, чем для хранения растровых изображений текстовых символов. Кроме того, текстовые символы стандарта Юникод можно редактировать, используя различные шрифты, а также обрабатывать всеми доступными в текстовых редакторах способами, в то время как оцифрованные изображения отсканированного документа можно изменить только с помощью специальных программ редактирования изображений.

На начальном этапе преобразования изображения отсканированного документа в электронный документ печатный документ (например, документ 100, показанный на рисунке 1) анализируется для определения в нем различных областей. Во многих случаях области могут быть логически упорядочены в виде иерархического ациклического

дерева, состоящего из корня, представляющего документ как единое целое, промежуточных узлов, представляющих области, содержащие меньшие области, и конечных узлов, представляющих наименьшие области. Дерево, представляющее документ, включает в себя корневой узел, соответствующий всему документу, и шесть конечных узлов, каждый из которых соответствует одной определенной области. Области можно определить, применяя к изображению разные методы, среди которых различные типы статистического анализа распределения пикселей или значений пикселей. Например, в цветном документе фотографию можно выделить по большему изменению цвета в области фотографии, а также по более частым изменениям значений яркости пикселей по сравнению с областями, содержащими текст.

Как только начальный анализ выявит различные области на изображении отсканированного документа, области, которые с большой вероятностью содержат текст, дополнительно обрабатываются подпрограммами OCR для выявления и преобразования текстовых символов в символы стандарта Юникод или любого другого стандарта кодировки символов. Для того чтобы подпрограммы OCR могли обработать содержащие текст области, определяется начальная ориентация содержащей текст области, благодаря чему в подпрограммах OCR эффективно используются различные способы сопоставления с эталоном для определения текстовых символов. Следует отметить, что изображения в документах могут быть не выровнены должным образом в рамках изображений отсканированного документа из-за погрешности в позиционировании документа на сканере или другом устройстве, формирующем изображение, из-за нестандартной ориентации содержащих текст областей или по другим причинам. Области, содержащие текст, затем делят на фрагменты изображений, содержащие отдельные знаки или символы, после чего эти фрагменты целиком масштабируются и ориентируются, а изображения символов центрируются внутри этих фрагментов для облегчения последующего автоматического распознавания символов, соответствующих изображениям символов.

Пример методов и систем OCR

Для перехода к конкретному обсуждению различных методов оптического распознавания символов в качестве примера будет использоваться набор символов для некоторого гипотетического языка. На рисунке 6 показан гипотетический набор символов. На рисунке 6 показаны 48 различных символов, расположенных в 48 прямоугольных областях, таких как прямоугольная область 602. В правом верхнем углу каждой прямоугольной области указан числовой индекс или код символа, вписанный в круг; например, индекс или код «1» 604, соответствует первому символу 606, показанному в прямоугольной области 602. Данный пример выбран для демонстрации работы как существующих в настоящее время способов и систем OCR, так и новых способов и систем, описанных в настоящем документе. Фактически для письменных иероглифических языков, включая китайский и японский языки, для печати и письма могут использоваться десятки тысяч различных символов.

На рисунках 7A-B показаны различные аспекты наборов символов для естественных языков. На рисунке 7A в столбце показаны различные формы изображения восьмого символа из набора, показанного на рисунке 6. В столбце 704 для восьмого символа 702 из набора символов, показанного на рисунке 6, представлены разные формы написания, встречающиеся в разных стилях текста. Во многих естественных языках могут использоваться различные стили текста, а также различные варианты написания каждого символа.

На рисунке 7B показаны разные подходы к распознаванию символов естественного

языка. На рисунке 7В конкретный символ естественного языка представлен узлом 710 на схеме 712. Конкретный символ может иметь множество различных общих письменных или печатных форм. В целях оптического распознавания символов каждая из этих общих форм представляется в виде отдельной графемы. В некоторых случаях определенный символ может содержать две или более графем. Например, китайские иероглифы могут содержать комбинацию из двух или более графем, каждая из которых присутствует в других иероглифах. Корейский язык, на самом деле, основан на алфавите, при этом в нем используются корейские морфо-слоговые блоки, содержащие ряд буквенных символов в различных позициях. Таким образом, корейский морфо-слоговой блок может представлять собой символ более высокого уровня, состоящий из множества компонентов графем. Для символа 710, показанного на рисунке 7В, существуют шесть различных графем 714-719. Кроме того, есть одна или более различных печатных или письменных форм начертания графем, каждая из которых представлена соответствующим эталоном. На рисунке 7В каждая из графем 714 и 716 имеет два возможных варианта начертания, представленных эталонами 720-721 и 723-724 соответственно. Каждая из графем 715 и 717-719 связана с одним эталоном 722 и 725-727 соответственно. Например, восьмой символ из набора, показанного в качестве примера на рисунке 6, может быть связан с тремя графемами, первая из которых соответствует начертаниям 702, 724, 725 и 726, вторая - 728 и 730, а третья - 732. В этом примере к первой графеме относятся начертания, в которых используются прямые горизонтальные элементы, ко второй графеме относятся начертания, в которых используются горизонтальные элементы и короткие вертикальные элементы с правой стороны, а к третьей графеме относятся начертания, включающие в себя изогнутые (а не прямые) элементы. В альтернативном варианте осуществления все начертания восьмого символа 702, 728, 724, 732, 725, 726 и 730 можно представить в виде эталонов, связанных с единственной графемой для восьмого символа. В определенной степени выбор графем осуществляется произвольно. В некоторых типах иероглифических языков много тысяч разных графем. Эталоны можно рассматривать в качестве альтернативного представления или изображения символа, при этом они могут быть представлены в виде набора пар «параметр - значение параметра», как описано ниже.

Хотя отношение между символами, графемами и эталонами показано на рисунке 7Б как строго иерархическое, при котором каждая графема связана с одним конкретным родительским символом, фактические отношения не могут быть так просто структурированы. На рисунке 7В показан несколько более сложный набор отношений, когда два символа 730 и 732 являются родительскими для двух разных графем 734 и 736. В качестве еще одного примера можно привести следующие символы английского языка: строчная буква «о», прописная буква «О», цифра «0» и символ градусов «°», которые могут быть связаны с кольцеобразной графемой. Отношения могут быть альтернативно представлены в виде графов или сетей. В некоторых случаях графемы (в отличие от символов или в дополнение к ним) могут отображаться на самых высоких уровнях в рамках выбранного представления отношений. В сущности, идентификация символов, графем, выбор эталонов для конкретного языка, а также определение отношений между ними осуществляются в большой степени произвольно.

На рисунках 8А-В показаны параметры и значения параметров, рассчитанные для изображений символов. Следует заметить, что словосочетание «изображение символа» может описывать печатный, рукописный или отображаемый на экране символ или графему. В следующем примере параметры и значения параметров рассматриваются применительно к изображениям символов, но в фактическом контексте реального

языка параметры и значения параметров часто применяются для характеристики и представления изображений графем. На рисунке 8A показано изображение прямоугольного символа 802, полученное из содержащего текст изображения, которое соответствует 22-му символу из набора, показанного в качестве примера на рисунке 6. На рисунке 8B показано изображение прямоугольного символа 804, полученное из содержащего текст изображения, которое соответствует 48-му символу из набора, показанного в качестве примера на рисунке 6. При печати и письме на гипотетическом языке, соответствующем набору символов, приведенному в качестве примера, символы размещаются в середине прямоугольных областей. Если это не так, системы OCR выполнят стадию начальной обработки изображений, изменив ориентацию, масштаб и положение полученных изображений символов относительно фоновой области для нормализации полученных изображений символов для дальнейших стадий обработки.

На рисунке 8A показаны три разных параметра, которые могут использоваться системой OCR для характеристики символов. Следует заметить, что область изображения символа, или окно символа, характеризуется вертикальным размером окна символа 806, обозначаемым сокращенно «vw», и горизонтальным размером окна символа 808, обозначаемым сокращенно «hw». Первым параметром является самый длинный в изображении символа непрерывный горизонтальный отрезок линии, обозначаемый «h» 810. Это самая длинная последовательность смежных темных пикселей на фоне по существу белых пикселей в окне символа. Вторым параметром является самый длинный в изображении символа непрерывный вертикальный отрезок линии 812. Третий параметр представляет собой отношение количество пикселей изображения символа к общему числу пикселей в окне символа, выраженное в процентах; в данном примере это процент черных пикселей в по существу белом окне символа. Во всех трех случаях значения параметров могут быть непосредственно рассчитаны сразу после того, как будет создано растровое отображение окна символа. На рисунке 8B показаны два дополнительных параметра. Первым параметром является число внутренних горизонтальных белых полос в изображении символа; изображение символа, показанного на рисунке 8B, имеет одну внутреннюю горизонтальную белую полосу 816. Вторым параметром является число внутренних вертикальных белых полос в изображении символа. В 48-м символе из набора, представленном изображением в окне символа 804 на рисунке 8B, имеется одна внутренняя вертикальная белая полоса 818. Число горизонтальных белых полос обозначается как «hs», а число внутренних вертикальных белых полос - «vs».

На рисунке 9 показана таблица значений параметров, рассчитанных для всех символов из набора, изображенного в качестве примера на рисунке 6. В каждой строке таблицы 902, показанной на рисунке 9, представлены значения параметров, рассчитанные для конкретного символа. Следующие параметры включают в себя: (1) отношение самого длинного непрерывного горизонтального отрезка линии к окну символа, $\frac{\mathbf{h}}{\mathbf{l...}}$, 904; (2) отношение самого длинного непрерывного вертикального отрезка линии к вертикальному размеру окна символа, $\frac{\mathbf{v}}{\mathbf{—}}$, 906; (3) выраженная в процентах общая площадь, соответствующая изображению символа или черной области, b, 908; (4) количество внутренних вертикальных полос, vs, 910; (5) количество внутренних горизонтальных полос, hs, 912; (6) общее количество внутренних вертикальных и

горизонтальных полос, vs+hs, 914; и (7) отношение самого длинного непрерывного вертикального отрезка к самому длинному непрерывному горизонтальному отрезку, $\frac{h}{l}$, 916. Как и следовало ожидать, в первой строке 920 таблицы 902, представленной на рисунке 9, первый символ набора (606 на рисунке 6) представляет собой вертикальную черту, и численное значение параметра $\frac{v}{-}$, равное 0,6, значительно больше численного значения параметра $\frac{h}{l...}$, равного 0,2. Символ 606 занимает всего 12 процентов всего окна символа 602. У символа 606 нет ни внутренних горизонтальных, ни внутренних вертикальных белых полос, поэтому значения параметров vs, hs и vs+hs равны 0. Соотношение $\frac{v}{l.}$ равно 3. Поскольку используемые в качестве примера символы имеют относительно простую блочную структуру, то значения каждого из параметров в таблице 902 отличаются незначительно.

Несмотря на то что значения каждого из параметров, рассмотренных выше в отношении рисунке 9, имеют относительно небольшие отличия для используемых в качестве примера 48 символов, всего трех параметров достаточно для разделения всех этих символов на 18 частей, или кластеров. На рисунке 10 показан трехмерный график для символов из набора, изображенного в качестве примера на рисунке 6, на котором каждое из измерений представляет значения одного из трех разных параметров. На рисунке 10 первая горизонтальная ось 1002 представляет параметр $\frac{v}{l.}$ (916 на рисунке 9), вторая горизонтальная ось 1004 представляет параметр vs+hs (914 на рисунке 9), а третья вертикальная ось 1006 представляет параметр b (908 на рисунке 9). На графике есть 18 различных точек (таких как нанесенная точка 1008), каждая из которых показана в виде небольшого черного диска с вертикальной проекцией на горизонтальную плоскость, проходящую через оси 1002 и 1004; эта проекция представлена в виде вертикальной пунктирной линии, такой как вертикальная пунктирная линия 1010, соединяющая точку 1008 с ее проекцией на горизонтальную плоскость 1012. Код или номер последовательности символов, которые соответствуют определенной точке на графике, перечислены в скобках справа от соответствующей точки. Например, символы 14, 20 и 37 (1014) соответствуют одной точке 1016 с координатами (1, 0, 0,32) относительно осей 1002, 1004 и 1006. Каждая точка связана с номером части или кластера, который указан в небольшом прямоугольнике слева от точки. Например, точка 1016 связана с кластером под номером «14» 1018. На рисунках 11A-B показаны символы, содержащиеся в каждом из кластеров, представленных точками трехмерного пространства, изображенного на рисунке 10. Рассмотрев символы, входящие в состав этих кластеров или частей, можно легко заметить, что три параметра, используемые для распределения символов в трехмерном пространстве, показанном на рисунке 10, эффективно разбивают 48 символов, используемых в качестве примера, на связанные наборы символов.

Можно использовать дополнительные параметры для однозначного распознавания каждого символа в каждом кластере или части. Рассмотрим, например, кластер 8 (1102),

показанный на рисунке ПА. Этот кластер символов включает в себя четыре угловых (L-образных) символа, отличающихся углом поворота и имеющих коды 26, 32, 38 и 44, а также Т-образный символ с кодом 43 и крестообразный символ с кодом 45. На рисунке 12А показан отдельный параметр, который можно использовать в комбинации с тремя параметрами, соответствующими каждому из измерений трехмерного пространства параметров, изображенного на рисунке 10, для полного распознавания каждого из символов в кластере 8. Как показано на рисунке 12А, окно символа 1202 делится на четыре квадранта: Q1 1204, Q2 1205, Q3 1206 и Q4 1207. После этого в каждом квадранте вычисляется площадь, занимаемая изображением символа, которая указывается рядом с квадрантом. Например, в квадранте Q1 1204 часть изображения символа занимает 13,5 единиц площади 1210. Затем вычисленные значения единиц площади каждого квадранта присваиваются переменным Q1, Q2, Q3 и Q4. Следовательно, в примере, представленном на рисунке 12А, переменной Q1 присвоено значение 13,5, переменной Q2 присвоено значение 0, переменной Q3 присвоено значение 18, а переменной Q4 присвоено значение 13,5. Затем согласно небольшому фрагменту псевдокода 1212, представленному на рисунке 12А под окном символа, рассчитывается значение нового параметра p. Например, если все четыре переменные Q1, Q2, Q3 и Q4 имеют одинаковые значения, то параметру p будет присвоено значение 0 (1214), что указывает на равенство четырех квадрантов в окне символа относительно количества единиц площади, занимаемой изображением символа. На рисунке 12 В показано значение дополнительного параметра для каждого символа из кластера 8, которое следует рассматривать со ссылкой на рисунок 12А. Как можно увидеть из значений параметров, связанных с символами на рисунке 12В, новый параметр, описанный выше касательно рисунка 12А, имеет разное значение для каждого из шести символов в кластере 8. Другими словами, можно использовать комбинацию трех параметров, используемых для создания трехмерного графика, показанного на рисунке 10, и дополнительного параметра, рассмотренного выше на рисунке 12А, для однозначной идентификации всех символов в кластере 8.

На рисунке 13 показано небольшое изображение, содержащее текст, которое было изначально обработано системой OCR для получения сетки окон символов 1300, в каждом из которых содержится изображение символа. Для большей наглядности на рисунке 13 показана сетка окон символов 1300, не содержащая изображений символов. Для упорядочивания окон символов используется вертикальный индекс i 1302 и горизонтальный индекс j 1304. Для облегчения понимания примера, рассматриваемого ниже, в нем будет идти речь о символах и изображениях символов, а не о графемах. В этом примере предполагается, что существует однозначное соответствие между символами, графемами и эталонами, используемыми для идентификации изображений символов в окнах символов. Кроме сетки окон символов 1300, на рисунке 13 также показан массив, или матрица, 1306 эталонов, каждая ячейка которой (например, ячейка 1308) включает в себя эталон. Эталоны представляют собой наборы пар «параметр-значение параметра», где параметры выбираются для однозначного распознавания изображений символов, как было описано выше со ссылкой на рисунки 8А-12В. На рисунке 13 также показан массив параметров 1310, представленный в виде набора пар фигурных скобок, таких как пара фигурных скобок 1312. Каждая пара фигурных скобок представляет собой функционал, который рассчитывает значение параметра относительно изображения символа.

На рисунке 14 показан общий подход к обработке сетки окон символов, показанной на рисунке 13. На самом высоком уровне обработка может рассматриваться как

вложенный цикл for 1402, в котором вызывается подпрограмма «process» 1404 для анализа каждого окна символа 1406 с целью формирования соответствующего кода символа 1408. Другими словами, в примере с псевдокодом сетка окон символов представляет собой двухмерный массив «page_of_text», а система OCR формирует двухмерный массив кодов символов «processed_text» на основе двухмерного массива окон символов «page_of_text». На рисунке 14 дугообразные стрелки, такие как дугообразная стрелка 1410, используются для демонстрации порядка обработки первой строки двухмерного массива или сетки окон символов 1300, а горизонтальные стрелки, такие как стрелка 1412, показывают обработку следующих строк, осуществляемую в цикле for 1402. Другими словами, сетка окон символов 1300 обрабатывается согласно указанному выше порядку обработки, при этом каждое окно символа в сетке обрабатывается отдельно для формирования соответствующего кода символа.

На рисунке 15 показан первый подход к реализации подпрограммы «process» (1404 на рисунке 14). Изображение символа, находящееся в окне символа 1502, используется в качестве входного параметра для подпрограммы «process». Подпрограмма «process» используется для расчета значений восьми разных параметров p1-p8, используемых в данном примере для получения отличительных признаков изображений символов путем восьмикратного вызова подпрограммы «parameterize» 1504, как показано на рисунке 15. Подпрограмма «parameterize» использует в качестве аргументов изображение символа и целочисленное значение, указывающее, для какого параметра необходимо рассчитать и вернуть рассчитанное значение. Значения параметров хранятся в массиве значений параметров «p_values». Затем, как показано дугообразными стрелками, такими как дугообразная стрелка 1506, подпрограмма «process» перебирает все эталоны 1508, соответствующие символам языка, сравнивая рассчитанные значения параметров для изображения символа, хранящиеся в массиве «p_values», с предварительно рассчитанными значениями параметров каждого эталона, как показано на иллюстрации операции сравнения 1510 на рисунке 15. Эталон, параметры которого больше всего соответствуют рассчитанным параметрам для изображения символа, выбирается в качестве эталона соответствия, а код символа, который соответствует этому эталону, используется в качестве возвращаемого значения подпрограммы «process». В качестве примера псевдокода, используемого для этого первого варианта осуществления подпрограммы «process», приведен псевдокод 1512 на рисунке 15. В первом цикле for 1514 рассчитываются значения параметров для входного символа s. Затем в нескольких вложенных циклах for внешнего цикла for 1516 анализируется каждый эталон из массива или вектора эталонов 1508 согласно порядку, указанному дугообразными стрелками, такими как дугообразная стрелка 1506. Во внутреннем цикле for 1518 вызывается подпрограмма «compare» для сравнения каждого рассчитанного значения параметра изображения символа с соответствующим предварительно рассчитанным значением параметра эталона, а общий результат сравнения записывается в локальную переменную t. Максимальное значение, накопленное в результате сравнения, хранится в локальной переменной score, а индекс эталона, который наиболее точно соответствует изображению символа, хранится в переменной p 1520. Код символа, связанный с эталоном p, возвращается подпрограммой «process» 1520 в качестве результата.

Наконец, на рисунке 15 показана грубая оценка вычислительной сложности первого варианта осуществления подпрограммы «process» 1522. Количество окон символов для содержащего текст изображения равно N=i×j. В текущем примере N=357. Конечно, количество изображений символов, которое необходимо обработать, зависит от типа документа и количества изображений в нем, а также от языка и других параметров.

Однако обычно количество изображений символов N может изменяться от нескольких десятков до нескольких сотен для каждого изображения документа. Количество эталонов, с которыми сравниваются изображения символов, представлено параметром P. Для многих алфавитных языков, включая большинство европейских языков, количество эталонов может быть относительно небольшим, что соответствует относительно небольшому множеству символов алфавита. Однако для таких языков, как китайский, японский и корейский количество эталонов может изменяться от десятков тысяч до сотен тысяч. Поэтому при обработке таких языков значение параметра P значительно превышает значение параметра N. Количество параметров, используемых для получения отличительных признаков каждого изображения символа и эталона, представлено параметром R. Следовательно, общая вычислительная сложность оценивается как NPR. Коэффициент N берется из внешних вложенных циклов for, показанных на рисунке 14. Коэффициенты PR берутся из вложенных циклов for 1516 и 1518 варианта осуществления подпрограммы «process» 1512, показанной на рисунке 15. Другими словами, подпрограмма «process» вызывается один раз для каждого из N изображений символов, при этом каждый вызов или обращение к подпрограмме «process» приводит к R сравнениям с каждым из P эталонов. При таком способе анализа изначально вычисленное значение параметра считается постоянной величиной. Вариант осуществления алгоритма, приведенного на рисунке 15, можно улучшить различными способами. Например, можно сравнивать только определенное подмножество параметров из общего количества параметров, необходимое для однозначного сопоставления изображения символа с конкретным эталоном. Таким образом, может потребоваться произвести среднее количество сравнений параметров $\frac{\mathbf{R}}{\cdots}$, а не R сравнений. Кроме того, вместо сравнения каждого изображения символа со всеми эталонами можно задать относительно высокий порог значения соответствия, при превышении которого последовательный перебор эталонов будет прекращаться. В этом случае количество эталонов, которые будут сравниваться с каждым изображением символа, будет равно $\frac{\mathbf{P}}{\cdots}$, а не P. Но даже при подобных улучшениях вычислительная сложность будет близка к значению наибольшего из параметров NPR.

На рисунках 16A-B показан второй вариант осуществления подпрограммы «process» (1404 на рисунке 14). Во втором варианте осуществления изображение символа 1602 также используется в качестве входного параметра подпрограммы «process». Однако в данном варианте осуществления алгоритма эталоны группируются в кластеры, такие как кластеры, рассмотренные ранее на примере рисунков 11A-B. Подпрограмма «process» используется для расчета определенного количества значений параметров 1604, достаточного для определения наиболее соответствующего кластера при переборе кластеров эталонов 1606. Таким образом, для выбора наиболее подходящего кластера эталонов сначала используется относительно простая операция сравнения 1608. Затем эталоны 1610 из выбранного кластера эталонов 1611 перебираются с помощью второй довольно простой операции сравнения 1612, для которой используются некоторые дополнительные значения параметров 1614, необходимые для определения наиболее подходящего эталона среди относительно малого числа эталонов 1610, содержащихся в кластере. Псевдокод для второго варианта осуществления подпрограммы «process» представлен на рисунке 16B. В первом вложенном цикле for 1620 выбирается наиболее подходящий или лучший среди имеющихся кластер эталонов, а во втором вложенном

цикле for 1622 определяется наиболее подходящий эталон среди представленных в выбранном кластере. Начальный набор параметров, используемых для определения наилучшего кластера, рассчитывается в цикле for 1624, а дополнительные параметры, необходимые для выбора эталона из числа эталонов выбранного кластера, рассчитываются в цикле for 1626. На рисунке 16B также представлена приблизительная оценка вычислительной сложности второго варианта осуществления подпрограммы «process» 1630. Как указано, оценочная вычислительная сложность для второго варианта осуществления подпрограммы «process» составляет:

$N(CR_1+P'R_2)$,

где количество символов на странице = N;
количество кластеров = C;
количество эталонов/кластер = P';
количество исходных параметров = R;
количество дополнительных параметров = R_2.

Поскольку значение P' по существу намного меньше значения P, а значение C еще меньше, то вычислительная сложность второго варианта осуществления подпрограммы «process» вполне приемлема по сравнению с вычислительной сложностью первого варианта осуществления подпрограммы «process».

Другим подходом к улучшению первого варианта осуществления подпрограммы «process», рассмотренного выше со ссылкой на рисунок 15, является сортировка эталонов в векторе, или массиве, эталонов, чтобы наиболее вероятные эталоны, соответствующие наиболее часто встречающимся символам, рассматривались в самом начале перебора вектора, или массива эталонов. Когда поиск соответствующего эталона прерывается вследствие нахождения эталона, результат сравнения которого превышает некоторое пороговое значение, и когда эталоны отсортированы по частоте употребления, соответствующей вероятности появления символов в обрабатываемом изображении, содержащем текст, вычислительная сложность значительно снижается. Однако частота появления символов в конкретных изображениях, содержащих текст, может сильно варьироваться в зависимости от типа документа или страницы, которые были отсканированы для получения изображения, и неизвестна до обработки системой OCR. Сортировка эталонов, приводящая к значительному снижению вычислительной сложности для одного типа документа, может значительно повысить вычислительную сложность для другого типа документа. Например, общий статистический анализ различных типов текстовых документов на определенном языке, включая романы, рекламные объявления, учебники и другие подобные документы, может позволить получить общий вариант сортировки эталонов по частоте появления символов. Однако в некоторых документах и текстах, относящихся к профильным сферам деятельности, частота появления символов может совершенно отличаться. В этом случае для документов, относящихся к определенным сферам деятельности, наиболее часто встречающиеся символы могут оказаться расположены ближе к концу обрабатываемого вектора, или матрицы, эталонов, отсортированных в соответствии с общей частотой появления символов. Второй вариант осуществления подпрограммы «process», рассмотренный выше со ссылкой на рисунки 16A-B, по существу приводит к значительному уменьшению сложности вычислений и соответствующему увеличению скорости обработки. Как правило, требуется произвести значительно меньше сравнений для нахождения соответствующего эталона для каждого изображения символа. Однако второй вариант осуществления связан с потенциально серьезной проблемой, которая состоит в том, что при неудачном выполнении первого вложенного цикла for, в котором

осуществляется выбор кластера, подпрограмма «process» не сможет найти правильный соответствующий символ. В этом случае правильный соответствующий символ будет находиться в другом кластере, который не будет анализироваться во втором вложенном цикле for. Хотя примеры наборов символов и кластеров, представленные выше, являются относительно простыми, как и параметры, используемые для определения их отличительных особенностей, для таких языков, как китайский и японский, подобная задача является более сложной и подверженной ошибкам из-за несовершенства печати, повреждения документа, а также из-за различных типов ошибок, которые могут возникнуть при сканировании и на начальных стадиях процесса OCR. Таким образом, вероятность неправильного выбора кластера в реальных условиях очень высока.

На рисунке 17 показан третий вариант осуществления подпрограммы «process», рассмотренной в предыдущем подразделе, с использованием тех же иллюстраций и условных обозначений в псевдокоде, которые использовались в предыдущем подразделе. Как показано на рисунке 17, третий вариант осуществления подпрограммы «process» применяет дополнительную структуру данных 1702, обозначаемую как «голоса» («votes»). Структура данных votes содержит целочисленное значение для каждого эталона. При начальной инициализации эта структура содержит нулевые значения для всех эталонов. После этого на первой стадии предварительной обработки, представленной двойным вложенным циклом for 1704 на рисунке 17, для каждого изображения символа в текстовом документе 1300 назначается новый набор кластеров, а эталоны в кластерах упорядочиваются согласно голосам, собранным в структуре данных «votes». Другими словами, эталоны упорядочиваются в заново выделенном наборе, или списке, кластеров, благодаря чему эталоны, с наибольшей вероятностью соответствующие изображению текущего символа, встречаются в начале перебора эталонов. Значения набора сравниваемых параметров, рассчитанные для текущего изображения символа, сравниваются со значениями параметров каждого эталона, при этом голоса отдаются тем эталонам, которые (по результатам сравнения) имеют сходство с изображением символа, превышающее установленный порог. В некоторых вариантах осуществления кластеры также могут быть отсортированы в пределах набора кластеров по накопленному сходству их эталонов с изображением символа.

После стадии предварительной обработки, осуществляемой вложенными циклами for 1704, каждое изображение символа обрабатывается третьим вариантом осуществления подпрограммы «process». Псевдокод для третьего варианта осуществления подпрограммы «process» 1710 представлен на рисунке 17. В данном варианте осуществления подпрограмма «process» принимает в качестве входных параметров изображение символа и набор кластеров, подготовленный для изображения символа на стадии предварительной обработки и хранящийся в массиве NxtLvlClusters, а возвращает указатель на список потенциально соответствующих эталонов. В первом цикле for 1712 рассчитываются значения параметров, которые используются для определения эталонов, соответствующих принятому изображению символа. Во втором внешнем цикле for 1714 рассматриваются все кластеры, пока не заполнится список потенциально соответствующих эталонов. Другими словами, когда находится максимальное количество потенциально соответствующих эталонов, этот внешний цикл for прерывается. Во внутреннем цикле for 1716 для каждого эталона в кластере вызывается функция «similar», осуществляющая определение того, является ли эталон достаточно похожим на изображение символа, чтобы его можно было добавить в список потенциально соответствующих эталонов. Когда список потенциально соответствующих эталонов заполнен, внутренний цикл for также прерывается. На

рисунке 17 представлена оценка вычислительной сложности третьего варианта осуществления подпрограммы «process» 1720. Поскольку как внешний, так и внутренний циклы for 1714 и 1716 прерываются, когда найдено достаточное количество потенциально соответствующих эталонов, а векторы, или списки, эталонов в каждом кластере отсортированы по частоте появления в обрабатываемом документе, то в третьем варианте осуществления подпрограммы «process» требуется выполнить лишь относительно небольшое количество сравнений по сравнению со вторым вариантом осуществления подпрограммы, что и показано дробью $\frac{1}{3}$ 1722. Существует, конечно, штраф за начальную предварительную обработку, представленный коэффициентом «e» 1744. Однако, как указывалось выше, количество обрабатываемых изображений символов N по существу значительно меньше значения параметров P или P' для таких языков, как китайский, японский и корейский, и, следовательно, третий вариант осуществления процедуры «process» обеспечивает значительное снижение вычислительной сложности по сравнению как с первым так и со вторым вариантами осуществления, рассмотренными выше. Что более важно, в третьем варианте осуществления подпрограммы «process» гарантируется просмотр всех кластеров, пока не будет обнаружено некоторое максимальное количество потенциально соответствующих символов. Когда порог сходства для кластеров имеет относительно низкое значение, а порог сходства для эталонов имеет относительно высокое значение, существует очень большая вероятность, что список потенциально соответствующих символов, возвращаемый подпрограммой «process», будет включать в себя именно тот символ, который наиболее точно соответствует входному изображению символа.

Рассмотренная выше информация, включая третий вариант осуществления, представленный на рисунке 17, создает основу для описания определенного аспекта этого обобщенного третьего варианта, к которому относится настоящее изобретение. Следует четко понимать, что описанные выше варианты представляют собой обобщенные варианты осуществления и что конкретный вариант осуществления системы OCR может применять любой из большого количества возможных альтернативных вариантов.

Настоящее изобретение относится к описанию логики управления и структур данных в системе OCR, которые можно использовать как для кластеризации эталонов, так и на стадии предварительной обработки, рассмотренной выше, в ходе которой графемы в эталонах могут быть отсортированы по частоте появления в отсканированном изображении, содержащем текст, или в наборе отсканированных изображений. Эти логика управления и структуры данных применяются в системе OCR для реализации стадий предварительной обработки/кластеризации, в ходе которых фиксированный набор параметров связывается с каждым кластером и применяется при сравнении изображения символа с эталонами, содержащимися в кластере. Кластеры можно использовать в различных локальных операциях или на разных этапах сложной задачи оптического распознавания символов, при этом конкретные параметры (а также количество этих параметров), используемые для сравнения изображения символа с эталонами, содержащимися в кластере, могут отличаться в различных локальных операциях и на разных этапах, а также часто могут быть различными в разных кластерах. На рисунке 18 показаны структуры данных, обеспечивающие кластеризацию и предварительную обработку в одном варианте осуществления системы OCR, включающей в себя третий вариант осуществления подпрограммы «process», описанный выше. Первой структурой данных является массив, или вектор, 1802, обозначенный

как «votes» («голоса»). В описанном варианте осуществления массив «votes» содержит один элемент для каждой графемы языка. Массив «votes» индексируется целыми значениями кодов графем. Другими словами, каждой графеме присваивается уникальный целочисленный идентификатор или код графемы, который используется в качестве индекса массива «votes». Как показано на рисунке 18, массив «votes» может содержать n элементов, где n представляет собой количество графем в языке, а коды графем монотонно возрастают от 0 до n. Конечно, структура данных «votes» может быть альтернативно реализована в виде разреженного массива, когда коды графем возрастают немонотонно, в виде списка, или с использованием других типов структур данных.

На рисунке 18 показана вторая структура данных 1804, которая представляется собой массив экземпляров класса «parameter» («параметр»). Как и в случае со структурой данных «votes», массив «parameters» может быть альтернативно реализован с использованием разных структур данных, включая списки, разреженные массивы и другие структуры данных. В текущем рассматриваемом варианте осуществления массив «parameters» включает в себя p записей или элементов, которые проиндексированы с помощью монотонно увеличивающихся чисел 0, 1, 2, …, p. Каждый экземпляр класса «parameter» представляет один из различных параметров, используемых для определения отличительных признаков изображений символов и эталонов, как описывалось выше.

На рисунке 18 также показана структура данных кластера 1806, которая представляет собой кластер или набор эталонов. Структура данных кластера включает в себя массив «clusterParameters» 1808, в котором содержатся параметры, применяемые для определения отличительных признаков эталонов в кластере в определенный момент времени, а также для определения отличительных признаков изображений символов для их сравнения с эталонами, содержащимися в кластере. Каждый элемент в массиве «clusterParameters» содержит индекс для массива «parameters» 1804. Используя индексы в массиве «parameters» 1804, можно легко изменить или переконфигурировать конкретные параметры, а также количество параметров, используемых для сравнения, вследствие чего конфигурация кластера может быть эффективно изменена для различных локальных операций или этапов. Структура данных кластера также включает в себя целочисленный параметр пит 1810, который указывает количество индексов параметров, содержащихся в массиве «clusterParameters». Структура данных кластера дополнительно содержит параметр «cutoff» («отсечка») 1812, имеющий формат с плавающей запятой (или формат двойной точности) и содержащий пороговое значение для оценки эталонов, находящихся в кластере, на предмет их соответствия изображению символа. Наконец, структура данных кластера 1806 содержит ряд структур данных эталона 1814-1822. Структуры данных эталона рассматриваются ниже.

На рисунках 19A-З показана предварительная обработка изображения символа с использованием структур данных, рассмотренных выше со ссылкой на рисунок 18. На рисунке 19A показана структура данных «votes» 1802, рассмотренная выше со ссылкой на рисунок 18, а также структура данных одного эталона 1902, выбранная из эталонов, содержащихся в структуре данных кластера 1806, также рассмотренной выше со ссылкой на рисунок 18. Каждая структура данных эталона содержит номер эталона 1904 и набор значений параметров 1905, рассчитанных для эталона с помощью параметров, ссылки на которые получены из индексов, содержащихся в массиве «clusterParameters» 1808, который находится в структуре данных кластера 1806. Как отмечалось выше, важно помнить, что изображения символов масштабируются, поворачиваются и преобразуются для создания нормированных изображений, чтобы облегчить процедуру сравнения

изображений символов и эталонов на основе параметров. Структура данных эталонов дополнительно содержит целочисленный параметр 1906, указывающий количество индексов в структуре данных эталона, а также значения самих индексов 1908. Эти индексы связаны с различными возможными весами, которые могут быть рассчитаны при сравнении изображения символа с эталоном. В одном варианте осуществления в структуре данных эталона может быть столько индексов, сколько имеется возможных рассчитанных весов, при этом каждый индекс содержит целочисленный индекс и рассчитанный вес, связанный с этим индексом. Возможны и другие варианты осуществления. Вес рассчитывается, когда рассчитываются параметры для изображения символа и значения параметров изображения символа сравниваются с эталоном, представленным структурой данных эталона. Чем больше значение веса, тем меньше изображение символа соответствует эталону. Этот вес применяется для выбора из указателей соответствующего индекса, который используется для выбора количества графем, соответствующих эталону, за которые необходимо проголосовать на стадии предварительной обработки. Каждая структура данных эталона включает в себя целочисленное значение, указывающее количество графем, соответствующих эталону 1910, а также коды всех графем набора, соответствующих эталону 1912. Во многих вариантах осуществления эти коды графем сортируются по сходству или близости к эталону в порядке убывания сходства.

На рисунках 19Б-З показана предварительная обработка изображения одного символа, выбранного из отсканированного изображения, содержащего текст. В примере, изображенном на рисунках 19Б-З, изображение символа 1914 представляет собой иероглиф одного из азиатских языков. На рисунке 19Б также показаны массив «parameters» 1804, рассмотренный выше со ссылкой на рисунок 18, и небольшой фрагмент структуры данных кластера 1806, включающей в себя массив «clusterParameters» 1808 и целочисленный параметр пит 1810.

Как показано на рисунке 19В, для всех параметров пит, индексы которых включены в массив «clusterParameters» 1808, индекс параметра 1916 извлекается из массива «clusterParameters» 1808 и используется для доступа к экземпляру класса «parameter» 1918 в массиве «parameters» 1804. Для формирования значения параметра 1920, которое затем хранится в локальной переменной 1922, вызывается метод «parameterize» экземпляра класса «parameter» 1918. На рисунке 19В показан расчет значения первого параметра для изображения символа. На рисунке 19Г показан расчет значения второго параметра для изображения символа. Когда все экземпляры пит класса «parameter» задействованы для формирования значений параметров пит для изображения символа, формируется список, или массив, значений параметров изображения символа 1924, как показано на рисунке 19Д.

Затем, как показано на рисунке 19Е, из предварительно рассчитанного значения параметра для эталона, представленного структурой данных эталона, вычитается соответствующий параметр для изображения символа с целью получения ряда рассчитанных значений, по одному для каждого параметра. Например, как показано на рисунке 19Е, из первого значения параметра 1926, хранящегося в структуре данных эталона 1902, вычитается первое значение параметра 1922, рассчитанное для изображения символа, с целью получения промежуточного значения 1928. Аналогичным образом из остальных предварительно определенных значений параметров для эталона 1930-1933 вычитаются остальные значения параметров для изображения символа 1934-1938 для получения дополнительных промежуточных расчетных значений 1940-1944. Абсолютные величины этих промежуточных значений 1928 и 1940-1944 суммируются

1946 для получения веса 1948, который численно представляет собой основанное на параметрах сравнение изображения символа и эталона, представленного структурой данных эталона 1902. И в этом случае, чем больше значение рассчитанного веса, тем меньше изображение символа похоже на эталон, поскольку вес представляет собой накопленное различие между значениями параметров для изображения символа и эталона.

Как показано на рисунке 19Ж, когда рассчитанный вес 1948 превышает вес отсечки «cutoff» 1812 для кластера, предварительная обработка изображения символа в отношении рассматриваемого эталона, представленного структурой данных эталона 1902, прекращается значением 1950. В противном случае, на этапе предварительной обработки изображения символа происходит голосование за одну или более графем, соответствующих эталону, представленному структурой данных эталона 1952.

На рисунке 193 показан случай, когда рассчитанный вес, представляющий собой сравнение изображения символа с эталоном, представленным структурой данных эталона, меньше или равен весу отсечки для данного кластера. В этом случае для выбора индекса 1954 из набора индексов 1908 используется рассчитанный вес 1948. Как описывалось выше, каждый из индексов 1908 может содержать индекс и связанный с ним вес, что позволяет выбрать один конкретный индекс 1954 согласно рассчитанному весу 1948 из индекса, который является индексом извлекаемых кодов графем. Этот извлеченный индекс указывает 1956 на конкретный код графемы 1958 в наборе кодов графем 1912, хранящемся в структуре данных эталона для представления тех графем, которые соответствуют эталону. Затем для всех кодов графем, начиная с первого кода графемы 1960 и до кода графемы 1958, на который указывает извлеченный индекс 1956, увеличивается соответствующий элемент из структуры данных «votes» 1802, как показано стрелками (такими как стрелка 1962), исходящими из элементов, содержащих коды графем между элементами 1960 и 1958 включительно.

Таким образом, если рассчитанный вес, используемый для сравнения изображения символа с эталоном, меньше веса отсечки, значит изображение символа достаточно похоже на эталон, чтобы по меньшей мере за некоторые из графем, соответствующих эталону, был отдан голос на стадии предварительной обработки. Графемы, достаточно похожие на изображение символа, выбираются на основе рассчитанного веса с использованием индекса, выбранного из индексов 1908 в соответствии с рассчитанным весом. Затем элементы структуры данных «votes», соответствующие этим графемам, увеличиваются для отражения количества голосов, отданных этим графемам во время предварительной обработки изображения символа.

Далее приводится псевдокод, напоминающий код на языке C++, для иллюстрации предварительной обработки изображения символа с учетом эталонов в кластере, как показано на рисунках 19А-З. Используется относительно простой псевдокод, похожий на язык C++, включающий массивы фиксированного размера и простые структуры управления. Конечно, в реальных системах OCR могут использоваться более эффективные, но и более сложные варианты осуществления, включающие итераторы, структуры данных, реализующие ассоциативную память, а также другие подобные структуры данных и связанные с ними способы.

Сначала определяется ряд структур данных, и объявляются классы:

```
1   int votes[NUM_GRAPHEMES];

2   class parameter
3   {
4           virtual double parameterize (symbolImage* s);
5   };

6   parameter Parameters [NUM_PARAMETERS];

7   class pattern
8   {
9           private :
10                  int patternNo;
11                  double parameters[MAX_PARAMETERS];
12                  int numIndices;
13                  int indices[MAX_INDICES];

14                  int numGraphemes;
15                  int graphemes[MAX_GRAPHEMES];
16          public :
17                  double getParameter (int i);
18                  int getIndex (double w);
19                  int getGrapheme (int i);
20                  pattern ();
21  };

22  class cluster
23  {
24          private :
25                  int num;
26                  int clusterParameters[MAX_CLUSTER_PARAMETERS];
27                  double cutoff;
28                  int numPatterns;
29                  pattern*patterns;
30          public :
31                  double getCutoff ( );
32                  int getNum ( );
33                  int getParameter (i);
34                  pattern* getPattern (i);
35                  int getNumPatterns ( );
36                  cluster ();
37  };
```

В строке 1 осуществляется объявление структуры данных голосов «votes» 1802. Частичное объявление класса «parameter» представлено в строках 2-5. В данном примере единственно важным аспектом класса «parameter» является то, что базовый класс включает в себя виртуальный метод «parameterize», который в качестве входных данных принимает изображение символа, а возвращает значение параметра с плавающей запятой. Конечно, в некоторых случаях конкретный параметр может иметь только целые значения, а не значения с плавающей запятой. Структура данных «parameters» 1804 объявляется в строке 6. Частичное объявление класса «pattern» происходит в строках 7-21. Класс «pattern» включает в себя следующие закрытые поля: «patternNo» (1904 на рисунке 19A), объявленный в строке 10, массив значений параметра «parameters»

(1906 на рисунке 19A), объявленный в строке 11, количество индексов «numIndices» (1906 на рисунке 19A), объявленное в строке 12, набор индексов (1908 на рисунке 19A) элементов множества «numIndices», объявленный в строке 13, целочисленное значение «numGraphemes» (1910 на рисунке 19A), а также набор кодов графем «graphemes» (1912 на рисунке 19A) элементов множества «numGraphemes». Класс «pattern» включает в себя метод «getParameter», объявленный в строке 17, который возвращает значение параметра из набора значений «parameters», метод «getIndex», объявленный в строке 18, который возвращает индекс, соответствующий рассчитанному весу, а также метод «getGrapheme», объявленный в строке 19, который возвращает код графемы из набора кодов «graphemes», объявленного в строке 15. Наконец, в строках 22-37 объявляется класс «cluster», который представляет структуру данных кластера 1806 на рисунке 18. Класс «cluster» включает в себя следующие закрытые поля: пит (1810 на рисунке 18), объявленный в строке 25, «clusterParameters» (1808 на рисунке 18), объявленный в строке 26, «cutoff» (1812 на рисунке 18), объявленный в строке 27, целочисленное значение «numPatterns», указывающее на количество эталонов в кластере, объявленное в строке 28, а также указатель на эталоны в кластере «patterns», объявленный в строке 29. Класс «cluster» включает в себя методы «getCutoff» и «getNum», используемые для получения веса отсечки и количества эталонов, объявленные в строках 31 и 32, метод «getParameter», используемый для получения индексов параметров из массива «clusterParameters», объявленный в строке 32, метод «getPattern», возвращающий определенный эталон, хранящийся в кластере, объявленный в строке 33, и метод «getNumPatterns», объявленный в строке 34, который возвращает количество эталонов, хранящееся в структуре данных кластера.

На примере следующего псевдокода подпрограммы «vote» показана реализация способа предварительной обработки для одного изображения символа и одного кластера:

```
36  void vote (symbolImage* s, cluster* c)
37  {
38      double params[MAX_PARAMETERS];
39      int i, j, k, l;
40      double weight, t;
41      pattern* p;
42
43      for (i = 0; i < c → getNum( ); i++)
44          params[i] = Parameters[c → getParameter (i)].parameterize(s);
45      for (j = 0; j < c → getNumPattern( ); j++)
46      {
47          p = c → getPattern(i);
48          weight = 0;
49          for (i = 0; i < c → getNum( ); c++)
50          {
51              t = p → getParameter (i) – params [i];
52              weight + = (t < 0) ? – t : t;
53          }
54          if (weight > c → getCutoff( )) continue;
55          k = p → getIndex(weight);
56          for (l = 0; l < k; l++)
57              votes[p → getGrapheme(l)]++;
58      }
59  }
```

Подпрограмма «vote» получает в качестве аргументов указатель на изображение

символа и указатель на кластер. Локальные переменные включают в себя массив «params», объявленный в строке 38, в котором хранятся рассчитанные значения параметров для изображения символа, итерационные целочисленные значения i, j, k и l, объявленные в строке 39, переменные с плавающей запятой «weight» и «t», используемые для хранения рассчитанного веса, полученного в результате сравнения входного изображения символа и эталона в кластере, а также указатель p, объявленный в строке 41 и указывающий на эталон во входном кластере. В цикле for, в строках 43-44, значения всех параметров, используемых кластером, рассчитываются для входного изображения символа и записываются в массив «params» (1924 на рисунке 19Д). Затем во внешнем цикле for, содержащем вложенные циклы for, представленные в строках 45-58, рассматривается каждый параметр входного кластера. В строке 46 получается указатель на рассматриваемый эталон путем вызова метода кластера «getPattern». В строке 48 локальной переменной «weight» присваивается значение 0. Затем в цикле for, в строках 49-53, рассчитывается вес, который представляет собой сравнение входного изображения символа с эталоном, как описывалось выше со ссылкой на рисунок 19F. Когда вес превышает вес отсечки для кластера, как определено в строке 54, текущая итерация, выполняемая во внешнем цикле for, в строках 44-58, прерывается, поскольку входное изображение символа недостаточно похоже на рассматриваемый эталон, чтобы можно было за него проголосовать. В противном случае локальная переменная k принимает значение последнего индекса кода графемы для голосования в строке 55. Затем в цикле for, в строках 56-57, выполняется голосование за все графемы до графемы с индексом k.

Существует множество различных альтернативных подходов к осуществлению стадии предварительной обработки и формированию описанных выше структур данных. Например, вместо использования веса отсечки для всего кластера можно использовать вес отсечки для конкретных эталонов, при этом вес отсечки может включаться в структуру данных эталона. В другом примере индексы, хранящиеся в эталоне, могут представлять собой экземпляры классов, которые содержат списки кодов графем, а не индексы, указывающие на упорядоченный список кодов графем, как это реализуется в последнем варианте осуществления. Также возможны и многие другие альтернативные варианты осуществления. Например, подпрограмма «vote» может принимать в качестве второго аргумента указатель на массив «params», а в цикле for, в строках 43-44, могут вычисляться значения параметров только в том случае, если они еще не были рассчитаны при обработке изображения символа для других кластеров. В других вариантах осуществления можно применять разные типы расчета веса и сравнения изображения символа с эталоном. В некоторых случаях большее значение веса может указывать на большее сходство изображения символа с эталоном в отличие от приведенного выше примера, когда увеличение значения веса означало уменьшение сходства изображения символа с эталоном. В ряде систем OCR с графемами могут связываться вещественные коэффициенты, позволяющие при голосовании использовать дробные значения и значения больше 1. В некоторых системах OCR графемы, эталоны и/или кластеры можно отсортировать на основании голосов, накопленных в течение предварительной обработки, для более эффективного последующего распознавания символов. В определенных вариантах осуществления структура данных кластера может включать в себя только количество структур данных эталона или ссылок на структуры данных эталона, при этом вес отсечки и эталоны, связанные с кластером, указываются в алгоритме управления, а не хранятся в структуре данных кластера.

Мультикластерная обработка документа

В предыдущем разделе на рисунке 18 и в комментариях к рисунку 18 были представлены структуры данных кластера, а также структуры данных эталона, описанные со ссылкой на рисунок 19А. Эти и другие структуры данных, описанные со ссылкой на рисунки 18 и 19А-Н, облегчают оптическое распознавание символов (OCR) и обработку изображений символов с применением OCR в документах, а также в других информационных источниках, содержащих изображения символов. В настоящем подразделе рассматривается способ обработки документа, в котором используется множество структур данных кластеров для обработки документа с применением OCR. Сначала на рисунках 20А-Нс помощью тех же иллюстраций, что были ранее использованы на рисунках 18 и 19А-З, показана мультикластерная обработка документа, содержащего изображения символов, с применением OCR. На дополнительных рисунках показаны расчет количественная оценка эталонов и сортировка эталонов внутри кластеров. И, наконец, на блок-схемах показан один из вариантов осуществления мультикластерной обработки документа с применением OCR.

На рисунке 20А показаны данные и структуры данных, использованные в примере мультикластерной обработки документа с применением OCR. Иллюстрации, использованные на рисунке 20А, также используются на последующих рисунках 20Б-П. На рисунке 20А снова показаны многие структуры данных с рисунков 18 и 19А-G. К ним относятся массив параметров 2002 (показан как массив 1804 на рисунке 18), массив голосов 2004 (показан как массив голосов 1802 на рисунке 18), три структуры данных кластера 2006-2008, каждая из которых идентична структуре данных кластера 1806 на рисунке 18, и массив рассчитанных значений параметров 2010, аналогичный массиву рассчитанных значений параметров 1924 на рисунке 19Е. Обратите внимание, что структурам данных при инициализации присваиваются соответствующие значения, в то время как массиву голосов присваиваются все нулевые значения. Каждая структура данных кластера, такая как структура данных кластера 2006, включает в себя массив параметров 2012, похожий на массив clusterPammeters 1808, показанный на рисунке 18, величину пит 2014 и величину cutoff'2016, аналогичные соответственно величинам пит и cutoff 2016 и 1812, показанным в кластере 1806 на рисунке 18, а также множество структур данных эталона, таких как структура данных эталона 2018, идентичных структуре данных эталона 1902 на рисунке 19А. Кроме того, на рисунке 20А показана структура данных 2020, которая представляет собой отсканированное изображение страницы документа. Эта структура данных является двумерным массивом, каждая ячейка которого, как, например, ячейка 2022, соответствует изображению символа. Также на рисунке 20А показана переменная 2024, содержащая обрабатываемое в настоящий момент изображение символа.

Структуры данных, показанные на рисунке 20А, могут быть реализованы различными способами с помощью различных языков программирования и технологий хранения данных. Структуры данных могут включать в себя дополнительные данные и подструктуры данных. В одном из вариантов осуществления, например, каждая структура данных эталона в каждом кластере представляет собой ссылки из отсортированного массива ссылок структуры данных кластера. В других вариантах осуществления каждая структура данных эталона в каждом кластере связывается с числовой последовательностью, что позволяет проходить по структурам данных эталона в определенном порядке. В некоторых осуществлениях структура данных кластера может включать в себя структуры данных эталона, в то время как в других осуществлениях структура данных кластера может ссылаться на структуры данных эталона. В большинстве осуществлений структуры данных могут быть динамически

расширены или сжаты, чтобы соответствовать изменениям способов OCR, в которых они используются. Таким образом, несмотря на то что для описания структуры данных голосов применяется термин «массив», данная структура может быть реализована с использованием структур данных, отличных от простых массивов, но позволяющих при этом индексировать элементы как в массивах.

Текстовая структура данных 2020 представляет собой страницу документа, которую необходимо обработать способом мультикластерной обработки документа с применением OCR для преобразования исходного отсканированного документа, содержащего изображения символов, в эквивалентный электронный документ, содержащий коды символов. Термины «документ», «страница» и «изображение символа» могут иметь различные значения в зависимости от контекста. В данном примере документ состоит из нескольких страниц, и каждая страница включает в себя множество изображений символов. Тот же самый или схожий способ мультикластерной обработки документов с применением OCR может использоваться для множества различных типов документов вне зависимости от того, содержат ли они страницы с одним или более изображениями символов или нет.

На первой стадии, показанной на рисунке 20Б, переменная текущего изображения символа 2024 считывает в себя или ссылается на первое изображение символа 2026, как это показано фигурной стрелкой 2027. Затем, как показано на рисунке 20С, каждая вычисляющая параметры функция или подпрограмма из массива параметров 2002 применяется к текущему изображению символа из переменной 2024 для формирования соответствующих значений параметров, которые затем записываются в массив рассчитанных значений параметров 2010, как показывают стрелки 2028-2031 и точки 2032-2033. Таким образом, в одном из осуществлений массив рассчитанных значений параметров 2010 включает в себя числовые значения параметров, соответствующие каждому из параметров массива 2002, представленных функциями или ссылками на них, и рассчитанные для текущего изображения символа. Вычисление значений параметров ранее описывалось со ссылкой на рисунки 19В-Г.

Затем, как показано на рисунке 20Г, в первой структуре данных кластера 2006 выбирается первая структура данных эталона 2034; значения параметров, связанных с первой структурой данных эталона, вместе с соответствующими значениями параметров из массива рассчитанных значений 2010 используются для расчета веса W 2035, как было описано выше со ссылкой на рисунок 19Е. Обратите внимание, что массив параметров структуры данных кластера (2012 на рисунке 20А) используется для индексации массива рассчитанных значений параметров. Как было описано выше со ссылкой на рисунок 19К, затем вычисленный вес сравнивается с весом отсечки (2016 на рисунке 20А) 2036. Это позволит определить, может ли структура данных эталона 2034 из первой структуры данных кластера 2006 отдать голос за графемы, как было описано выше со ссылкой на рисунок 193. В настоящем примере, как показано на рисунке 20Д, рассчитанный вес 2035 ниже веса отсечки, в результате чего происходит накопление голосов, формируемых первой структурой данных эталона 2034 из первой структуры данных кластера 2006, в массиве голосов 2024. Как было описано выше со ссылкой на рисунок 19Н, рассчитанный вес 2035 используется для выбора индекса из набора индексов 2038 в первой структуре данных эталона 2034, а содержимое выбранного элемента является указателем 2039 на некоторый код графемы из первой структуры данных эталона 2040. Голоса формируются для графем, соответствующих кодам графем из сегмента кодов графем первой структуры данных эталона, начиная с первого кода и заканчивая кодом графемы, на которую указывает индекс, выбранный

из сегмента индексов структуры данных эталона. На рисунке 20Д голоса, создаваемые первой структурой данных эталона 2034 из первой структуры данных кластера 2006, показаны фигурными стрелками 2042-2046. Пустые значения в массиве голосов (2004 на рисунке 20А) представляют собой нулевые значения (0). Начальное голосование для первой структуры данных эталона 2034 из первой структуры данных кластера 2006 увеличивает значения накопленных голосов 2047-2051 с 0 до 1 для тех графем в массиве голосов, для которых выбраны соответствующие коды из первой структуры данных эталона. В других вариантах осуществления при голосовании к значениям элементов массива голосов могут прибавляться числа, отличные от 1.

Далее, как показано на рисунке 20Е, выбирается вторая структура данных эталона 2052 из первой структуры данных кластера 2006, и процесс, описанный со ссылкой на рисунок 20Г, повторяется снова. Однако в настоящем примере вес, рассчитанный для второй структуры данных эталона, выше веса отсечки для первого кластера, так что голосование не производится. Как показано на рисунке 20К, следующей для обработки выбирается третья структура данных эталона 2053 из первой структуры данных кластера 2006, и стадии, показанные на рисунках 20Г-Д, повторяются. В данном примере, как показано на рисунке 2К, вес, рассчитанный для структуры данных эталона, меньше либо равен весу отсечки. Это значит, что третья структура данных эталона 2053 формирует голоса, которые накапливаются в массиве голосов 2004, что и показано фигурными стрелками 2054-2057. Обратите внимание, на рисунке 20Д показано, что третья структура данных эталона отдает голос 2055 за графему 2048, за которую уже ранее отдавала голос первая структура данных эталона 2034. Таким образом, общее число накопленных голосов для данной графемы теперь равно 2.

Как показано на рисунке 203, тот же самый процесс повторяется для каждой последующей структуры данных эталона из первой структуры данных кластера 2006 для всех голосов, формируемых оставшимися структурами данных эталонов и накапливаемыми в массиве голосов 2004, на что указывает волнистая стрелка 2058. Затем, как показано на рисунке 201, выбирается первая структура данных эталона 2059 из второй структуры данных кластера 2007, и стадии, показанные на рисунках 20Г-Д, выполняются для первой структуры данных эталона 2059 из второй структуры данных кластера 2007, при этом структура данных эталона формирует голоса, которые в данном примере показаны фигурными стрелками 2060-2063. Необходимо отметить, что параметры, на которые ссылается массив параметров в каждом кластере, как, например, массив параметров 2012 в кластере 2006, могут отличаться от параметров, на которые ссылаются другие кластеры. Другими словами, каждая структура данных кластера включает в себя массив параметров, который может ссылаться на уникальный набор параметров, связанных с данным кластером. Между параметрами, на которые ссылаются два разных кластера, может не быть никаких совпадений или же могут быть частичные или существенные совпадения. Каждый кластер представляет собой специализированный механизм обнаружения семейств или наборов похожих символов. Поскольку различные семейства и наборы символов наилучшим образом обнаруживаются с помощью соответствующих наборов параметров, каждый кластер включает в себя массив параметров, который ссылается на конкретные параметры из глобального массива параметров (2002 на рисунке 20А) и используется кластером для обнаружения тех символов из семейства или множества символов, для обнаружения которых этот кластер предназначается. Кроме того, каждый кластер может использовать разное число структур данных эталонов, а значит, как и в случае с параметрами, на которые ссылаются кластеры, между структурами данных параметров, на которые

ссылаются два разных кластера, может не быть никаких совпадений или же могут быть частичные или существенные совпадения.

Как показано волнистой стрелкой 2065 на рисунке 20Ж, обработка продолжается для каждой последующей структуры данных эталона из второй структуры данных кластера 2007, а также всех остальных структур данных кластера, включая последнюю структуру данных кластера 2008. Все голоса, формируемые структурами данных эталона, накапливаются в массиве голосов 2004. Так выглядит обработка первого изображения символа, выбранного из текстовой страницы 2020.

Далее, как показано на рисунке 20Л, голоса, накопленные в массиве голосов для первого изображения символа, выбранного из текстовой страницы, используются для подготовки отсортированного списка кодов графем, которые чаще всего соответствовали изображению символа в процессе обработки, описанной выше со ссылкой на рисунки 20Б-И, согласно накопленным голосам для кодов графем в массиве голосов. На рисунке 20Л массив голосов 2004 показан в верхней части рисунка. Каждая ячейка в массиве голосов содержит число голосов; индексами ячеек являются коды графем. Голоса и индексы кодов графем затем сортируются в порядке убывания чисел голосов, формируя, таким образом, отсортированный массив 2067, в котором каждая ячейка содержит код графемы, а индексы слева направо монотонно возрастают, упорядочивая коды графем по количеству голосов, которые они получили в процессе обработки, описанной на рисунках 20Б-И. Например, наибольшее число голосов, равное 16, получил код графемы «9» 2066, а значит, код графемы «9» будет стоять на первой позиции 2068 в отсортированном массиве кодов графем 2067. Затем отсортированный массив 2067 урезается, формируя усеченный отсортированный массив кодов графем 2069. Усеченный отсортированный массив кодов графем включает в себя отсортированный список кодов графем, которые получили голоса в процессе обработки, описанной выше со ссылкой на рисунки 20Б-И. В процессе, описанном на рисунках 20Б-И, голоса получили только 14 кодов графем, а значит, усеченный отсортированный массив кодов графем 2069 содержит только 14 элементов. Это первые 14 элементов в отсортированном массиве кодов графем 2067. Остальные элементы отсортированного массива кодов графем 2067, следующие за четырнадцатым элементом с индексом 13, содержат коды графем, для которых голоса не были получены. Далее, как показывает фигурная стрелка 2070, усеченный массив кодов графем включается в первый элемент 2071 таблицы обработанных изображений символа 2072. Каждый элемент таблицы обработанных изображений символа включает в себя поле (представлено как первый столбец 2073), указывающее на число или порядок символа внутри текстовой структуры данных (2020 на рисунке 20А); поле с количеством кодов графем, получивших голоса в процессе обработки символа (второй столбец 2074 в таблице обработанных изображений символа), и отсортированный усеченный массив кодов графем (третий столбец 2075 таблицы обработанных изображений символа 2072).

В некоторых осуществлениях отсортированный усеченный массив кодов графем немедленно используется в дополнительном алгоритме обнаружения символов, который формирует соответствующий символ для изображения символа. Этот обнаруженный символ затем может быть немедленно помещен в обработанную страницу, соответствующую странице, содержащей изображения символов, которые были обработаны вышеописанным способом со ссылкой на рисунки 20А-И. Однако в настоящем осуществлении отсортированные усеченные массивы кодов графем для изображений символов накапливаются в таблице обрабатываемых изображений символа для каждой страницы документа. Отсортированные усеченные массивы кодов графем

затем используются на втором этапе для преобразования изображений символов со страницы документа в символы, помещаемые затем на обработанную страницу документа. В любом случае, отсортированный усеченный массив кодов графем представляет собой результат начальной мультикластерной обработки, которая определяет набор кодов графем, наиболее вероятно связанных с изображением символа внутри изображения страницы документа. В настоящем осуществлении все коды графем, заработавших голоса, включены в отсортированные усеченные массивы кодов графем. В других осуществлениях в отсортированные усеченные массивы кодов графем включаются только те коды графем, число голосов которых превышает заданный порог.

Как только завершается обработка первого изображения символа, извлеченного из страницы документа, и в таблице обработанных изображений символов создается соответствующая запись, второе изображение символа 2078 выбирается из текстовой структуры данных (2020 на рисунок 20А) и помещается в переменную 2024. Затем рассчитываются значения параметров для следующего символа и сохраняются в массиве рассчитанных значений параметров 2010, как показано на рисунке 20М и описано выше со ссылкой на рисунок 20 В. Затем, как показано на рисунок 20Н, обрабатывается второе изображение символа с использованием способа, описанного выше со ссылкой на рисунки 20 В-И. Таким образом, формируется новый набор накопленных голосов в массиве голосов 2004 для второго изображения символа. Как показано на рисунке 20О, накопленные голоса для второго изображения символа 2004 сортируются по числу голосов, формируя отсортированный массив кодов графем 2080, как это было описано выше со ссылкой на рисунок 20Л. Отсортированный массив кодов графем 2080 урезается, формируя усеченный отсортированный массив 2081, который, как показывает фигурная стрелка 2082, включается во второй элемент 2083 таблицы обработанных изображений символа 2072, представляющий второе изображение символа 2078.

Как показано на рисунок 20П, каждое изображение символа в текстовом документе 2020 последовательно обрабатывается набором стадий, описанных выше со ссылкой на рисунки 20Б-Ж. При этом для каждого изображения символа в массиве голосов 2004 накапливаются голоса, как это показано стрелками 2084-2093. Накопленные голоса для каждого изображения символа затем формируют отсортированные усеченные массивы кодов графем для каждого изображения символа, после чего эти массивы добавляются в состав соответствующих элементов таблицы обработанных изображений символа 2072. В результате обработки изображений символа текстового документа, использующей кластеры, формируется таблица обработанных изображений символа 2072, в которой имеется запись для каждого изображения символа текстового документа. Каждый элемент таблицы обработанных изображений символа представляет собой начальный набор графем, потенциально соответствующих изображению символа. Набор потенциально соответствующих графем помещается в отсортированный усеченный массив кодов графем, отсортированный в порядке убывания накопленных голосов таким образом, что коды графем, набравших наибольшее количество голосов, располагаются первыми в отсортированном усеченном массиве кодов графем. Набор потенциально соответствующих кодов графем, представленный в виде отсортированного усеченного массива кодов графем, может быть в дальнейшем использован в дополнительном алгоритме распознавания символов для определения наилучших кодов символов для того изображения символа, для которого был сформирован отсортированный усеченный массив кодов графем в процессе обработки, описанной выше со ссылкой на рисунки 20Б-Ж.

На рисунках 21А-Г с помощью блок-схем показан один из вариантов осуществления способа мультикластерной обработки документа с применением OCR. На рисунке 21А показан общий алгоритм способа мультикластерной обработки документа с применением OCR. На стадии 2102 происходит получение документа и инициализация структуры данных, соответствующей обработанному документу (PD), которая будет хранить коды символов, сформированных в процессе обработки документа, для соответствующих изображений символов из полученного документа. Далее, на стадии 2104 инициализируются структуры данных, описанные выше со ссылкой на рисунки 18-20А, для подготовки к обработке документа. Затем в цикле for на стадиях 2106-2112 происходит обработка каждой страницы документа для замены в обработанном документе PD изображений символов из полученного документа на коды символов или другие вычисленные представления символов. На первой стадии внешнего цикла for 2106-2112 таблица обработанных изображений символа, описанная выше со ссылкой на рисунки 20Л и 20О (2072 на рисунках 20Л и 20О), очищается и инициализируется повторно. Затем во внутреннем цикле for на стадиях 2108-2111 каждый символ текущей страницы обрабатывается путем вызова подпрограммы «process symbol image» 2109. По завершении вложенных циклов for на стадиях 2106-2112 память, занятая структурами данных, использованными при обработке полученного документа с применением OCR, освобождается на стадии 2114, и возвращается обработанный документ PD.

На рисунке 21Б показана подпрограмма ((initialize data structures)), вызываемая на стадии 2104 рисунка 21А. На стадии 2116 происходит выделение памяти для массива параметров (2002 на рисунке 20А) и его инициализация. На стадии 2118 происходит выделение памяти для массива голосов (2004 на рисунке 20А). На стадии 2120 происходит выделение памяти и инициализация структур данных кластеров (2006-2008 на рисунке 20А) и структур данных, которые содержат в себе или на которые ссылаются структуры данных кластеров, включая массивы локальных параметров (2012 на рисунке 20А) и структуры данных эталонов (2018 на рисунке 20А). Как говорилось выше, каждая структура данных кластера может включать в себя набор ссылок на параметры, описанные в глобальном массиве параметров (2002 на рисунке 20А), а также структуры данных эталонов и вес отсечки, отличные от других структур данных кластеров. Каждая структура данных кластера специализируется на распознавании подмножества или семейства символов конкретного языка или набора родственных языков. Наконец, на стадии 2122 происходит выделение памяти для таблицы обработанных изображений символа (2072 на рисунке 20Л). Кроме того, происходит статическое либо динамическое выделение памяти и для других различных переменных и массивов и их инициализация.

На рисунке 21В показана блок-схема подпрограммы «process symbol image», вызываемой на стадии 2109 на рисунке 21 А. На стадии 2124 подпрограмма «process symbol image» вычисляет значения для всех параметров, представленных функциями, рассчитывающими значения параметров, или ссылками на такие функции в массиве параметров 2008. Затем во вложенных циклах for на стадиях 2126-2135 подпрограмма «process symbol image» обрабатывает изображение символа, как было описано выше со ссылкой на рисунки 20Г-Ж. Внешний цикл for на стадиях 2126-2135 выполняется для каждой структуры данных кластера. В первом внутреннем цикле for на стадиях 2127-2134 перебираются все структуры данных эталона внутри текущей структуры данных кластера. На стадии 2128 для текущей структуры данных эталона рассчитывается вес W на основе значений параметров эталона, содержащихся в структуре данных эталона, и соответствующих значений параметров, рассчитанных для текущего изображения символа, как было описано выше со ссылкой на рисунок 20D. Если рассчитанный вес

структуры данных кластера больше веса отсечки, как определено на стадии 2129, то обработка текущей структуры данных эталона прекращается и осуществляется переход к стадии 2134, описанной ниже. В противном случае на стадии 2130 рассчитанный вес используется для выбора индекса из сегмента индексов структуры данных эталона, указывающего на код графемы в структуре данных эталона. Затем во внутреннем цикле for на стадиях 2131-2133 происходит добавление голосов к ячейкам массива голосов, на которые указывает каждый код графемы в структуре данных эталона, начиная с первого и заканчивая кодом графемы, индекс которой был выбран на стадии 2130. В текущем осуществлении каждый голос добавляет 1 к числу накопленных голосов за графему. В других осуществлениях голоса могут представлять собой вещественные числа из некоторого диапазона, например, [0.0, 1.0]. Также в других осуществлениях целочисленные значения голосов из некоторого диапазона целых чисел могут использоваться в качестве значения голосования, характеризующего степень сходства графемы и эталона, представляющего структуру данных эталона. На стадии 2136, как было описано выше со ссылкой на рисунок 20Л, коды графем, набравших голоса в процессе обработки текущего изображения символа, сортируются, формируя отсортированный усеченный массив кодов графем (2069 на рисунке 20Л). Затем на стадии 2138 отсортированный усеченный массив кодов графем добавляется в соответствующий текущему изображению символа элемент таблицы обработанных изображений символа (2072 на рисунке 20Л).

На рисунке 21Г в виде блок-схемы показана подпрограмма «process page», вызываемая на стадии 2111 рисунке 21А. На стадии 2140 происходит инициализация новой структуры данных для обрабатываемой страницы, которая будет хранить коды символов или их другие представления для дальнейшей передачи в обработанный документ PD. Затем в цикле for на стадиях 2142-2146 обрабатывается каждое изображение символа внутри изображения страницы содержащего текст документа, полученного на стадии 2102 рисунка 21А. На стадии 2143 происходит обращение к записи таблицы обработанных изображений символа, соответствующей текущему изображению символа; затем на стадии 2144 эта запись используется для выбора символа, наиболее похожего на изображение символа, из числа графем, полученных для изображения символа в процессе обработки подпрограммой «process symbol image» (показанной на рисунке 21В). Существует множество способов определения символа, наиболее соответствующего изображению символа. На стадии 2145 символ или представляющий его код помещается на место в структуре данных обрабатываемой страницы, соответствующее расположению изображения символа на обрабатываемой в настоящий момент странице полученного документа, содержащего текст. После того как были обработаны все изображения символа, содержимое структуры данных обрабатываемой страницы помещается в обработанный документ PD на стадии 2148, после чего память, занятая структурой данных обрабатываемой страницы, освобождается на стадии 2150.

Хотя настоящее изобретение описывается на примере конкретных вариантов осуществления, предполагается, что оно не будет ограничено только этими вариантами осуществления. Специалистам в данной области будут очевидны возможные модификации сущности настоящего изобретения. Например, любой из множества возможных вариантов осуществления структур данных и методов, используемых для предварительной обработки в соответствии с обобщенным третьим вариантом осуществления системы OCR, описанным выше, может быть достигнута путем изменения любого из различных параметров проектирования и осуществлении, среди которых: структуры данных, структуры управления, модульное исполнение, язык

программирования, используемая операционная система и аппаратное обеспечение, а также многие другие подобные параметры проектирования и осуществлении.

Следует понимать, что представленное выше описание раскрытых вариантов осуществления предоставлено для того, чтобы дать возможность любому специалисту в данной области повторить или применить настоящее описание. Специалистам в данной области будут очевидны возможные модификации представленных вариантов осуществления, при этом общие принципы, представленные здесь, могут применяться к другим вариантам осуществления без отступления от сущности или объема описания. Таким образом, настоящее описание не ограничено представленными здесь вариантами осуществления, оно соответствует широкому кругу задач, связанных с принципами и новыми отличительными признаками, раскрытыми настоящем описании.

(57) Формула изобретения

1. Система оптического распознавания символов,

содержащая: один или более процессоров;

один или более модулей памяти;

одно или более запоминающих устройств; и

команды в машинном коде, хранящиеся в одном или более из одного или более запоминающих устройств при выполнении одним или более из одного или более процессоров управляют системой оптического распознавания символов для обработки содержащего текст отсканированного изображения документа за счет:

идентификации изображений символов в содержащем текст отсканированном изображении документа;

для каждой страницы документа,

для каждого изображения символа на странице,

идентификации каждой графемы из набора графем, которая соответствует нормированному изображению символа относительно эталона символа из набора эталонов символов,

сортировки идентифицированных графем по частоте, с которой идентифицированные графемы соответствуют нормированному изображению символа относительно эталонов символа в наборе эталонов символов, и

использования отсортированных идентифицированных графем для выбора кода символа, который представляет нормированное изображение символа; и

подготовки обработанного документа, содержащего коды символов, которые представляют нормированные изображения символов из отсканированного изображения документа, и сохранения обработанного документа на одном или более из одного или более запоминающих устройств и модулей памяти.

2. Система оптического распознавания символов по п. 1, дополнительно содержащая набор структур данных, который хранится в одном или более из одного или более модулей памяти, причем набор структур данных включает:

структуру данных голосов, в которой хранятся накопленные голоса, где каждый накопленный голос связан с кодом графемы и накопленные голоса в структуре данных голосов проиндексированы кодами графем;

структуры данных эталона, каждая из которых представляет эталон символа и включает в себя упорядоченный набор значений параметров, упорядоченный набор индексов и упорядоченный набор кодов графем; и

две или более структур данных кластера, причем каждая структура данных кластера включает в себя упорядоченный набор ссылок на параметры, вес отсечки и одну из

множества структур данных эталона или множества ссылок на структуры данных эталона.

3. Система оптического распознавания символов по п. 2, дополнительно содержащая упорядоченный набор функций, которые формируют значения параметров для нормированных изображений символов.

4. Система оптического распознавания символов по п. 3, в которой идентификация каждой графемы из набора графем, соответствующих нормированному изображению символа, относительно эталона символа из набора эталонов символов дополнительно включает:

инициализацию структуры данных голосов; и

накопление голоса в структуре данных голосов для каждого соответствия графемы нормированному изображению символа относительно эталона символа.

5. Система оптического распознавания символов по п. 4, в которой инициализация структуры данных голосов дополнительно содержит присвоение нулевого значения каждому накопленному голосу из структуры данных голосов.

6. Система оптического распознавания символов по п. 4, в которой накопление голоса в структуре данных голосов для каждого соответствия графемы нормированному изображению символа относительно эталона символа дополнительно содержит:

для каждой структуры данных кластера,

выбор в качестве упорядоченного набора значений параметров, связанного с кластером, значений параметров, формируемых в результате применения к нормированному изображению символа функции, соответствующей каждому параметру, на который ссылается упорядоченный набор ссылок на параметры в структуре данных кластера,

для каждой структуры данных эталона, которая содержится внутри структуры данных кластера или на которую она ссылается,

оценку значений параметров из набора значений параметров в структуре данных эталона относительно значений параметров в упорядоченном наборе значений параметров, связанном с кластером, для формирования значения, и

добавление голосов к одному или более накопленным голосам, хранящимся в структуре данных голосов, для каждой из одной или более графем в том случае, если сравнение сформированного значения и порогового значения указывает на потенциальное соответствие нормированного изображения символа эталону символа.

7. Система оптического распознавания символов по п. 6, в которой оценка значений параметров в наборе значений параметров структуры данных эталона относительно значений параметров в упорядоченном наборе значений параметров, связанном с кластером, формирующая значение, дополнительно содержит добавленные абсолютные значения разностей между каждым значением параметра из набора значений параметров структуры данных эталона и соответствующим значением параметра в упорядоченном наборе значений параметров, связанном с кластером.

8. Система оптического распознавания символов по п. 6, в которой добавление голосов к одному или более накопленным голосам, хранящимся в структуре данных голосов, для каждой из одной или более графем дополнительно содержит:

использование сформированного значения для выбора индекса из упорядоченного набора индексов в структуре данных эталона;

использование выбранного индекса для выбора кода графемы из упорядоченного набора кодов графемы в структуре данных эталона; и

для каждого кода графемы в упорядоченном наборе кодов графем в структуре

данных эталона, начиная с первого кода графемы и заканчивая выбранным кодом графемы,

индексацию структуры данных голосов при помощи кода графемы для доступа к накопленному голосу для графемы, соответствующей коду графемы, и добавление значения к накопленному голосу для графемы.

9. Система оптического распознавания символов по п. 2, в которой каждая структура данных кластера включает в себя структуры данных эталона, которые вместе распознают изображения символов, соответствующие символам из набора или семейства родственных символов.

10. Способ, используемый в системе оптического распознавания символов, имеющей один или более процессоров, один или более модулей памяти, одно или более запоминающих устройств и команды в машинном коде, которые хранятся на одном или более из одного или более запоминающих устройств и выполняются одним или более из одного или более процессоров, причем способ содержит:

идентификацию изображений символов в содержащем текст отсканированном изображении документа;

для каждой страницы документа,

для каждого изображения символа на странице,

идентификацию каждой графемы из набора графем, которая соответствует нормированному изображению символа относительно эталона символа из набора эталонов символов,

сортировку идентифицированных графем по частоте, с которой идентифицированные графемы соответствуют нормированному изображению символа относительно эталонов символа в наборе эталонов символов, и

использование отсортированных идентифицированных графем для выбора кода символа, который представляет нормированное изображение символа; и

получение обработанного документа, содержащего коды символов, которые представляют нормированные изображения символов из отсканированного изображения документа, и сохранение обработанного документа на одном или более из одного или более запоминающих устройств и модулей памяти.

11. Способ по п. 10, в котором система оптического распознавания символов дополнительно включает в себя набор структур данных, хранимых в одном или более из одного или более модулей памяти, включая:

структуру данных голосов, в которой хранятся накопленные голоса, где каждый накопленный голос связан с кодом графемы и накопленные голоса в структуре данных голосов проиндексированы кодами графем;

структуры данных эталона, каждая из которых представляет эталон символа и включает в себя упорядоченный набор значений параметров, упорядоченный набор индексов и упорядоченный набор кодов графем; и

две или более структур данных кластера, причем каждая структура данных кластера включает в себя упорядоченный набор ссылок на параметры, вес отсечки и одну из множества структур данных эталона или множества ссылок на структуры данных эталона.

12. Способ по п. 11, в котором система оптического распознавания символов дополнительно включает в себя упорядоченный набор функций, формирующих значения параметров из нормированных изображений символов.

13. Способ по п. 12, в котором идентификация каждой графемы из набора графем, соответствующих нормированному изображению символа относительно эталона

символа из набора эталонов символов, дополнительно включает:

инициализацию структуры данных голосов; и

накопление голоса в структуре данных голосов для каждого соответствия графемы нормированному изображению символа относительно эталона символа.

14. Способ по п. 13, в котором инициализация структуры данных голосов дополнительно содержит присвоение нулевого значения каждому накопленному голосу из структуры данных голосов.

15. Способ по п. 13, в котором накопление голоса в структуре данных голосов для каждого соответствия графемы нормированному изображению символа относительно эталона символа дополнительно содержит:

для каждой структуры данных кластера,

выбор в качестве упорядоченного набора значений параметров, связанного с кластером, значений параметров, формируемых в результате применения к нормированному изображению символа функции, соответствующей каждому параметру, на который ссылается упорядоченный набор ссылок на параметры в структуре данных кластера,

для каждой структуры данных эталона, которая содержится внутри структуры данных кластера или на которую она ссылается,

оценку значений параметров из набора значений параметров в структуре данных эталона относительно значений параметров в упорядоченном наборе значений параметров, связанном с кластером, для формирования значения, и

добавление голосов к одному или более накопленным голосам, хранящимся в структуре данных голосов, для каждой из одной или более графем в том случае, если сравнение сформированного значения и порогового значения указывает на потенциальное соответствие нормированного изображения символа эталону символа.

16. Способ по п. 15, в котором оценка значений параметров из набора значений параметров структуры данных эталона относительно значений параметров из упорядоченного набора значений параметров, связанного с кластером, формирующая значение, дополнительно содержит добавленные абсолютные значения разностей между каждым значением параметра из набора значений параметров структуры данных эталона и соответствующего значения параметра в упорядоченном наборе значений параметров, связанном с кластером.

17. Способ по п. 15, в котором добавление голосов для каждой из одной или более графем к одному или более накопленным голосам, хранящимся в структуре данных голосов, дополнительно содержит:

использование сформированного значения для выбора индекса из упорядоченного набора индексов в структуре данных эталона;

использование выбранного индекса для выбора кода графемы из упорядоченного набора кодов графемы в структуре данных эталона; и

для каждого кода графемы в упорядоченном наборе кодов графем в структуре данных эталона, начиная с первого кода графемы и заканчивая выбранным кодом графемы,

индексацию структуры данных голосов при помощи кода графемы для доступа к накопленному голосу для графемы, соответствующей коду графемы, и добавление значения к накопленному голосу для графемы.

18. Способ по п. 11, в котором каждая структура данных кластера включает в себя структуры данных эталона, которые вместе распознают изображения символов, соответствующие символам из набора или семейства родственных символов.

19. Запоминающее устройство, хранящее команды в машинном коде, которые, при выполнении одним или более процессорами в системе оптического распознавания символов, управляют системой оптического распознавания символов для:

идентификации изображений символов в содержащем текст отсканированном изображении документа;

для каждой страницы документа,

для каждого изображения символа на странице,

идентификации каждой графемы из набора графем, которая соответствует нормированному изображению символа относительно эталона символа из набора эталонов символов,

сортировки идентифицированных графем по частоте, с которой идентифицированные графемы соответствуют нормированному изображению символа относительно эталона символа из набора эталонов символов, и

использования отсортированных идентифицированных графем для выбора кода символа, который представляет нормированное изображение символа; и

подготовки обработанного изображения, содержащего коды символов, которые представляют нормированные изображения символов в отсканированном изображении документа, и сохранения обработанного документа на одном или более из одного или более запоминающих устройств или модулей памяти.

20. Запоминающее устройство, хранящее команды в машинном коде по п. 19, в которых система оптического распознавания символов дополнительно содержит набор структур данных, который хранится в одном или более из одного или более модулей памяти, причем

набор структур данных включает:

структуру данных голосов, в которой хранятся накопленные голоса, где каждый накопленный голос связан с кодом графемы и накопленные голоса в структуре данных голосов проиндексированы кодами графем;

структуры данных эталона, каждая из которых представляет эталон символа и включает в себя упорядоченный набор значений параметров, упорядоченный набор индексов и упорядоченный набор кодов графем; и

две или более структур данных кластера, причем каждая структура данных кластера включает в себя упорядоченный набор ссылок на параметры, вес отсечки и одну из множества структур данных эталона или множества ссылок на структуры данных эталона.

1

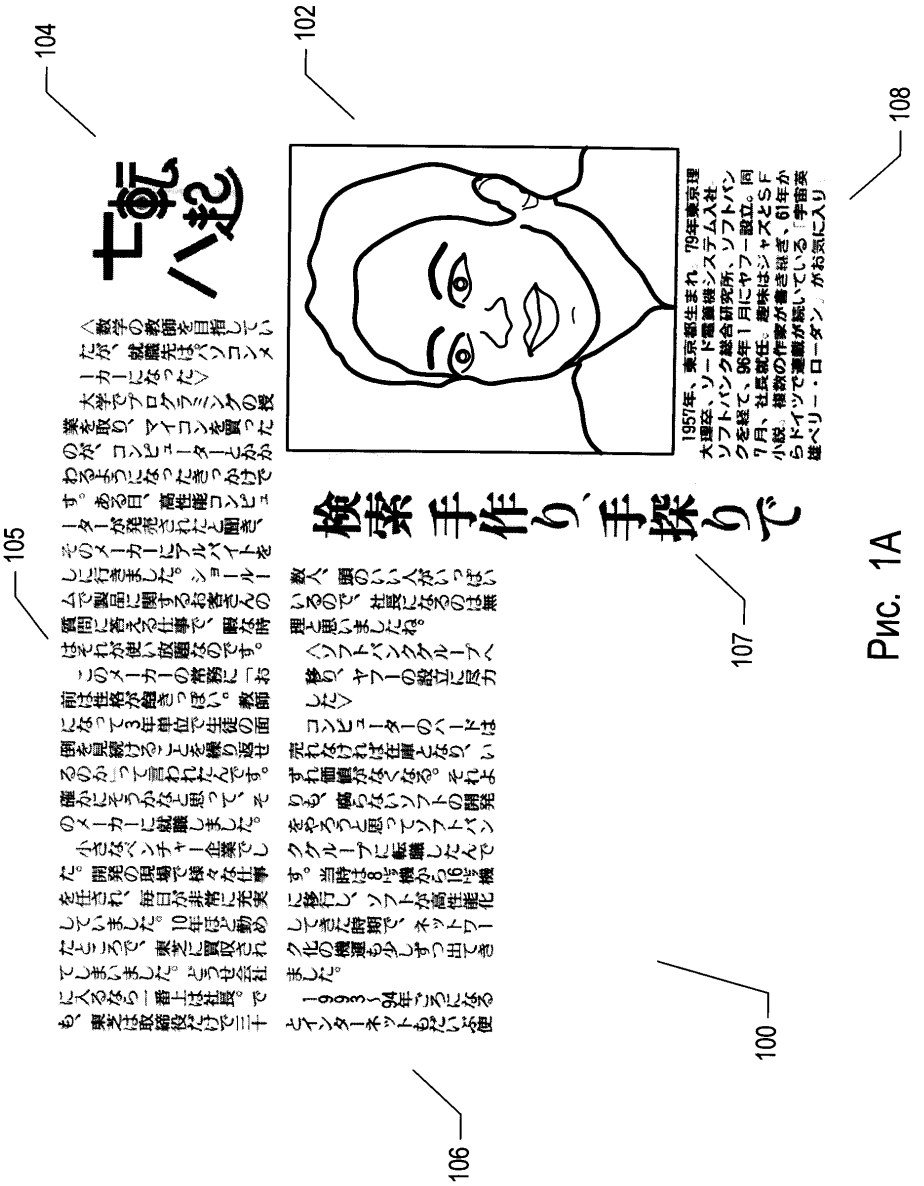

104
102
108
105
107
106
100
検索手作り、手探りで
1957年、東京都生まれ。79年東京理大理学部、ソード電算機システム入社。ソフトバンク総合研究所、ソフトバンクを経て、96年1月にヤフー設立。同7月、社長就任。趣味はジャズとSF小説。複数の作家が書き継ぎ、61年からドイツで連載が続いている「宇宙英雄ペリー・ローダン」がお気に入り

Рис. 1A

2

<Я хотел стать учителем математики, однако в итоге меня взяла на работу компания по производству персональных компьютеров.>

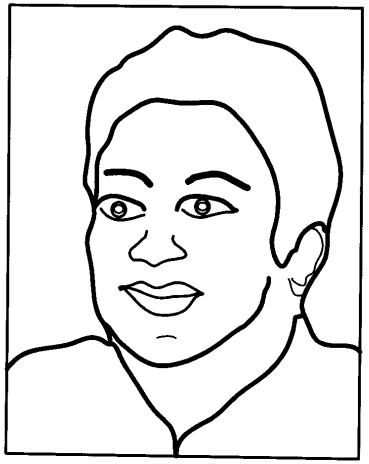
Уроки программирования в колледже и покупка микрокомпьютера сформировали мое желание работать с компьютерами. Однажды, узнав, что в продажу поступил новый высокопроизводительный компьютер, я устроился на временную работу в компьютерную компанию. В мои обязанности входило консультирование клиентов в демонстрационном зале, и я мог пользоваться компьютерами в свободное время.

«Тебе скоро надоест однообразие. Я не могу представить, как ты сможешь заниматься учениками через три года преподавания», — сказал один из руководителей компании, и его слова заставили меня задуматься. Затем я принял решение устроиться на постоянную работу в компании. Это было небольшое коммерческое предприятие, где я уже работал ранее. За время, проведенное в компании, мне представилась возможность поработать на различных должностях, связанных с научно-производственной деятельностью, и каждый день я получал истинное наслаждение от своей работы. На десятом году моей работы в штате компанию приобрела корпорация TOSHIBA, Inc. Если вы работаете в корпорации, вашей целью должно стать «президентское кресло». Однако в TOSHIBA на тот момент работало более тридцати руководителей; многие из них были очень умными людьми, поэтому заполучить президентское кресло было для меня непосильной задачей.

Рис. 1B

Семь раз упасть, восемь раз подняться

Поисковая система ручной работы от Grope



Родился в Токио в 1957 г. Окончил Токийский научный университет в 1979 г., был принят на работу в компанию Sword Machine Systems. Переведен в Soft Bank Laboratory, Soft Bank, затем в январе 1996 г. основал Yahoo Inc. С января 1996 г. президент и генеральный директор Yahoo Inc. Хобби и интересы: джазовая музыка и научно-фантастическая литература. Любимая серия книг: «Перри Родан — герой вселенной», была написана несколькими немецкими писателями-романистами.

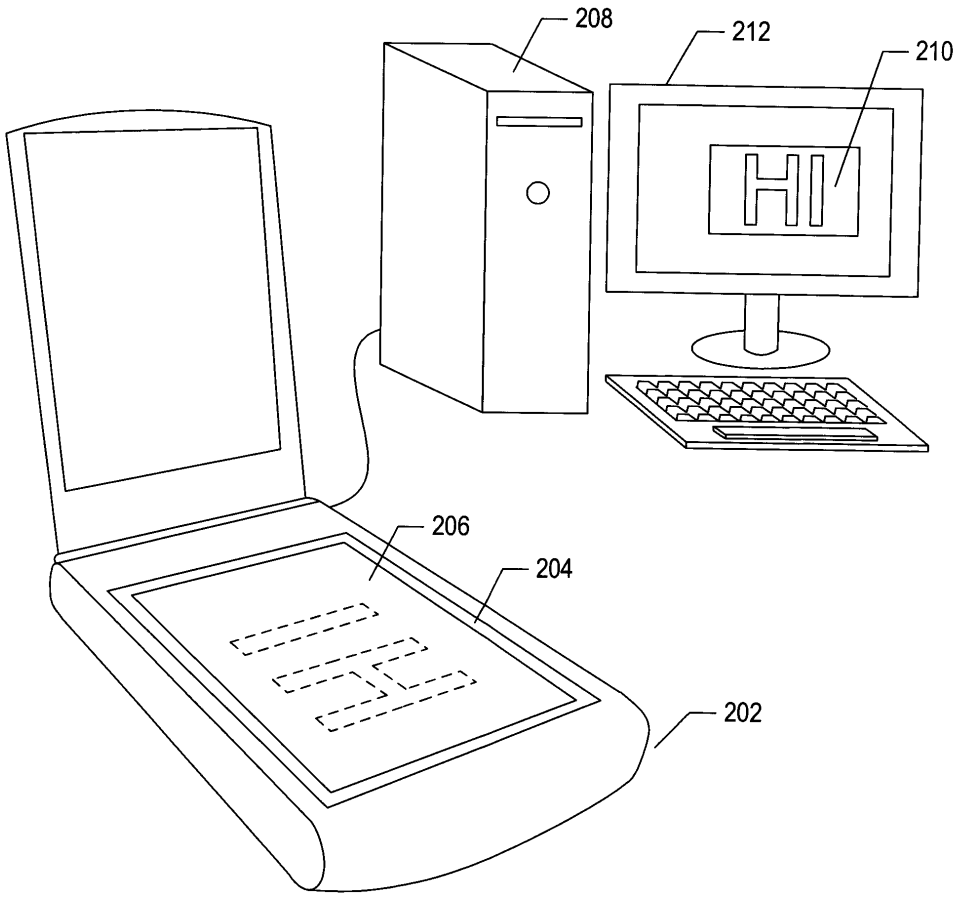


Рис. 2

4

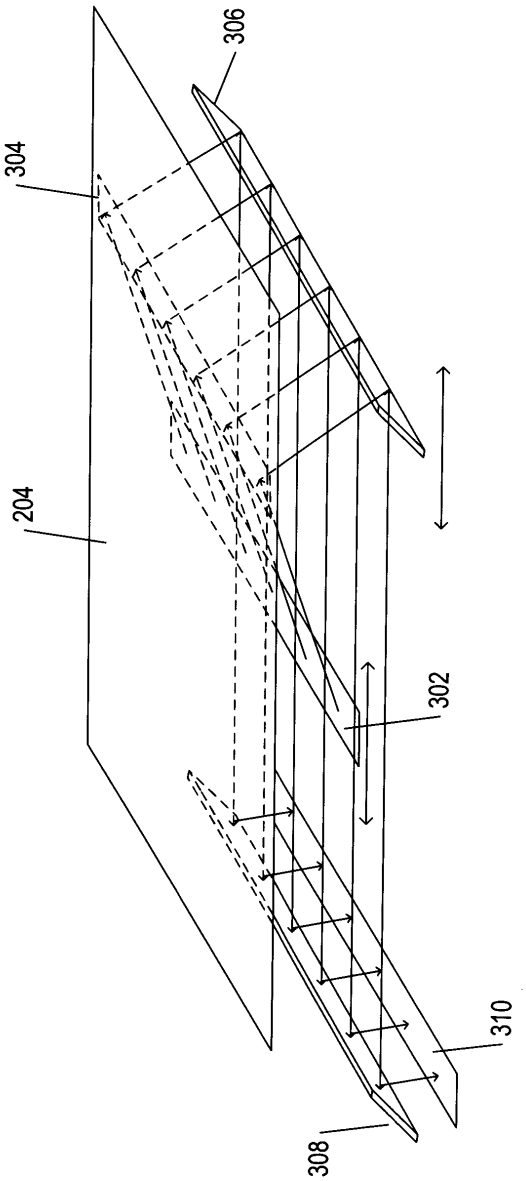


Рис. 3

5

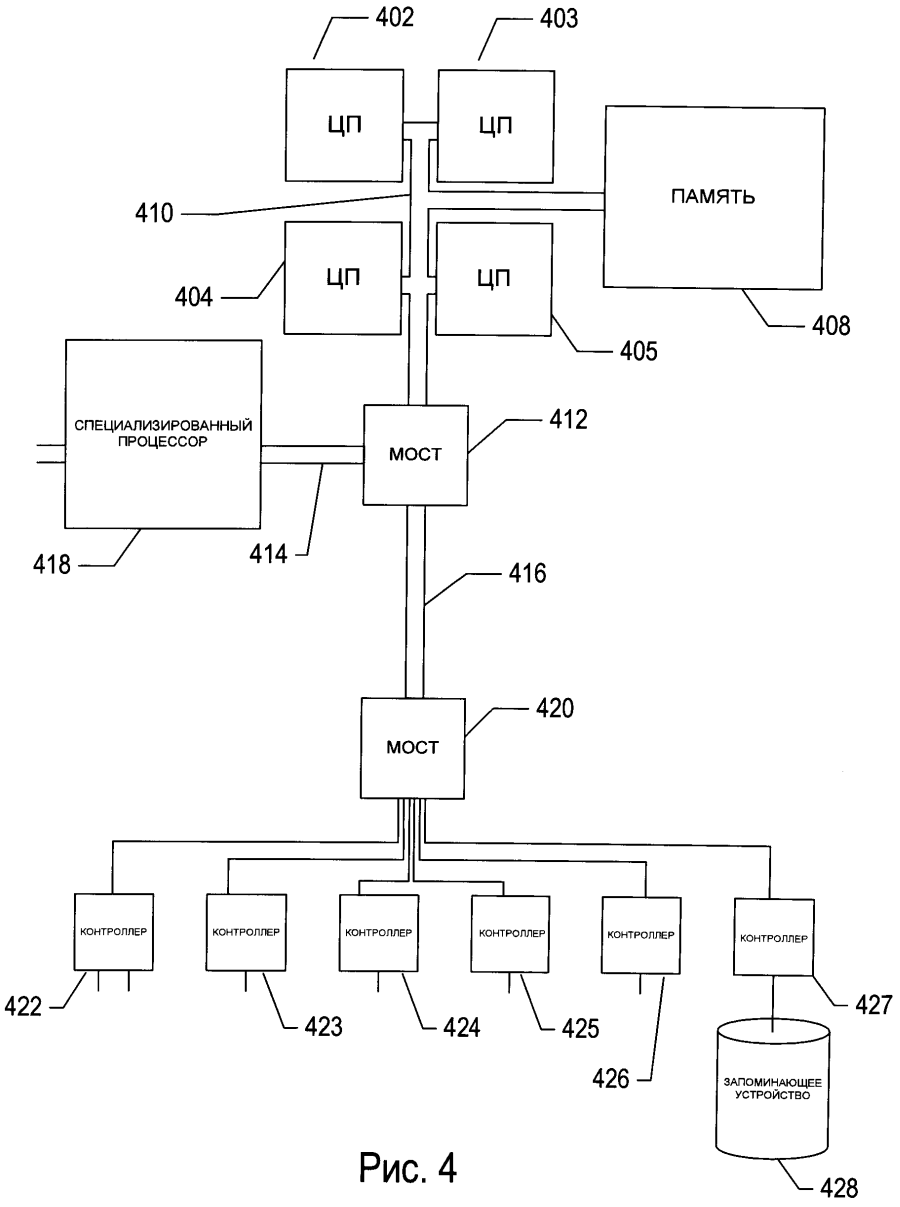

402
403
ЦП
ЦП
ПАМЯТЬ
410
ЦП
ЦП
404
408
405
СПЕЦИАЛИЗИРОВАННЫЙ ПРОЦЕССОР
МОСТ
412
418
414
416
МОСТ
420
КОНТРОЛЛЕР
КОНТРОЛЛЕР
КОНТРОЛЛЕР
КОНТРОЛЛЕР
КОНТРОЛЛЕР
КОНТРОЛЛЕР
422
423
424
425
426
427
ЗАПОМИНАЮЩЕЕ УСТРОЙСТВО
428

Рис. 4

6

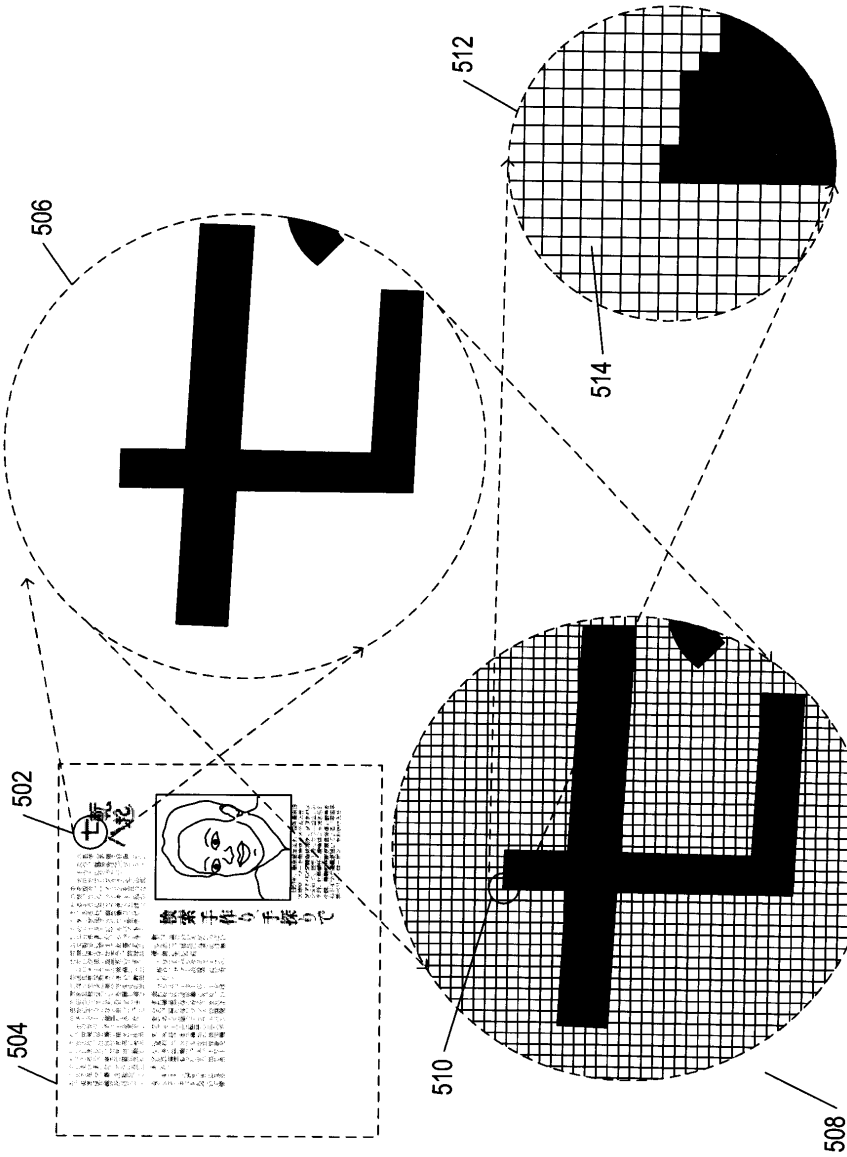


Рис. 5

7

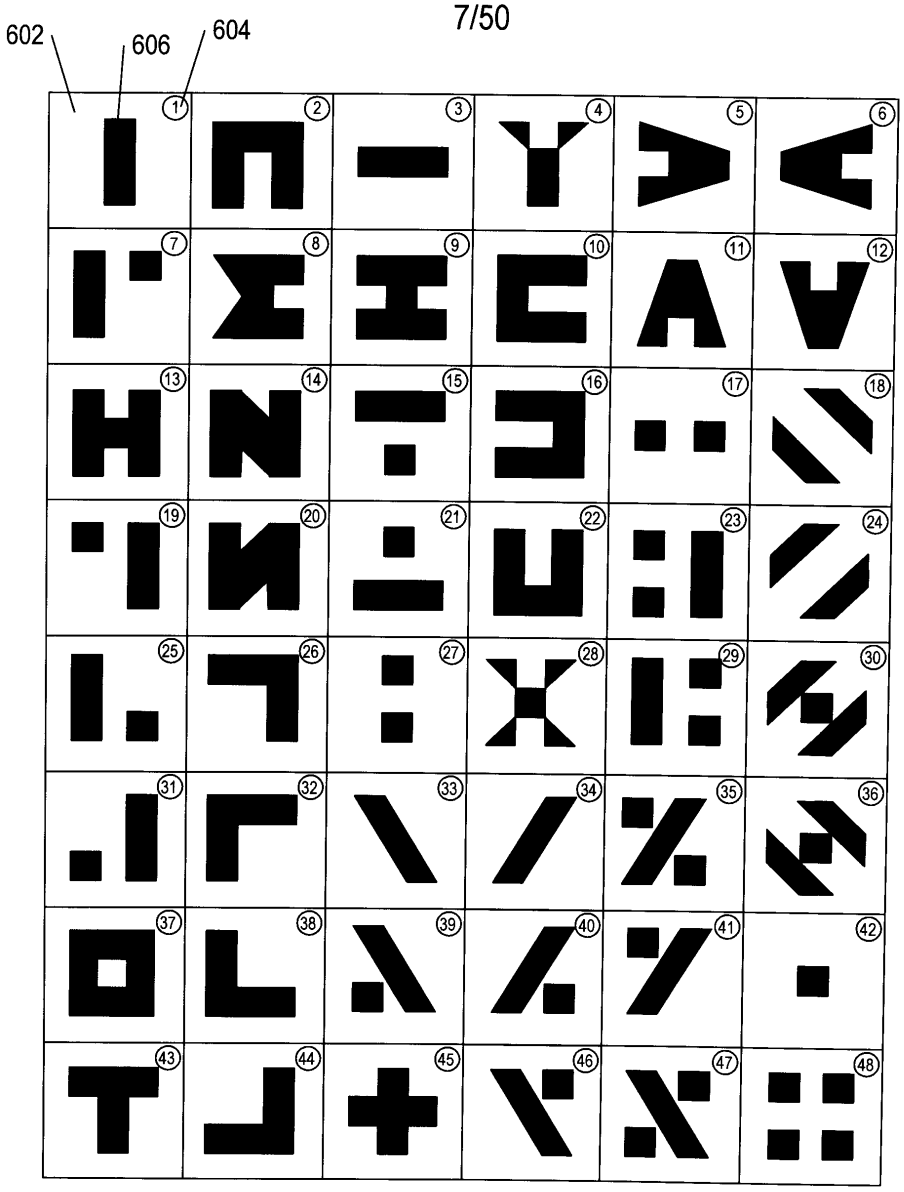

602
606
604
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48

Рис. 6

8

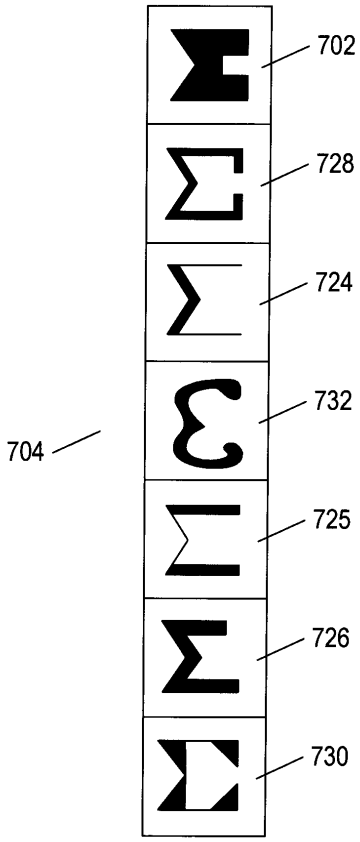

702
728
724
732
704
725
726
730

Рис. 7А

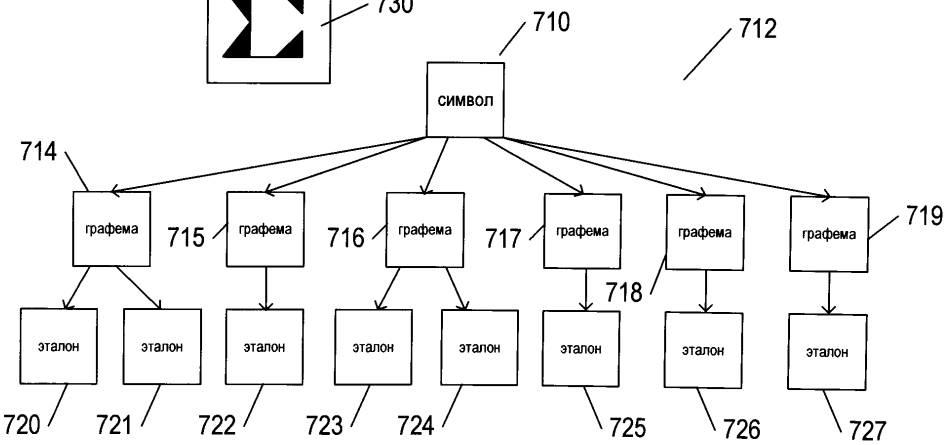

710
712
символ
714
графема
715
графема
716
графема
717
графема
718
графема
719
графема
эталон
720
721
722
723
724
725
726
727

Рис. 7Б

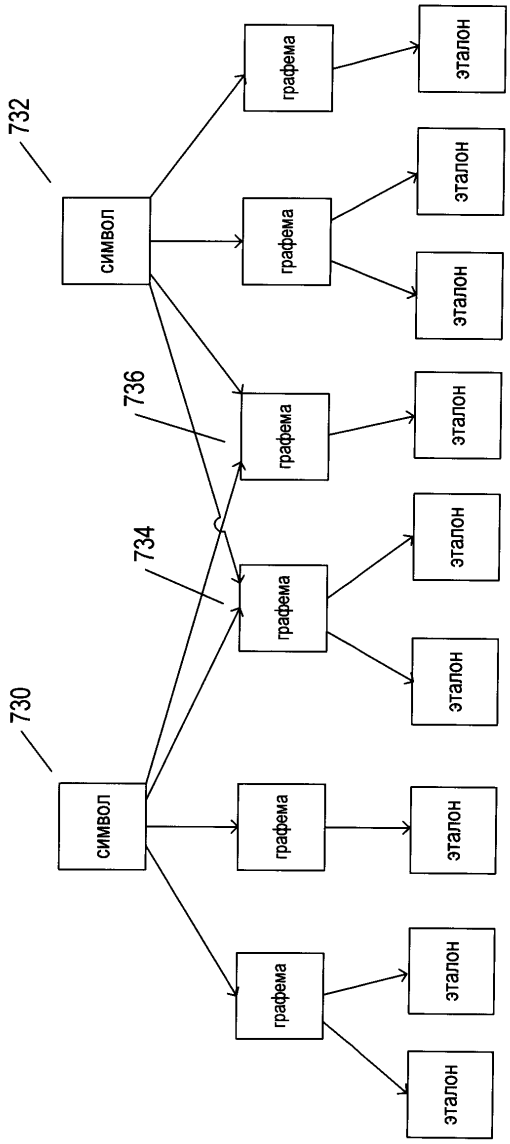

732
734
736
730
символ
символ
графема
графема
графема
графема
графема
графема
графема
эталон
эталон
эталон
эталон
эталон
эталон
эталон
эталон
эталон

Рис. 7B

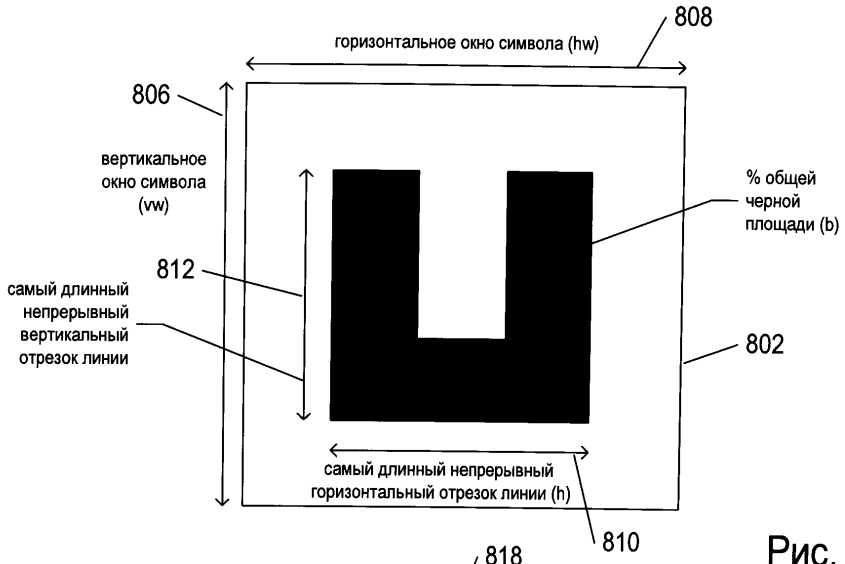

808
горизонтальное окно символа (hw)
806
вертикальное окно символа (vw)
% общей черной площади (b)
812
самый длинный непрерывный вертикальный отрезок линии
802
самый длинный непрерывный горизонтальный отрезок линии (h)
810

Рис. 8A

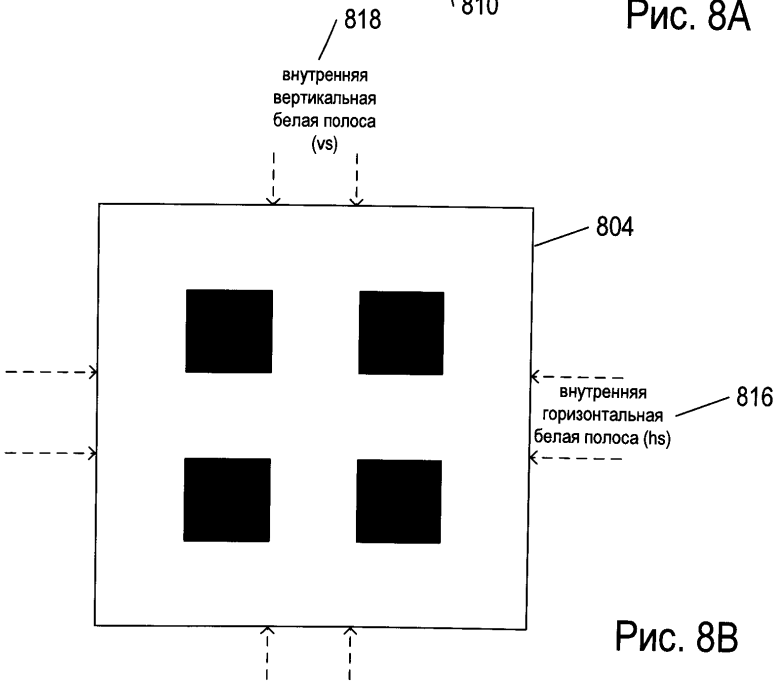

818
внутренняя вертикальная белая полоса (vs)
804
внутренняя горизонтальная белая полоса (hs)
816

Рис. 8B

11/50

символ	904 h/hw	906 v/vw	908 b	910 число vs	912 число hs	914 число vs + число hs	916 v/h
1	0,2	0,6	0,12	0	0	0	3
2	0,6	0,6	0,28	0	0	0	1
3	0,6	0,2	0,12	0	0	0	0,33
4	0,2	0,6	0,12	0	0	0	3
5	0,6	0,46	0,2	0	0	0	0,77
6	0,6	0,46	0,2	0	0	0	0,77
7	0,2	0,6	0,6	1	0	1	3
8	0,6	0,6	0,28	0	0	0	1
9	0,6	0,6	0,28	0	0	0	1
10	0,6	0,6	0,28	0	0	0	1
11	0,46	0,6	0,2	0	0	0	1,3
12	0,46	0,6	0,2	0	0	0	1,3
13	0,6	0,6	0,28	0	0	0	1
14	0,6	0,6	0,32	0	0	0	1
15	0,6	0,2	0,16	0	1	1	0,33
16	0,6	0,6	0,28	0	0	0	1
17	0,2	0,2	0,08	1	0	1	1
18	0,2	0,2	0,12	0	0	0	1
19	0,2	0,6	0,16	1	0	1	3
20	0,6	0,6	0,32	0	0	0	1
21	0,6	0,2	0,16	0	1	1	0,33
22	0,6	0,6	0,28	0	0	0	1
23	0,2	0,6	0,2	1	0	1	3
24	0,2	0,2	0,12	0	0	0	1
25	0,2	0,6	0,16	1	0	1	3
26	0,6	0,6	0,2	0	0	0	1
27	0,2	0,2	0,08	0	1	1	1
28	0,2	0,6	0,08	0	1	1	1
29	0,2	0,6	0,2	1	0	1	3
30	0,2	0,2	0,16	0	0	0	1
31	0,2	0,6	0,16	1	0	1	3
32	0,6	0,6	0,2	0	0	0	1
,33	0,2	0,33	0,15	0	0	0	1,7
34	0,2	0,33	0,15	0	0	0	1,7
35	0,2	0,33	0,23	0	0	0	1,7
36	0,2	0,2	0,16	0	0	0	1
37	0,6	0,6	0,32	0	0	0	1
38	0,6	0,6	0,2	0	0	0	1
39	0,2	0,33	0,19	0	0	0	1,7
40	0,2	0,33	0,19	0	0	0	1,7
41	0,2	0,33	0,19	0	0	0	1,7
42	0,2	0,2	0,04	0	0	0	1
43	0,6	0,6	0,2	0	0	0	1
44	0,6	0,6	0,2	0	0	0	1
45	0,6	0,6	0,2	0	0	0	1
46	0,2	0,33	0,19	0	0	0	1,7
47	0,2	0,33	0,23	0	0	0	1,7
48	0,2	0,2	0,16	1	1	2	1

920

902

Рис. 9

12

12/50

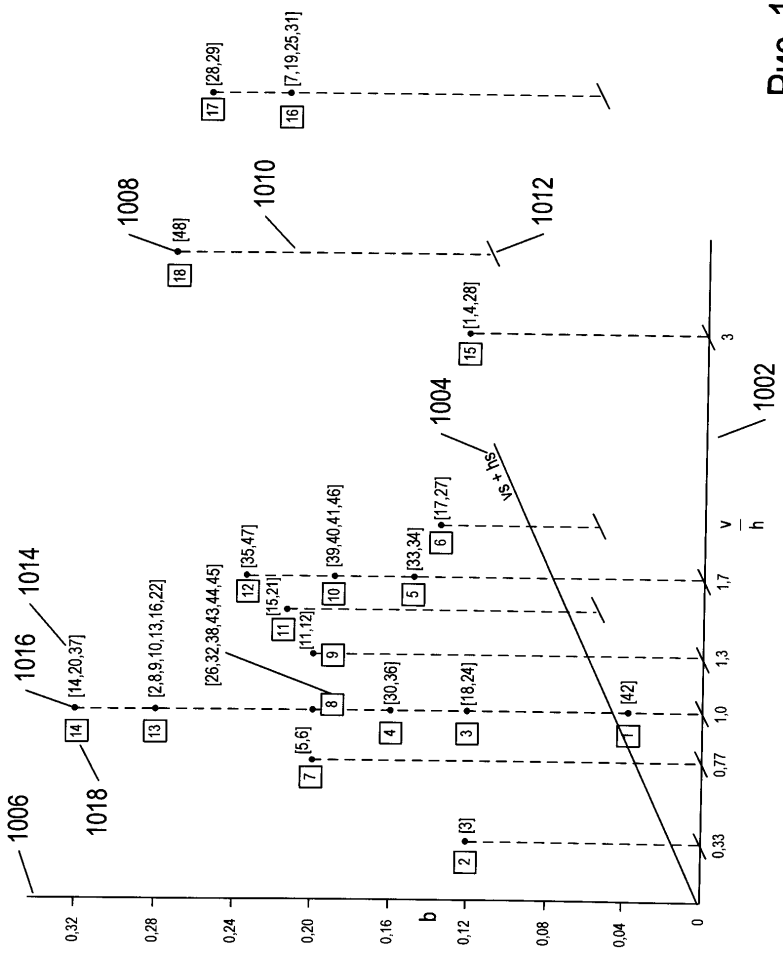


Рис. 10

13

13/50

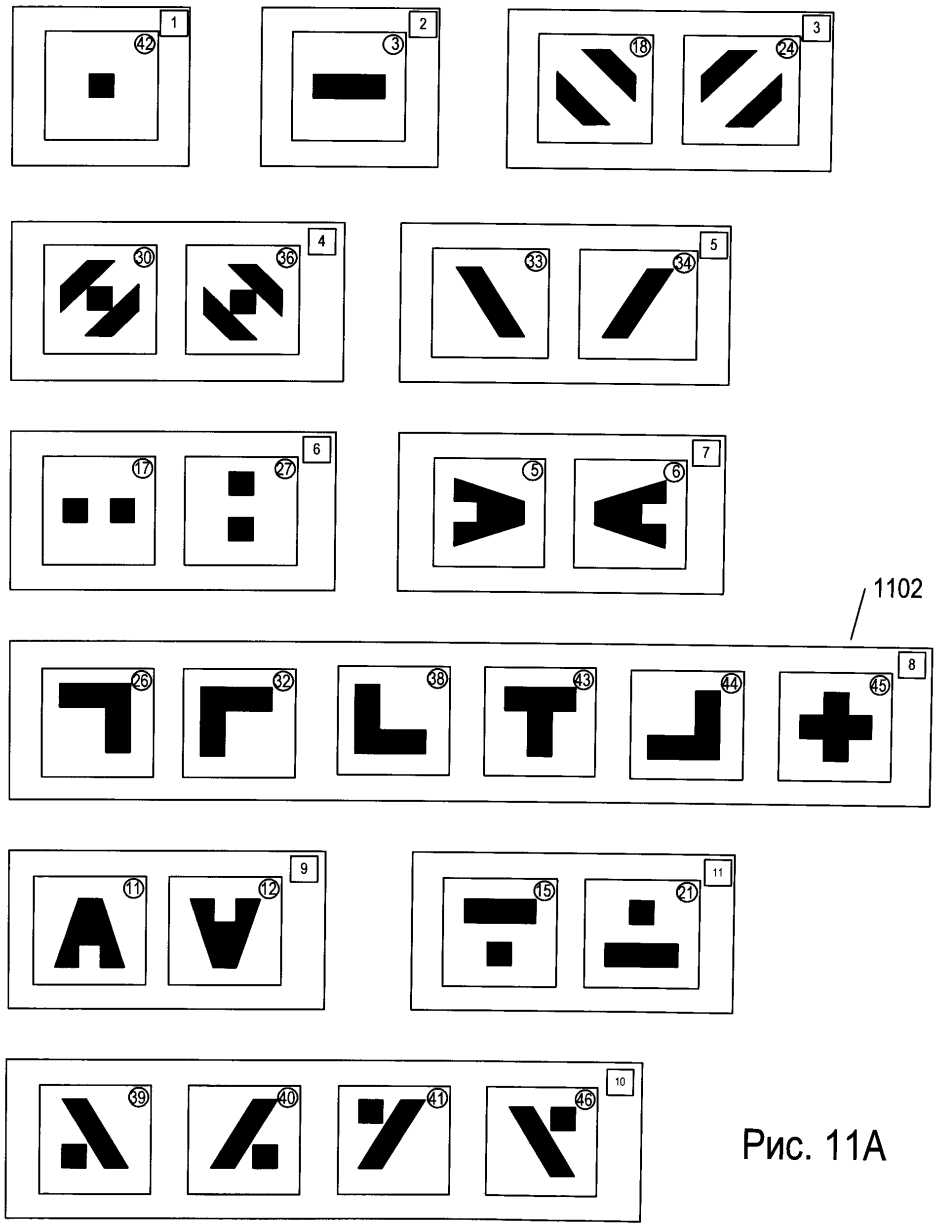


Рис. 11A

14

14/50

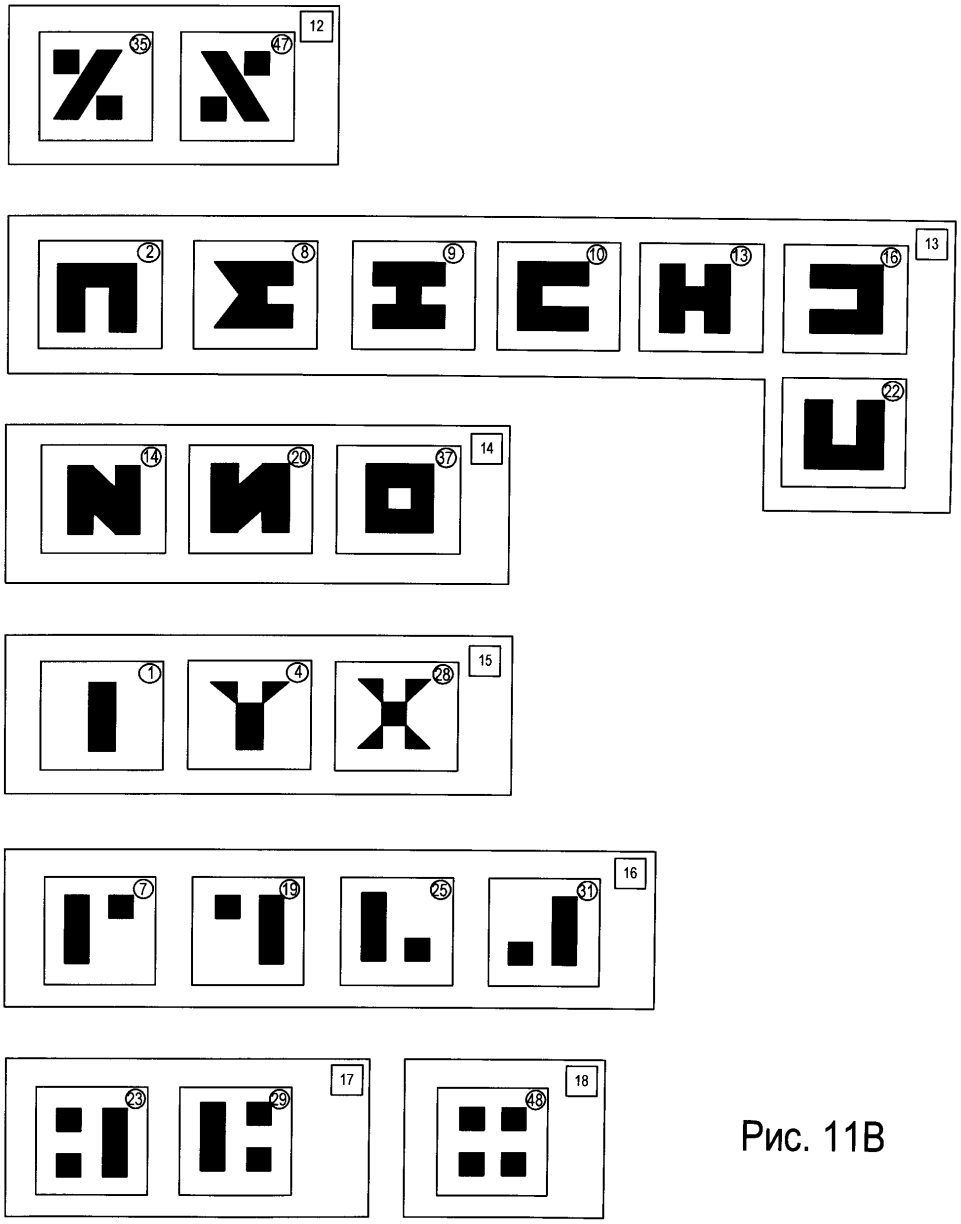
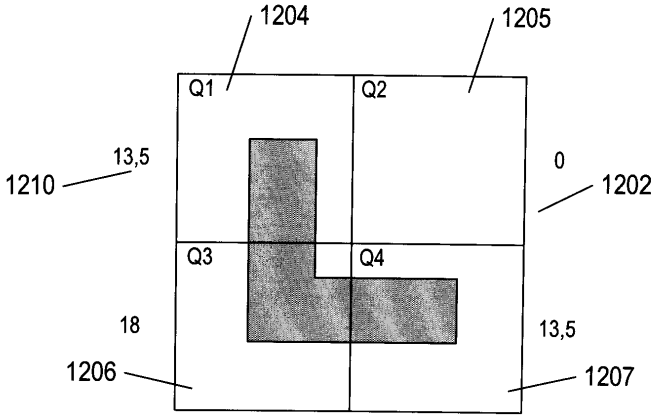


Рис. 11В

15



1214

1212

```
if (Q1 == Q2 == Q3 == Q4)   p = 0   /* симметрия в четырех квадрантах
else if (Q1 == Q2 and Q3 == Q4 and Q3 > Q1)   p = 1   /* вертикальная симметрия в нижней
части
else if (Q1 == Q2 and Q3 == Q4 and Q3 < Q1)   p = 2   /* вертикальная симметрия в верхней части
else if (Q1 = Q3 and Q2 = Q4 and Q1 > Q2)   p = 3   /* горизонтальная симметрия в левой части
else if (Q1 = Q3 and Q2 = Q4 and Q1 < Q2)   p = 4   /* горизонтальная симметрия в правой части
else if (Q1 > Q2 and Q1 > Q3 and Q1 > Q4)   p = 5   /* верхний левый квадрант
else if (Q2 > Q1 and Q2 > Q3 and Q2 > Q4)   p = 6   /* верхний правый квадрант
else if (Q3 > Q1 and Q3 > Q2 and Q3 > Q4)   p = 7   /* нижний левый квадрант
else if (Q4 > Q1 and Q4 > Q2 and Q4 > Q3)   p = 8   /* нижний правый квадрант
else   p = 9                                       /* диагональ
```

Рис. 12A

16

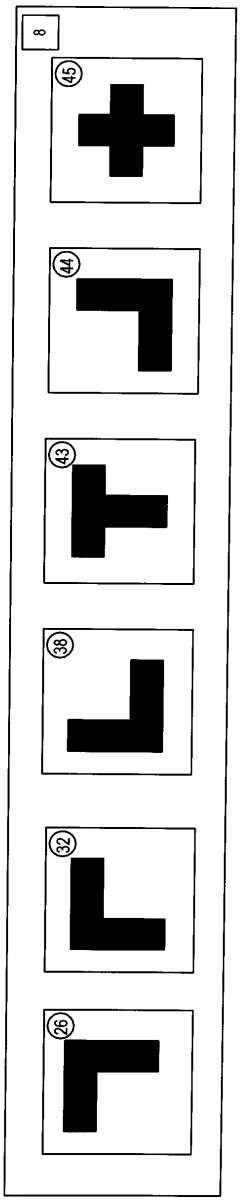


Рис. 12B

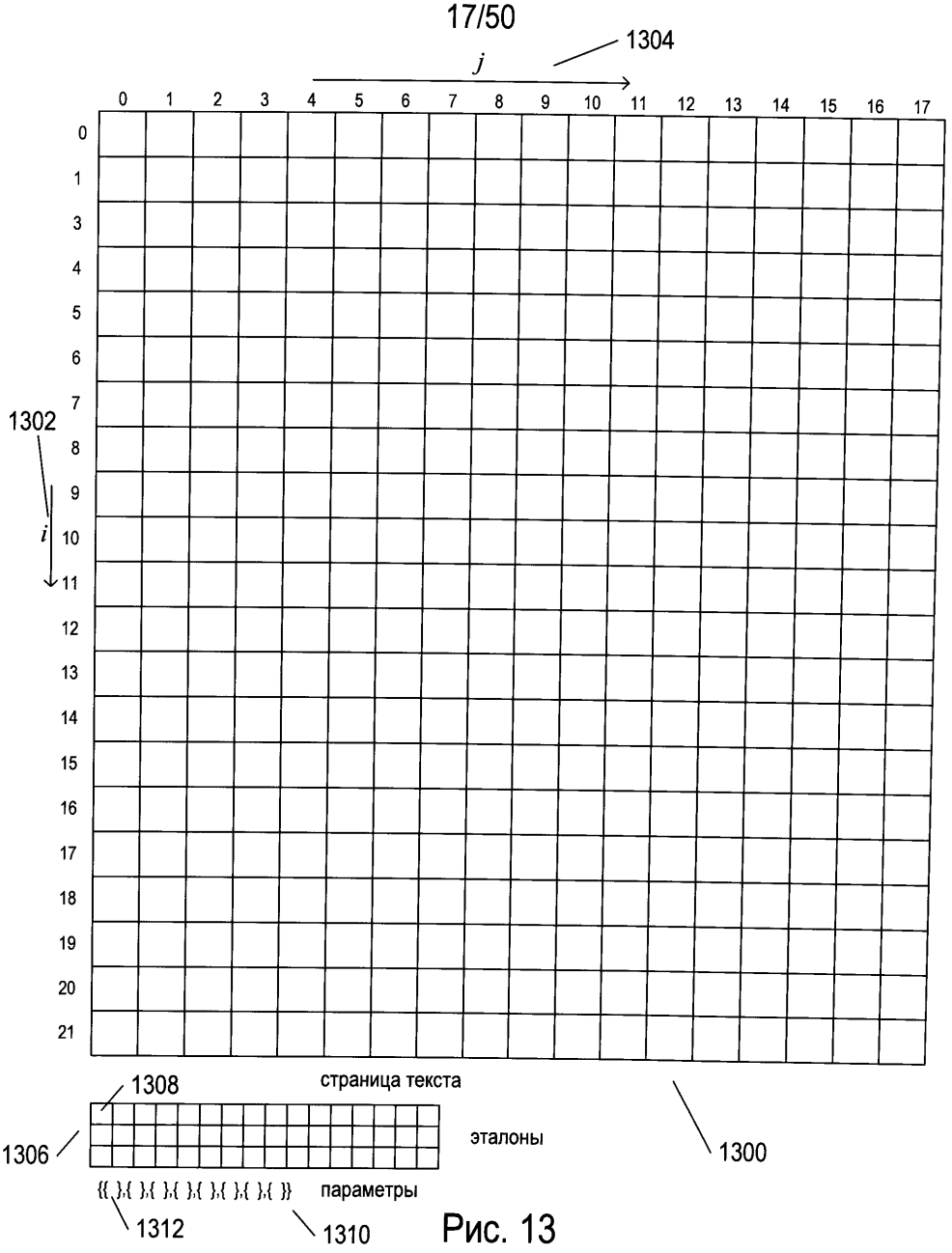

1304
j
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
1302
i
1308
страница текста
1306
эталоны
1300
{{ },{ },{ },{ },{ },{ },{ },{ }}
параметры
1312
1310

Рис. 13

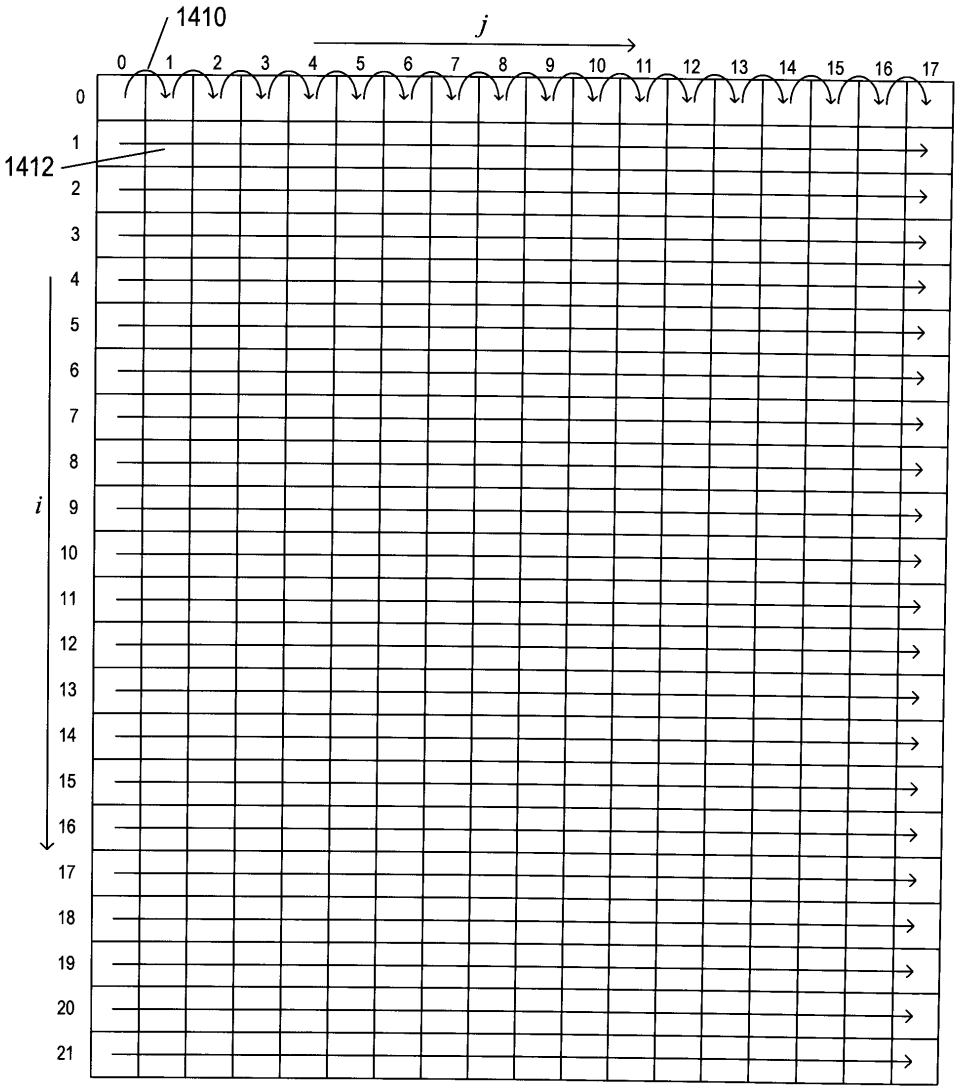
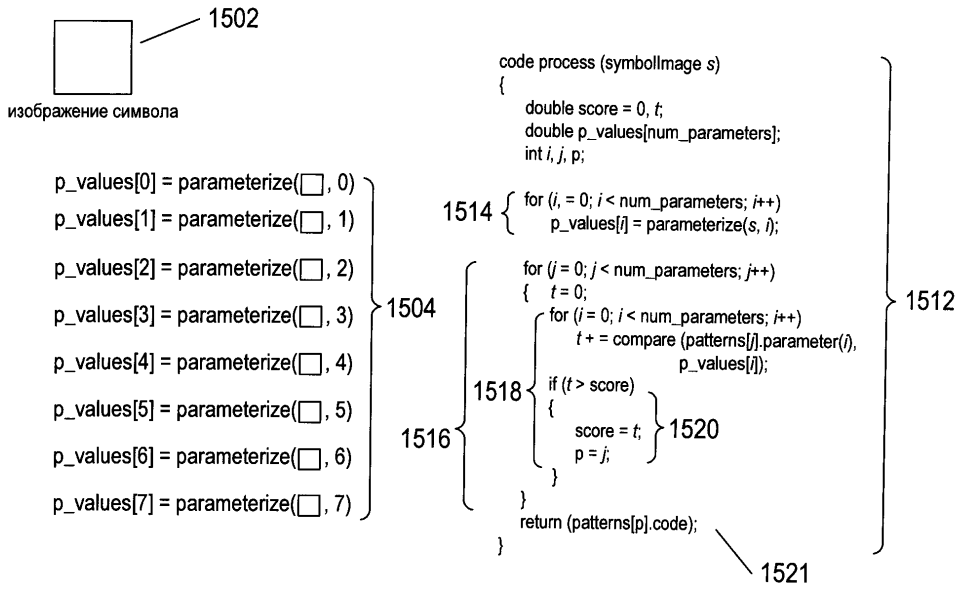
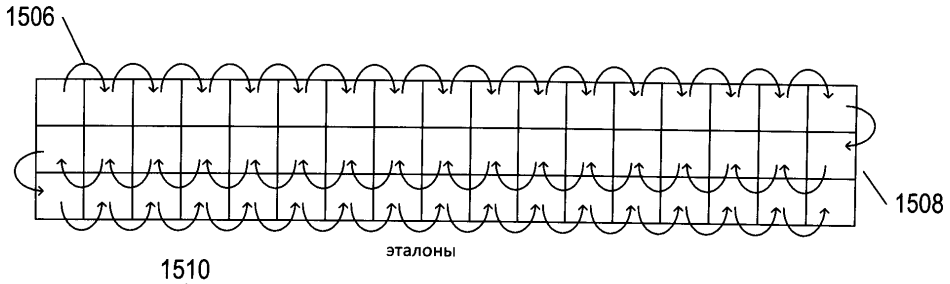

1410
j
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
1412
i
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
страница текста
1402
for (i = 0; i < 22; i++)
for (j = 0; j < 18; j++)
processed_text[i,j] = process (page_of_text[i,j]);
1404
1406
1300

Рис. 14


1502
изображение символа
p_values[0] = parameterize(□, 0)
p_values[1] = parameterize(□, 1)
p_values[2] = parameterize(□, 2)
p_values[3] = parameterize(□, 3)
p_values[4] = parameterize(□, 4)
p_values[5] = parameterize(□, 5)
p_values[6] = parameterize(□, 6)
p_values[7] = parameterize(□, 7)
1504
code process (symbolImage s)
{
double score = 0, t;
double p_values[num_parameters];
int i, j, p;
1514
for (i, = 0; i < num_parameters; i++)
p_values[i] = parameterize(s, i);
for (j = 0; j < num_parameters; j++)
{ t = 0;
for (i = 0; i < num_parameters; i++)
t + = compare (patterns[j].parameter(i),
p_values[i]);
1516
1518
if (t > score)
{
score = t;
p = j;
1520
}
}
return (patterns[p].code);
}
1512
1521


1506
1508
эталоны

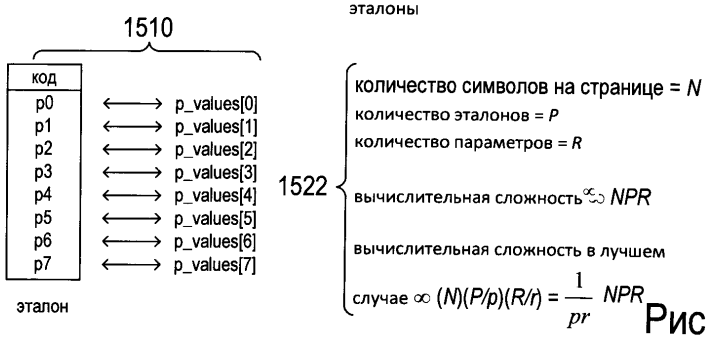

1510
код
p0 ⟷ p_values[0]
p1 ⟷ p_values[1]
p2 ⟷ p_values[2]
p3 ⟷ p_values[3]
p4 ⟷ p_values[4]
p5 ⟷ p_values[5]
p6 ⟷ p_values[6]
p7 ⟷ p_values[7]
эталон
1522
количество символов на странице = N
количество эталонов = P
количество параметров = R
вычислительная сложность ∝ NPR
вычислительная сложность в лучшем случае ∝ (N)(P/p)(R/r) = 1/pr NPR

Рис. 15

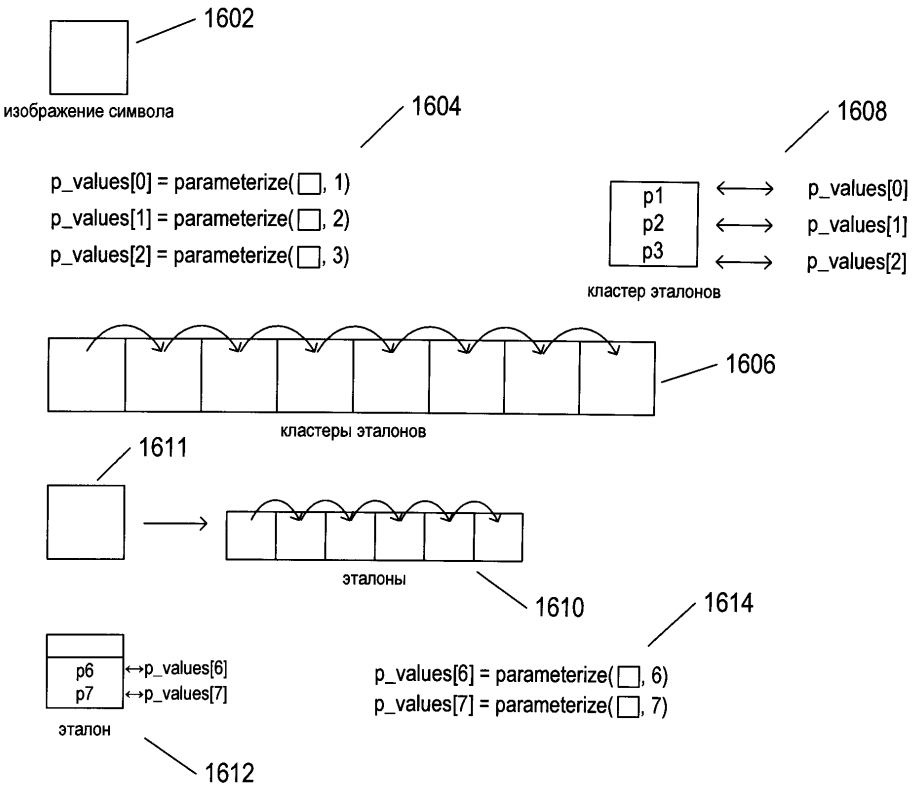

1602
изображение символа
1604
p_values[0] = parameterize(, 1)
p_values[1] = parameterize(, 2)
p_values[2] = parameterize(, 3)
1608
p1
p2
p3
p_values[0]
p_values[1]
p_values[2]
кластер эталонов
1606
кластеры эталонов
1611
эталоны
1610
1614
p6
p7
p_values[6]
p_values[7]
p_values[6] = parameterize(, 6)
p_values[7] = parameterize(, 7)
эталон
1612

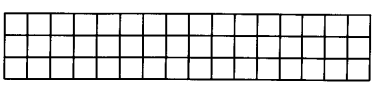
Рис. 16A

```
code process (symbolImage s)
{
    double score = 0, t;
    double p_values[num_parameters];
    int i, j, k, c, p;
1624
    for (i = 0; i < num_InitialParameters; i++) p_values[i] = parameterize(s, i);
    for (j = 0; j < num_clusters; j++)
    {
        t = 0;
        for (i = 0; i < num_InitialParameters; i++) t + = compare(clusters[j].parameter(i), p_values[i]);
1620    if (t > score)
        {
            score = t;
            c = j;
        }
    }
    score = 0;
    for (i = num_InitialParameters; i < clusters[c].numAdditionalParameters; i++)
    {
1626    k = clusters[c].additionalParameter(i);
        p_values[i] = parameterize(s, k);
    }
    for (j = 0; j < clusters[c].numPatterns; j++)
    {
        t = 0;
        for (i = numInitialParameter; k = 1; i < clusters[c].numAdditionalParameters; i++, k++)
            t + = compare(clusters[c].patterns[j].parameter(k), p_values[i]);
1622    if (t > score)
        {
            score = t;
            p = j;
        }
    }
    return (clusters[c].patterns[p].code);
}
```

количество символов на странице = N
количество кластеров = C
количество эталонов/кластер = P'
количество исходных параметров = R_1
количество дополнительных параметров = R_2

вычислительная сложность = $N(CR_1 + P'R_2)$ } 1630

Рис. 16B




```
for (i = 0; i < 22; j++)
    for (j = 0; j < 18; j++)
        {
            NxtLvlClusters[(i*18)+j] = new cluster [num_clusters];
            orderByVoting (page_of_text [i,j], NxtLvlClusters[(i*18)+j])
        }
```

1704, 1706, 1708

```
list* process (symbolImage s, cluster* clusters)
{
    list* similarPatterns;
    double score = 0, t;
    double p_values[num_parameters];
    int i,j;
    similarPatterns = new (list);
    for (i = 0; i < num_parameters; i++) p_values[i] = parameterize(s, i);
    for (j = 0; j < num_clusters &&!similarPatterns → full( ); j++)
    {
        if (similar (p_values, clusters[j]))
        {
            for (k = 0; k < clusters[j].numPatterns( ); k++)
            {
                if (similar(p_values, clusters[j].patterns[k]))
                    similarPatterns → add(clusters[j].patterns[k]);
                if (similarPatterns→ full( )) break;
            }
        }
    }
    return similarPatterns;
}
```

1712, 1710, 1714, 1716

Рис. 17

вычислительная сложность = $N\left(e + \frac{1}{d}(CR_1 + P'R_2)\right)$ — 1720, 1722, 1744

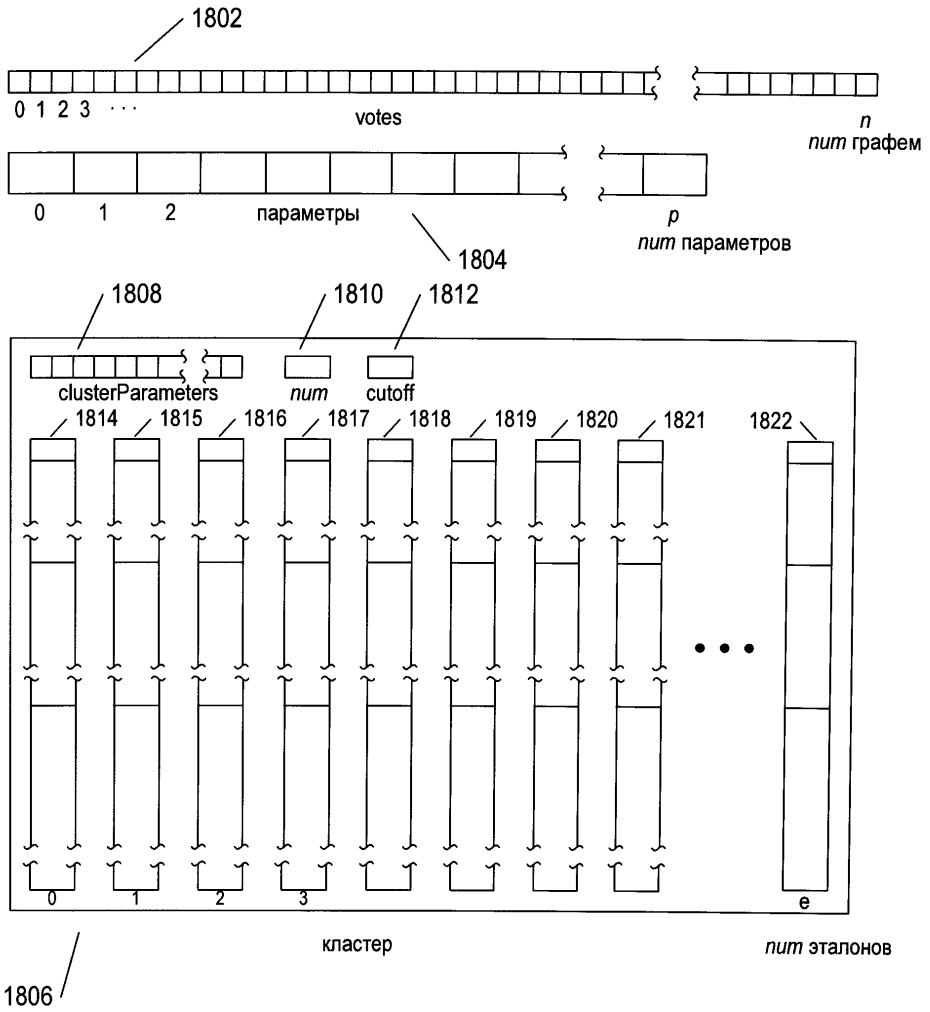

1802
0 1 2 3 ···
votes
n
num графем
0
1
2
параметры
p
num параметров
1804
1808
1810
1812
clusterParameters
num
cutoff
1814
1815
1816
1817
1818
1819
1820
1821
1822
0
1
2
3
e
кластер
num эталонов
1806

Рис. 18

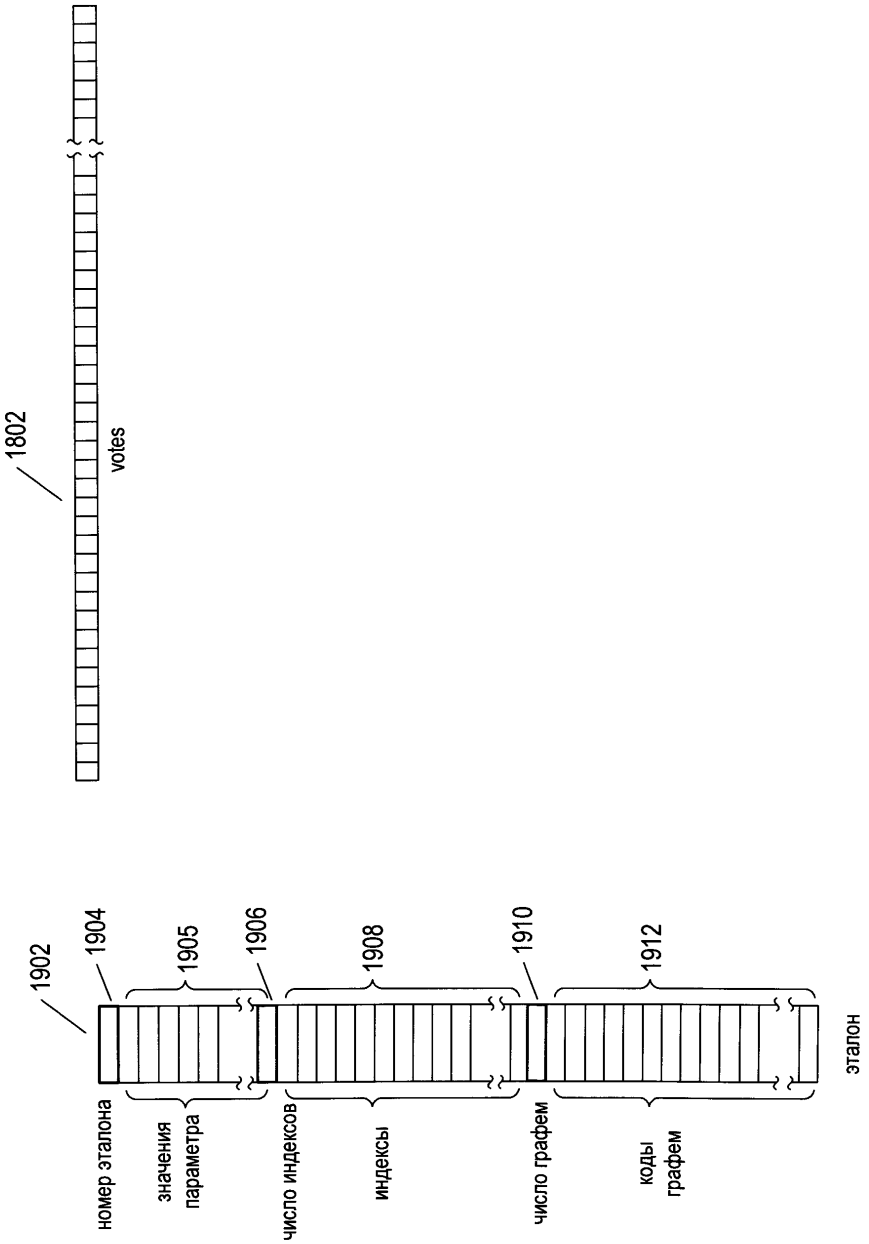

1802
votes
1902
1904
1905
1906
1908
1910
1912
эталон
номер эталона
значения параметра
число индексов
индексы
число графем
коды графем

Рис. 19А

25/50

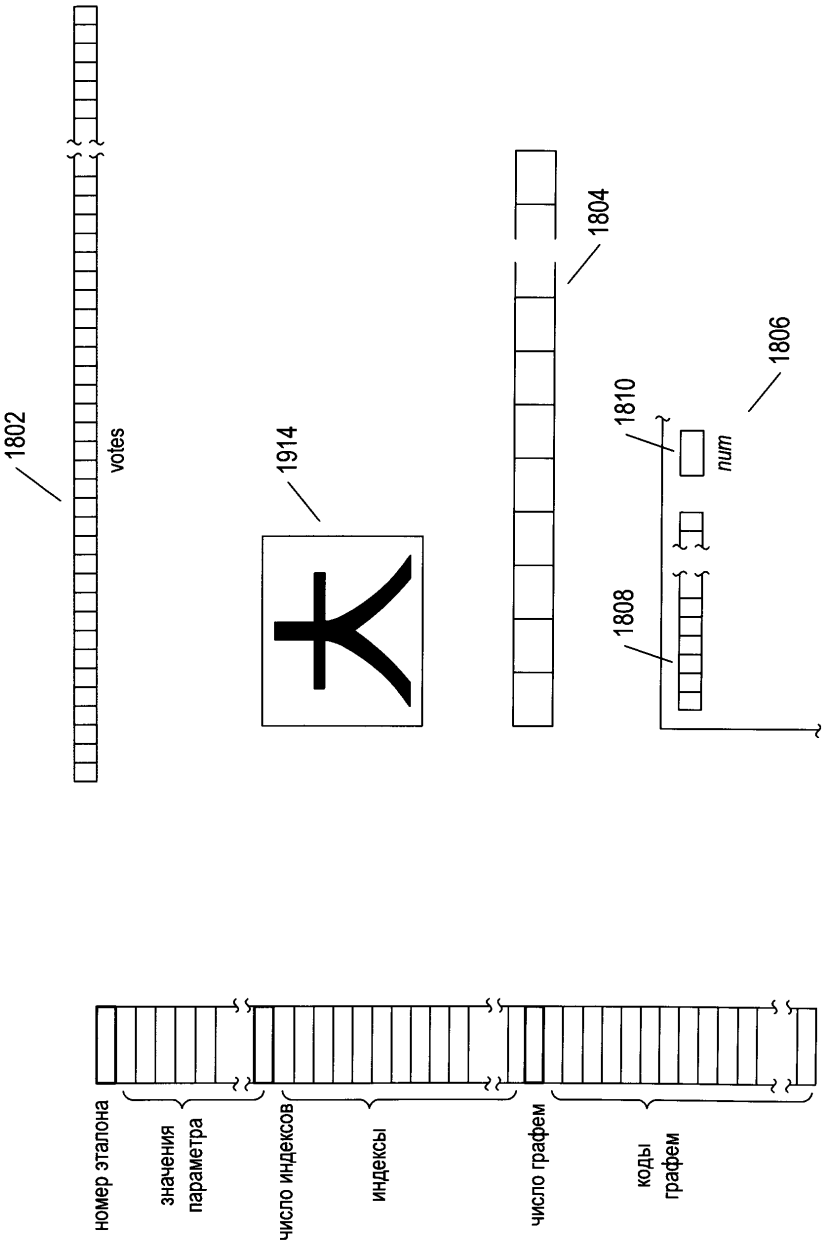


Рис. 19Б

26

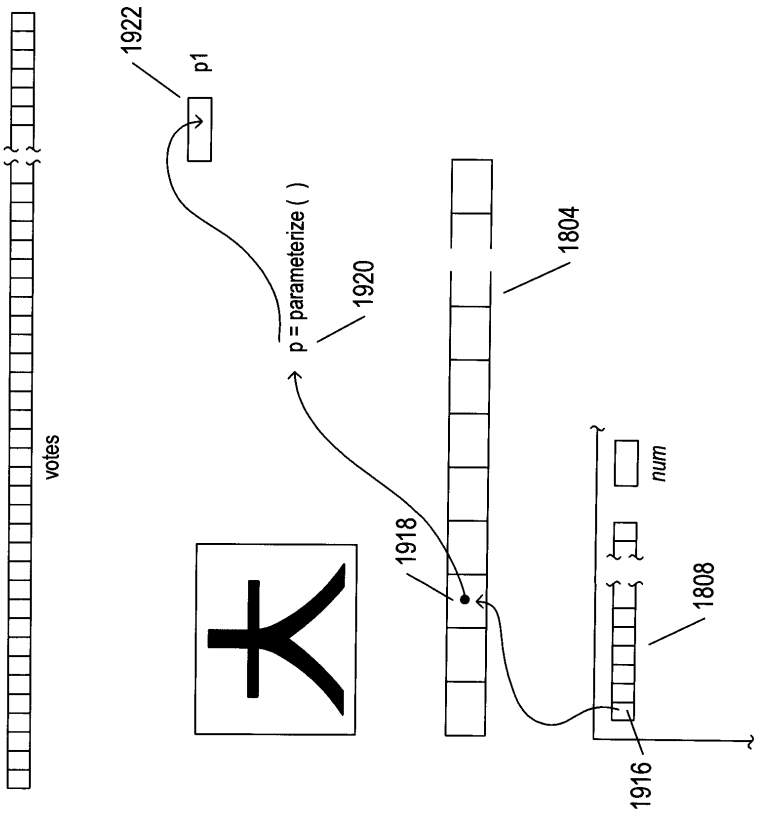
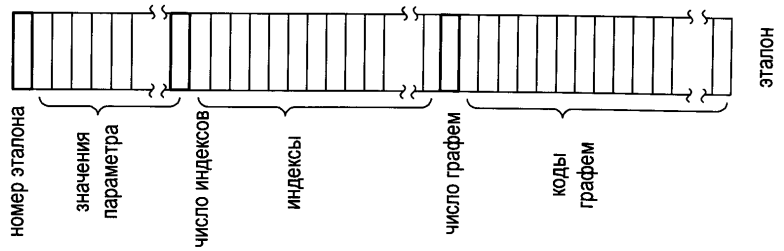

votes
1922
p1
p = parameterize ( )
1920
1804
1918
num
1808
1916

Рис. 19B


номер эталона
значения параметра
число индексов
индексы
число графем
коды графем
эталон

27/50

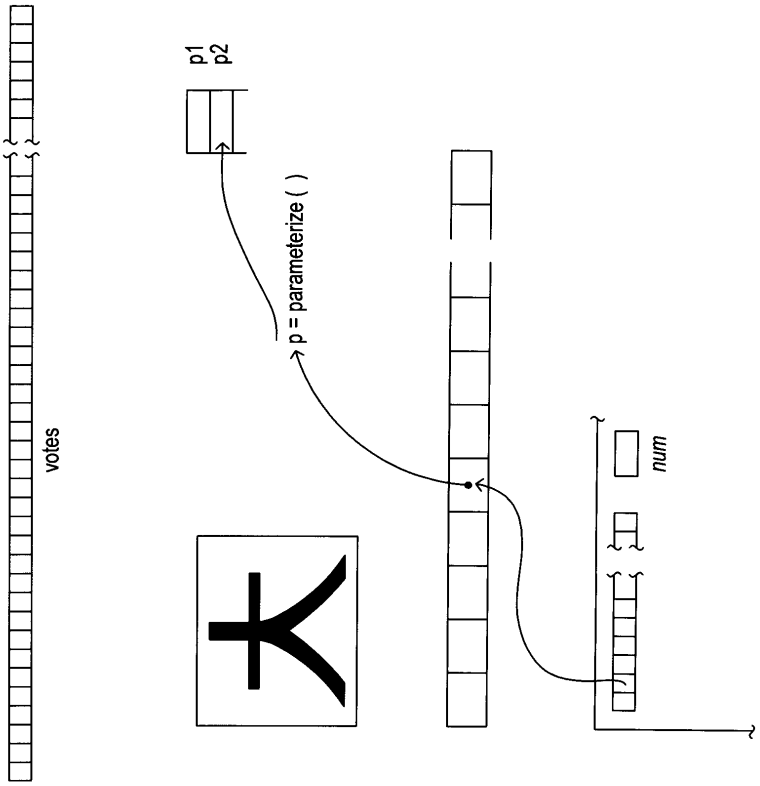
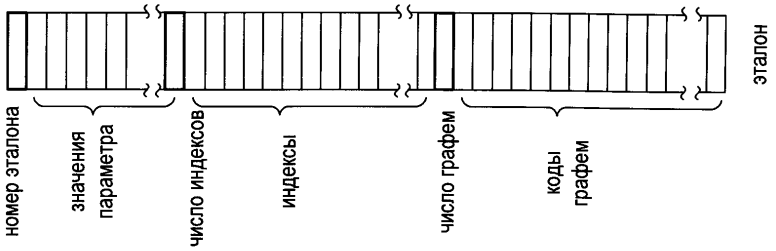


Рис. 19Г



28

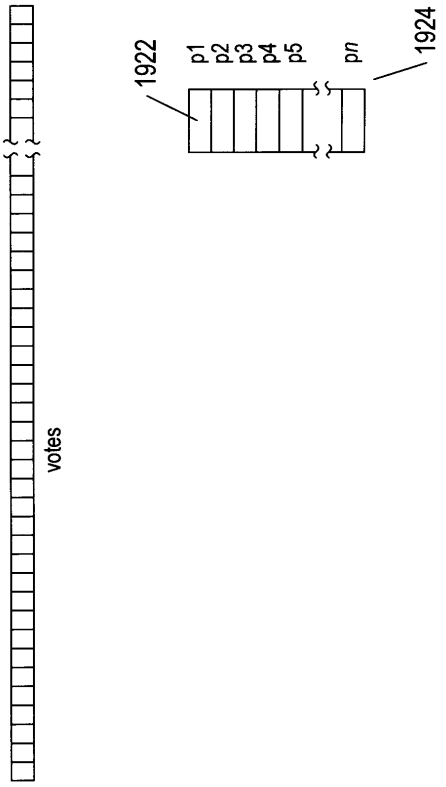
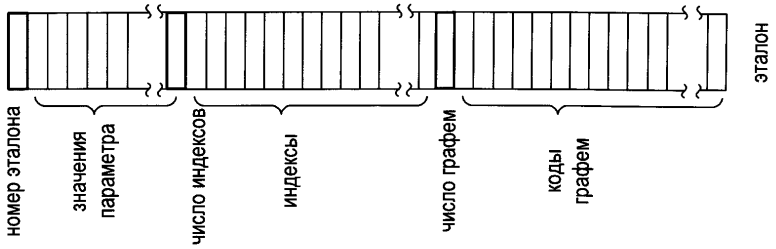

1922
p1
p2
p3
p4
p5
pn
1924
votes

Рис. 19Д


номер эталона
значения параметра
число индексов
индексы
число графем
коды графем
эталон

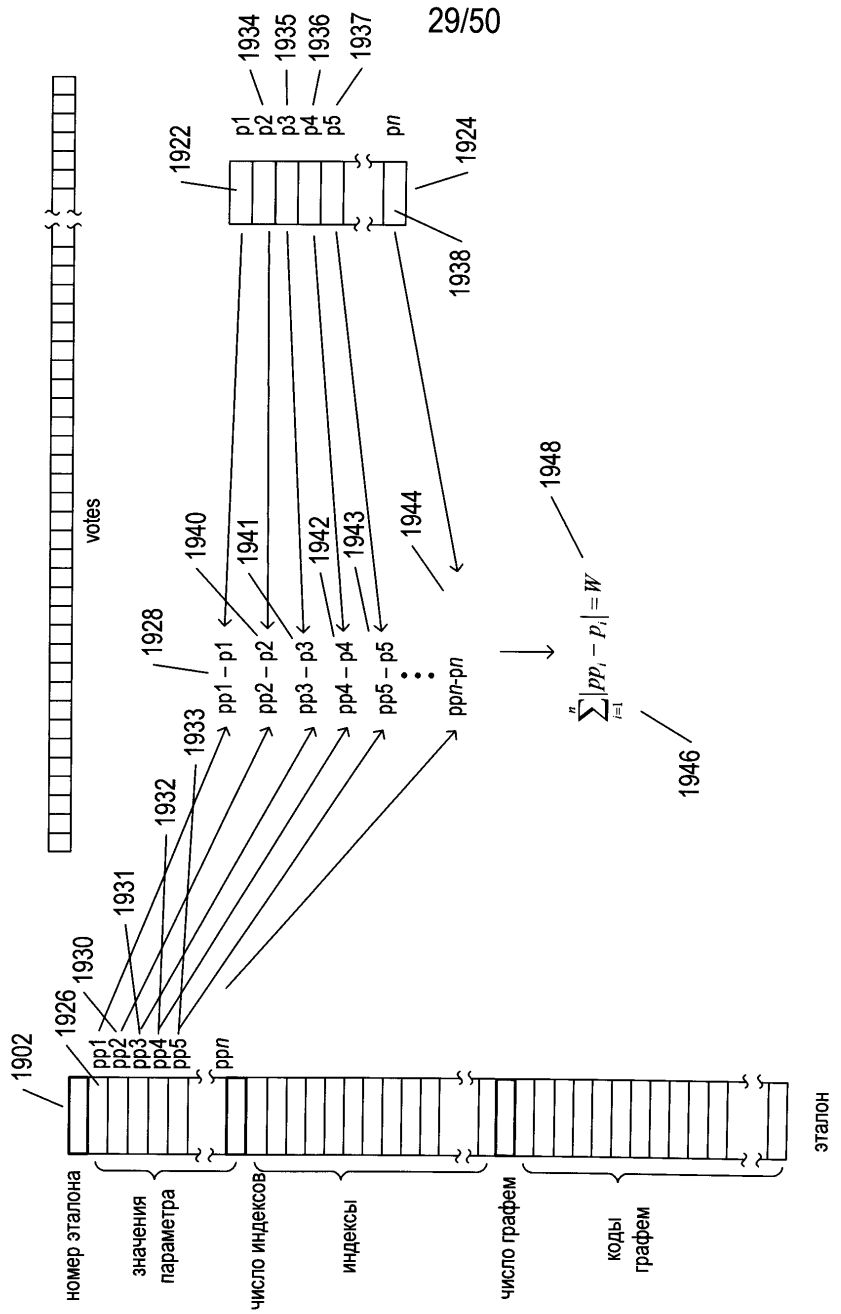

1902
1926
1930
1931
1932
1933
1928
1940
1941
1942
1943
1944
1922
1934
1935
1936
1937
1924
1938
1948
1946
votes
pp1
pp2
pp3
pp4
pp5
ppn
p1
p2
p3
p4
p5
pn
pp1 – p1
pp2 – p2
pp3 – p3
pp4 – p4
pp5 – p5
ppn-pn
$\sum_{i=1}^{n} |pp_i - p_i| = W$
номер эталона
значения параметра
число индексов
индексы
число графем
коды графем
эталон

Рис. 19E

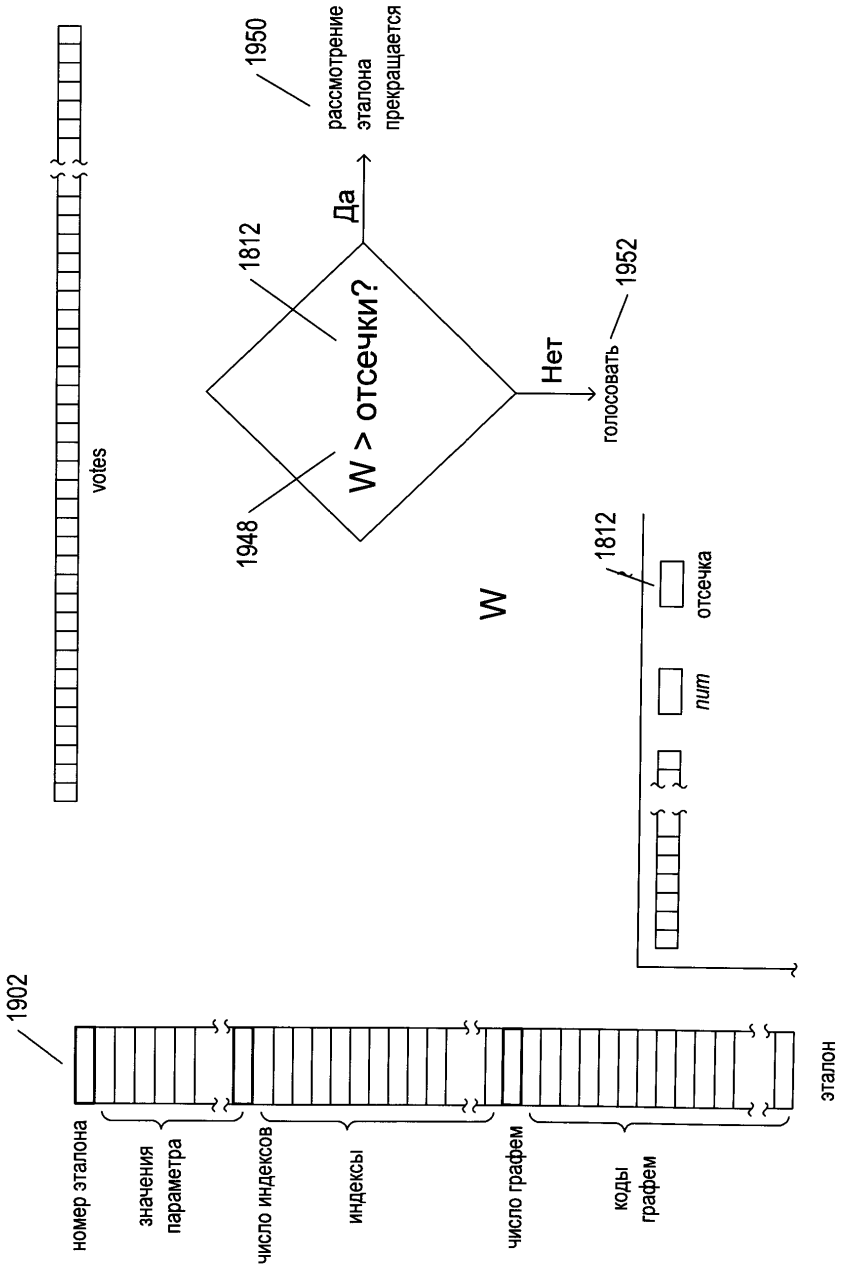

1902
номер эталона
значения параметра
число индексов
индексы
число графем
коды графем
эталон
W
1812
отсечка
num
1948
W > отсечки?
1812
Да
1950
рассмотрение эталона прекращается
Нет
1952
голосовать
votes

Рис. 19Ж

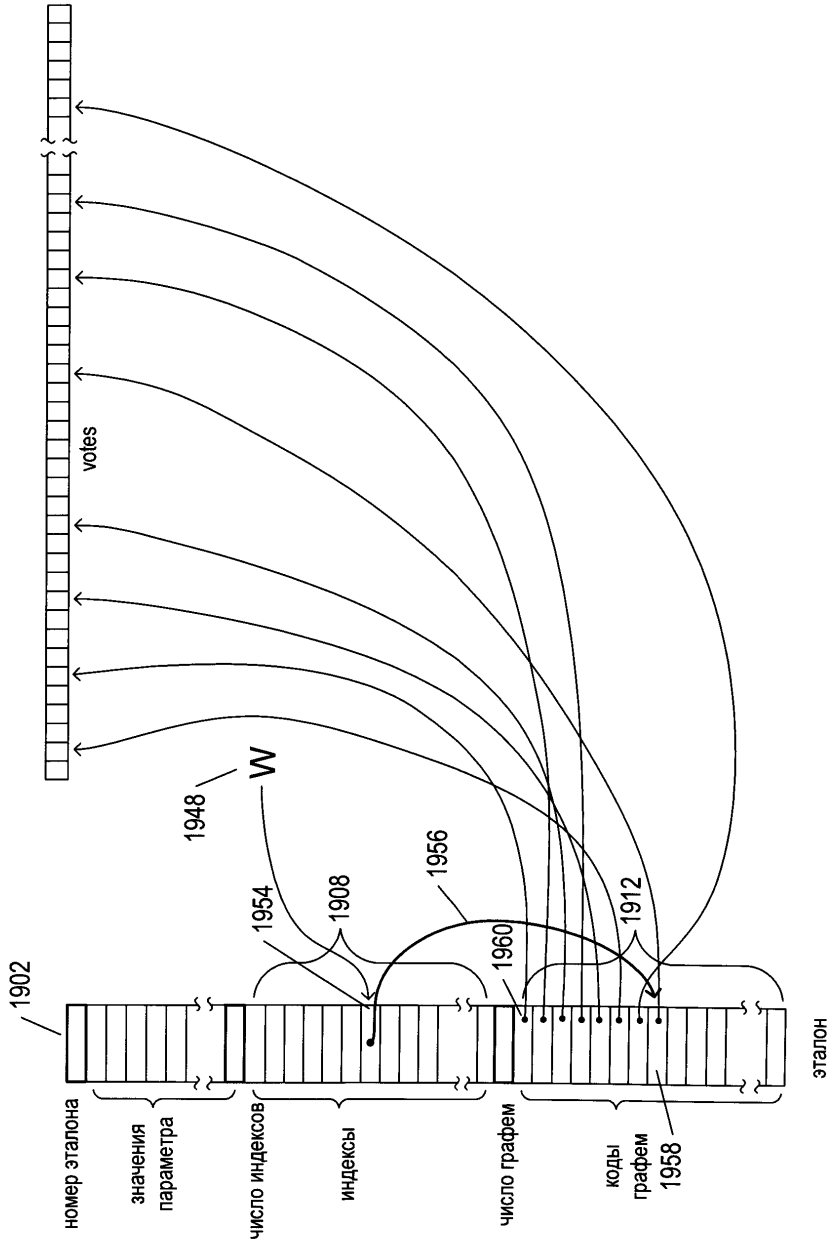

votes
W
1948
1954
1908
1956
1912
1960
1902
номер эталона
значения параметра
число индексов
индексы
число графем
коды графем
1958
эталон

Рис. 19 3

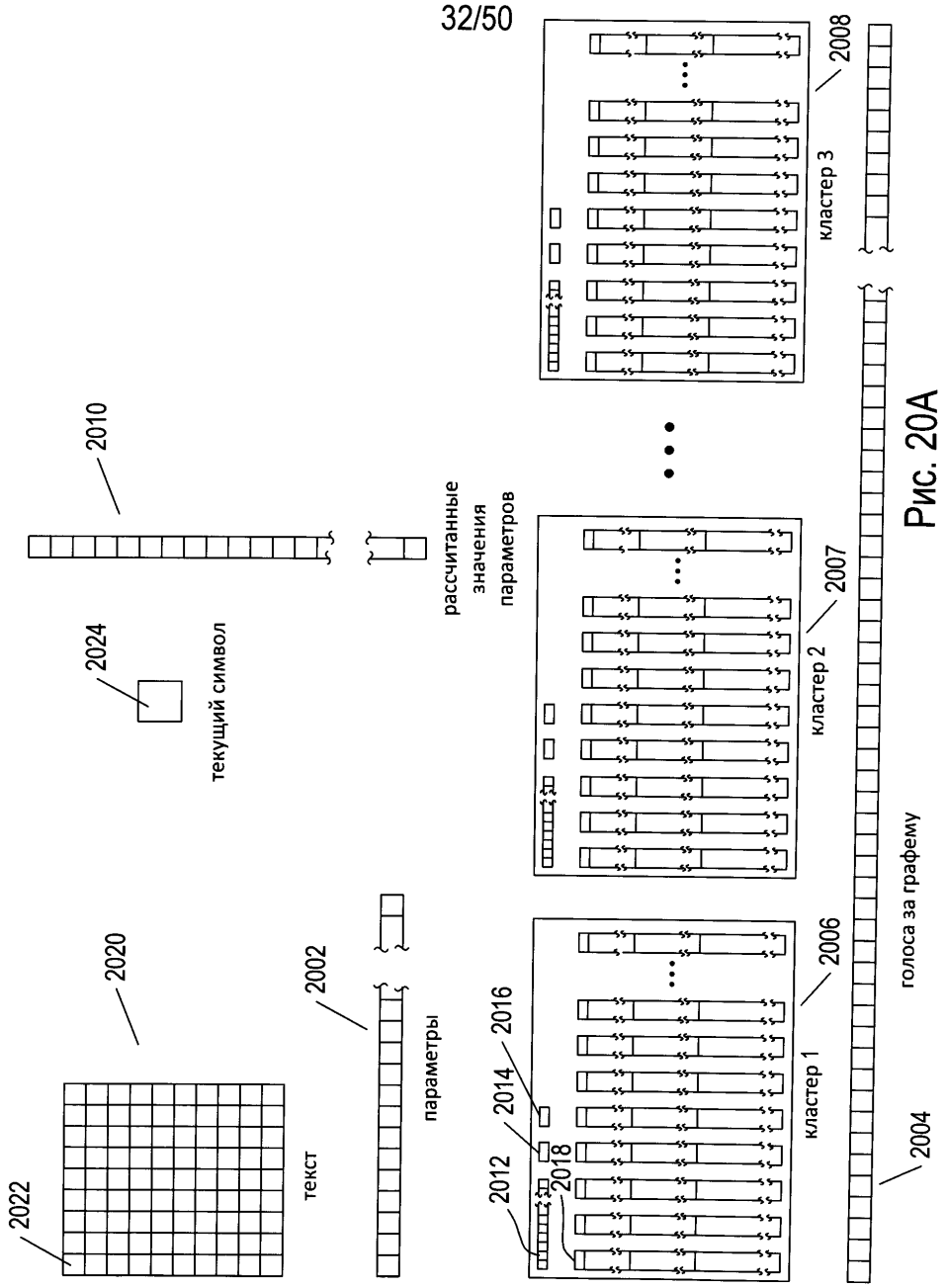

2010
2024
текущий символ
рассчитанные значения параметров
2008
кластер 3
2007
кластер 2
2020
2002
параметры
2016
2014
2018
2012
2006
кластер 1
2022
текст
2004
голоса за графему

Рис. 20A

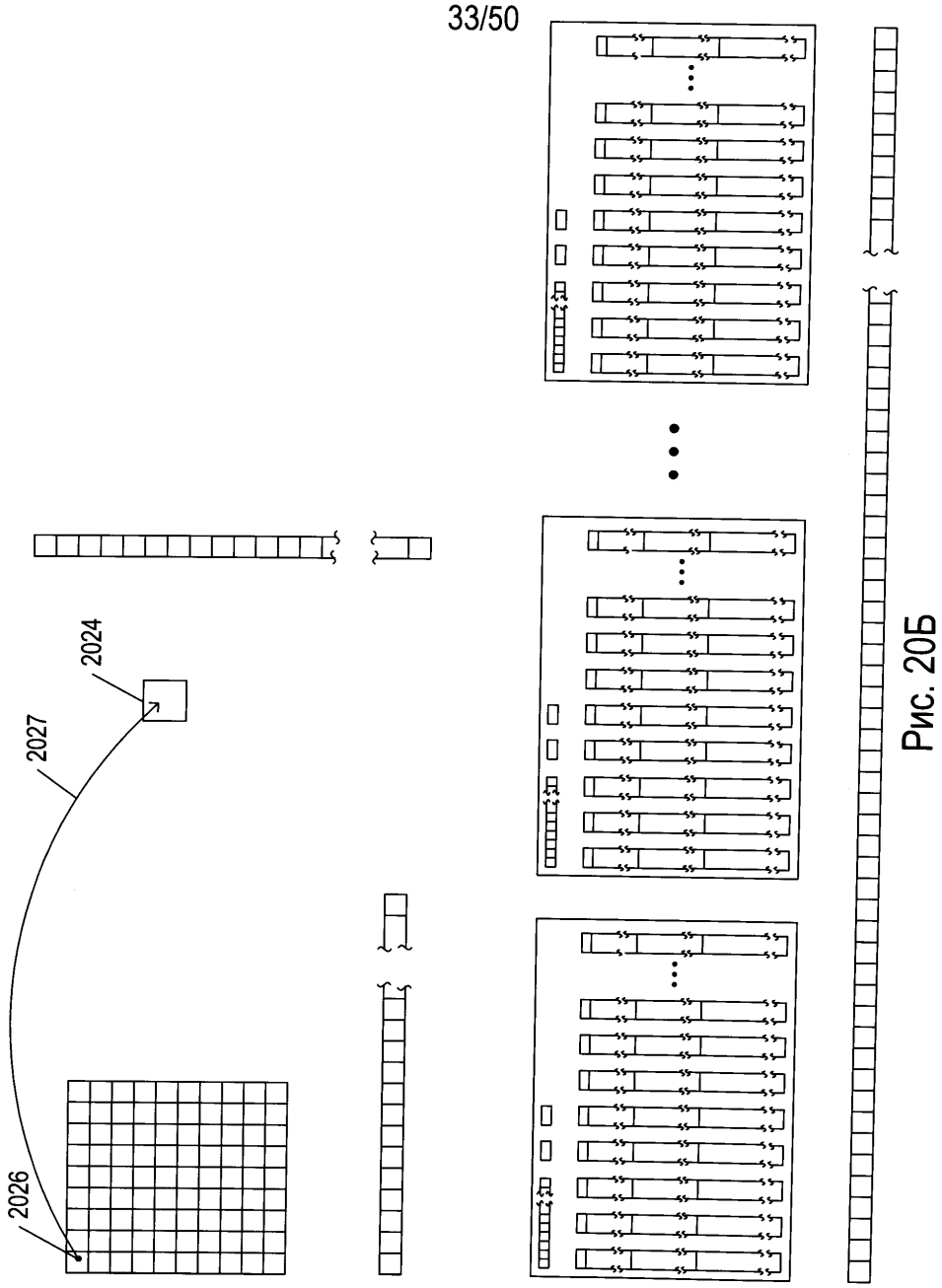


Рис. 20Б

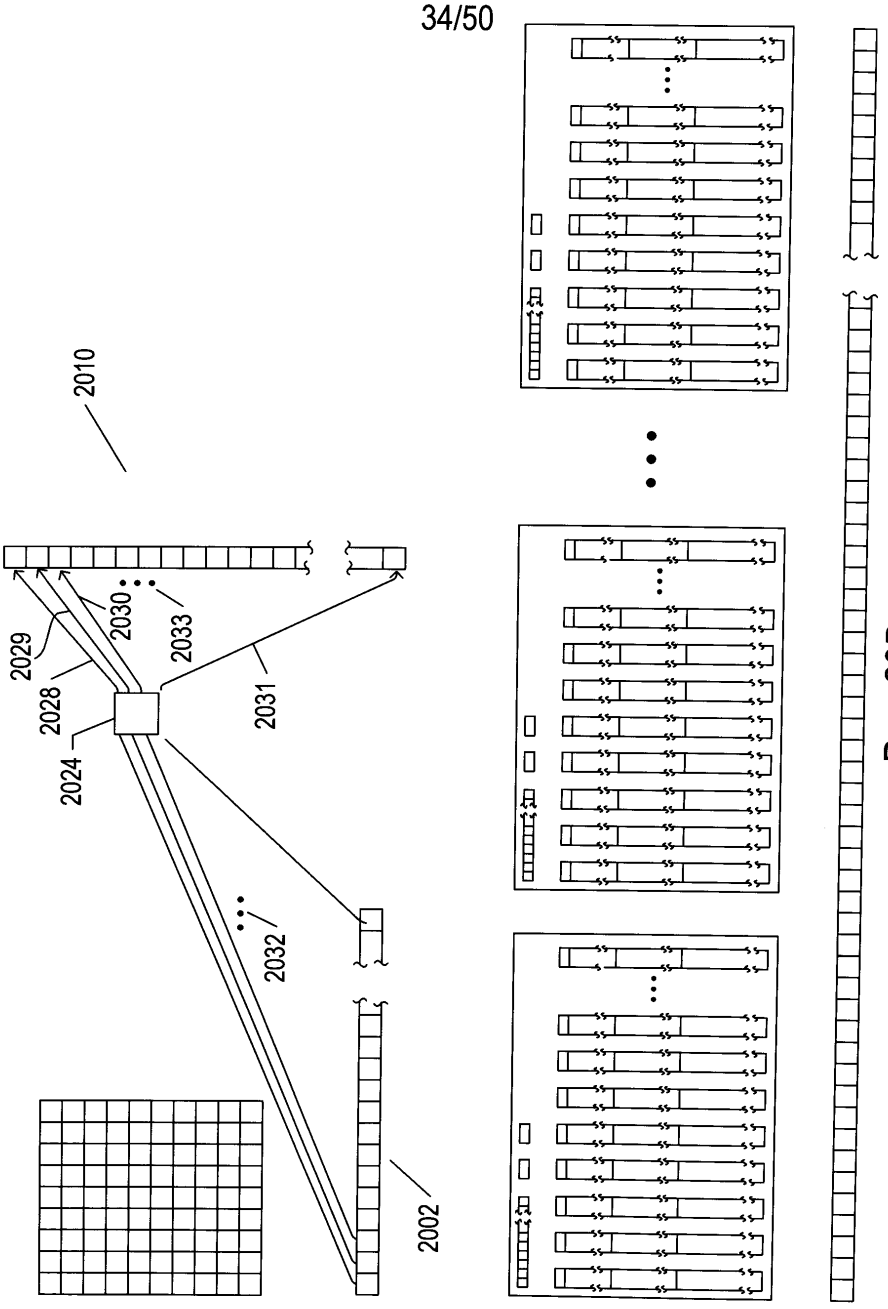


Рис. 20B

35

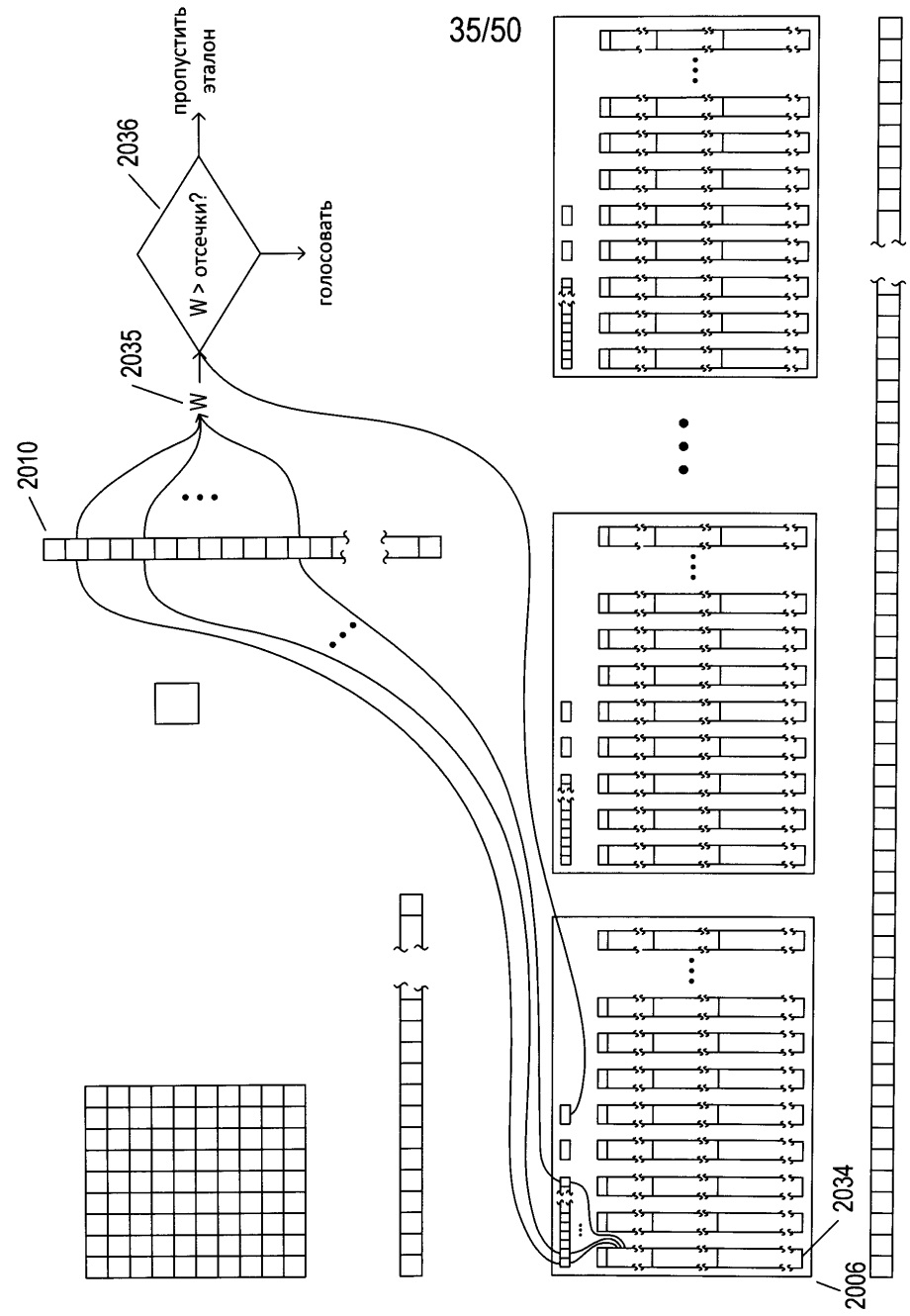

35/50
пропустить эталон
2036
W > отсечки?
голосовать
2035
W
2010
2034
2006

Рис. 20Г

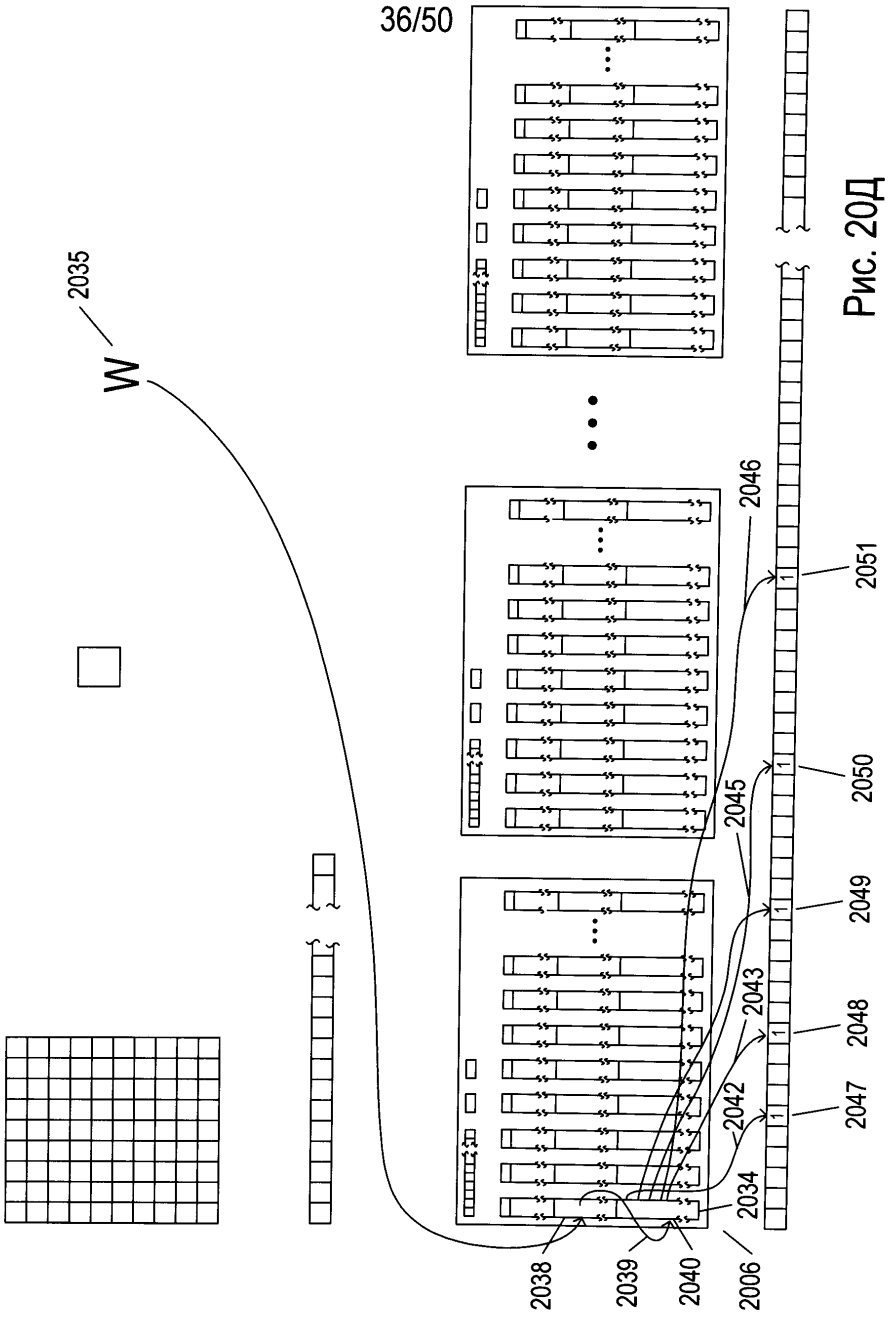

36/50
2035
W
2046
2051
2045
2050
2049
2043
2048
2042
2047
2034
2038
2039
2040
2006

Рис. 20Д

37

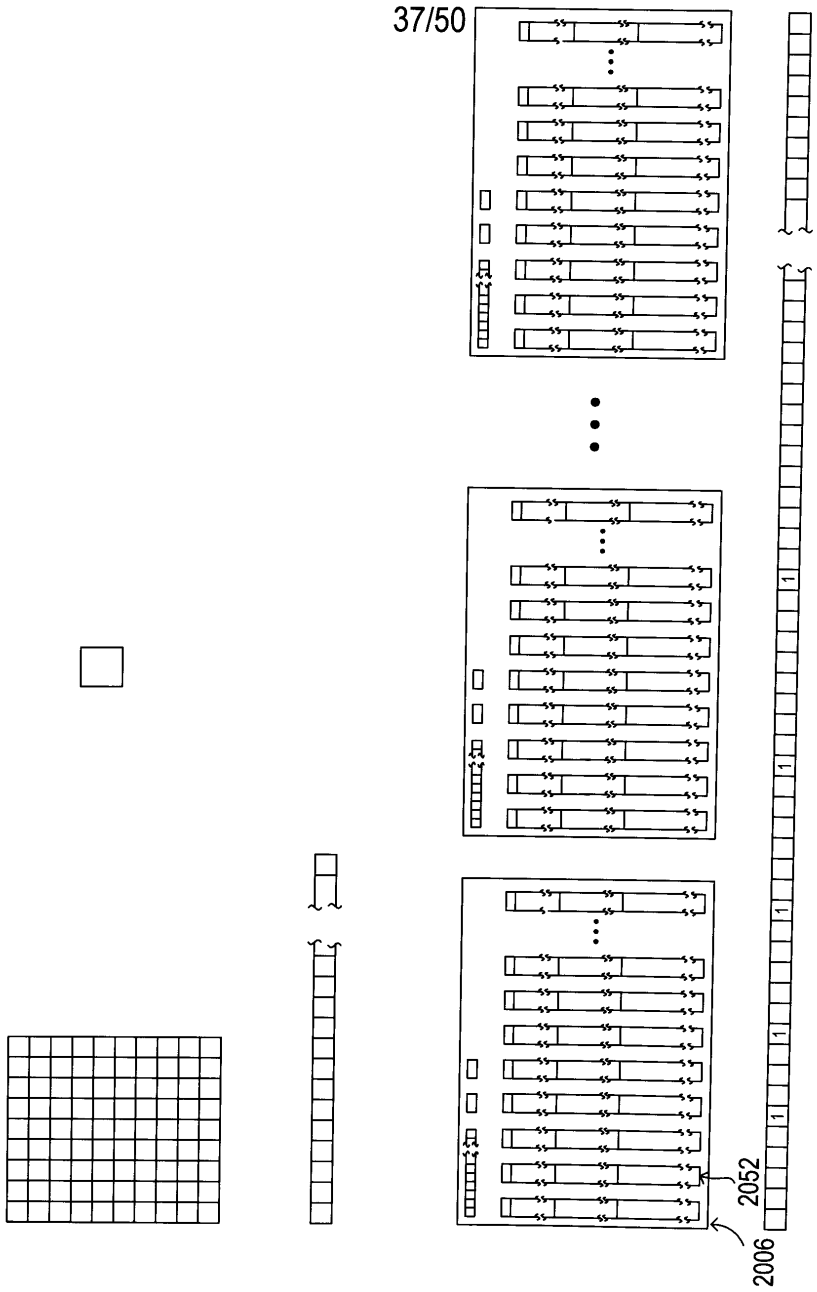


Рис. 20E

38

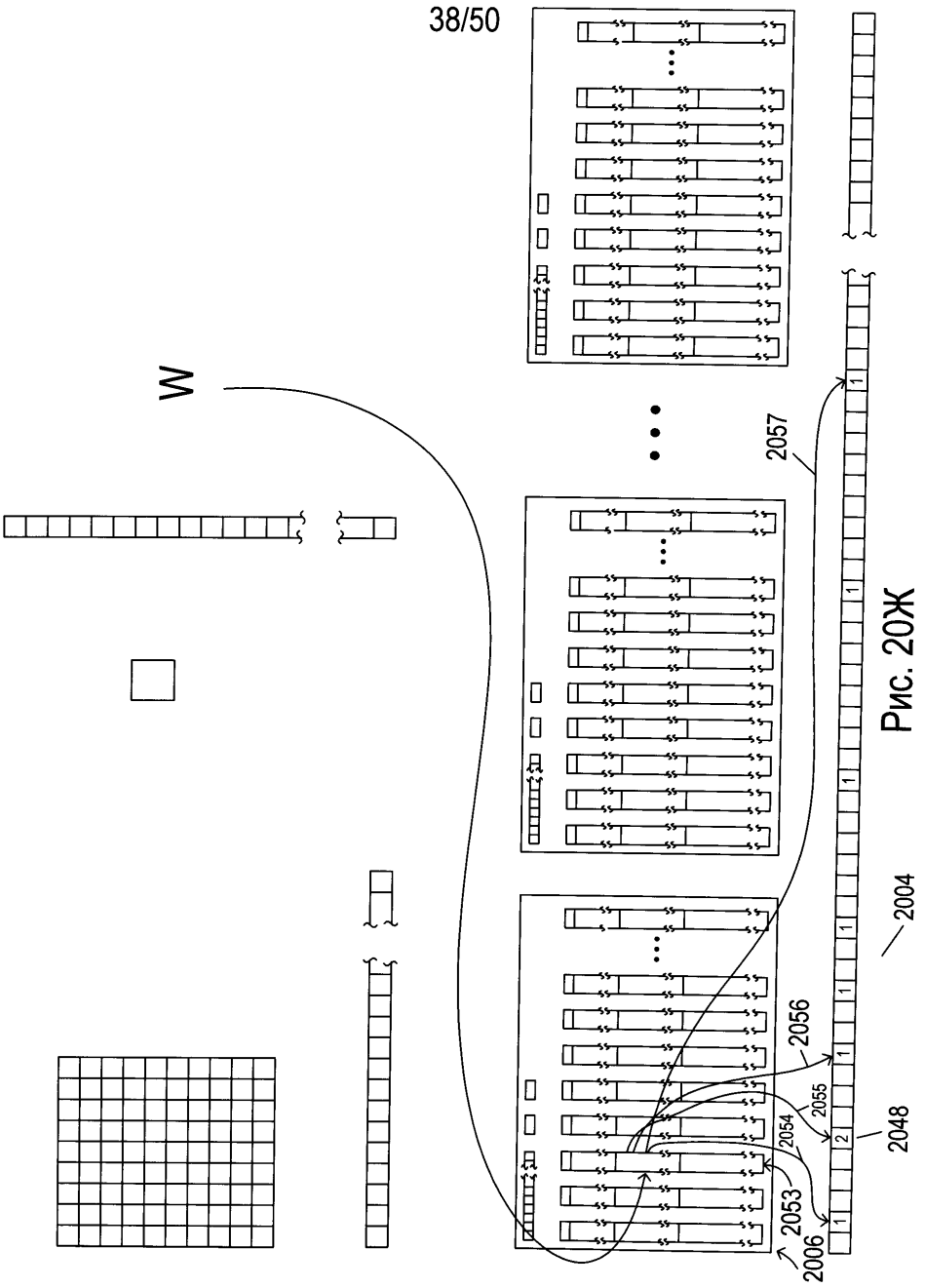


Рис. 20Ж

39

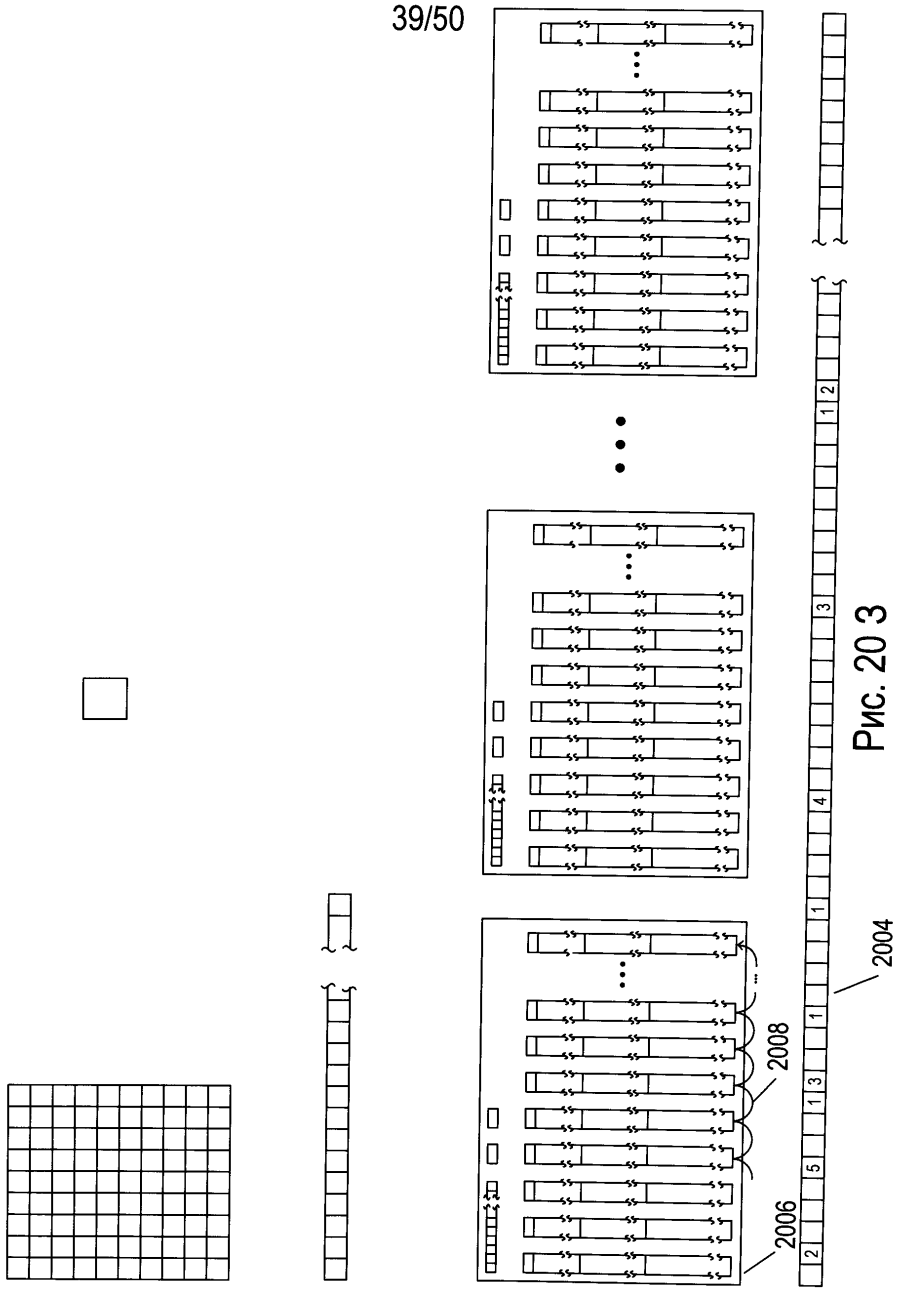


Рис. 20 3

40

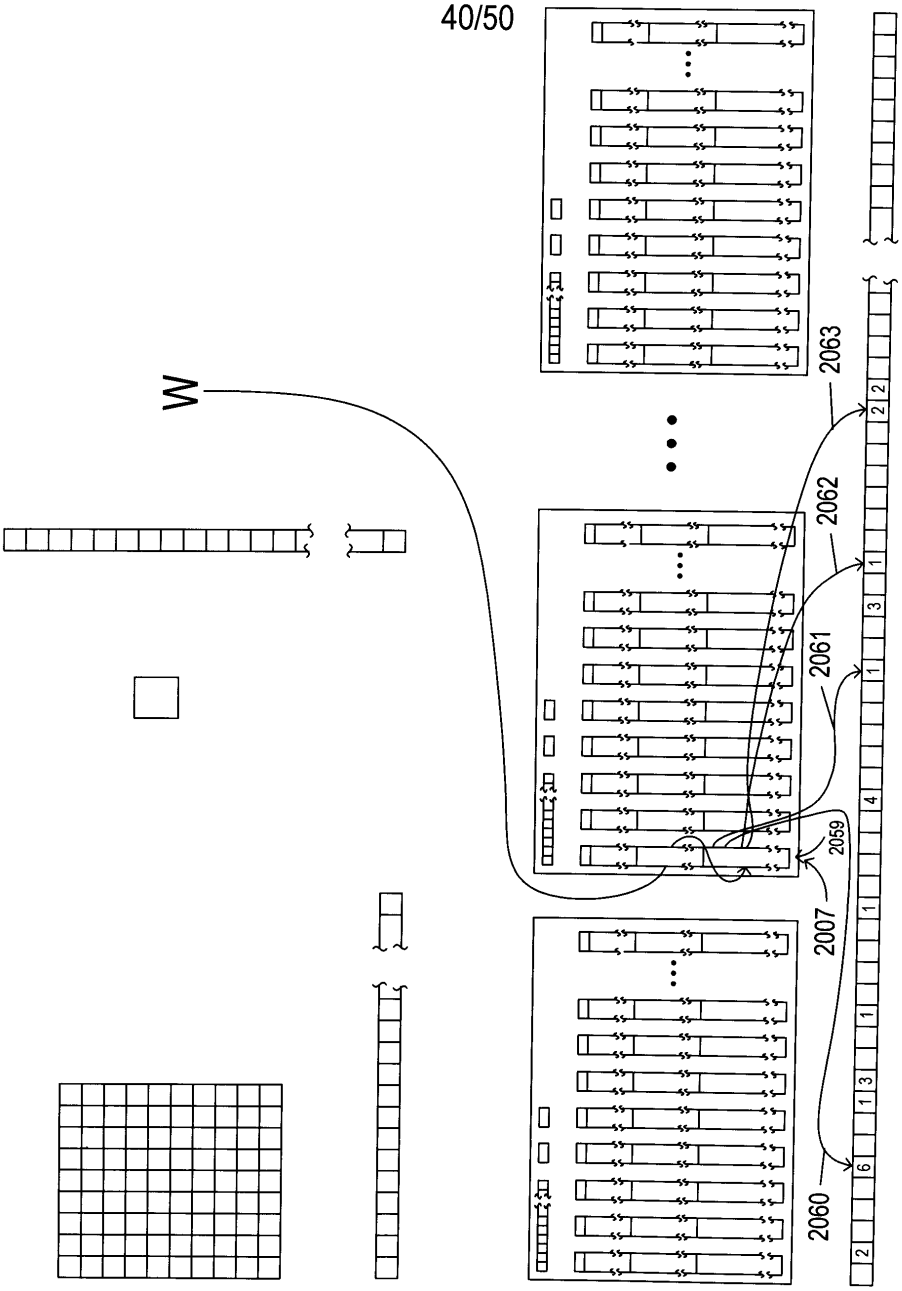


Рис. 20И

41

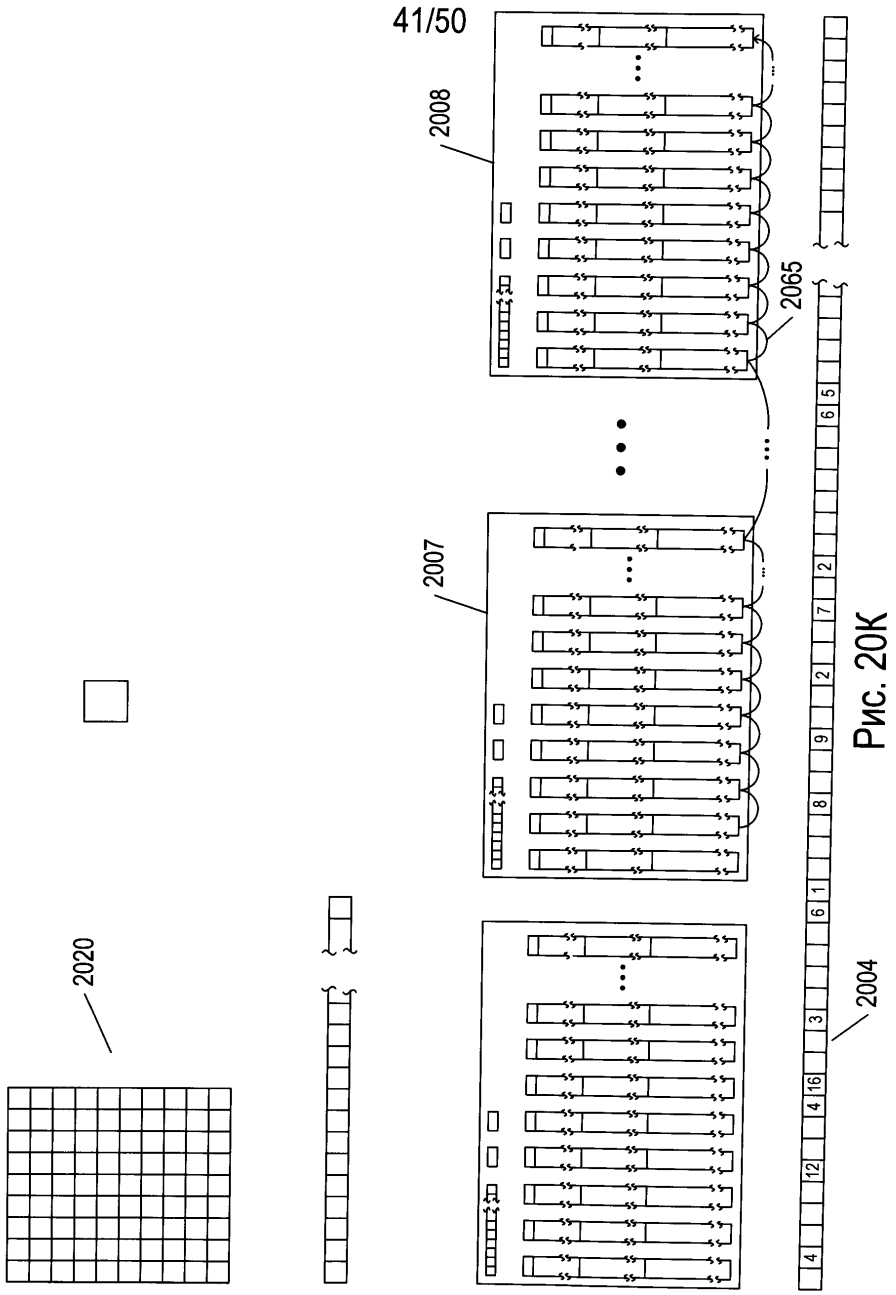


Рис. 20K

42

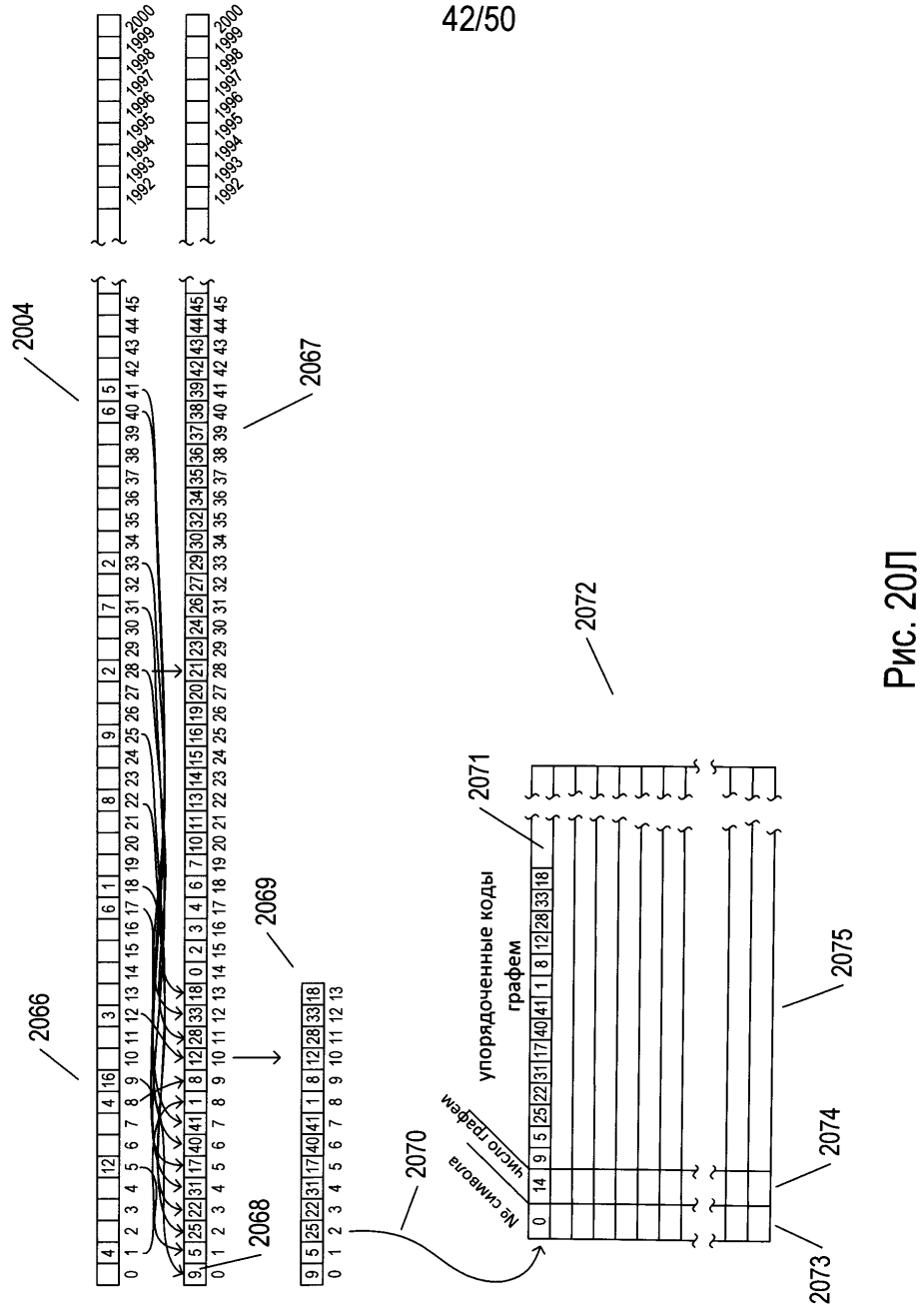

2004
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
упорядоченные коды графем
№ символа
число графем

Рис. 20Л

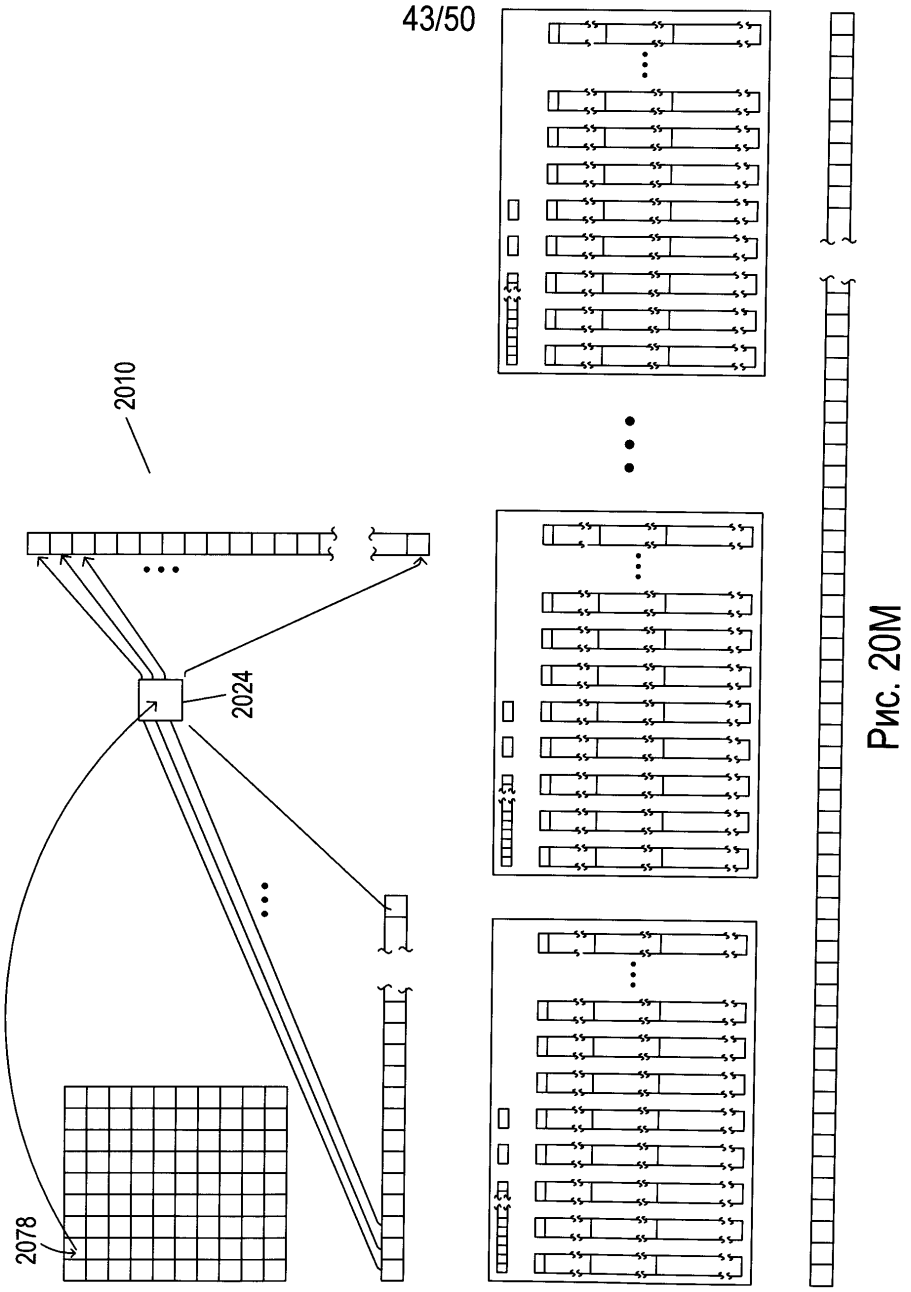


Рис. 20M

44

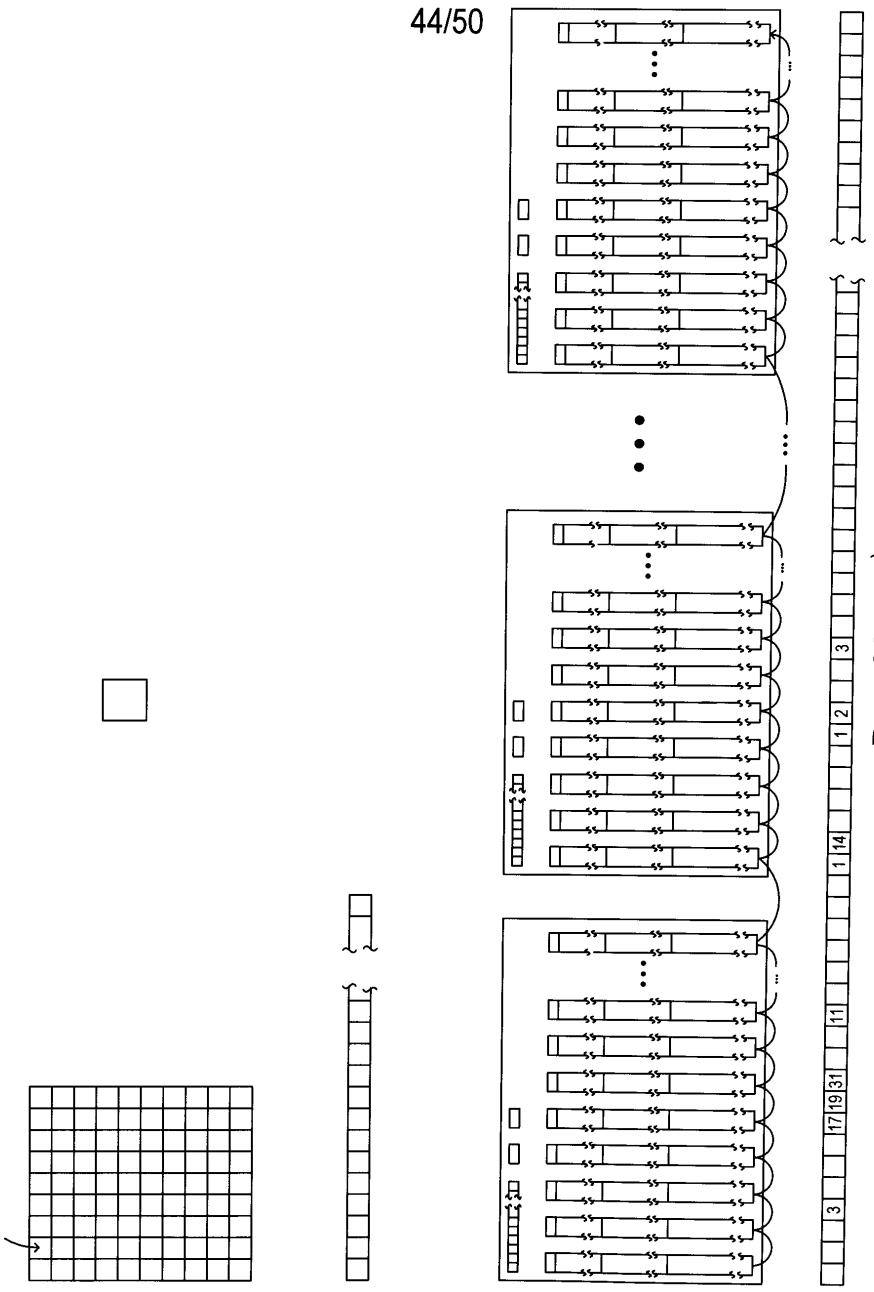


Рис. 20H

45

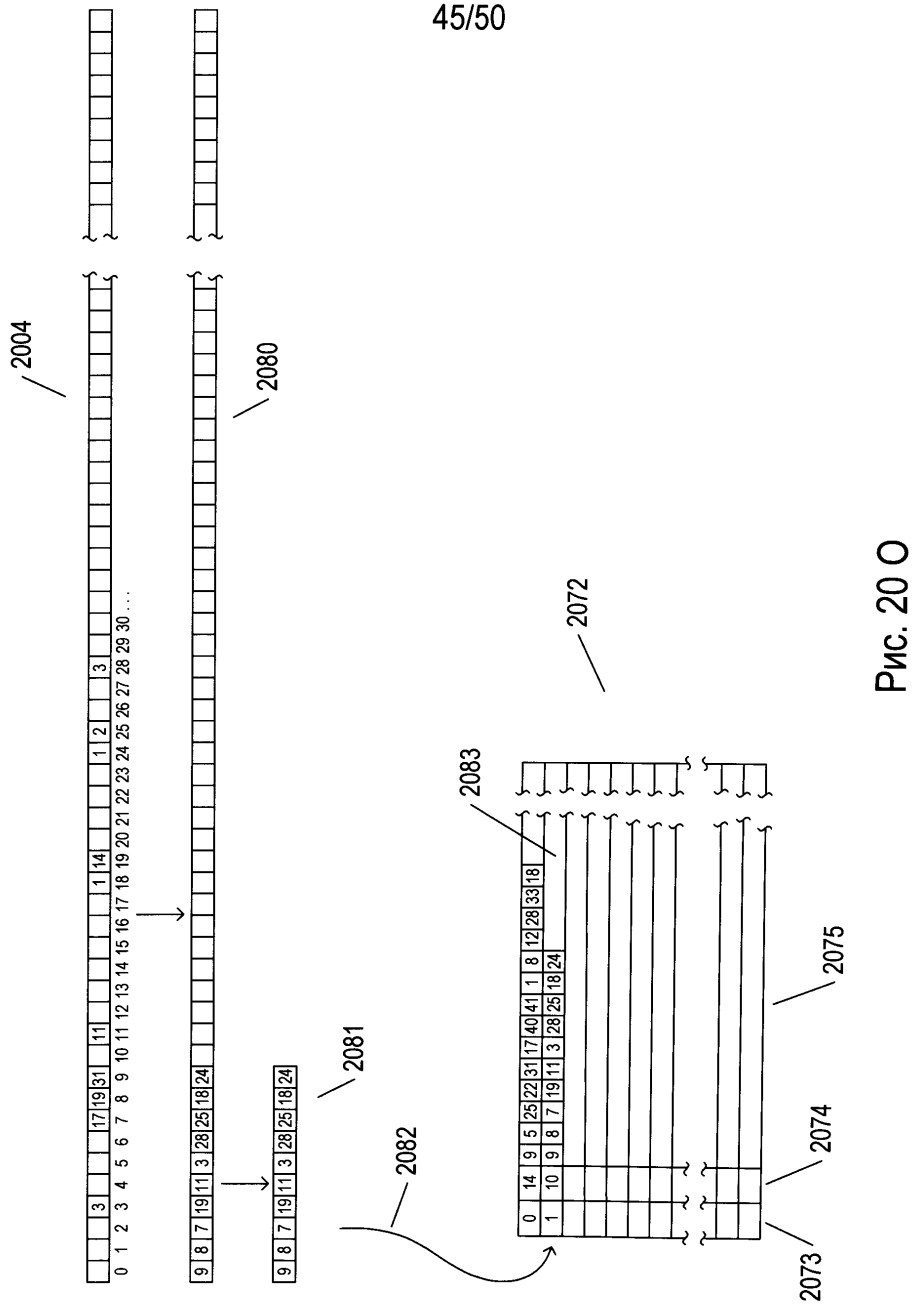


Рис. 20 O

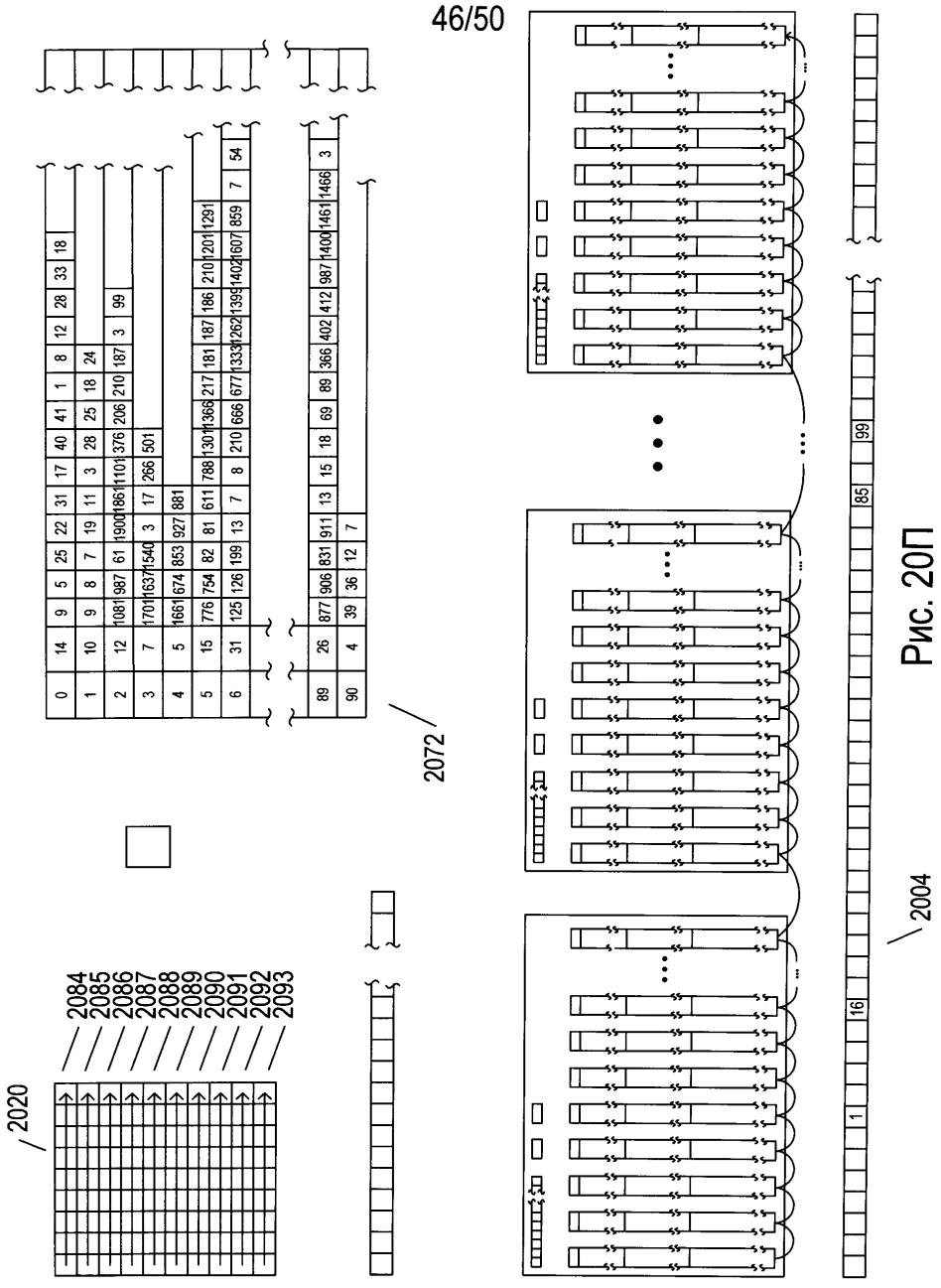

46/50
2072
2020
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2004
1
16
85
99

Рис. 20П

47

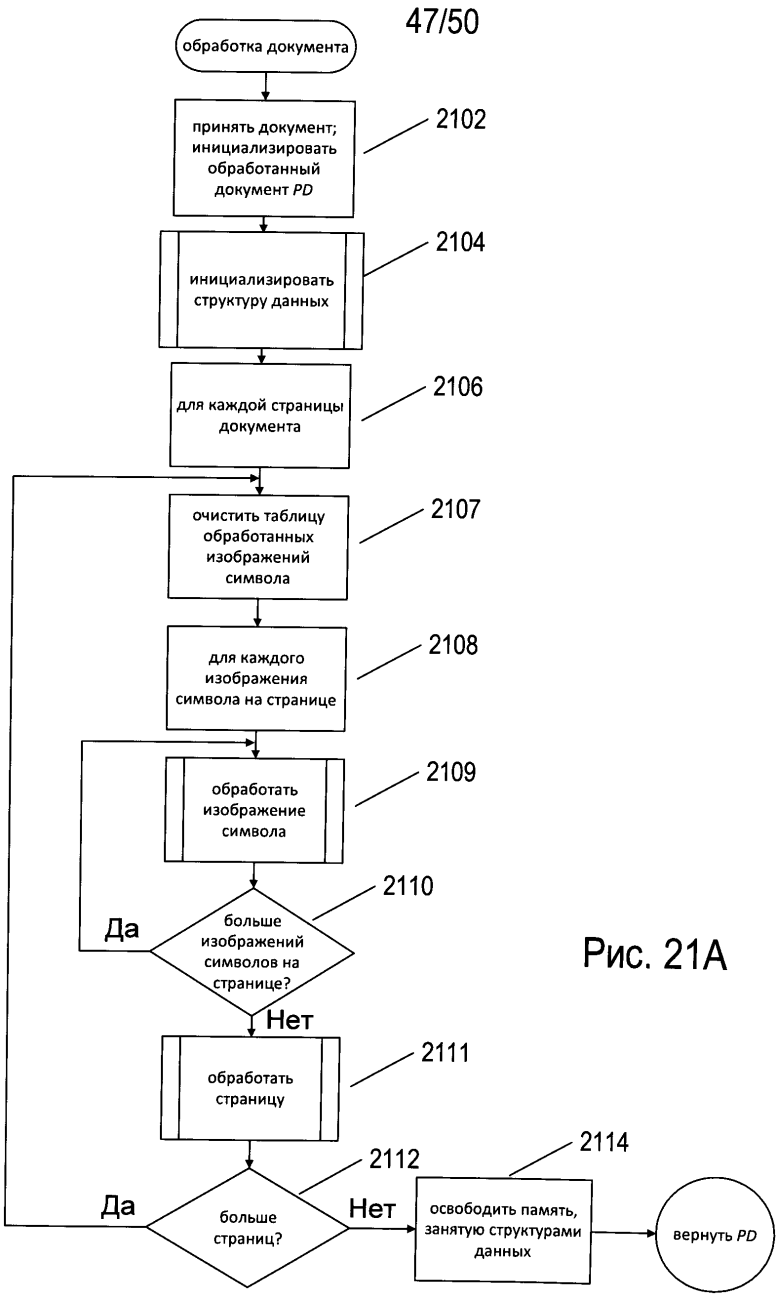

обработка документа
принять документ; инициализировать обработанный документ PD
2102
инициализировать структуру данных
2104
для каждой страницы документа
2106
очистить таблицу обработанных изображений символа
2107
для каждого изображения символа на странице
2108
обработать изображение символа
2109
больше изображений символов на странице?
2110
Да
Нет
обработать страницу
2111
больше страниц?
2112
Да
Нет
освободить память, занятую структурами данных
2114
вернуть PD

Рис. 21A

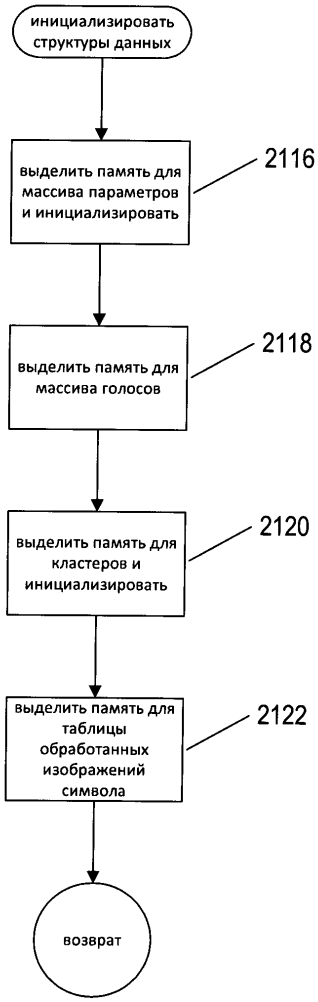

инициализировать структуры данных
выделить память для массива параметров и инициализировать
2116
выделить память для массива голосов
2118
выделить память для кластеров и инициализировать
2120
выделить память для таблицы обработанных изображений символа
2122
возврат

Рис. 21Б

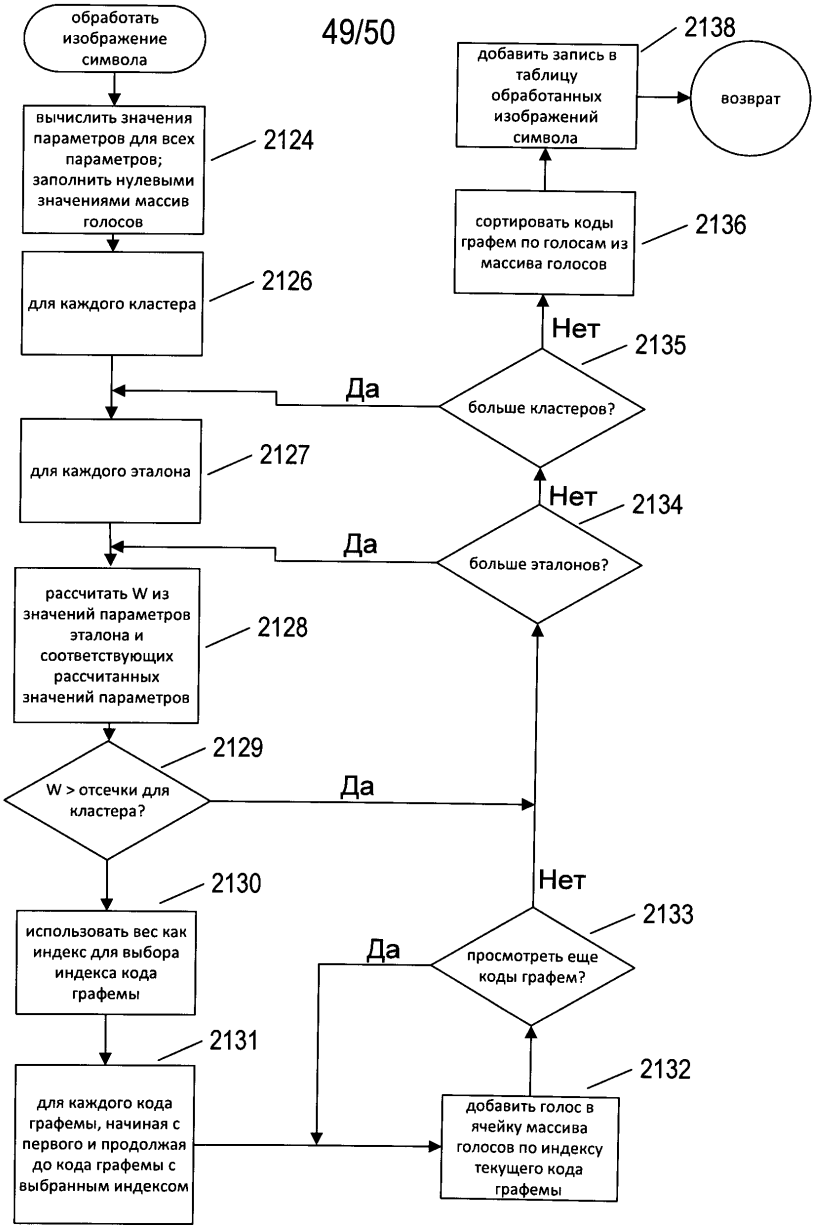

49/50
обработать изображение символа
вычислить значения параметров для всех параметров; заполнить нулевыми значениями массив голосов
2124
для каждого кластера
2126
для каждого эталона
2127
рассчитать W из значений параметров эталона и соответствующих рассчитанных значений параметров
2128
W > отсечки для кластера?
2129
Да
использовать вес как индекс для выбора индекса кода графемы
2130
для каждого кода графемы, начиная с первого и продолжая до кода графемы с выбранным индексом
2131
добавить голос в ячейку массива голосов по индексу текущего кода графемы
2132
просмотреть еще коды графем?
2133
Нет
Да
больше эталонов?
2134
Нет
Да
больше кластеров?
2135
Нет
сортировать коды графем по голосам из массива голосов
2136
добавить запись в таблицу обработанных изображений символа
2138
возврат

Рис. 21B

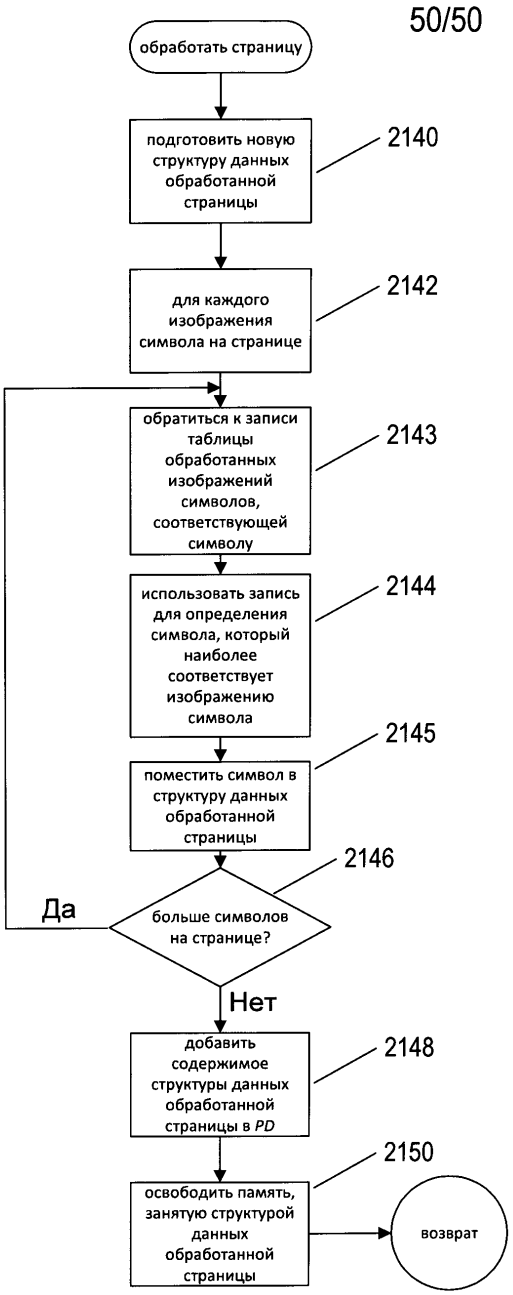

обработать страницу
подготовить новую структуру данных обработанной страницы
2140
для каждого изображения символа на странице
2142
обратиться к записи таблицы обработанных изображений символов, соответствующей символу
2143
использовать запись для определения символа, который наиболее соответствует изображению символа
2144
поместить символ в структуру данных обработанной страницы
2145
больше символов на странице?
2146
Да
Нет
добавить содержимое структуры данных обработанной страницы в PD
2148
освободить память, занятую структурой данных обработанной страницы
2150
возврат

Рис. 21Г