УДК 004932:621.396

Т.М. ВЛАСОВА, В.Г. КАЛМЫКОВ

АЛГОРИТМ И ПРОГРАММА РАСПОЗНАВАНИЯ КОНТУРОВ ИЗОБРАЖЕНИЙ КАК ПОСЛЕДОВАТЕЛЬНОСТИ ОТРЕЗКОВ ЦИФРОВЫХ ПРЯМЫХ

Abstract: *In the given work the algorithm of the recognition of the digital direct line segment in contours of the binary images and the software implementation of the algorithm is considered. Utilization of this algorithm to process the images will result to more natural and economical description in comparison with known ways of the description of the images. The considered algorithm and the software implementation can be used also for the description of contours when processing the half-tone and colour images.*
Key words: *image, contour, digital straight segments, algorithm, program.*

Анотація: *У даній роботі наводиться алгоритм розпізнавання відрізків цифрових прямих у контурах бінарних зображень, а також програмна реалізація алгоритму. Використання цього алгоритму для оброблення зображень призведе до того, що опис зображень буде більш натуральним та економним порівняно з відомими засобами кодування зображень. Запропоновані алгоритм і програмна реалізація можуть застосовуватись для кодування контурів при обробленні напівтонових та кольорових зображень.*
Ключові слова: *зображення, контур, відрізки цифрових прямих, алгоритм, програма.*

Аннотация: *В данной работе рассматриваются алгоритм распознавания отрезков цифровых прямых в контурах бинарных изображений и программная реализация алгоритма. Использование этого алгоритма при обработке изображений приведет к более естественному и экономному по сравнению с известными способами описанию изображений. Рассматриваемый алгоритм и программная реализация могут быть использованы также и для описания контуров при обработке полутоновых и цветных изображений.*
Ключевые слова: *изображение, контур, отрезки цифровых прямых, алгоритм, программа.*

1. Введение

Структурный анализ контуров изображений как последовательностей отрезков прямых и дуг кривых является одной из основных задач при обработке изображений с целью их интерпретации в системах искусственного интеллекта.

В большинстве случаев изображение можно рассматривать как часть плоскости, разделенную на области с постоянными или меняющимися по некоторому закону параметрами, например, оптической плотностью, цветом, текстурой, определяющие фон и объекты изображения. Неотъемлемым свойством каждой из этих областей является ее граница, то есть контур – односвязная последовательность, состоящая из отрезков прямых и дуг кривых. При обработке растрового изображения обычно выделяются контуры объектов. Однако контуры объектов, представленные в виде совокупности отдельных, граничных пикселов мало пригодны для дальнейшей обработки, поскольку недостаточно выражают его геометрическую сущность.

Распознавание контуров изображений в виде последовательности отрезков прямых можно считать одной из основных задач в процессе обработки растрового изображения. Решение задачи представления контура в виде последовательности отрезков прямых позволяет получить описание изображения в компактном виде, естественном для человеческого восприятия, инвариантном относительно аффинных преобразований, удобном, в частности, для обработки нейросетями. Отрезки прямых являются основными элементами контура. Дуга кривой также часто заменяется вписанной в нее ломаной линией как в основных положениях математического анализа, так и в большом количестве практических приложений.

Известные методы и алгоритмы, в частности, предложенные в работе [1], позволяют получить приближенное решение, что приемлемо не для всех приложений.

В настоящей работе рассматривается распознавание контура бинарного изображения как последовательности отрезков цифровых прямых без потери информации.

2. Контур как последовательность отрезков цифровых прямых

В данном разделе рассматривается структурный анализ контуров изображений как последовательности отрезков цифровых прямых, которые являются исходными данными для сегментации контура на дуги цифровых кривых и отрезки цифровых прямых.

Ограничимся рассмотрением бинарных изображений, объекты которых полностью определяются ограничивающими их контурами. Дуги цифровых кривых, как и отрезки цифровых прямых, образуются при дискретизации изображений, содержащих контуры, образованные отрезками прямых и дугами кривых.

Характерные признаки отрезков прямых и дуг кривых в процессе преобразования утрачиваются. Рассматривая дискретизованное изображение при достаточном увеличении, часто трудно бывает узнать отдельные отрезки прямых и дуги кривых в последовательности вертикальных и горизонтальных отрезков. Дополнительные трудности возникают при обработке из-за того, что линии контуров – математические линии без толщины – отображаются на экране монитора связными последовательностями пикселов, то есть визуальными линиями, имеющими толщину.

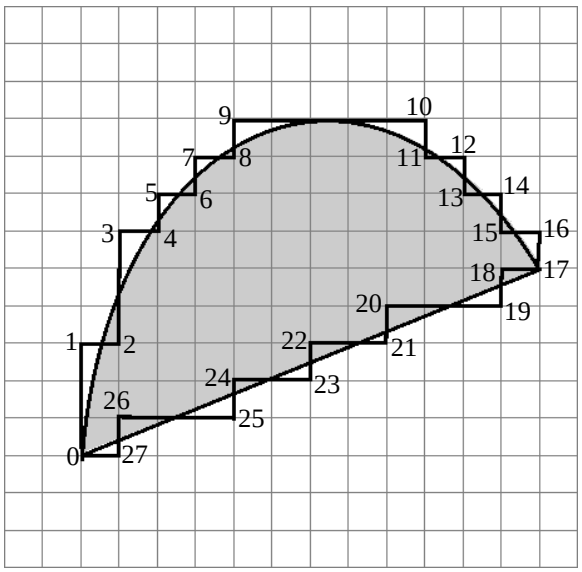

Рис. 1. Контур объекта и его цифровой эквивалент

Чтобы исключить возникающие из-за этого проблемы, будем рассматривать изображение, полученное из исходного в результате дискретизации, как двумерный клеточный комплекс [1]. В таком случае пикселы являются двумерными элементами этого клеточного комплекса. Помимо пикселов, имеются *крэки (crac)* и точки. Крэки – это стороны пикселов, являющиеся одномерными элементами. Точки есть конечными точками крэков и угловыми точками пикселов. Точки являются нольмерными элементами. Таким образом, в рассматриваемом случае контур объекта – это связная замкнутая последовательность *контурных* крэков, граничных между пикселами объекта и фоном. Контур может быть описан как последовательность целочисленных координат точек, ограничивающих контурные крэки. Как показано в [1], представление плоскости изображения как

клеточного комплекса дает много преимуществ. В частности, граница области становится тонкой кривой с нулевой площадью.

На рис. 1 приведены пример исходного контура объекта, образованного дугой кривой и отрезком прямой, а также его цифровой эквивалент как последовательность крэков. Пронумерованы точки, принадлежащие крэкам разных направлений. Как и в работах [2,3], под L-элементом будем понимать связную последовательность крэков одного и того же направления, выходящую из некоторой точки и заканчивающуюся крэком того же или перпендикулярного направления. На рис. 1 приведено одно из возможных разбиений контура на L-элементы, которые образованы крэками между точками: (0-2), (2-4), (4-6), (6-8), (8-9), (9-11), (11-13), (13-15), (15-17), (17-19), (20-21), (21-23), (23-25), (25-27), (27-0). Каждый L-элемент характеризуется такими параметрами: направлением относительно начальной его точки g (принято g=0 – для направления вверх, 1 – вправо, 2 – вниз, 3 – влево); *l* – количеством крэков направления g (*l* = 1,2,…); направлением последнего крэка *q* относительно направления g предыдущих крэков (q = – 1 – последний крэк направлен влево относительно направления g, +1 – вправо, 0 – совпадает с направлением g). Количество крэков *l* условно будем называть "длиной" L-элемента Для L-элемента (0-2) g=0, *l*=3, q=+1. Для L-элемента (27-0) g=3, *l*=1, q=0.

Метод выделения отрезков цифровых прямых в контуре использует следующее свойство последовательности L-элементов, образующих отрезок. Такая последовательность включает L-элементы с одинаковыми значениями g, q; их длины принимают значения *l*, *l*+1. Причем чередование L-элементов длин *l*, *l*+1 определяется цепной дробью, получаемой при делении целых чисел $n = \Delta x = |x_1 - x_2|$ и $m = \Delta y = |y_1 - y_2|$, где (x_1, y_1), (x_2, y_2) – координаты начальной и конечной точек отрезка: $[l; \kappa_1, \kappa_2, ..., k_t]$ или

$$\frac{n}{m} = l + \frac{r}{m} = l + \frac{1}{k_1 + \dfrac{r_1}{r}} = ... = l + \frac{1}{k_1 + \dfrac{1}{k_2 + \dfrac{r_2}{r_1}}} = ... = l + \frac{1}{k_1 + \dfrac{1}{k_2 + \dfrac{1}{k_3 + ... + \dfrac{1}{k_t}}}}. \qquad (1)$$

Положим для определенности, что $n > m$. Как следует из формулы (1), *l* – целая часть от деления n на m – соответствует в отрезке цифровой прямой количеству из *l* подряд идущих крэков одного направления. Вместе с примыкающим перпендикулярным крэком они образуют L-элемент длины *l*. κ_1 подряд идущих L-элементов длины *l* и один L-элемент длины *l*+1 (или κ_1 подряд идущих L-элементов длины *l*+1 и один L-элемент длины *l*) образуют K_1-элемент "длины" κ_1 (по аналогии с "длиной" L-элемента). L-элемент, отличающийся по длине на 1 от подряд идущих L-элементов, будем называть ***измененным*** L-элементом данного K_1-элемента. Аналогично, κ_2 подряд идущих K_1-элементов “длины” κ_1 и один К$_1$-элемент “длины” κ_1+1 (или κ_2 подряд идущих K_1-элементов "длины" κ_1+1 и один K_1-элемент “длины” κ_1) образуют К$_2$-элемент “длины” κ_1. И так

далее до исчерпания членов цепной дроби. K_1-элемент (вообще K_{t-1}-элемент), отличающийся по длине на 1 от подряд идущих K_1-элементов (K_{t-1}-элемент), будем называть ***измененным*** K_1-элементом (K_{t-1}-элементом) данного K_2-элемента (K_t-элемента). Таким образом, каждому цифровому отрезку прямой соответствует цепная дробь, элементы которой определяют структуру этого отрезка.

В контуре на рис. 1 могут быть выделены следующие отрезки цифровых прямых: 0-3, 3-9, 9-10, 10-17, 17-0.

3. Выделение отрезков цифровой прямой в контуре

При обработке контуров изображений, в частности, бинарных изображений, в последовательности крэков, образующей контур, необходимо выделить части последовательности, образующие отрезки прямых. Эту задачу можно рассматривать как задачу определения элементов цепной дроби по последовательности L-элементов контура. Данная задача является обратной по отношению к задаче определения структуры отрезка прямой по последовательности членов цепной дроби, получаемой как отношение разностей координат начала и конца отрезка.

Метод выделения отрезков цифровой прямой состоит в последовательном выполнении следующих действий.

1. Выделение последовательности L-элементов в последовательности крэков. Это действие соответствует определению целой части *l* цепной дроби (1).

2. Выделение последовательности K_t-элементов при $t=1$ в последовательности L-элементов, причем один из L-элементов каждого K_1-элемента должен содержать на 1 крэк больше или меньше других. Это действие соответствует определению κ_1-го элемента цепной дроби (1). После его выполнения значение t должно быть увеличено на 1.

3. Выделение последовательности K_t-элементов в последовательности K_{t-1}-элементов, причем один из K_{t-1}-элементов каждого K_t-элемента должен содержать на один K_{t-2}-элемент больше или меньше других. Это действие соответствует определению k_t-го элемента цепной дроби (1). После его выполнения значение t должно быть увеличено на 1.

4. Пункт 3 повторяется до тех пор, пока из идущих подряд K_t-элементов еще возможно образовать K_{t+1}-элемент.

5. Определяют граничные точки между двумя соседними L-элементами, которые не входят в один и тот же K_t-элемент. Эти точки являются конечными точками отрезков цифровых прямых, образующих контур.

Рассмотрим алгоритм выделения отрезков прямых в последовательности L-элементов контура.

Пусть $\{L_s(/_s, g_s, q_s)\}$; $s = 0.1,...,S$ – последовательность L-элементов, образующих контур; x_s, y_s – координаты начала s-го L-элемента; $\{x_j, y_j\}$; $j = s_i$; $i = 0,1,...,$ I; I $< S$ – множество точек излома контура. Точки излома определяют конечные точки отрезков прямых, которые образуют контур. Найти точки излома – значит, определить отрезки прямой, образующие контур.

Каждый рассматриваемый отрезок характеризуется K_t-элементом, а также цепной дробью $[/; k_1, \kappa_2, ..., k_t]$. В начальный момент распознавания отрезка прямой элементы соответствующей цепной дроби равны 0. Отрезок можно считать распознанным, если распознаны параметры K_t-элемента, включая его порядок t и значения элементов соответствующей цепной дроби.

1. Начальные условия.

Заданы последовательности $\{L_s(/_s, g_s, q_s)\}$ и $\{x_s, y_s\}$.

Необходимо найти координаты точек излома $\{x_j, y_j\}$.

$k_{0p} := 0$, $k_{1p} := 0$, $k_{2p} := 0,\ldots$, $k_{tp} := 0$ – рабочие значения элементов цепной дроби.

Примем в качестве начальной точки первого отрезка точку $s = 0$; $j = 0$; $t = 0$.

2. Принимают первый L-элемент в последовательности началом первого отрезка прямой. Начальная его точка x_s, y_s. Длина $l = l_0$ является также значением первого элемента цепной дроби. $s := s+1$; $k_{1p} := k_{1p}+1$.

3. Выполняют проверку для следующего L-элемента, образуют ли они вместе с предыдущими K_0-элемент.

3.1. Если $((g_s == g_{s-1}) \,\&\&\, (q_s == q_{s-1}) \,\&\&\, (l_s == l_{s-1}))$, то продолжение K_1-элемента $k_{0p} := k_{0p}+1$; $s := s+1$; и продолжение отрезка прямой. Переход к п.3.

3.2. Если $((g_s \neq g_{s-1}) \,||\, (q_s \neq q_{s-1}) \,||\, (|l_s - l_{s-1}| > 1))$, то конец отрезка прямой. Переход к п.5.

3.3. Если $((g_s == g_{s+1}) \,\&\&\, (q_s == q_{s+1}) \,\&\&\, ((l_s+1 == l_{s+1}) \,||\, (l_s - 1 == l_{s+1})))$, то завершение K_0-элемента; $t = t+1$.

4. Проверка продолжения/завершения K_t-элемента.

4.1. Если $(k_t == 0)$, то $k_t := k_{tp}$; $k_{tp} := 0$; $k_{t+1p} := k_{t+1p}+1$; $s := s+1$; начало K_{t+1}-элемента. Переход к п.3.

4.2. Если $((k_t \neq 0) \,\&\&\, (k_t == k_{tp}))$, то $k_{t+1p} := k_{t+1p}+1$; $s := s+1$; продолжение K_{t+1}-элемента. Переход к п.3.

4.3. Если $((k_t \neq 0) \,\&\&\, ((k_t+1 == k_{tp}) \,||\, (k_t-1 == k_{tp})))$, то $t := t+1$; конец K_{t+1}-элемента. Переход к п.4.

4.4. Если $((k_t \neq 0) \,\&\&\, (|k_t - k_{tp}| > 1))$, то конец отрезка прямой переход к п.5.

5. Конец отрезка.

$x_j = x_s$; $y_j = y_s$; $k_{1p} := 0$, $k_{2p} := 0,\ldots$, $k_{tp} := 0$; $k_1 := 0$, $k_2 := 0,\ldots$, $k_t := 0$.

Если ($s < S$), то $s := s + 1$; переход к п. 2.

Иначе – конец последовательности L-элементов. Конец работы алгоритма.

По сути, предлагаемый алгоритм находит элементы цепной дроби и для каждого полученного K_t-элемента и проверяет, является ли цепная дробь вновь построенного K_t-элемента подходящей к уже построенной.

4. Программа выделения отрезков цифровой прямой

Как видно из описания алгоритма, в нем содержится значительное количество условных переходов, применение которых противоречит рекомендациям структурного программирования из-за трудностей, возникающих при отладке программ. Кроме того, количество параметров κ_t заранее определить невозможно, поскольку переменная t заранее не ограничена. Ограничить величину t – значит заранее ограничить размеры изображения. Программная реализация, особенно отладка предложенного алгоритма тривиальными средствами по указанным причинам существенно затруднена. Уменьшить трудности разработки и отладки программной реализации можно, если использовать современные средства объектно-ориентированного программирования.

Предложенный алгоритм реализован в виде программы LINESEGM, которая входит в состав лабораторного программного комплекса по обработке изображений в среде Visual C++.

В качестве исходной информации программа LINESEGM использует последовательности L-элементов, построенные для каждого из контуров обрабатываемого изображения.

Результатом работы программы является построенная для каждого контура связная последовательность отрезков цифровых прямых, представленная координатами конечных точек отрезков.

Как видно из алгоритма, операции построения K_t-элементов из K_{t-1}-элементов одинаковы для всех значений t. Отметим, что начальное значение t=0 и в процессе работы алгоритма увеличивается каждый раз на 1. Специальный класс CKForLn включает методы, соответствующие операциям алгоритма. В процессе работы программы, реализующей алгоритм при каждом увеличении t на 1, создается новый объект, содержащий функции, выполняющие операции алгоритма для каждого значения t.

Учитывая, что на нулевом уровне K_0-элементы образуются не из K-элементов, а из L-элементов, для реализации алгоритма на нулевом уровне создана специальная модификация класса CKForLn – класс Cmini.

Принцип работы программы заключается в том, что для каждого значения t в программе реализуется объект класса CKForLn t–того уровня, содержащий функции, определяющие параметры K_t-элемента. Исходными параметрами K_t-элемента являются параметры уже завершенного K_{t-1}-элемента, параметры которого были определены объектом класса CKForLn t-1 – ого уровня.

Объекты класса CKForLn реализуются по мере возникновения условий, а именно: необходимости построения K-элемента очередного уровня, – и накапливаются в специальном динамическом массиве. Объект нулевого уровня создается сразу же в начале работы программы.

Реализация объектов в динамическом массиве по мере увеличения *t* позволяет не накладывать ограничений на размеры изображения. Ограничения на размер изображения определяются только ресурсами используемого компьютера.

При описании работы программы будет использовано понятие ***завершенного*** K_t-элемента. Каждый завершенный K_t-элемент содержит k_t K_{t-1}-элементов и один измененный K_{t-1}-элемент, который содержит $k_{t-1}\pm 1$ K_{t-2}-элементов, в отличие от незавершенного K_t-элемента, который не содержит незавершенного K_t-элемента.

Класс CKForLn включает следующие методы.

1. Метод *DK*(), (define K-element) – определить K-элемент.

Определить K_t-элемент – значит, определить количество K_{t-1}-элементов, образующих данный K_t-элемент.

2. Метод *VK*(), (verify K-element) – проверить идентичность рассматриваемого K-элемента с К-элементом того же уровня, определенным предварительно функцией метода *DK*() .

3. Метод *DE*(), (define the end of K-element) – определить завершение K-элемента, то есть при определении K_t-элемента найти его измененный K_{t-1}-элемент. Функция метода *DE*() уровня *t*–1 вызывается функцией метода *DK*() уровня *t.*

4. Метод *VE*(),(verify the end of K-element) – проверить идентичность завершения рассматриваемого K-элемента измененным K-элементом, определенным предварительно функцией метода *DE*().

Класс Cmini включает такие же методы, отличающиеся от методов класса CKForLn тем, что методы класса Cmini работают с L-элементами и определяют или проверяют K_0-элементы.

Методы класса Cmini

Методы класса Cmini используют в качестве исходных данных последовательности L-элементов, построенные для каждого из контуров обрабатываемого изображения, начиная с текущего номера L-элемента в момент вызова функции метода.

Метод *DK*()

Функция метода *DK*() класса Cmini сравнивает параметры каждого следующего L-элемента с параметрами начального L-элемента до тех пор, пока они совпадают. В случае несовпадения параметров функция *DK*() проверяет, завершен ли K_0-элемент и заканчивает работу. Завершенным считается K_0-элемент, заканчивающийся измененным L-элементом, таким, длина которого отличается от других L-элементов K_0-элемента на 1 (операция 3.1 для начала отрезка – *t* = 0).

Метод *VK*()

Функция метода *VK*() проверяет, совпадают ли параметры следующих k_0 L-элементов с параметрами L-элементов K_0-элемента, предварительно определенного функцией метода *DK*() того же уровня. В случае совпадения параметров текущего K_0-элемента с предварительно определенным функция *VK*() формирует признак продолжения отрезка и завершает работу (операция 3.1 для продолжения отрезка – $t > 0$).

В противном случае функция *VK*() формирует признак завершения отрезка и завершает работу.

Метод *DE*()

Функция метода *DE*() сравнивает параметры текущего K_0-элемента с параметрами K_0-элемента, предварительно определенного функцией *DK*(), чтобы определить, является ли текущий K_0-элемент измененным. При равенстве других параметров количество L-элементов k_0 измененного K_0-элемента сравнительно с K_0-элементом, предварительно определенным функцией *DK*(), должно отличаться на 1 (операция 3.2, 3.3 для определения завершения начального K_0-элемента отрезка – $t = 0$). Результат – параметры измененного K_0-элемента используются в методе *VE*() класса Cmini.

Метод *VE*()

Функция метода *VE*() сравнивает параметры текущего K_0-элемента с параметрами предварительно определенного функцией *DE*() измененного K_0-элемента, чтобы определить, совпадают ли они (операция 3.2, 3.3 для продолжения отрезка – $t > 0$). Результат – признак совпадения или несовпадения – используется в методе *VK*() класса CKForLn.

Методы класса CKForLn

Методы используют в качестве исходных данных параметры К-элементов, построенные для низшего уровня. То есть, для определения параметров K_t-элемента используются параметры уже построенных K_{t-1}-элементов.

Метод *DK*()

Функция метода *DK*() уровня *t* класса CKForLn при определении параметров K_t-элемента вызывает функцию *VK*() уровня *t*-1 класса CKForLn, которая проверяет, следует ли за уже определенным K_{t-1}-элементом K_{t-1}-элемент с такими же параметрами. Если да, вызов функции *VK*() повторяется. При этом происходит подсчет количества повторений, то есть определяется параметр k_t.

В противном случае функция *DK*() уровня *t* вызывает функцию *DE*() уровня *t*-1 для определения измененного K_{t-1}-элемента и заканчивает работу. По окончании работы функция *DK*() уровня *t* класса CKForLn определяет параметры и формирует признаки завершенного или незавершенного K_t-элемента (операция 4.1, 4.2 при текущем максимальном значении *t*).

Метод *VK*()

Функция метода *VK*() уровня *t* класса CKForLn проверяет, совпадают ли параметры следующих k_t K_t-элементов с параметрами K_t-элемента, предварительно определенного функцией метода *DK*() того же уровня. В случае совпадения параметров текущего K_t-элемента с предварительно определенным функцией *DK*() K_t-элементом того же уровня, функция *VK*() формирует признак продолжения отрезка и завершает работу.

В противном случае функция *VK*() формирует признак завершения отрезка и завершает работу (операция 4.1,4.2 при текущем значении *t*, меньшем максимального).

Метод *DE*()

K_t-элементФункция метода *DE*()уровня *t* класса CKForLn при определении параметров K_t-элемента сравнивает параметры текущего K_t-элемента с параметрами K_t-элемента, предварительно определенного функцией *DK*(), чтобы определить, является ли текущий K_t-элемент измененным. При равенстве других параметров их значения k_{t-1} должны отличаться на 1. В случае выполнения этого условия функция *DE*() формирует признак измененного K_t-элемента и завершает работу (операция 4.3, 4.4 при текущем максимальном значении *t*).

Метод *VE*()

Функция метода *VE*() уровня *t* класса CKForLn сравнивает параметры текущего K_t-элемента с параметрами предварительно выделенного функцией *DE*() измененного K_t-элемента, чтобы определить, совпадают ли значения их параметров.

В случае совпадения значений параметров текущего K_t-элемента с предварительно определенным функцией *DK*() того же уровня, функция *VK*() формирует признак совпадения значений параметров и завершает работу (операция 4.3,4.4 при текущем значении *t*, меньшем максимального).

Временная диаграмма (рис. 2) иллюстрирует работу программы LINESEGM на примере распознавания отрезка прямой. В нижней части рисунка изображена часть цифровой линии, состоящая из L-элементов одних и тех же основного и вспомогательного направлений и различных длин.

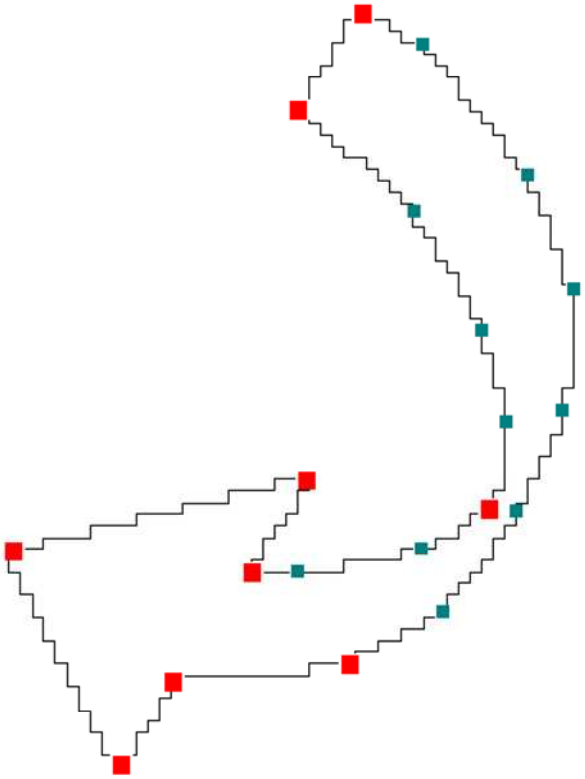

Рис. 3. Пример работы программы структурного анализа контура – сегментация контура отрезками цифровых прямых

На шаге 0 создан объект класса Cmini, который определяет параметры K_0-элемента.

На шаге 10 завершено определение параметров K_0-элемента и создается объект 1 класса CKForLn, который использует функции ранее созданного объекта для определения параметров K_1-элемента. На шаге 19 завершено определение параметров K_1-элемента и создается объект 2 класса CKForLn, который использует функции ранее созданных объектов для определения параметров K_2-элемента. На шаге 49 завершено определение параметров K_2-элемента и создается объект 3 класса CKForLn, который использует функции ранее созданных объектов для определения параметров K_3-элемента. На шаге 79 выполняется условие завершения отрезка. Подробно работа программы описана в приложении.

На участке 0-6 два L-элемента образуют незавершенный K_0-элемент. Очевидно, что L-элемент 3-6 длины 3 завершает отрезок прямой, так как L-элемент 6-7 длины 1 не может быть его продолжением. Таким образом, L-элемент 6-7 является началом отрезка цифровой прямой.

На рис. 3 представлен пример работы программы. Контур бинарного изображения фигурной стрелки разделен квадратиками на отрезки прямых. Результат работы программы – последовательность отрезков прямых – был использован для выделения дуг цифровых кривых [4]. Большие квадратики показывают границы дуг цифровых кривых.

Работа программы проверена на значительном количестве (более 2000) примеров и используется при исследовании структурного анализа полутоновых изображений.

5. Работа программы распознавания отрезков прямых

Рассмотрим работу программы LINESEGM на примере рис. 4. В нижней части рисунка изображена часть цифровой линии, состоящая из L-элементов одних и тех же основного и вспомогательного направлений и различных длин. На участке 0-6 два L-элемента образуют незавершенный K_0-

элемент. Очевидно, что L-элемент 3-6 длины 3 завершает отрезок прямой, так как L-элемент 6-7 длины 1 не может быть его продолжением. Таким образом, L-элемент 6-7 является началом отрезка цифровой прямой.

Работу программы по определению очередного отрезка прямой начинает функция *DK*() нулевого уровня, которая определяет завершенный K_0-элемент 6-10, состоящий из L-элементов длин 1,1,2; k_0=2. Этот K_0-элемент является начальным для K_1-элемента. Программа образовывает объект первого уровня и передает управление функции *DK*() этого объекта. Функция *DK*() уровня 1 вызывает функцию *VK*() уровня 0. Функция *VK*() сравнивает параметры L-элементов K_0-элемента 6-10 с последующими L-элементами и подтверждает наличие K_0-элемента 10-14, идентичного K_0-элементу 6-10. Продолжая работу, функция *VK*() обнаруживает, что следующие L-элементы не образуют такого же K_0-элемента, завершает работу и передает управление функции *DK*() уровня 1. Функция *DK*() уровня 1 вызывает функцию *DE*() уровня 0. Эта последняя, начиная с L-элемента 6-7, определяет наличие измененного K_0-элемента 14-19, состоящего из L-элементов длин 1,1,1,2; k_0=3, завершает работу и передает управление функции *DK*() уровня 1. Эта функция определяет наличие завершенного K_1-элемента 6-19, состоящего из двух K_0-элементов 1,1,2, (k_1=2) и одного измененного 1,1,1,2 (k_0=3). Программа образовывает объект второго уровня и передает управление функции *DK*() этого объекта. Функция *DK*() уровня 2 вызывает функцию *VK*() уровня 1, которая, в свою очередь, вызывает функцию *VK*() уровня 0. Функция *VK*() сравнивает параметры L-элементов K_0-элемента 6-10 с последующими L-элементами и подтверждает наличие K_0-элементов 19-23, 23-27, идентичных K_0-элементу 6-10, то есть столько же, сколько таких K_0-элементов содержится в K_1-элементе 6-19. Далее функция *VK*() уровня 0 возвращает управление с признаком продолжения отрезка функции *VK*() уровня 1. Функция *VK*() вызывает функцию *VE*() уровня 0, которая определяет наличие измененного K_0-элемента 27-32, идентичного K_0-элементу 14-19. Таким образом, определен K_1-элемент 19-32, идентичный K_1-элементу 6-19. Далее функция *VK*() уровня 1 не определяет очередной K_1-элемент, идентичный K_1-элементу 6-19, из-за того, что функция *VE*() уровня 0 не определяет измененный K_1-элемент, идентичный K_1-элементу 6-19, начиная с L-элемента 40-41, и возвращает управление функции *DK*() уровня 2. Функция *DK*() уровня 2 вызывает функцию *DE*() уровня 1, которая определяет наличие измененного K_1-элемента 32-49, состоящего из K_0-элементов 32-36, 36-40, 40-44, 44-49. Далее определяется K_2-элемент 6-49, образуется объект уровня 3, определяется измененный K_2-элемент 49-79. Эти два K_2-элемента образуют K_3-элемент 6-79. Этим построение отрезка завершается, поскольку следующие L-элементы 79-81 и 81-83 не образуют K_0-элемент, идентичный K_0-элементу 6-10, и функция *VK*() уровня 0 не формирует признак продолжения

отрезка. В последовательности L-элементов выделен отрезок цифровой прямой 6-79. Программа начинает определение следующего отрезка, начиная с L-элемента 80-82.

6. Выводы

1. Предложен новый алгоритм выделения отрезков прямой в контурах изображений и нетривиальная программная реализация алгоритма, которые позволяют получить точное решение задачи распознавания контуров изображений как последовательностей отрезков прямых.
2. Программная реализация алгоритма выделения отрезков прямой в контурах изображений выполнена с применением современных средств объектно-ориентированного программирования, что позволило не накладывать явных ограничений на размеры обрабатываемого изображения при максимальном использовании ресурсов применяемого компьютера.
3. На основе предложенного алгоритма и его программной реализации получено теоретическое решение и проведены эксперименты по распознаванию дуг цифровых кривых и сегментации контура бинарных изображений на отрезке цифровых прямых и дуги цифровых кривых.

СПИСОК ЛИТЕРАТУРЫ

1. Kovalevsky V.A. Applications of Digital Straight Segments to Economical Image Encoding, In Proceedings of the 7th International Workshop, DGCI'97, Montpellier. – France, 1997. – December 3-5. – P. 51–62.
2. Калмыков В.Г. Структурный метод описания и распознавания отрезков цифровых прямых в контурах бинарных изображений // Штучний інтелект. – 2002. – № 4. – С. 450–457.
3. Калмыков В.Г., Вишневский В.В. Анализ контуров объектов в бинарных изображениях // Математические машины и системы. – 1997. – № 2. – С. 68 – 71.
4. Калмиков В.Г. Дуга цифрової кривої – визначення і застосування // Оброблення сигналів і зображень та розпізнавання образів. Праці сьомої Всеукраїнської міжнародної конференції. – Київ. – 2004. – 11 – 15 жовтень. – С. 205–208.