ИНФОРМАТИКА

2006 **январь-март** **№ 1**

ОБРАБОТКА СИГНАЛОВ, ИЗОБРАЖЕНИЙ И РЕЧИ

УДК 681.3

В.В. Буча, С.В. Абламейко

АЛГОРИТМ ИНТЕРАКТИВНОЙ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ, ОСНОВАННЫЙ НА МЕТОДЕ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Предлагается интерактивный алгоритм сегментации изображений, основанный на методе динамического программирования и быстром поиске в хеш-таблице. Данный алгоритм отличается от существующих повышенной скоростью, устойчивостью работы, возможностью выделения как площадных объектов с определением их контура, так и линейных с выделением их скелетного представления. При тестировании на аэрокосмических снимках алгоритм обеспечил более удобное, быстрое и точное выделение картографических объектов по сравнению с другими ручными или автоматическими методами.

Введение

В последние годы многие исследователи сконцентрировали свое внимание на методах автоматической сегментации [1]. Реализация полностью автоматического метода сегментации изображений до сих пор является неразрешенной проблемой, поэтому существует практическая необходимость в сегментации с помощью интерактивных методов, которые успешно применяются в различных приложениях обработки и анализа изображений [2−5]. В отличие от автоматических методов интерактивная сегментация активно использует знания пользователя об обрабатываемом объекте для выделения его границы. Выделяют два вида интерактивных технологий сегментации. Первая технология основывается на выделении регионов изображения, вторая − на определении границ выделяемого объекта.

Типичным примером подхода выделения регионов является инструмент magic-wand [6], реализованный в большинстве коммерческих систем обработки изображений. Данный инструмент позволяет пользователю выбрать затравочную точку, к которой добавляются соседние точки, удовлетворяющие некоторому критерию подобия. Результирующий регион часто подвергается дополнительной обработке с помощью других инструментов сегментации, так как данный метод не обеспечивает достаточной обратной связи с пользователем для правильного и полного выделения объекта.

Технология активных контуров, или снейков [7], является наиболее популярной и хорошо проработанной технологией сегментации второго типа. Данный подход требует от пользователя ввода кривой, которая будет являться первоначальной аппроксимацией выделяемого контура. Заданная кривая «стягивается» и обхватывает объект в процессе минимизации энергии кривой. Однако данным процессом тяжело управлять интерактивно. Поэтому если результаты выделения неудовлетворительны, необходимо задать новую кривую или обработать полученный контур дополнительными средствами.

Интерактивные методы сегментации типа live-wire также относятся ко второму типу и являются яркими представителями своего класса. Данные методы сочетают в себе удобное управление процессом сегментации, высокие точность результатов и скорость сегментации. Типичными представителями технологии live-wire являются методы сегментации intelligent scissors [2] и live-wire [3], которые рассматривают изображение в виде ориентированного графа и используют алгоритмы поиска глобального оптимального пути для нахождения границы выделяемого объекта.

Основанные на технологии live-wire подходы обладают существенным недостатком – скорость работы алгоритмов существенно замедляется при незначительном увеличении размеров изображения. Для ускорения работы алгоритмов был предложен ряд схем, например live-lane [3] и enhanced lane [4], ограничивающих область поиска маленьким окном, в котором

ищется локальный оптимальный путь. Набор таких путей, получаемых в процессе интерактивной сегментации, задает результирующий сегментирующий контур. Методы live-lane жертвуют качеством сегментации для получения достаточного для комфортной работы оператора времени отклика алгоритма.

Предлагаемый алгоритм сегментации является дальнейшим развитием работы [5], в которой предлагается эффективная по времени реализация парадигмы live-wire, сохраняющая концепцию поиска глобального оптимального пути. Для этого поиска предлагается использовать модифицированный алгоритм Дейкстры [8] с добавлением эвристических правил, позволяющих находить оптимальный путь с меньшими временными затратами.

Для реализации быстрого алгоритма поиска пути с минимальной стоимостью между двумя произвольными узлами ориентированного графа предлагается использовать эффективную схему организации внутренних структур хранения графа и быстрого поиска на базе технологии хеширования, или хеш-таблицы [9]. Данная схема обеспечивает сравнительно постоянное время вставки, удаления и поиска узла графа с минимальной стоимостью в сортированном списке, который является неотъемлемой частью используемого алгоритма Дейкстры.

В отличие от всех похожих технологий интерактивной сегментации, предлагаемый подход позволяет выделять как площадные объекты, определяя их контур, так и линейные, выделяя их скелетное представление. Для решения последней задачи используется процедура назначения весов, основанная на пиксельном силовом преобразовании (ПСП) [10], которое выступает в качестве аналога дистанционного преобразования для полутоновых и цветных изображений. Алгоритм был протестирован на задаче выделения картографических объектов на аэрокосмических снимках. Было получено более удобное, быстрое и точное выделение объектов по сравнению с другими ручными или автоматическими методами.

1. Предлагаемый подход

В работах [2, 3] были предложены интерактивные методы сегментации intelligent scissors и live-wire, основанные на глобальном поиске оптимального пути на ориентированном графе. Под оптимальным путем понимается путь с наименьшей стоимостью. Изображение или часть изображения рассматривается как ориентированный граф, в котором пиксели соответствуют узлам графа, а связи с соседями соответствуют дугам графа. Каждой дуге графа назначается цена или весовой коэффициент, который вычисляется с помощью ряда свойств, характеризующих принадлежность пикселя к границе объекта. В работах [2–4] авторы используют следующие характеристики для вычисления локальной цены связи: модуль и направление градиента, интенсивность пикселя, значение второй производной и другие характеристики, в том числе вычисляемые в процессе тренировки и обучения алгоритма.

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	(2)	5	9
12	4	2	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

Рис. 1. Матрица локальных весов

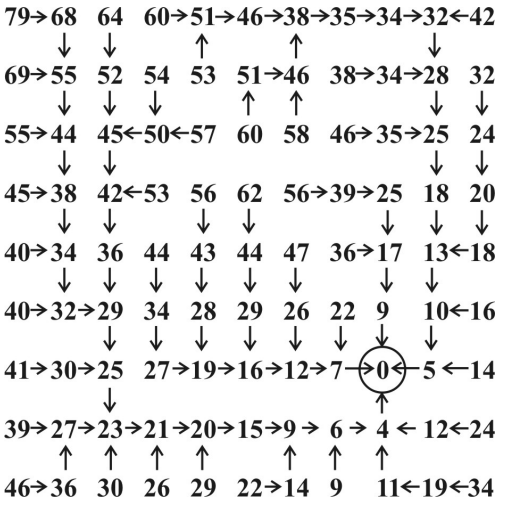


Рис. 2. Матрица накопленных весов и карта направлений

На рис. 1 изображена локальная матрица весов взвешенного графа, характеризующая локальные свойства пикселей изображения. После указания оператором затравочной точки (на рис. 1, 2 выделена кружком) вычисляется карта направлений (рис. 2), с помощью которой строится оптимальный путь от любой точки изображения до затравочной. Каждое число на рис. 2 соответствует стоимости пути от данной точки до затравочной. Для вычисления карты направлений, задающей оптимальный путь, использовался поиск в четырехсвязной области. Оптимальный путь вычисляется и отображается в реальном времени на дисплее от текущего положения курсора до затравочной точки. Если изображение сильно зашумлено или содержит объекты сложной формы, то может понадобиться несколько граничных сегментов для задания сегментирующего контура. Если полученный сегмент адекватно описывает часть границы объекта, то указывается новая затравочная точка для последующего выделяемого граничного сегмента.

Данная технология является интерактивной технологией сегментации реального времени, позволяющей точно определять сложные контуры на цветных и полутоновых изображениях (рис. 3) [4], однако вычисление карты направлений для всех точек изображения является крайне ресурсоемкой операцией, особенно когда размер изображения большой. Время отклика алгоритма после указания пользователем затравочной точки может измеряться десятками секунд, что неприемлемо для метода интерактивной сегментации.

Рис. 3. Выделение граничного сегмента с помощью технологии live-wire на аэрокосмическом изображении

Основные временные издержки скрываются в алгоритме вычисления карты направлений. В работах [2–5] используется алгоритм Дейкстры [8], основанный на методе динамического программирования. На каждом шаге он ищет необработанные узлы, близкие к стартовому, затем просматривает соседей текущего узла и устанавливает или обновляет их соответствующие расстояния от старта. Для понимания предлагаемого подхода приведем алгоритм Дейкстры и обозначим наиболее проблемные места и причины падения производительности алгоритма.

Алгоритм Дейкстры

Вход: w(p,q) – функция, задающая цену перехода от узла графа p к q; p_s – начальный узел; $N(p)$ – функция, возвращающая соседей узла p.

Выход: карта направлений *MD*.

Промежуточные структуры: AL – активный список узлов; E – массив обработанных узлов; S – массив суммарной стоимости; S_{tmp} – промежуточное значение стоимости.

1. $S(p_i)=\infty$ //*Установка максимальной суммарной стоимости для всех узлов графа*
2. $S(p_s)=0$; $AL \leftarrow p_s$ //*Установка минимальной стоимости для* p_s *и сохранение в списке*
3. While $AL \neq \varnothing$ do begin //*Пока активный список не пустой*

```
4.      p←min(AL)                    //Извлечение узла с минимальной стоимостью S(p_i)
5.      E(p)=TRUE                    //Пометка узла p как обработанного
6.      For each ( q ∈ N( p) AND NOT E(q)) begin   //Перебор всех необработанных соседей p
7.              S_tmp=S(p)+w(p,q)                 //Вычисление суммарной стоимости
8.              if S_tmp<S(q) begin               //Если найден более дешевый путь
9.              S(q)=S_tmp                        //Обновление стоимости для узла q
10.             MD (q)=p                          //Установка ссылки на узел p
11.                     if q ∉ AL  begin          //Если узел не в списке
12.                     AL←q;                     //Сохранение узла в списке
                        endif
                endif
        endfor
    endwhile
```

Как показывает практика, наиболее проблемными местами алгоритма являются шаги 4, 9, 12, выполняющие операции с активным списком узлов графа. Сложность состоит в поддержании данного списка в отсортированном состоянии и выполнении операций вставки, удаления и извлечения узла с минимальной суммарной стоимостью из списка с постоянной сложностью. Для быстрого доступа к элементам активного списка предлагается использовать технологию хеширования [9] с элегантным решением коллизий, которое основывается на доказываемых в последующих разделах утверждениях.

В классическом алгоритме глобального поиска оптимального пути карта направлений вычитается для всех узлов изображения. Основываясь на свойствах ориентированных графов и алгоритма Дейкстры, можно ограничить область поиска узлами, у которых суммарная стоимость меньше стоимости узла p_e, соответствующего текущему положению курсора [5]. Данное нововведение позволит значительно повысить производительность алгоритма (более чем в 1,3−31 раз быстрее [5], чем live-wire) для больших изображений.

Еще одним существенным нововведением является использование эвристического приближения наилучшего пути, которое позволяет алгоритму быстрее (в 2−15 раз) находить оптимальный путь. Вводимое эвристическое правило позволяет рассматривать в первую очередь узлы с маленькой суммарной стоимостью, которые находятся ближе к конечному узлу p_e. Таким образом фронт просмотра узлов графа становится более сфокусированным, значительно сужая область поиска.

Все ранее предлагаемые технологии интерактивной сегментации, основанные на парадигмах live-wire и live-lane, использовались для выделения площадных объектов. В настоящей работе впервые предлагается использовать данную технологию для непосредственного выделения и получения скелета линейных объектов. Для назначения весов графа применяются дистанционное преобразование [11] для бинарных изображений и предложенное в работе [10] ПСП для цветных и полутоновых изображений.

2. Эффективная реализация активного списка

Основная идея алгоритма Дейкстры сводится к извлечению из активного списка *AL* вершины графа с минимальной суммарной стоимостью. Эффективная реализация структуры хранения и доступа к элементам активного списка является первоочередной задачей, так как даже эффективный алгоритм может не дать ожидаемого эффекта производительности из-за неправильно выбранного контейнера активного списка.

Рассмотрим простейший случай, когда в качестве структуры данных *AL* используется динамический массив. Если узлы в этом массиве не будут упорядочены, то операция добавления узла может выполняться с постоянной сложностью $O(1)$. Однако операции удаления и извлечения узла с минимальной стоимостью не могут обеспечить постоянное время выполнения. Так, в худшем случае необходимо просматривать весь массив для поиска необходимого элемента и сложность этих операций будет как минимум линейной $O(N)$.

Если поддерживать массив в сортированном состоянии, то операция извлечения минимального узла будет постоянной $O(1)$, а вставка и удаление могут быть выполнены за $O(lg(N)+1)$ операций, где N − число записей в массиве. Данный вариант мог бы быть неплохим решением, однако возникают накладные расходы при вставке и удалении элемента в произвольное место массива. Накладные расходы связаны с перераспределением памяти для поддержания произвольного доступа в динамическом массиве, и время выполнения данной операции прямо пропорционально количеству элементов в массиве, поэтому сложность операций вставки и удаления будет также линейной $O(N)$.

Перераспределений памяти можно избежать, если использовать в качестве контейнера двухсвязный или односвязный список. Операции вставки, удаления и извлечения минимального узла могут быть выполнены с постоянной скоростью $O(1)$, однако при такой структуре поиск позиции для вставки/удаления элемента будет выполняться с линейной сложностью $O(N)$, так как данный тип контейнера не поддерживает произвольный доступ и для поиска необходимо перебрать все элементы от конца/начала списка до искомого.

Первым приемлемым решением является использование бинарного дерева в качестве контейнера активного списка, которое позволит обеспечить логарифмическую сложность выполняемых операций $O(\log N)$. Бинарное дерево не всегда может гарантировать устойчивое время выполнения операций с элементами, так при операциях вставки/удаления может получаться вырожденное дерево. Логично было бы использовать алгоритм поддержания бинарного дерева в некотором оптимальном состоянии. Эффективное решение было предложено советскими математиками Г.М. Андельсон-Вельским и Е.М. Ландисом. Данный метод, предоставляющий преимущества устойчивого выполнения операций с элементами дерева со сложностью $O(\log N)$, называется методом сбалансированных деревьев, или AVL-деревьев [9]. Таким образом, постоянная сложность $O(1)$ для операции извлечения минимального узла графа и логарифмическая сложность $O(\log N)$ для вставки/удаления элемента достигаются с использованием AVL-деревьев.

Для дальнейшего улучшения быстродействия операций с элементами активного списка необходимо рассмотреть основные свойства ориентированного графа и алгоритма Дейкстры и доказать утверждение о диапазоне значений суммарной стоимости узлов, находящихся в активном списке в произвольный момент времени.

Взвешенный граф G может быть представлен в виде пары (A, w), где A − ограниченная двухмерная матрица весов графа; $w(p):A\rightarrow[L, H]$ − функция, задающая вес узла p в A, значение которого лежит в интервале $[L, H]$. Ориентированный граф может быть получен из матрицы весов графа присвоением всем дугам, исходящим из узла p, веса $w(p)$ (рис. 4). Отметим, что вес дуги графа $w(p_i,q_i)$ не равняется весу $w(q_j,p_i)$.

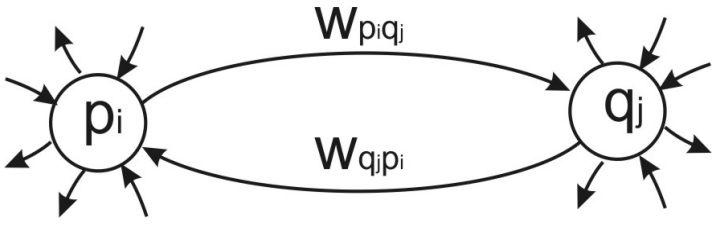


Рис. 4. Модель ориентированного графа

Утверждение 1. *Значения суммарной стоимости для всех узлов, принадлежащих активному списку алгоритма Дейкстры, лежат в следующем диапазоне:*

$$\min(S(p)) \le S(p_j) \le \min_{p\in Al}(S(p)) + H,\ \forall p_j \in AL,\ j \in [0, N], \tag{1}$$

где $\min(S(p))$ *− минимальное значение суммарной стоимости для всех узлов, принадлежащих активному списку; N − количество узлов в активном списке; H − максимальное значение, принимаемое w(p).*

Воспользуемся методом математической индукции. Для t=1, когда в активном списке находится один элемент, утверждение свойства очевидно. Докажем выполнение выражения (1) для произвольного времени t. На шаге 4 алгоритма Дейкстры в произвольный момент времени t происходит извлечение узла с минимальной суммарной стоимостью $\min_t = \min(S(p)), p \in AL_t$. Так как на шаге 7 стоимость может только увеличиться, то в активный список добавятся узлы q_i со стоимостью, большей или равной стоимости текущего узла p. На следующей итерации t+1 из активного списка будет извлечен минимум:

$$\min_{t+1} \geq \min_t . \tag{2}$$

Все элементы активного списка в момент времени t подчиняются следующему неравенству, которое следует из шагов 4 и 7:

$$\min_{t-1} + L \leq \underset{j=[0,N]}{S(p_j) \in AL_t} \leq \min_{t-1} + H. \tag{3}$$

Учитывая неравенство (2) и тот факт, что $\min_{t-1}$ был удален из списка на предыдущей итерации, неравенство (3) примет вид

$$\min_t + L \leq \underset{\forall p_j \in AL_t, ; j=[0,N]}{S(p_j)} \leq \min_t + H,$$

что является эквивалентом формулы (1) при L=0. Утверждение 1 доказано.

Утверждение 1 может быть использовано для эффективного решения коллизий в предлагаемой схеме хеширования активного списка, которая обеспечивает среднее время $O(1)$ и наихудшее время $O(c)$ для операций вставки, удаления и извлечения минимального значения.

В технологии хеширования используется хеш-таблица, состоящая из набора списков, которые хранят элементы массива (рис. 5), знак # означает отсутствие дальнейших элементов. Адресация в хеш-таблице задается специальной хеш-функцией $h(K)$, определяющей индекс списка, в котором хранится элемент. Хеш-функция $h(K)$ должна удовлетворять двум требованиям: быть быстрой и порождать хорошие ключи для распределения элементов по таблице в диапазоне $0 \leq h(K) \leq M$. Последнее требование минимизирует коллизии (случаи, когда два разных элемента имеют одинаковое значение хеш-функции) и предотвращает случай, когда элементы данных с близкими значениями попадают только в одну часть таблицы.

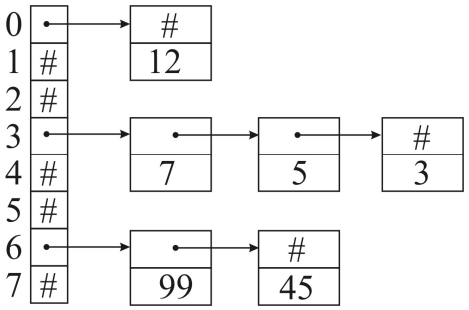


Рис. 5. Пример хеш-таблицы

Многочисленные тесты показали хорошую работу двух основных типов хеш-функций, один из которых основан на делении, другой – на умножении [9]. Метод деления весьма прост: используется остаток от деления на M, а именно

$$h(K) = K \bmod M.$$

Если *H* является дискретной величиной, то можно организовать хеш-таблицу с *H* списками. Причем, принимая во внимание утверждение 1, можно констатировать, что коллизий не возникнет в любой момент времени выполнения алгоритма. Если веса графа и суммарная стоимость представлены вещественными числами и происходит существенная потеря точности при их переводе в целочисленную систему, предлагаемая схема хеширования не может обеспечить отсутствие коллизий и, соответственно, правильную работу алгоритма. В таком случае необходимо использовать методы решения коллизий в хеш-таблицах [9]. Использование данных методов приводит к потере производительности метода хеширования. Поэтому применение AVL-деревьев с логарифмическим временем выполнения операций с элементами окажется более эффективным.

Опишем функции вставки, модифицирования и извлечения узла с минимальным значением суммарной стоимости. Заметим, что если *H* (авторы полагают *H*=256 или *H*=4096) − дискретное значение, полученное возведением числа 2 в степень *n*, операция *mod* может быть заменена на более быструю побитовую операцию *AND*:

$$h(K) = K \ \ AND \ (H-1).$$

Операция вставки элемента AL←q (шаг 12 алгоритма Дейкстры)

Вход: q – вставляемый элемент; *AL* − хеш-таблица; *S*(*q*) − функция суммарной стоимости *q*.
Промежуточные данные: h − индекс списка в таблице.

1. *h*= *S*(*q*) *AND* (*H*−1) //Расчет хеш-индекса
2. *AL*[*h*] ←*q* //Сохранение в списке *h* элемента *q*

Время выполнения этой операции постоянно *O*(1). Основываясь на утверждении 1, реализуем алгоритм извлечения узла из хеш-таблицы с минимальным значением стоимости. Будем исходить из того факта, что на этапе *t* известно положение узла со значением суммарной стоимости $\min_{t-1} = \min(S(p)), p \in AL_{t-1}$. Узел со стоимостью $\min_{t-1}$ был обработан на предыдущей итерации и находился в списке с номером $h_{\min(t-1)} = \min_{t-1} AND(H-1)$.

Если делать обход хеш-таблицы в сторону увеличения индекса $h_{\min(t-1)}$, а при $h_{\min(t-1)} = H$ совершать переход на индекс 0, то будут выбираться узлы с минимальной стоимостью, так как $h_{\min(t)} = \min_t AND(H-1) \geq h_{\min(t-1)}$ (при выполнении правила обхода). Заметим также, что в каждом списке хеш-таблицы хранятся элементы с равной суммарной стоимостью и порядок извлечения из списка не имеет значения. Тогда можно предложить следующий алгоритм с постоянным временем выполнения O(1) для извлечения элемента с минимальным значением стоимости.

Операция извлечения элемента с минимальным значением стоимости (шаг 4 алгоритма Дейкстры)

Вход: $\min_{t-1}$ − значение предыдущего извлеченного минимума; *AL* − хеш-таблица активного списка; *S*(*q*) − функция суммарной стоимости *q*.
Промежуточные данные:h − индекс списка в хеш-таблице; s − промежуточное значение стоимости.
Выход: $\min_t$ − узел с минимальной стоимостью.

1. *s*= $\min_{t-1}$
2. *h*= *s AND* (*H*−1) //Расчет хеш-индекса
3. while ($AL[h] \neq \varnothing$) begin //До тех пор, пока не встретим непустой список
4. *s*=*s*+1 //Увеличение стоимости на 1
5. *h*= *s AND* (*H*-1) //Расчет хеш-индекса
6. endwhile

7. $\min_t \leftarrow AL[h]$ //Извлечение любого элемента из списка

Последняя операция связана с изменением стоимости на шаге 9 алгоритма Дейкстры. Так как на данном шаге изменяется стоимость узла, которая используется для вычисления хеш-индекса, необходимо просто удалить элемент из хеш-таблицы и добавить его вновь с новой стоимостью. Процедура удаления становится не нужна, если используется функция *E(p)*, которая показывает, был обработан узел или нет. В этом случае обновленный узел будет обработан и помечен как обработанный раньше необновленного узла. Когда очередь дойдет до необновленного узла, он уже будет помечен как обработанный и алгоритм его проигнорирует. Таким образом, данная операция имеет постоянную сложность и время ее выполнения равно времени выполнения алгоритма вставки. Постоянное время выполнения операций с элементами активного списка позволяет реализовать алгоритм поиска оптимального пути с минимальными временными затратами.

3. Улучшенный алгоритм поиска оптимального пути

Следующие изменения коснутся структурных изменений алгоритма Дейкстры, которые прореживают дерево решений и фокусируют фронт исследования узлов в наиболее вероятном направлении. Основываясь на утверждении 1, доказанном в предыдущем разделе, можно вывести следующее утверждение 2, которое используется в работе [5].

Утверждение 2. *Алгоритм Дейкстры формирует оптимальный глобальный путь от затравочного узла p_s до конечного p_e на временном этапе t, когда будет выполняться следующее условие* $S_t(p_e) < \min(S(p)), p \in AL_t$, *где* $S_t(p_e)$ − *значение стоимости узла* p_e *на итерации t;* $\min(S(p)), p \in AL_t$ − *минимальное значение стоимости узла, принадлежащего активному списку на итерации t.*

Доказательство утверждения 2 следует из утверждения 1, так как в активном списке при выполнении условия $S_t(p_e) < \min(S(p)), p \in AL_t$ будут находиться узлы с большей стоимостью и стоимость последующих узлов будет только увеличиваться с увеличением *t*. Поэтому путь с меньшей стоимостью не может быть найден для узла p_e на следующих за *t* итерациях. Утверждение 2 может быть использовано для ограничения числа итераций в процессе поиска, что позволяет находить оптимальный путь до точки p_e быстрее в 1,3−31 раз [5].

В активном списке узлы отсортированы по стоимости, однако при одинаковой стоимости они могут находиться на разном геометрическом расстоянии до узла p_e. Существует большая вероятность нахождения оптимального пути, если начать исследования с узла, который находится геометрически ближе к p_e. Этот факт используется для задания эвристической функции $f(n) = S(n) + d(n)$, которая вычисляет приближение стоимости пути к цели p_e от узла *n*, где $S(n)$ − суммарная стоимость узла *n*, $d(n)$ − эвристическое приближение.

Данная идея была предложена в работе [12] и реализована в алгоритме, известном как Nilson A*. Он гарантированно находит кратчайший путь до тех пор, пока эвристическое приближение $d(n)$ является допустимым, т. е. он никогда не превышает действительного оставшегося расстояния до цели.

Функция эвристического приближения $d(n)$ обычно равняется минимальному расстоянию между текущим узлом и целевым, умноженному на минимальную стоимость передвижения между узлами, что гарантирует допустимость эвристики $d(n)$. В настоящей работе предлагается улучшенная эвристическая функция, которая использует среднюю цену единичного перехода, позволяя более точно оценивать стоимость:

$$d(n) = dist(p_e, n) \cdot C(n),$$

где $dist(p_e, n)$ − расстояние от узла p_e до *n* на изображении; $C(p)$ − средняя цена единичного перехода между узлами на пути от затравочного узла p_s до узла *n*.

Однако такая функция не может быть напрямую использована с предлагаемой схемой хеширования, так как нарушается утверждение 1. Если реализовать метод решения коллизий при нарушенном утверждении 1, то эффективность хеширования падает и использование AVL деревьев является более предпочтительным решением. Для соблюдения утверждения 1 предлагается использовать функцию $O(d(n))$, которая отображает значения функции $d(n)$ в диапазон значений $[0, c_1]$, где c_1 − некоторое дискретное значение, причем $0 < c_1$.Чтобы использовать предлагаемую эффективную схему хеширования без изменений, необходимо положить $M = H + c_1$ и в качестве K применять значение функции $f(n) = S(n) + O(d(n))$. Функция отображения может быть записана следующим образом:

$$O(d(n)) = \begin{cases} c1, \min_{d(n)} = \max_{d(n)}; \\ 0, d(n) \le \min_{d(n)}; \\ c1, d(n) \ge \max_{d(n)}; \\ \dfrac{(d(n) - \min_{d(n)})}{\max_{d(n)} - \min_{d(n)}} c1, \min_{d(n)} < d(n) < \max_{d(n)}, \end{cases}$$

где $\min_{d(n)}$ и $\max_{d(n)}$ − минимальное и максимальное значения функции $d(n)$ для узлов, принадлежащих активному списку соответственно. С учетом изменений предлагаемый алгоритм поиска оптимального пути будет выглядеть следующим образом.

Улучшенный алгоритм поиска оптимального пути

Вход: w(p,q) − функция, задающая цену перехода от узла графа *p* к *q*; p_s − начальный узел; p_e − конечный узел; *N*(*p*) – функция, возвращающая соседей узла *p*.

Выход: карта направлений *MD.*

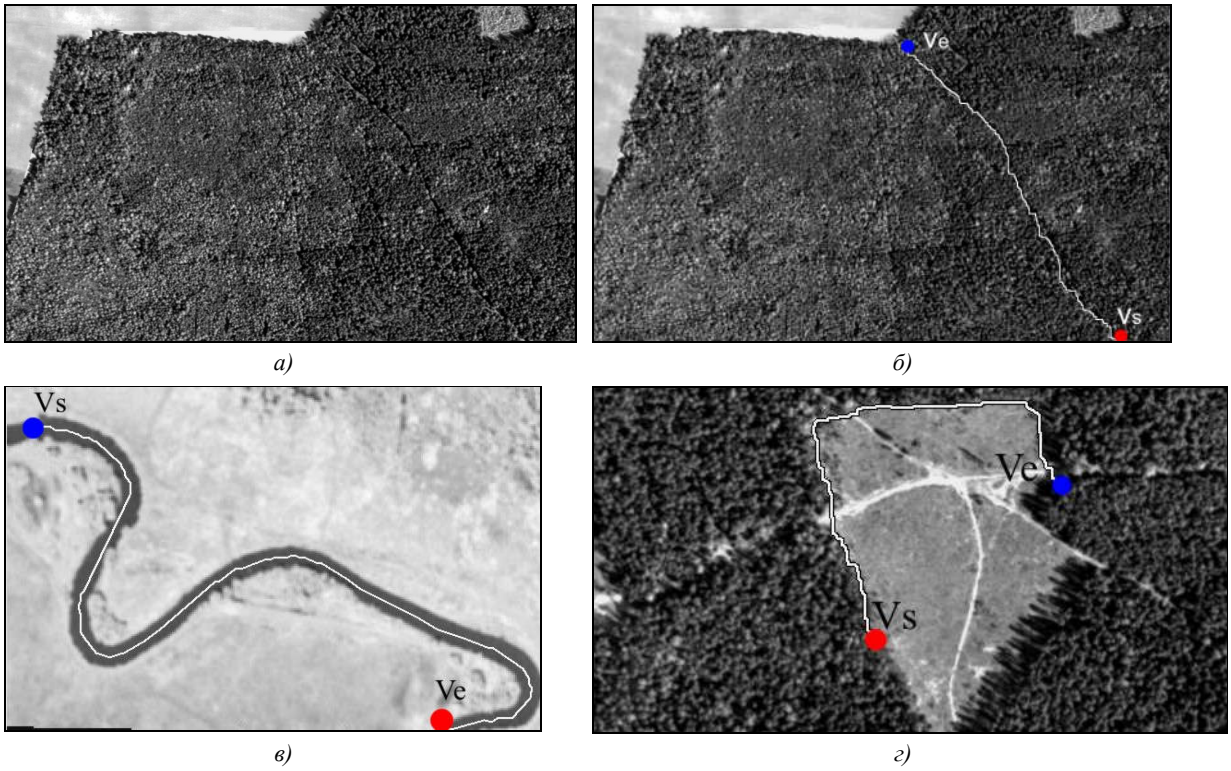
Промежуточные структуры: AL − активный список узлов; *E* − массив обработанных узлов; *S* − массив суммарной стоимости; S_{tmp} − промежуточное значение стоимости; f_{tmp} − промежуточное значение приближенной стоимости; *f* − массив приближенной стоимости; $\min_f$ − минимальное значение эвристического приближения.

1. $S(p_i)=\infty; f(p_i)=O(p_i)$ //*Инициализация*
2. $S(p_s)=0$; $AL \leftarrow p_s$; $\min_f=0$ //*Установка минимальной стоимости для* p_s *и сохранение в списке*
3. While ($AL \neq \varnothing$) OR (E(p_e) AND f(p_e)<$\min_f$) do begin
//*пока AL не пустой или не найден оптимальный путь до* p_e
4. $p \leftarrow \min(AL)$ //*Извлечение узла с минимальной стоимостью* $S(p_i)$
5. $\min_f=f(p)$ //*Получение минимального значения приближенного пути*
6. $E(p)=TRUE$ //*Пометка узла p как обработанного*
7. For each ($q \in N(p)$ AND NOT E(q)) begin // *Перебор всех необработанных соседей p*
8. $S_{tmp}=S(p)+w(p,q)$ //*Вычисление суммарной стоимости*
f_{tmp}= O(p)+S_{tmp} //*Вычисление приближенной стоимости до узла* p_e
9. If $f_{tmp} < f(q)$ begin //*Если найден более дешевый путь*
10. $S(q)= S_{tmp}$ //*Обновление стоимостей для узла q*
11. $f(q)= f_{tmp}$
12. $MD(q)=p$ //*Установка ссылки на узел p*
13. If $q \notin AL$ begin //*Если узел не в списке*
14. $AL \leftarrow q$; //*Сохраняем узел в списке*
endif
endif
endfor
endwhile

Использование эвристического правила позволяет сфокусировать фронт исследования узлов в направлении, ближайшем до точки $p_{e,}$ и тем самым значительно ограничивает область поиска.

Заключение

По сравнению с самой быстрой реализаций live-wire on-the fly [5], предложенный алгоритм показывает рост производительности в 2−15 раз на больших аэрокосмических изображениях размерами 8000×9000 пикселей, а по сравнению с первыми реализациями [2, 3] − в 10−120 раз. С помощью предложенного алгоритма сегментации успешно выделяются как площадные, так и линейные объекты (рис. 6).



а) *б)* *в)* *г)*

Рис. 6. Пример выделения объектов на аэрокосмических изображениях:
а) исходное изображение со скрытой дорогой; *б)* результат выделения скрытой дороги;
в), г) выделение линейного и площадного объектов соответственно

Алгоритм показывает отличные результаты по сегментации скрытых или зашумленных объектов. Так, на рис. 6, *а* изображена лесная дорога, которая скрыта в лесном массиве тенями деревьев, а на рис. 6, *б* результат нахождения оптимального пути между точками V_s (затравочная точка) и V_e (текущее положение курсора).

Предлагаемый подход отличается от аналогичных способом задания графа, методом вычисления весовых коэффициентов, алгоритмом поиска оптимального пути на графе. При этом результатом поиска может являться как скелет объекта, так и его контур. Данный алгоритм обеспечивает лучшее время и качество оцифровки в тех случаях, когда автоматическая сегментация не дает удовлетворительных результатов. Высокая интерактивность позволяет пользователю изменять результаты выделения объекта в реальном времени с помощью простого движения курсора мыши, получая необходимые результаты сегментации и выделения.

Список литературы

1. Color image segmentation: advances and prospects / H. Cheng, X. Jiang, Y. Sun et al. // Pattern Recognition. – 2001. – № 34. – P. 2259–2281.

2. Mortensen E.N., Barrett W.A. Interactive segmentation with intelligent scissors // Graphical Models and Image Processing. – 1998. – № 60. – P. 349–384.

3. User-Steered image segmentation paradigms: live wire and live lane / X.A. Falco, J.K. Udapa et al. // Graphical Models and Image Processing. – 1998. – № 60. – P. 233–260.

4. Kang H.W., Sung Y.S. Enhanced lane: interactive image segmentation by incremental path map construction // Graphical Models. – 2003. – № 64. – P. 282–303.

5. Falcao A.X., Udupa J.K., Miyazawa F.K. An ultra fast user-steered linage segmentation paradigm: live wire on the fly // IEEE Trans. Med. Imaging. – 2000. – Vol. 19. – № 1. – P. 55–62.

6. Dayton L., Davis J. The Photoshop 6 Wow! Book. – Peachpit press, 2001. – 640 p.

7. Kass M., Witkin A., Terzopoulos D. Snakes: active contour models // Int. J. Computer Vis. – 1988. – Vol. 1. – № 4. – P. 321–331.

8. Dijkstra E.W. A note on two problems in connexion with graphs // Numer. Math. – 1959. – № 1. – P. 269–271.

9. Кнут Д.Э. Искусство программирования: сортировка и поиск. – СПб.: Вильямс, 2000. – 832 с.

10. Bucha V.V., Ablameyko S.V. Image pixel interaction and application to image processing // Pattern Recognition and Image Analysis. – 2005. – Vol. 15. – № 1. – P. 136–138.

11. Danielson P.E. Euclidian distance mapping // Computer Graphics and Image Processing. – 1980. – № 14. – P. 227–248.

12. Nilson N.J. Principles of artificial intelligence. – Palo Alto: Tioga, 1980. – 425 p.

Поступила 22.09.05

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: bucha@newman.bas-net.by

V.V. Bucha, S.V. Ablameyko

ALGORITHM OF INTERACTIVE IMAGE SEGMENTATION BASED ON DYNAMIC PROGRAMMING

Interactive segmentation technique based on dynamic programming and fast search in hash table is proposed. The proposed algorithm differs from published work in speed and robustness. It allows extracting contour and skeleton as well. The proposed algorithm was tested on remote sensing images for segmentation task. In comparison with hand digitization and semiautomatic approaches the results of segmentation of line and area cartographic objects are achieved in more convenient and accurate way.