Генеративное распознавание штрихкодов с применением аппарата быстрых обобщенных преобразований Хафа

Павел Безматерных

Институт системного анализа РАН

bezmpavel@gmail.com

Сергей Гладилин

Институт проблем передачи информации имени А.А. Харкевича РАН

gladilin@iitp.ru

Дмитрий Николаев

Институт проблем передачи информации имени А.А. Харкевича РАН

dimonstr@iitp.ru

Аннотация

В работе предлагается метод быстрого вычисления оценки близости матрицы распознаваемого штрихкода с матрицами идеалов, использующий алгоритм построения быстрых обобщенных преобразований Хафа (далее БОПХ), также рассматриваются оптимизации данного алгоритма по скорости и памяти, и проводится сравнение с другими способами реализации генеративного распознавания.

1. Введение

Задача распознавания штрихкодов, отсканированных или сфотографированных в неизвестных заранее условиях, актуальна во многих промышленных системах. Большинство штрихкодов (и мы будем рассматривать именно их) представляют собой совокупность ячеек, каждая из которых может быть черной или белой. У одномерных штрихкодов ячейки представляют собой отрезки, у двумерных – прямоугольники или шестиугольники (рис. 1). По аналогии с минимальными элементами изображения, называемыми пикселями (от англ. *picture cell*), ячейки штрихкодов мы будем называть *барселями* (от англ. *barcode cell*).

Как правило, работа систем распознавания штрихкодов подразделяется на несколько этапов. На первых этапах происходит локализация штрихкода и выделение его из остального изображения, затем штрихкод разделяется на отдельные барсели. Пример исходного изображения для данной подсистемы приведен на рис. 2.

а б в

Рисунок 1. Примеры штрихкодов:
а) штрихкод формата UPC-A;
б) штрихкод формата Aztec;
в) штрихкод формата MaxiCode.

Далее происходит непосредственно распознавание. Все распространенные форматы двумерных штрихкодов характеризуются наличием нескольких зон, различающихся по способу использования. К первой зоне относятся ориентационные барсели – их значение фиксировано для каждого формата штрихкода, а назначение – служить метками для алгоритмов локализации и ориентации штрихкода. Такие барсели одинаковы на всех штрихкодах данного формата, информация в них не кодируется. Алгоритм распознавания ориентирует штрихкод, основываясь на значениях барселей в этой зоне.

Ко второй зоне относятся барсели, содержащие метаинформацию – алфавит кодируемого значения, длину закодированной строки и т.д. Алгоритм распознавания настраивает свои параметры по этой зоне перед началом работы.

Третья зона содержит непосредственно

Рисунок 2. Пример исходного изображения в системе распознавания штрихкодов.

полезную информацию и (возможно) контрольные суммы, необходимые для ее восстановления в случае повреждения отдельных барселей.

Важно отметить, что, как правило, уже на самых ранних стадиях выполняется бинаризация изображения [3, 4]. Это упрощает работу алгоритмов ценой потери части информации. Другой недостаток описанного класса алгоритмов – потеря возможности распознания штрихкода в случае повреждения метаинформации, т.к. в отличие от полезной информации метаинформация, как правило, не защищена контрольными суммами [5].

В некоторых случаях возможны другие методы распознавания, позволяющие не выполнять бинаризацию. Рассмотрим случай, когда распознаваемый штрихкод кодирует не произвольное значение, а лишь одно из некоторого сравнительно небольшого набора (к примеру – одно из миллиона возможных). В этом случае можно применить генеративный метод, суть которого заключается в том, что генерируется весь миллион возможных матриц штрихкода (называемых *идеалами*), после чего матрица распознаваемого штрихкода сравнивается с каждым идеалом в поисках наименьшего отклонения, заданного определённым образом. Отклонение вычисляется между штрихкодом в оттенках серого и монохромным идеалом, т.е. без проведения бинаризации и вероятной потери части значимой информации. Генеративный метод одинаково хорошо работает как при повреждениях барселей полезной информации, так и метаинформации. Основным его недостатком является потенциально невысокая скорость работы при большом количестве возможных значений штрихкода.

В данной работе предлагается метод быстрого вычисления близости матрицы распознаваемого штрихкода с матрицами образцов, использующий алгоритм построения быстрых обобщенных преобразований Хафа [1], рассматриваются оптимизации данного алгоритма по скорости и памяти, а также проводится сравнение с другими способами реализации генеративного распознавания.

2. Постановка задачи генеративного распознавания штрихкода

На вход системы поступает изображение локализованного, правильно ориентированного штрихкода, разделенного на отдельные барсели. Для каждого барселя вычислено среднее значение яркости, нормированное от 0 до 1 (Рис. 3).

Имеется набор идеалов – изображений штрихкодов такого же формата, каждый барсель которых имеет яркость строго 0 (черный) или 1 (белый). Необходимо выбрать среди идеалов наиболее близкий к поступившему на вход штрихкоду (будем называть его *тестом)*.

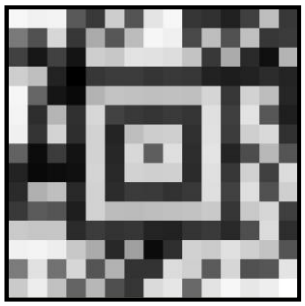

Рисунок 3. Пример штрихкода, попадающего на вход системы.

Пронумеруем все барсели штрихкода значениями от 1 до n (это значение одинаково для теста и идеалов, т.к. по условию они все одинакового формата). Обозначим A_j яркость j-ого барселя теста: $0 \le A_j \le 1$. Весь тест задается набором значений $A = \{ A_j \}$, $j=1,\ldots,n$.

Пусть $B_i = \{ B_{ij} \}$, $j=1,...,n$; $i=1,...,N$ – i-ый идеал (всего идеалов N), $B_{ij} \in \{0,1\}$. Определим отклонение A от B_i по следующей формуле:

$$d(A,B_i) = \frac{\sum_{j=1}^{n} A_j B_{ij}}{\sum_{j=1}^{n} B_{ij}} - \frac{\sum_{j=1}^{n} A_j (1-B_{ij})}{\sum_{j=1}^{n} (1-B_{ij})} = \frac{\sum_{j=1}^{n} A_j B_{ij}}{\sum_{j=1}^{n} B_{ij}} - \frac{\sum_{j=1}^{n} A_j - \sum_{j=1}^{n} A_j B_{ij}}{n - \sum_{j=1}^{n} B_{ij}}$$

Таким образом, задача поиска самого похожего идеала сводится к вычислению $d(A, B_i)$ для всех $i=1,...,N$ и выбору значения i, при котором величина $d(A, B_i)$ минимальна.

Поскольку количество идеалов N может быть достаточно велико (сотни тысяч), актуальной является задача оптимизации вычислений. Так, число белых барселей $\sum_{j=1}^{n} B_{ij}$ можно вычислить для каждого идеала заранее. Вычисление суммарной яркости теста $\sum_{j=1}^{n} A_j$ не требует значительных затрат, ведь может быть выполнено единственный раз независимо от числа идеалов. Поэтому основной задачей является быстрое вычисление величин

$$s_i = \sum_{j=1}^{n} A_j B_{ij} ,$$

которые, учитывая, что $B_{ij} \in \{0,1\}$, могут быть переписаны следующим образом:

$$s_i = \sum_{j:B_{ij}=1} A_j . \qquad (1)$$

Вычисление сумм s_i методом "грубой силы" требует значительного количества операций, при этом многие подсуммы могут быть вычислены многократно. Поэтому возможной оптимизацией является организация такой схемы вычислений, которая позволила бы использовать ранее вычисленные подсуммы, не считая их повторно.

3. Генеративное распознавание штрихкодов с использованием ОПХ

Рассмотрим более общую задачу: пусть $A = \{ A_j \}$ – множество переменных, содержащее n элементов, а $C = \{ C_i \}$ – множество сумм ($|C|=N$, $\sum|C_i|=K$), каждый элемент которого содержит некоторое подмножество переменных из A. Вычисление сумм C называется обобщенным преобразованием Хафа (ОПХ). Очевидно, что вычисление всех сумм (1) есть ОПХ на множестве переменных A по набору подмножеств $C_i=\{j: B_{ij} = 1\}$. Сложностью ОПХ называется общее число суммаций, равное $K–N$. Назовем ОПХ тривиальным, если все $|C_i|=1$. Сложность такого ОПХ равна нулю.

Поставим задачу поиска оптимальной по количеству операций схему вычисления всех сумм, используя только операцию сложения. Данная задача известна как задача оптимального вычисления ОПХ или задача построения быстрого ОПХ (БОПХ). Формально, решением задачи построения БОПХ является последовательность операций $z_k \leftarrow x_k + y_k$, $k \le M$ (где в качестве x и y могут выступать A_j или z_l, $l<k$), вычисляющая все суммы C_i, причём не существует подобной последовательности с меньшим M.

Решение этой задачи позволило бы ускорить метод генеративного распознавания, минимизируя число требуемых суммаций. К сожалению, задача построения БОПХ является NP-трудной, т.к. она по сути совпадает и строго сводима к задаче вычисления семейства [7].

Таким образом, встает вопрос об эффективных методах построения субоптимальных БОПХ. Один из подобных алгоритмов изложен в работе [1]. Однако представленная в указанной работе реализация алгоритма построения оказывается неработоспособной при большом количестве подмножеств суммации. Это связано с тем, что с целью ускорения алгоритм использует информационные таблицы размером $O(M^2)$. Из исходной постановки (распознавание штрихкодов) следует, что количество сумм N имеет порядок не менее 10^5 – 10^6. Поскольку очевидно, что $M \ge N$, алгоритм из [1] неприменим в данном случае. Далее будет предложен алгоритм оптимизации вычисления ОПХ, не требующий квадратичного от M расхода памяти.

4. Жадный алгоритм оптимизации вычисления ОПХ

Для решения задачи построения субоптимальных БОПХ будем строить итерационный алгоритм, на каждом шаге которого будет порождаться одна операция БОПХ и ОПХ со сложностью, меньше исходной. Алгоритм завершается, когда после очередного шага ОПХ становится тривиальным, при этом БОПХ оказывается построенным. Для выбора шага будем применять жадную стратегию. Шаг данной стратегии может быть описан следующим образом:

- осуществляется поиск такой пары индексов k и l, что подсумма A_k+A_l входит в наибольшее число сумм, которые необходимо вычислить (т.е. пара k, l входит в наибольшее число множеств C_i, $i=1,...N$).
- вместо суммы этой пары вводится новая переменная A_m, после чего в каждом множестве C_i вместо элементов k и l вводится единственный элемент m.

Очевидно, что данный алгоритм сходится, т.к. суммарная мощность множеств C_i на каждом шаге уменьшается. Также очевидно, что на каждом шаге алгоритма комбинация, состоящая из частичного БОПХ и перестроенного ОПХ, эквивалентна исходной задаче.

В [1] справедливо отмечено, что оценка уменьшения сложности ОПХ при сокращении элементов k и l на каждом шаге делает алгоритм слишком медленным. Поэтому там же предложена оптимизация, заключающаяся в поддержании в памяти для каждых k и l списка

$$i : k \in C_i, l \in C_i$$

требующего квадратичного объема памяти от суммы исходных переменных с количеством введенных суммаций, что как было показано выше, существенно ограничивает применимость метода. В связи с этим были разработаны другие методы оптимизации оценки пары (k, l), которые и будут описаны ниже.

5. Варианты оптимизации

Определим понятие *цены хода* алгоритма как количество множеств C_i, мощность которых в результате шага алгоритма уменьшится (как было описано выше, если мощность C_i уменьшается, то всегда на 1). Задача жадного алгоритма – в качестве очередного шага всегда делать шаг с максимальной возможной ценой. Суть предполагаемых оптимизаций заключается в предварительной оценке максимальной возможной цены каждого шага, чтобы прекращать перебор, как только хотя бы один шаг с максимальной возможной ценой найден. Для дальнейшего описания нам понадобятся следующие тривиальные утверждения:

Утверждение 1: После каждого шага алгоритма цена хода жадного алгоритма монотонно нестрого убывает;

Следствие 1.1: Если цена рассматриваемого хода совпадает с ценой предыдущего хода, то можно делать ход;

Следствие 1.2: Если наилучшая цена хода равна единице, то ОПХ не может быть оптимизирован и следует перечислить оставшиеся операции в терминах БОПХ и прекратить процесс;

Утверждение 2: Цена хода не может быть больше минимального из числа вхождений в множества C_i среди элементов пары;

Утверждение 3: Если найдено 2 наилучших хода, не пересекающихся по элементам пары, то после выполнения первого хода оценка второго будет одной из наилучших;

Следствие 3.1: Последовательное применение непересекающегося множества наилучших ходов не нарушает жадную стратегию.

На основании сформулированных ранее утверждений было предложено две оптимизации: отсечение по цене хода, и пакетное исполнение наилучших непротиворечивых ходов.

Первая оптимизация опирается на следствия 1.1 и 1.2 и состоит в следующем: отсортируем пары по верхней грани, определяемой из утверждения 2. По мере перебора пар будем проверять, превышает ли цена наилучшей из уже рассмотренных пар верхнюю грань текущей пары. Если это так, то проверка остальных пар избыточна.

Вторая оптимизация непосредственно опирается на следствие 3.1 и может быть применена без увеличения сложности алгоритма: по мере оценки ходов будем поддерживать вектор наилучших ходов, добавляя варианты непересекающихся ходов той же цены и обнуляя вектор, если нашелся ход с лучшей ценой. Чтобы проверить, пересекается ли очередной ход с уже отобранными, будем дополнительно поддерживать вектор использованных индексов. К сожалению, обнуление этого булевского вектора требует более чем O(1) операций. Однако можно воспользоваться следующим приёмом: будем считать, что индекс занят, только если в ячейке вектора хранится определённая константа h. Вместо обнуления вектора будем инкрементировать h.

Для оценки эффективности указанных оптимизаций были исследованы три варианта жадного алгоритма:

1. метод «грубой силы»: перебор всех вариантов в поисках шага с наилучшей ценой хода;
2. метод «грубой силы» с отсечением по цене хода.
3. метод поиска набора непротиворечивых шагов с одинаковой ценой хода (и отсечением по цене хода).

Перечисленные выше варианты были реализованы программно, и были испытаны при обработке 1000 сумм. Время работы составило для первого метода – 81 сек, второго – 10 сек, и 3 сек для послойного метода соответственно.

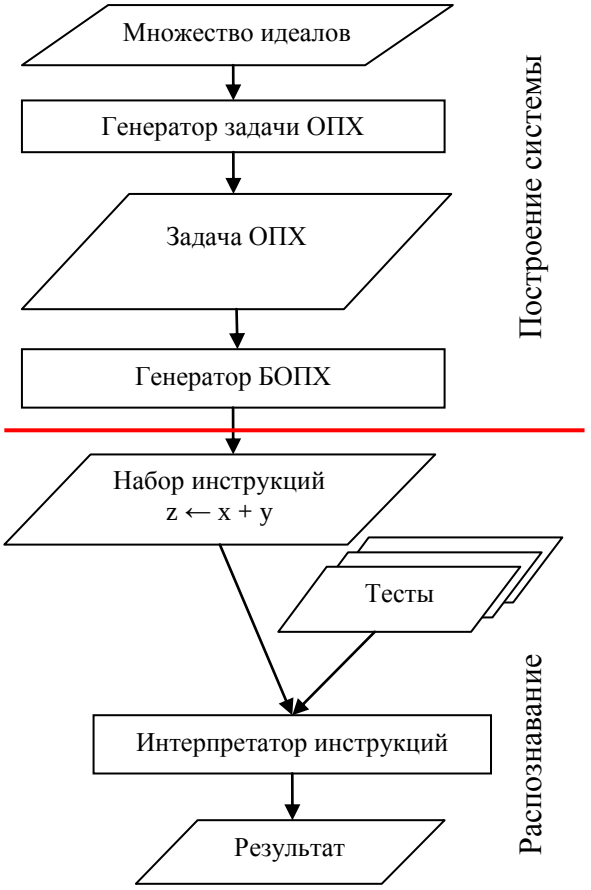

Рисунок 4. Схема работы системы.

6. Схема работы системы

Опишем общую схему работы системы распознавания, использующей аппарат БОПХ.

На подготовительном этапе на вход системе подается всё множество идеалов, на основании которого строится задача ОПХ вычисления сумм s_i. Система на основании данной задачи ОПХ строит БОПХ – набор инструкций для близкого к оптимальному вычислению. Данные инструкции записываются на специализированном языке (БОПХ-ассемблер). Кроме того, для каждого идеала вычисляется $\sum_{j=1}^{n} B_{ij}$.

Для распознавания поданного на вход теста производится вычисление всех s_i путем интерпретации инструкция БОПХ-ассемблера. На основании них остается вычислить значения $d(A, B_i)$ и выбрать i, при котором $d(A, B_i)$ минимально.

7. Экспериментальная оценка эффективности генеративного распознавания с использованием БОПХ

В ходе работы было реализовано несколько вариантов метода генеративного распознавания штрихкодов и проведена экспериментальная оценка их эффективности. Напомним, что задача распознавания сводится к поиску минимума $d(A, B_i)$ для всех $i=1,...,N$, а основные затраты приходятся на вычисление величин $s_i = \sum_{j=1}^{n} A_j B_{ij}$.

Во всех методах $d(A, B_i)$ вычислялось совершенно идентично, разница заключалась лишь в методе вычисления s_i.

Далее приведено краткое описание реализованных вариантов и полученные результаты.

1. IF – для каждого барселя идеала проверяется его яркостное значение, и если оно не 0, то осуществляется операция сложения.
2. MUL – скалярное произведение образца с идеалом.
3. MULASM – скалярное произведение образца с идеалом с использованием SSE-команд x86 ассемблера (использовалась команда psadbw [6])
4. IND – заранее строится список индексов белых барселей каждого идеала и суммирование осуществляется только по данному списку индексов.
5. FGHT – интерпретация вычисленной схемы БОПХ (записанной на БОПХ-ассемблере).
6. FGHTASM – интерпретация вычисленной схемы БОПХ с использованием динамической генерации ассемблерного кода целевой машины с последующим исполнением.

Таблица 1. Сравнение различных вариантов реализации генеративного распознавания.

Метод	Общее время, с	Среднее время на тест, с
IF	1180.9	0.2455
MUL	232.1	0.0483
MULASM	66.8	0.0139
IND	282.1	0.0586
FGHT	75.9	0.0157
FGHTASM	32.6	0.0067

Все перечисленные методы были опробованы на тестовом наборе данных, состоящем примерно из 5 000 образцов и 230 000 идеалов. Схема ОПХ, подготовленная по данным идеалам, содержала 24 444 348 суммаций, в то время как построенная в результате работы программы схема БОПХ состояла всего лишь из 1 041 330 операций суммирования, т.е. удалось сократить объем операций почти в 24 раза. Исходные тексты программ были написаны на языке C++ и собраны компилятором MSVC 6.0. Тестирование проводилось на машине с конфигурацией Intel(R) Core(TM) 2 Quad CPU Q9400 @ 2.66 ГГц, 3 Гб ОЗУ. Результаты времени работы программы на данной машине представлены в таблице 1.

Заметим теперь, что результатом работы алгоритма перевода ОПХ в БОПХ является набор инструкций вида $z \leftarrow f(x, y)$, причем результат каждой команды заносится всегда в новую ячейку. Подобная форма записи вычислений известна как простой блок в SSA-форме [2]. Непосредственное выполнение программы в SSA-форме неэффективно с точки зрения механизмов кэширования и объемов требуемой памяти. К счастью, для программ в SSA-форме существуют полиномиальные алгоритмы оптимизации объема используемой памяти [2]. Фактически, для простых блоков задача сводится к задаче интервальной раскраске графа, для которой оптимальное решение находится наивным жадным алгоритмом [8]. Применение алгоритма оптимизации памяти в нашем случае позволило уменьшить число требуемых ячеек с 1041330 до 254978 и обеспечило ускорение вычисления БОПХ схемы (см. Таблицу 2).

Таблица 2. Влияние числа используемых ячеек памяти на быстродействие.

Метод	Общее время, с	Среднее время на тест, с
FGHT	75.9	0.0157
FGHT_OPT	68.5	0.0142
FGHTASM	32.6	0.0067
FGHTASM_OPT	26.8	0.0055

8. Апробация в индустриальной системе

Созданная система была опробована в задаче распознавания штрихкодов, возникающей при проведении детского конкурса «Ярмарка идей на Юго-Западе». В конкурсе применяются штрихкоды типа Aztec формата 15x15. Всего распечатывается порядка 230 000 стикеров со штрихкодом на каждом, которые наклеиваются на работы участников, а затем фотографируются и распознаются генеративным методом.

9. Заключение

В работе рассматривается генеративный метод распознавания штрихкодов с использованием аппарата ОПХ. Преимуществом генеративного метода является возможность распознавания без проведения бинаризации. Однако алгоритм генеративного распознавания с опорой на ОПХ требует значительной подготовительной стадии расчетов, поэтому

применение этого метода оправдано именно в тех случаях, когда набор выделяемых идеалов известен заранее и в дальнейшем не меняется. Для оптимизации указанных предварительных расчетов было предложено две модификации жадного алгоритма, позволяющие существенно сократить время построения БОПХ. Также было реализовано множество альтернативных вариантов метода генеративного распознавания. Поставленный эксперимент показал, что оптимальным методом по скорости работы является алгоритм, базирующийся на интерпретации вычисленной схемы БОПХ с использованием динамической генерации ассемблерного кода целевой машины. Полученные результаты подтверждают возможность использования генеративного метода распознавания штрихкодов с использованием аппарата ОПХ в промышленных системах.

Работа частично финансово поддержана грантом РФФИ 09-07-00473-а.

10. Список литературы

[1] *С.М. Карпенко, Д.П.Николаев, П.П. Николаев, В.В. Постников* Общий метод построения быстрых обобщенных преобразований Хафа // В сб.: "Интеллектуальные системы (IEEE AIS"05)", "Интеллектуальные САПР (CAD-2005)": Труды международной конференции / М.: Изд-во Физматлит, 2005. Т. 2. С. 313-318.

[2] *S. Hack, D. Grund, and G. Goos.* Register allocation for programs in SSA-form // In Proceedings of the International Conference on Compiler Construction, pages 247-262. LNCS 3923, Springer Verlag, 2006.

[3] *Rafal Tarlowski, Michal Choras* Digital Analysis of 2D Code Images Based on Radon Transform // Computer Recognition Systems 3, AISC 57, pp. 143-150, Springer-Verlag Berlin Heidelberg, 2009

[4] *Michael Trummer, Joachim Denzler* Reading out 2D barcode PDF417 // Progress in pattern recognition, Springer, 2007, ISBN 1846289440, 9781846289446

[5] Information technology - Automatic identification and data capture techniques Aztec Code bar symbology specification ISO/IEC 24778:2008

[6] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z (http: // www.intel.com / Assets / PDF / manual / 253667.pdf)

[7] *М.Гэри, Джонсон Д.* Вычислительные машины и труднорешаемые задачи, 1982

[8] *Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн* Алгоритмы. Построение и анализ, Вильямс, 2010