Точная локализация опорных решеток полей заполнения в анкетах методами динамического программирования и морфологической фильтрации

Андрей Куроптев
Институт системного анализа РАН
kavseverstahl@mail.ru

Дмитрий Николаев
Институт проблем передачи информации имени А. А. Харкевича РАН
dimonstr@iitp.ru

Василий Постников
Институт системного анализа РАН
vassili.postnikov@gmail.com

Аннотация

В данной работе рассматривается проблема распознавания полей заполнения в анкетах, снабженных опорной разграфкой. Ставится задача детектирования и удаления квазипериодических решеток. Предлагается и описывается алгоритм локализации решеток методами динамического программирования и морфологической фильтрации. Приводятся результаты тестирования реализации предложенного алгоритма на значительном объеме документов.

1. Введение

Задача точной локализации решеток знакомест рукопечатного ввода возникает в контексте применения технологий оптического распознавания (OCR, ICR) к документам, построенным по заранее известной форме. В таких случаях расположение полей ввода данных в документе известно заранее и описано в модели документа, которая включает описание статических элементов (линий разграфки, заголовки полей), состав элементов, их взаимное расположение, размеры и другие характеристики. Для бланков документов, подразумевающих заполнение от руки, поля ввода оформляются как специального вида решетка, в которой для каждой буквы отведена отдельная клетка (знакоместо).

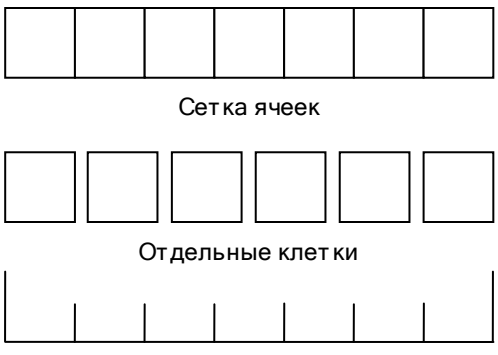
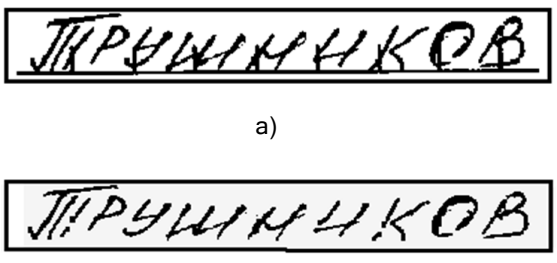


Рисунок 1. Виды решеток знакомест.

Решетки знакомест бывают различного типа. На практике наиболее часто используются три основных типа решеток: в виде сетки ячеек, отдельных клеток и в виде строки с засечками (Рисунок 1). Все они имеют характерную прямоугольную структуру (состоят из горизонтальных и вертикальных линий).

На Рисунок 2а приведен пример заполненного рукопечатного поля с решеткой типа «строка с засечками». С точки зрения распознающей системы, решетка знакомест по сути является структурированным шумом, наложенным на текстовые данные, порождающим большое количество ошибок распознавания. В частности, в отличие от некоррелированного шума типа «соль/перец», наложенные решетки делают полностью непригодными топологические методы распознавания символов. Таким образом, попытка удаления этих «искусственных» помех может привести к существенному росту качества распознавания.

Знания о точном расположении опорных элементов позволяют подавить их, почти не искажая при этом текста. Несмотря на то, что



а)

б)

Рисунок 2. Пример заполненного поля в присутствии засечек (а) и после снятия регулярных помех (б).

Рисунок 3. Примеры деформаций решеток знакомест.

теоретически эта информация известна (заложена в модель документа), на практике она часто бывает неполной или неверной. В процессе печати, заполнения и последующей оцифровки бланков привносятся различного рода деформации (сдвиг, наклон, незначительные растяжения, связанные с протяжкой в принтере и/или сканере) и шумовые сигналы (пятна, тени от сгиба бумаги и другие). Поэтому эталонное описание решеток часто не соответствует их реализации: теряется свойство прямоугольной формы, нарушается периодичность между засечками и прочее (Рисунок 3).

Простейшим способом выделения горизонтальных линий является определение пиков на проекции изображения поля на вертикальную ось. Вертикальные же линии могут быть найдены с помощью частотного анализа на аналогичной горизонтальной проекции. Однако на реальных бланках часто в непосредственной близости к текстовым полям располагаются различного рода линии разграфки, которые имеют более толстое и контрастное начертание (Рисунок 4). Поэтому при поиске пиков на вертикальной гистограмме алгоритм может неправильно находить горизонтальные линии.

Дополнительные проблемы к поиску составляющих решетки знакомест линии привносятся на этапе заполнения из-за «линиеподобных» символов, например «1», «Т», «П» и прочие.

Подводя итог вышесказанному, перечислим

Рисунок 4.

проблемы, которые возникают в процессе локализации опорных решеток:

- Искажение прямолинейности линий.
- Нарушение периодичности в решетках знакомест.
- Наличие посторонних статических элементов в непосредственной близости к полям ввода.
- Наличие элементов заполнения, морфологически похожих на отрезки прямых.

Первые две проблемы могут быть устранены путем применения алгоритма динамической трансформации временной оси (DTW, Dynamic time warping) [1], который является классическим примером использования метода динамического программирования. Исключить из внимания линии разграфки можно с помощью дополнительных знаний о структуре решеток знакомест: наличии стыков горизонтальных и вертикальных соединительных линий. Такие структурные особенности легко детектируются с помощью морфологических фильтров типа «hit-or-miss» [2].

В данной работе изложен алгоритм локализации опорных решеток, построенный на применении этих двух инструментов.

2. Алгоритм локализации решетки знакомест

В данном разделе вначале опишем краткую схему предлагаемого алгоритма, после чего рассмотрим подробно каждый ключевой пункт.

Методы предварительной идентификации документов позволяют определить приблизительные области полей заполнения, а также угол наклона документа в целом. К сожалению, из-за деформаций изображения это не гарантирует точной ориентации отдельных полей. Кроме того, на анкетах встречаются так называемые «надпечатки» – наносимые с помощью принтера вариативные части анкеты. По понятным причинам поля надпечатки могут иметь наклон, отличный от глобального наклона документа. Предлагаемый алгоритм локализации ожидает «на входе» прямо ориентированное изображение, поэтому первым шагом следует довернуть изображение поля, определив его угол наклона. Для этого используется алгоритм, опирающийся на быстрое преобразование Хафа, подробно изложенный в [3]. Далее, на основании знаний о числе и типе стыков с вертикальными составляющими решетки, определяется положение горизонтальных опорных линий. Следующим этапом находится расположение вертикальных опорных элементов. В заключение, с помощью метода динамического программирования

корректируются расположение горизонтальных линий для каждой ячейки.

Отметим, что задачи определения ориентации изображения и снятия точно найденной решетки являются отдельными задачами и в рамках данной работы не рассматриваются.

Теперь рассмотрим более подробно алгоритмы поиска и корректировки линий и стыков.

2.1 Поиск горизонтальных линий с известным шаблоном стыков

Итак, для того, чтобы выделять опорные горизонтальные линии решеток на фоне посторонних контрастных линий, предлагается опереться на информацию о сопряжении опорной линии с вертикальными элементами решетки.

Будем описывать искомый объект простейшим регулярным выражением – строкой, содержащей смысловые символы и символ возможного повтора (‘*’). Назовем это описание шаблоном линии.

В качестве смысловых символов будем использовать символы типа стыка (например, ‘└’, ‘⊥’, ‘┘’), а также символы наличия линии без стыков (‘—’) и отсутствия линии (‘○’). Например, нижние опорные линии всех полей, изображенных на Рис. 4, описываются паттерном « └—* ⊥—* ┘», а полей с решеткой типа «отдельные клетки» – « └—* ┘ ○ * └—* ┘ ○ * └—* ┘ …».

Рассмотрим теперь метод оценивания пикселя изображения как центрального пикселя стыка линий. Фактически, нам необходимо оценить, насколько темным является центральный пиксель, а также пиксели, располагающиеся в тех направлениях, в которых мы ожидаем наличия линий. Кроме того, мы должны проверить, что в остальных направлениях пиксели достаточно светлые (иначе на крестообразный стык будут реагировать все детекторы), а, кроме того, светлыми должны быть диагонали (иначе детектор будет реагировать на углы черных прямоугольников). Будем рассматривать паттерн из 9 пикселей в узлах регулярной решетки с шагом L, соразмерным ожидаемой толщине линии. На рис. 5 изображена ожидаемая конфигурация для стыка «⊥». Пиксель, для которого производятся вычисления, соответствует центру приведенной схемы.

Построим теперь морфологический фильтр, соответствующий рассматриваемой конфигурации. По пикселям, помеченным на схеме белым, будем искать минимум, а по черным – максимум. Затем найдем разность полученных значений, и обнулим, если она меньше 0. Полученное значение W будем считать оценкой качества искомого стыка в данном пикселе:

$$W_{white} = \min\left(p(-l,-l), p(-l,0), p(-l,l), p(0,-l), p(0,l), p(l,-l), p(l,0), p(l,l)\right)$$

$$W_{black} = \max\left(p(0,0), p(-l,-l), p(-l,0), p(-l,l), p(0,-l), p(0,l), p(l,-l), p(l,0), p(l,l)\right)$$

$$W = \max\left(0, W_{white} - W_{black}\right)$$

Аналогичным образом строятся детекторы всех остальных типов стыков, а также детектор прямой без стыков. Для детектора отсутствия линии способ построения немного отличается:

$$W_{bg} = \max\begin{pmatrix} p(-l,-l), p(0,-l), p(l,-l), \\ p(-l,0), p(0,0), p(l,0), \\ p(-l,l), p(0,l), p(l,l) \end{pmatrix},$$

$$W = \min\left(255 + p(0,0) - W_{bg}, 255\right)$$

где W_{bg} несет смысл оценки яркости фона в присутствии возможных загрязнений в любом из пикселей паттерна.

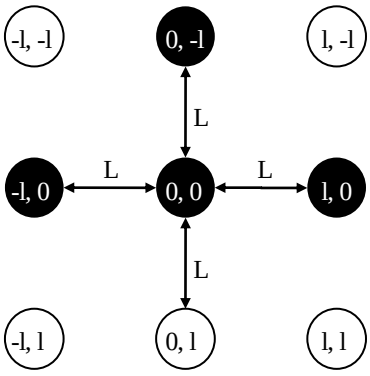


Рисунок 5. Паттерн фильтра.

Итак, применяя соответствующие морфологические фильтры к изображению поля, мы получаем в каждом пикселе набор оценок, говорящих, насколько данный пиксель похож на центральный пиксель стыка, линии или пробела (Рис. 5).

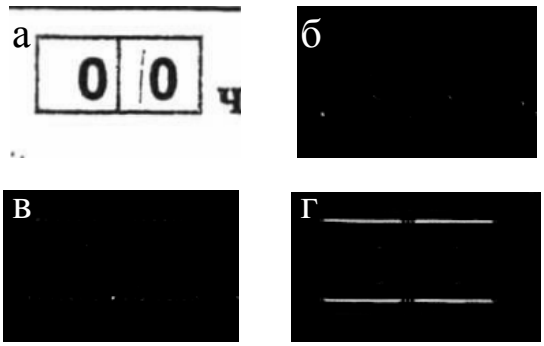


Рисунок 6. Результат детекции стыков ‘└’, ‘⊥’ и ‘—’ (б, в, г) на изображении поля (а).

Теперь для каждой строки изображения поля определим, насколько она соответствует искомому паттерну, и среди всех строк выберем наилучшую. Вполне очевидно, что задача оценки одной строки – это задача динамического программирования [4]. Действительно, создадим таблицу, каждый столбец которого соответствует смысловому символу шаблона либо символу, сопровождаемому звездочкой, а каждая строка – пикселю строки изображения. Каждый элемент таблицы заполним оценкой пикселя, задаваемого строкой, на соответствие стыку, задаваемому столбцом (Табл. 1). Нашей задачей будет найти путь с наибольшим весом в таблице от левого до правого края, при условии, что в столбцах без звездочки мы можем двигаться только на одну ячейку вправо вниз по диагонали, а в столбцах со звездочкой – так же или на одну ячейку вниз. При этом весом пути будет считаться сумма значений ячеек, по которым проходит путь. (В Табл. 1 оптимальный путь выделен голубым цветом.)

Решив эту задачу последовательно для каждой строки изображения, найдем координаты нижней и (если есть у данного вида решетки) верхней опорной линии решетки.

	└	—*	┴	—*	┴	—*	┘
1	0	10	2	10	2	10	10
2	201	27	30	27	30	27	12
3	20	204	2	204	2	204	0
4	3	203	5	203	5	203	0
5	0	180	10	180	10	180	2
6	1	10	205	10	205	10	20
7	2	107	3	107	3	107	0
8	10	240	0	240	0	240	0
9	2	202	2	202	2	202	10
10	0	50	2	50	2	50	0
11	0	10	220	10	220	10	20
12	0	191	20	191	20	191	2
13	3	203	2	203	2	203	2
14	2	205	0	205	0	205	0
15	1	2	0	2	0	2	250
16	1	7	0	7	0	7	0

Таблица 1. Пример поиска оценивания соответствия шаблону методом динамического программирования.

2.2 Поиск вертикальных линий

Вертикальные линии решетки можно было бы искать с помощью частотного анализа горизонтальной проекции. Но из-за неидеальной системы подачи бумаги принтеров и протяжных сканеров в решетках знакомест часто нарушается периодичность между засечками (Рис. 6). Кроме того, при экстренном увеличении числа ячеек поля на этапе разработки бланка (при внезапном знакомстве с фамилией «Гедиминайте-Бержанскайте-Клаусутайте») случается, что добавленные ячейки имеют нестандартный размер. Проблема усугубляется тем, что даже в случае идеальной геометрии решетки из-за символов, имеющих выраженные вертикальные штрихи, на проекции отображается много ложных пиков.

Последнюю проблему, впрочем, можно частично ослабить: так как на данном этапе детектирования решетки нам уже известно расположение горизонтальных линий, можно рассматривать не проекцию всего изображения на горизонтальную ось, а проекцию лишь части, которая находится выше нижней линии решетки, но ниже верхней линии или высоты засечек в случае решеток соответствующего типа. Тем самым обеспечивается, что пики, образованные высокими буквами, не будут превышать пики, образованные элементами решетки. Кроме того, разумно добавить в гистограмму вес «┴» и «┬» стыков, чтобы усилить истинные пики.

Зная предположительный период и возможную погрешность в расстоянии между вертикальными опорными элементами, можно быстро посчитать всевозможные варианты расположения таких элементов на заданном нам участке, и выбрать наилучший вариант. Эта задача хорошо решается методом динамического программирования, причем в данном случае для решения не потребуется таблица.

Будем двигаться вдоль гистограммы слева направо. На каждом шагу рассмотрим отрезок гистограммы $[x-p-\Delta, x-p+\Delta]$, где x – наше текущее положение, p – ожидаемый период, а Δ – максимальное отклонение. Найдём максимум среди значений гистограммы по указанному отрезку и прибавим к текущей ячейке, а в параллельном массиве сохраним индекс максимума. Тогда глобальный максимум полученной гистограммы будет соответствовать положению последней вертикальной линии, а соответствующий индекс будет указывать на предыдущий элемент списка найденных линий. Пройдя по списку, восстановим положения всех вертикальных опорных элементов.

2.3 Корректировка расположения горизонтальных линий

После нахождения вертикальных и горизонтальных линий решетка знакомест

считается сформированной. Однако, для алгоритма снятия решётки требуется пиксельная точность, а при протягивании бумаги на этапе печати и сканирования изображение может подвергнуться веерной дисторсии (если лист будет поворачиваться по мере прохождения по тракту). Кроме того, алгоритм определения угла наклона изображения поля также может внести небольшие ошибки. Поставим задачу следующим образом: требуется найти последовательность смещений по вертикали элементов горизонтальной линии от ячейки к ячейке (Рис. 3), при условии, что каждое отдельное смещение не может быть больше, чем k пикселей (как правило, $k=1$).

Эта задача также решается методом динамического программирования.

Соберем вертикальную гистограмму размера, равного высоте поля (что превышает размер решетки), для каждой ячейки решетки и заполним этими гистограммами таблицу шириной, равной числу ячеек. Будем максимизировать путь от левой до правой границы таблицы, разрешая при смещении вправо сдвигаться вверх или вниз не более чем на k элементов. Найденный максимальный путь будет проходить по положениям элементов опорной горизонтальной линии.

3. Результаты работы

Разработанный подход к решению задачи сопоставления был программно реализован и в настоящий момент промышленно используется для потокового ввода анкет ОСАГО, заявлений на выдачу кредитов, а также анкет сетевого маркетинга.

Качество работы алгоритма оценивалось на наборе из 395 полисов ОСАГО, содержащих 12245 полей (Табл. 2). Для этого набора вручную были введены правильные значения всех полей, затем система распознавания была запущена в трех режимах: с выключенной подсистемой снятия решеток, с включенным «наивным» алгоритмом (опирающимся исключительно на анализ гистограмм) и с предложенным алгоритмом. Качество распознавания замерялось как процент правильно (без единой ошибки) распознанных полей. Ценой двухкратного замедления работы системы удалось поднять качество распознавания с 30% до 80%. Следует отметить, что системы распознавания документов с качеством распознавания ниже 70% как правило нерентабельны – заполнение оператором всех полей оказывается быстрее, чем исправление ошибок.

Как показал детальный просмотр обработанных изображений, предложенный алгоритм обеспечивает:

- Корректную работу с полями, захватывающими посторонние контрастные линии разграфки.
- Корректную обработку полей с переменной шириной ячеек.
- Корректную обработку решеток знакомест с нарушенной геометрией (включая веерные искажения).

	Без снятия	Наивный метод	Предложенный алгоритм
Распознанных полей, %	32,34	58,92	80,24
Общее время работы, сек	735	963	1323
Время работы алгоритма, сек	0	231	592

Таблица 2. Результаты работы алгоритма.

4. Список литературы

[1] П.В. Безматерных, Д.П. Николаев, В.В. Постников, “Метод идентификации типа документа по структуре его проекций на координатные оси”, *Труды конференции «Информационные технологии и системы»*, 2008, с. 498–501.

[2] Р. Гонсалес, Р. Вудс, **“**Цифровая обработка изображений”, М.: Техносфера, 2005, 1072 с.

[3] D.P. Nikolaev, S.M. Karpenko, I.P. Nikolaev, P.P. Nikolayev, **“**Hough Transform: Underestimated Tool in the Computer Vision Field”, *22st European Conference on Modelling and Simulation (ECMS)*, 2008, pp. 238–243.

[4] Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн, **“**Алгоритмы: построение и анализ”, М.: Вильямс, 2005, 1296 с.