Система быстрого обнаружения параметрических кривых на серых и цветных изображениях с контролем достоверности

Алексей Левашов, Дмитрий Юрин
Факультет вычислительной математики и кибернетики
Московский государственный университет, Москва, Россия
alexeylevashov89@gmail.com , yurin@cs.msu.su

Аннотация

Предлагается многоэтапная система поиска параметрических кривых на изображении. Сначала производится поиск граничных линий, затем, после ряда подготовительных процедур, результат сохраняется в виде цепочек связанных пикселей. Эти цепочки анализируются целиком и по-фрагментно. Анализ каждого фрагмента на соответствие модели выполняется сначала на основе рандомизированных методов, что позволяет достичь высокого быстродействия, и, в случае успешного прохождения такого теста, производится точная оценка параметров модели методом наименьших квадратов и достоверности гипотезы на основе критерия хи-квадрат. Пофрагментный анализ, фазы роста фрагмента и слияния фрагментов, постоянный контроль достоверности в смысле хи-квадрат позволяют находить параметрические кривые различных типов одновременно, а из набора моделей выбирать наиболее простую, описывающую кривую с точностью, соответствующей погрешности исходных данных.

Ключевые слова: *edge detector, параметрические кривые, векторизация.*

ВВЕДЕНИЕ

Зачастую в задачах компьютерного зрения целесообразно применять упрощенный подход, основанный на поиске известных ориентиров. В роли таких ориентиров могут выступать штанги, люки, ребра корпуса – т.е. объекты, видимые в виде простых геометрических фигур. Таким образом, задача сводится к быстрому и достоверному поиску на изображении кривых, описываемых аналитическим выражением с небольшим количеством параметров и оценке этих параметров.

Существует множество методов, решающих данную проблему. Первым является метод Хафа [4], который оказался очень не эффективным при поиске многопараметрических кривых. Существенным улучшением данного подхода является использование разреженных массивов, а также применение рандомизированных выборок [1, 5, 8]. Однако рандомизированные методы пропускают кривые на сложных изображениях из-за низкой вероятности того, что случайная выборка точек принадлежит одной кривой. Оба подхода плохо отслеживают кривые малой длины, а также ориентированы на поиск кривых только одного типа. Другой более точный подход к решению данной задачи основан на использовании бимлетов [2]. Однако подобные методы также ориентированы на нахождение кривых только одного типа.

В настоящей работе предлагается метод, анализирующий цепочки связанных пикселей – граничных линий, что становится возможным при использовании высококачественных детекторов граничных точек. Путем рандомизированных проверок отбрасываются граничные линии, заведомо не удовлетворяющих ни одной из рассматриваемых гипотез. Согласно оставшимся гипотезам оценка параметров выполняется методом наименьших квадратов, достоверность контролируется методом хи-квадрат.

1. АЛГОРИТМ

Общая схема алгоритма состоит из нескольких шагов. Сначала применяется детектор граничных линий Канни – для изображений в оттенках серого, Дизензо-Кумани – для цветных изображений [6] с вычислением производных путем свертки с производными функции Гаусса и процедурой подавления не максимальных точек [3]. При использовании этих детекторов получаются линии с малым числом разрывов и, за исключением незначительного числа точек, – шириной в 1 пиксель. После выполнения данного шага получаем изображение с граничными линиями. Следующим этапом является векторизация, т.е. граничные точки собираются в виде списка кривых, а кривые – в виде списка точек. Каждая точка представляется в памяти как пара координат, а также записывается направление и величина градиента в данной точке. Весь дальнейший анализ ведется только с таким списком границ. Тут есть две проблемы, которые мешают этапу векторизации. Первая – граничные линии могут быть не единичной толщины, что представляет трудность в записывании их в массив. Вторая – возможны точки ветвления, тем самым придется хранить список векторов в виде дерева или даже графа. Однако в точках ветвления точность детектирования границ обычно невысока. Поэтому точки ветвления удаляются вместе с небольшой окрестностью, а линии не единичной толщины утончаются.

Таким образом, получаем схему алгоритма:

1) Детектор граничных линий [6].
2) Утончение граничных линий [9].
3) Удаление точек ветвления.
4) Векторизация изображения.
5) Анализ каждой граничной линии в отдельности.
6) Объединение параметрических кривых с близкими значениями параметров.

Шаг 6 не рассматривается подробно, из-за ограничений объема статьи, но он важен, если кривые имеют разрывы.

1.1 Удаление точек ветвления

Пусть P – пиксель на изображении. Считаем, что $P=1$, если P – граничная точка, 0 – иначе. Вокруг каждой граничной точки P берется окно 3x3 пикселя $P_1, P_2, ..., P_9$ (Рис 1)

и вычисляются характеристики $J(P_1)=\sum_{i=2}^{9}P_i$ и $K(P_1)=$количеству паттернов 01 в последовательности $P_2,...,P_9,P_2$. Точка удаляется вместе с ее окрестностью, если выполняется условие $J(P_1)\geq 5 \ \vee \ K(P_1)>2$. В отличие от [5], точки удаляются сразу при сканировании. Если в процессе сканирования хотя бы одна точка была удалена, следует потом просканировать еще раз.

$P_9(i-1,j-1)$	$P_2(i-1,j)$	$P_3(i-1,j+1)$
$P_8(i,j-1)$	$P_1(i,j)$	$P_4(i,j+1)$
$P_7(i+1,j-1)$	$P_6(i+1,j)$	$P_5(i+1,j+1)$

Рис 1: Индексация пикселей в окне 3x3

1.2 Векторизация

Переход от пиксельных данных к векторному представлению существенно упрощает дальнейшую работу. Целесообразно хранить данные как список списков – список кривых, каждая из которых является списком точек. Здесь требуется произвольный доступ и хранение в виде массива. Было применено следующее решение. Память выделялась массивами по 2^{16}=65536 элементов по мере исчерпания. Непрерывная адресация (обобщенный массив) осуществляется путем использования старших бит индекса для выбора массива, а 16 младших – для выбора элемента. На этой памяти выделялись элементы для всех списков. По окончании сбора точек, данные сортировались так, что точки различных кривых оказывались последовательно размещенными в обобщенных массивах, последовательно одна кривая за другой.

Если предположить, что мы ищем исключительно гладкие кривые, то, если на граничной линии есть уголки, имеет смысл анализировать только сегменты между уголками. Находить такие уголки можно с помощью детектора уголков Харриса или Форстнера со скользящим окном, однако можно ускорить алгоритм, рассматривая уже собранные списки. Для каждого граничного пикселя по его соседям строим ковариационную матрицу градиентных направлений и по второму собственному числу, если оно больше некоторого порога и является локальным максимумом, выделяем его как уголок. Как показано на Рис 2 карты уголков, построенные таким методом, остаются плавными.

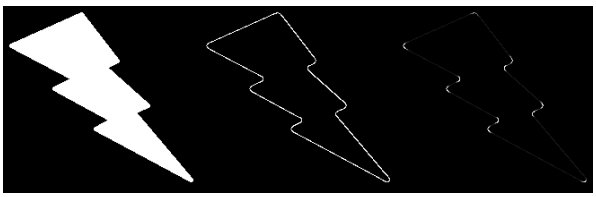

Рис 2: Карта угоков

1.3 Алгоритм анализа граничной линии

Т.к. одна граничная линия может состоять из нескольких параметрических кривых, то важной задачей является быстро и достоверно разбить данную линию на модели. Эту задачу выполняет *Алгоритм 1*.

Пусть P – массив точек данной граничной линии. Под сегментом $P[n_1,...,n_2]$ будет пониматься набор точек из P с индекса n_1 и заканчивая индексом n_2. Под функцией $\texttt{size}(P)$ понимается число элементов массива P.

Пусть $\mathbf{M}$ – это заданный априори набор математических моделей параметрической кривой, на соответствие которым проверяются фрагменты кривой. В выражении $h\in\mathbf{M}$ под h может пониматься окружность, прямая, эллипс и любая другая аналитическая кривая. Под конкретной параметрической кривой понимается $\{h, params\}$, где $params$ – это параметры кривой модели h. Для каждого такого h необходимо знать следующие алгоритмы:

1) $\texttt{LeastSquareMatching}(h,n_1,n_2)$ – метод наименьших квадратов для точек из сегмента $P[n_1,...,n_2]$, который находит оптимальные параметры для h.
2) $\texttt{ChiSquareFitting}(\{h, params\},n_1,n_2)$ – критерий достоверности χ^2 модели h с параметрами $params$ для сегмента $P[n_1,...,n_2]$.
3) $B(h)$ – минимальное необходимое количество точек для построения модели h.
4) $\texttt{FindParams}(P',h)$ – нахождение параметров модели h по P' точкам, при чем $\texttt{size}(P')=B(h)$.
5) $a\in m$ – способ определить лежит ли точка a на параметрической кривой $m=\{h, params\}, h\in\mathbf{M}$.

Пусть S – упорядоченный набор $S_0,...,S_{n-1}$ разделителей, которые делят массив на сегменты. Разделитель это индекс элемента в массиве. Множество точек между двумя ближайшими разделителями – это сегмент массива. Считаем, что $S_{-1}=S_0$ и $S_n=S_{n-1}$. Изначально S состоит из 2-х элементов: начального и конечного индексов P.

Все найденные кривые будут записаны в M. M – это множество троек $\{m,n_1,n_2\}$, где n_1 – индекс начала сегмента, n_2 – индекс конца сегмента, а $m-\{h, params\}, h\in\mathbf{M}$ параметрическая кривая, описывающая данный сегмент.

Идея ***Алгоритма 1*** заключается в следующем: сначала рассматривается все точки P, и пробуется с помощью ***Алгоритма 3.*** найти параметрическую кривую описывающий данный сегмент. Если такого сделать не удается, то сегмент делится пополам и процедура повторяется для 2-х половинок. Как только был найден сегмент, для которого нашлась оптимальная кривая, то выполняются ***Алгоритм 2a*** и ***Алгоритм 2b*** для определения точных границ сегмента для найденной кривой. Деление пополам продолжается до тех пор, пока сегмент не будет меньше $C_{\min}$ – константы, определяющей минимальное количество точек в сегменте.

Алгоритм 1. $\texttt{Separation}()$: Разбиение P на сегменты, удовлетворяющие моделям.

1. $S \leftarrow [0, size(P)-1]$
2. $M \leftarrow \{\}$
3. **while** $\exists i : (S_{i+1} - S_i > C_{\min} \wedge !\exists m : \{m, S_i, S_{i+1}\} \in M)$
4. **do for** $i = 0, i < size(S) - 1$
5. $m \leftarrow$ `FindParamCurve` (S_i, S_{i+1})
6. **if** m найдена **then**
7. $n_1 \leftarrow$ `ExpandLeft` (m, S_{i-1}, S_i)
8. $n_2 \leftarrow$ `ExpandRight` (m, S_{i+1}, S_{i+2})
9. $S_i \leftarrow n_1$
10. $S_{i+1} \leftarrow n_2$
11. $M \leftarrow M \cup \{m, n_1, n_2\}$
12. **else**
13. $S \leftarrow [S_0, ..., S_i, \frac{S_i + S_{i+1}}{2}, S_{i+1}, ..., S_{n-1}]$

В процессе деления сегментов пополам, скорее всего будет обнаружена только часть кривой, и целесообразно расширить границы найденного сегмента так, чтобы расширенный сегмент описывал кривую максимальной длины. C_r – количество случайных точек, которое берется при расширении сегмента. В *Алгоритме 2a* индекс n_2 – текущее положение левой границы сегмента для кривой, а n_1 - граница максимально возможного расширения слева.

Алгоритм 2a. `ExpandLeft`(m, n_1, n_2):

1. **if** $(n_2 - n_1) \le 1$
2. **then return** n_2
3. $i \leftarrow \frac{n_1 + n_2}{2}$
4. **while** $i < n_2 - 1$ **do**
5. **if** $P_i \in m$ **then**
6. $T_r \leftarrow C_r$ случайных точек из $P[i, ..., n_2]$
7. **if** $\forall a \in T_r : a \in m$ **then**
8. **if** $\forall a \in P[i, ..., n_2] : a \in m$ **then**
9. $n_1 \leftarrow 2i - n_2$
10. $n_2 \leftarrow i$
11. **goto 1**
12. $i \leftarrow \frac{i + n_2}{2}$
13. **return** n_2

В цикле 4-12 пытаемся сдвинуть правую границу на $(n_2 - n_1)/2^n$, проверяя при этом, принадлежат ли присоединяемые точки к m. После добавления в строчках 9 – 11 повторяем алгоритм для уточнения границы слева от новой найденной.

Алгоритм 2b `ExpandRight`(m, n_1, n_2) Аналогичен *Алгоритму 2a*: n_1 - индекс правой границы найденного сегмента, а n_2 - индекс максимально возможного смещения границы.

В *Алгоритме 3* находится оптимальная кривая для сегмента $P[n_1, ..., n_2]$. C_{χ^2} – степень достоверности критерия χ^2.

Алгоритм 3. `FindParamCurve`(n_1, n_2): Определение модели, описывающий сегмент $P[n_1, ..., n_2]$

1. $m \leftarrow$ *пустая модель*
2. $T_{\chi^2} \leftarrow 0$
3. **for each** $h \in \mathbf{M}$ **do**
4. **if** QuickTest(h, n_1, n_2) **then**
5. $params \leftarrow$ LeastSquareMatching(h, n_1, n_2)
6. $T_{\chi^2}' \leftarrow$ ChiSquareFitting$(\{h, params\}, n_1, n_2)$
7. **if** $T_{\chi^2}' > C_{\chi^2}$ **then**
8. **if** $T_{\chi^2}' > T_{\chi^2}$ **then**
9. $T_{\chi^2} \leftarrow T_{\chi^2}'$
10. $m \leftarrow \{h, params\}$
11. **return** m

В *Алгоритме 4* происходит быстрая проверка, может ли сегмент $P[n_1, ..., n_2]$ описываться кривой типа h. Используются константы C_q – количество тестов быстрой проверки и C_q^{accept} – минимальное необходимое количество пройденных тестов быстрой проверки. В алгоритме берется случайным образом $B(h)+1$ точек из текущего сегмента, по $B(h)$ точкам строится кривая типа h и проверяется, лежит ли дополнительная точка на кривой. Так делается C_q раз.

Алгоритм 4. `QuickTest` (h, n_1, n_2): Быстрая проверка гипотезы h на сегменте $P[n_1, ..., n_2]$

1. $count \leftarrow 0$
2. **for** $i = 0, i < C_q$ **do**
3. $P' \leftarrow B(h)$ точек из $P[n_1, ..., n_2]$, выбранных случайно
4. $c \leftarrow$ случайно выбранная точка из $P[n_1, ..., n_2]$

5. $params \leftarrow$ `FindParams` (P', h)

6. **if** $c \in \{h, params\}$ **then**

7. $count \leftarrow count + 1$

8. **return** $count > C_q^{accept}$

2. РЕЗУЛЬТАТЫ

Временные эксперименты были сделаны на процессоре Intel Core 2 Duo 2.00 GHz, оперативной памяти 2 Гб. Искались одновременно две модели: прямая и окружность. Времена работы алгоритма и его частей представлены в Таблице 1 в сравнении с алгоритмом [1]. Прочерком показаны тесты, которые алгоритм [1] не выполнил. Не было замечено чтобы разработанный алгоритм пропускал искомые кривые за исключением случаев когда их форма была сильно искажена на стадии детектирования граничных линий.

3. ЗАКЛЮЧЕНИЕ

Разработана система поиска параметрических кривых на изображениях. Сравнение с алгоритмом [1] показало близкие времена работы, но предложенный метод находит одновременно различные типы кривых и не имеет ограничений по сложности изображения. Комбинация рандомизированных проверок с методом наименьших квадратов и критерием хи-квадрат обеспечивает высокое быстродействие без потери надежности. Алгоритм не имеет ограничений на длины искомых кривых.

4. БЛАГОДАРНОСТИ

Работа выполнена при поддержке ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009 – 2013 годы и гранта РФФИ 09-07-92000-ННС_а.

ЛИТЕРАТУРА

[1] Chen T.C., Chung K.L., *An Efficient Randomized Algorithm or Detecting Circles. // CVIU 2001, V.83, P.172–191.*

[2] Donoho D.L., Huo X., Jermyn I., Jones P., Lerman G., Levi O., Natterer F. *Beamlets and Multiscale Image Analysis //In Multiscale and Multiresolution Methods, Springer 2001, P.149-196.*

[3] Devernay F. *A Non-Maxima Suppression Method for Edge Detection with Sub-Pixel Accuracy //INRIA Tech. Report RR-2724, 20 p, 1995.*

[4] Duda R., Hart P. *Use of the Hough Transformation to Detect Lines and Curves in Pictures. Comm. ACM 15: 1 – 15, 1972.*

[5] McLaughlin R.A. *Randomized Hough Transform: better ellipse detection // IEEE TENCON - Digital Signal Processing Applications, V. 1, P. 409-414, Nov. 1996.*

[6] Pratt W.K. *Digital Image Processing: PIKS Scientific inside (4th ed.) // Wiley-Interscience, John Wiley & Sons, Inc., Los Altos, California, 2007, 782 p.*

[7] Triggs B., Sdika M. *Boundary Conditions for Young - van Vliet Recursive Filtering // IEEE Transactions on Signal Processing, V.54, No.5, May 2006.*

[8] Xu L., Oja E. *Randomized Hough Transform // in Encyc. of Artif. Intell., Ed.By: J.Ramón, R.Dopico; J.Dorado; A.Pazos, IGI Global publishing comp., 2008, P.1354—1361.*

[9] Zhang T.Y., Suen C.Y. *A fast parallel algorithm for thinning digital patterns // Communications of the ACM, V. 27, No. 3, P. 236--239, 1984.*

Об авторах

Левашов Алексей Евгеньевич – студент магистратуры факультета Вычислительной математики и кибернетики Московского государственного университета. alexeylevashov89@gmail.com

Юрин Дмитрий Владимирович, к.ф.-м.н., с.н.с. лаборатории Математических методов обработки изображений факультета Вычислительной математики и кибернетики Московского государственного университета им. М.В.Ломоносова. yurin_d@inbox.ru

	Размеры Изображения	Кол-во точек	Кол-во списков	Время работы, мс									Изображения
				1	2	3	4	5	6	7	8	9	
1	256×256	13063	1511	11	4	3	6	8	21	5	54	3	
2	256×256	11839	1473	11	3	3	5	7	18	7	55	4	
3	256×256	9167	1000	11	2	3	4	5	14	6	20	0	
4	256×192	7938	689	9	2	3	8	5	18	5	20	2	
5	559×559	39076	3034	58	14	10	13	8	45	3	-	-	
6	375×486	27881	2544	34	7	6	11	17	41	2	-	-	
7	1000×667	137387	21235	139	68	40	59	74	241	2	-	-	
8	1024×1024	228526	30932	372	106	88	85	128	407	0	-	-	

Таблица 1. Временные показатели. Пронумерованные колонки содержат времена работы (1) детектор граничных линий (2) утончение линий, (3) удаление узлов, (4) векторизация, (5) нахождение кривых, (6) общее время исполнения предложенного алгоритма без детектора граничных линий, (7) количество окружностей найденных нашим алгоритмом, (8) поиск окружностей [1], (9) количество окружностей найденных алгоритмом [1].