УДК 004.93

Поиск параметрических кривых на изображениях

А.Е. Левашов[1]*, Д.В. Юрин*[2]

Факультет вычислительной математики и кибернетики (ВМиК),

Московский государственный университет им. М.В.Ломоносова (МГУ)

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 частям. По-фрагментный 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: *детектор границ, параметрические кривые, векторизация.*

Введение

Одна из ключевых задач в обработке изображений и компьютерном зрении – это поиск на изображениях объектов, представляющих интерес. Зачастую целесообразно применять упрощенный подход, основанный на поиске известных ориентиров. В роли таких ориентиров могут выступать балки, люки, окна, купола – т.е. объекты, видимые в виде прямых, окружностей, эллипсов и других простых геометрических фигур.

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

[1] Левашов Алексей Евгеньевич – студент магистратуры факультета ВМиК МГУ им. М.В.Ломоносова. Email: alexeylevashov89@gmail.com.

[2] Юрин Дмитрий Владимирович, к.ф.-м.н., с.н.с. лаборатории Математических методов обработки изображений факультета ВМиК МГУ им. М.В.Ломоносова. Email: yurin_d@inbox.ru, yurin@cs.msu.su .

оставшимся гипотезам оценка параметров выполняется методом наименьших квадратов, достоверность контролируется методом хи-квадрат.

Алгоритм

Схема алгоритма выглядит следующим образом:

1) Детектор граничных линий (*Pratt*, 2007).
2) Утончение граничных линий (*Zhang, Suen,* 1984).
3) Удаление точек ветвления.
4) Векторизация изображения.
5) Анализ каждой граничной линии в отдельности.
6) Объединение параметрических кривых с близкими значениями параметров.

На первом шаге применяется детектор граничных линий Канни – для изображений в оттенках серого, Дизензо-Кумани – для цветных изображений (*Pratt*, 2007). Используется реализация этих алгоритмов с вычислением производных путем свертки с производными функции Гаусса и процедурой подавления не максимальных точек (*Pratt*, 2007). Для гарантии того, чтобы все граничные линии были единичной толщины и не содержали точек ветвления выполняется шаги 2-3. Следующим этапом является векторизация, т.е. граничные точки собираются в виде списка кривых, а кривые – в виде массива 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 такими массивами граничных точек.

Удаление точек ветвления

Удаление точек ветвления основано на той же морфологической операции, что и утончение граничных линий, описываемой в (*Zhang, Suen,* 1984). Поэтому в этом разделе используются обозначения из (*Zhang, Suen,* 1984). Вокруг каждой граничной точки P_1 строится окно 3x3 пикселя и вычисляются характеристики $A(P_1)$ и $B(P_1)$. Точка удаляется вместе с ее

окрестностью, если выполняется условие $B(P_1) \geq 5 \vee A(P_1) > 2$. В отличие от (*Zhang, Suen,* 1984), точки удаляются сразу при сканировании. Если в процессе сканирования хотя бы одна точка была удалена, следует потом просканировать еще раз. Мотивировка удаления точек ветвления границ двоякая: с одной стороны это сильно упрощает структуру данных и ее представление в памяти, а с другой – вблизи точек ветвления погрешность детекторов границ наиболее высока, и неточность определения границ будет скорее затруднять правильную оценку параметров кривых.

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.*

P – массив точек данной граничной линии. **M** – это заданный априори набор математических моделей параметрической кривой, на соответствие которым проверяются фрагменты кривой. Под конкретной параметрической кривой понимается $\{h, params\}$, где $params$ – это параметры кривой модели $h \in \mathbf{M}$. Для каждой такой модели h задаются следующие алгоритмы:

1) `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) `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) `FindParams`(P', h) – нахождение параметров модели h по минимально необходимому количеству $B(h)$ точек P'.
5) $a \in m$ – способ определить лежит ли точка a на параметрической кривой $m = \{h, params\}$, $h \in \mathbf{M}$.

S – упорядоченный набор $S_0,...,S_{n-1}$ разделителей, которые делят массив на сегменты. Разделитель это индекс элемента в массиве. Все найденные кривые будут записаны в M, где $M-$ это множество троек $\{m,n_1,n_2\}$, где n_1, n_2 – индексы начала и конца сегмента.

Алгоритм 1. `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. $S \leftarrow [S_0,...,S_i, \frac{S_i + S_{i+1}}{2}, S_{i+1},...,S_{n-1}]$

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 *Алгоритме 1* 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. В *Алгоритме 3* C_{χ^2} – степень достоверности критерия χ^2. В *Алгоритме 4* C_q – количество тестов быстрой проверки и C_q^{accept} – минимальное необходимое количество пройденных тестов быстрой проверки.

Алгоритм 3.

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

```
1.  m ← пустая модель
2.  T_χ² ← 0
3.  for each h ∈ M do
4.    if QuickTest(h, n1, n2) then
5.      params ← LeastSquareMatching(h, n1, n2)
6.      T_χ²' ← ChiSquareFitting({h, params}, n1, n2)
7.      if T_χ²' > C_χ² then
8.        if T_χ²' > T_χ² then
9.          T_χ² ← T_χ²'
10.         m ← {h, params}
11. return m
```

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

```
1. count ← 0
2. for i = 0, i < C_q do
3.     P' ← B(h) точек из P[n1,...,n2],
       выбранных случайно
4.     c ← случайно выбранная
       точка из P[n1,...,n2]
5.     params ← FindParams(P', h)
6.     if c ∈ {h, params} then
7.         count ← count + 1
8. return count > C_q^accept
```

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 (*Chen, Chung,* 2001). Прочерком показаны тесты, которые алгоритм (*Chen, Chung,* 2001) не выполнил.

Заключение

Разработана система поиска параметрических кривых на изображениях. Сравнение с алгоритмом (*Chen, Chung,* 2001) показало близкие времена работы. Предложенный алгоритм находит все имеющиеся на изображении параметрические кривые указанных типов, в то время как алгоритм (*Chen, Chung,* 2001) попускает часть кривых, тем большую, чем сложнее изображение. Таким образом, комбинация рандомизированных проверок с методом наименьших квадратов и критерием хи-квадрат обеспечивает высокое быстродействие без потери надежности. В отличие от преобразования Хафа, алгоритм не имеет ограничений на длины искомых кривых и может находить одновременно и короткие и длинные кривые.

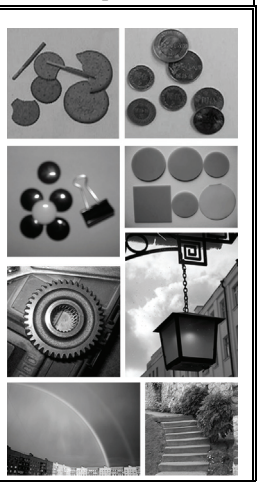
	Размер изобра-жения	Кол-во гранич-ных точек	Кол-во гранич-ных линий	Время работы, мс									Тестируемые изображения
				1	2	3	4	5	6	7	8	9	
1	256×256	13063	1511	11	4	3	6	8	21	5	54	3	
2	256×256	11839	1473	11	3	3	5	7	18	7	55	4	
3	256×256	9167	1000	11	2	3	4	5	14	6	20	0	
4	256×192	7938	689	9	2	3	8	5	18	5	20	2	
5	375×486	27881	2544	34	7	6	11	17	41	2	-	-	
6	559×559	39076	3034	58	14	10	13	8	45	3	-	-	
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) определение модели кривых и оценка параметров модели, (6) общее время исполнения предложенного алгоритма без детектора граничных линий, (7) количество окружностей найденных предложенным алгоритмом, (8) поиск окружностей (*Chen, Chung,* 2001), (9) количество окружностей найденных алгоритмом (*Chen, Chung,* 2001).

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

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

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

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.*

Searching parametric curves in images

A.E. Levashov[1], *D.V. Yurin*[2]

Faculty of Computational Mathematics and Cybernetics

Moscow State University, Moscow, Russia

Multi-stage search of parametric curves in the image is proposed. Firstly, we search the edge curves. Then after several preparatory procedures the result is stored as chains of connected pixels. These chains are analyzed in whole and by fragments. Analysis of each fragment whether it matches to model or not is performed on the basis of randomized methods. It allows to get high performance by excluding impossible alternates. If this test was passed successfully the accurate model parameters estimation is performed by least squares method and reliability of a hypothesis is checked based on the chi-square criterion. Piecemeal analysis of the curve, the growth phase of the fragment and the fusion of fragments, continuous monitoring of reliability in terms of chi-square allow to find the parametric curves of different types simultaneously and to choose from a set of models the most simple one that describes a curve with an accuracy corresponding to the error of the edge localization.

Keywords: *edge detector, parametric curves, vectorization.*

[1] Alexey E. Levashov is a student at Chair of Mathematical Physics, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia. Email alexeylevashov89@gmail.com.

[2] Dmitry V. Yurin (PhD) is a senior researcher at laboratory of Mathematical Methods of Image Processing, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia. Email yurin_d@inbox.ru