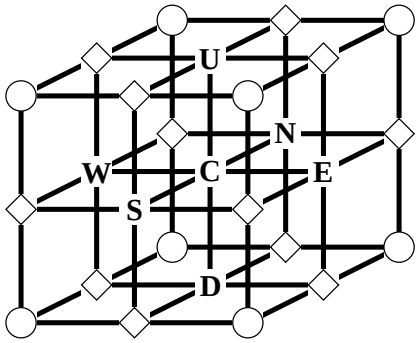


Рис. 1: Все 26 соседей являются 26-смежными с точкой C. Все точки не отмеченные окружностями являются 18-смежными с точкой C. Точки U, D, N, S, E, W 6-смежны с точкой C.

1 Основные определения

Пусть $x, y \in Z^3$ – две точки дискретного пространства. Точки x и y называются *6-смежными*, если $\|x-y\| \leq 1$, *18-смежными*, если $\|x-y\| \leq \sqrt{2}$ и *26-смежными*, если $\|x-y\| \leq \sqrt{3}$.

На рисунке 1 показаны какие отношения смежности возникают между соседними точками. Как видно из рисунка, у каждой точки есть шесть 6-смежных соседей; двенадцать 18-смежных, но не 6-смежных соседей; и восемь 26-смежных, но не 18-смежных соседей.

Точка принадлежащая объекту называется граничной, если она 6-смежна хотя бы с одной точкой фона. Граничная точка называется U-граничной, если точка, отмеченная как U на рисунке 1, является точкой фона. Аналогично определяются D-, N-, S-, E-, W- граничные точки.

Будем говорить, что изображение просматривается сверху вниз или в направлении *UD*, если сначала просматривается плоскость соответствующая верхней грани куба с точкой U (рис.1), затем плоскость с точками N, S, E, W, C, и затем точки плоскости нижней грани куба с точкой D. Аналогично определяются такие направления: *DU* – снизу вверх (D $\Longrightarrow$ <N,S,E,W,C> $\Longrightarrow$ U); *NS* – с севера на юг (N $\Longrightarrow$ <U,E,W,D,C>$\Longrightarrow$ S); *SN* – с юга на север (S $\Longrightarrow$ <U,E,W,D,C>$\Longrightarrow$ N); *EW* – с востока на запад (E $\Longrightarrow$ <U,N,S,D,C>$\Longrightarrow$ W); *WE* – с запада на восток (W $\Longrightarrow$ <U,N,S,D,C>$\Longrightarrow$ E);

2 Алгоритм скелетизации

Итерационный алгоритм извлекает тонкий скелет (среднюю линию) объектов трёхмерных бинарных изображений. Алгоритм основан на [1]. Итерации выполняются до тех пор, пока не будет построен скелет. Каждая итерация состоит из шести просмотров изображения (подитераций) в различных направлениях. Во время просмотра удаляются (перекрашивается в цвет фона) только те точки скелетизируемого объекта, которые удовлетворяют определённым условиям. Алгоритм конечен, так как входное изображение конечно (алг. 1).

INPUT: Image;
OUTPUT: Skeleton;
1: Skeleton := Image;
2: **repeat**
3: Skeleton := SubIteration(UD, Skeleton);
4: Skeleton := SubIteration(DU, Skeleton);
5: Skeleton := SubIteration(NS, Skeleton);
6: Skeleton := SubIteration(SN, Skeleton);
7: Skeleton := SubIteration(EW, Skeleton);
8: Skeleton := SubIteration(WE, Skeleton);
9: **until** *есть точки, которые можно удалить*;

Алгоритм 1: Итерационный процесс скелетизации из шести подитераций.

Во время просмотра изображения в направлении сверху вниз (UD) алгоритм пытается удалить U-граничные точки. Во время просмотра изображения в направлении снизу вверх алгоритм пытается удалить D-граничные точки. Аналогично во время остальных четырёх просмотров в направлении юг-север, север-юг, запад-восток и восток-запад удаляются некоторые N- , S- , E- , W- граничные точки. Таким образом, за одну итерацию удаляются шесть различных типов точек.

Во время итерационного процесса значение точки объекта зависит от 26-смежных ей точек. Это соседи текущей точки. Назовём шаблоном (маской) определённую комбинацию соседних точек. Тогда точка объекта удаляется (перекрашивается в цвет фона), если как минимум один шаблон из заданного множества шаблонов соответствует рассматриваемой комбинации соседних точек. Для каждого направления просмотра задаётся своё множество шаблонов (масок).

Множество шаблонов при просмотре изображения в направлении сверху вниз (UD) состоит из 24 элементов: шесть базовых шаблонов (ма-

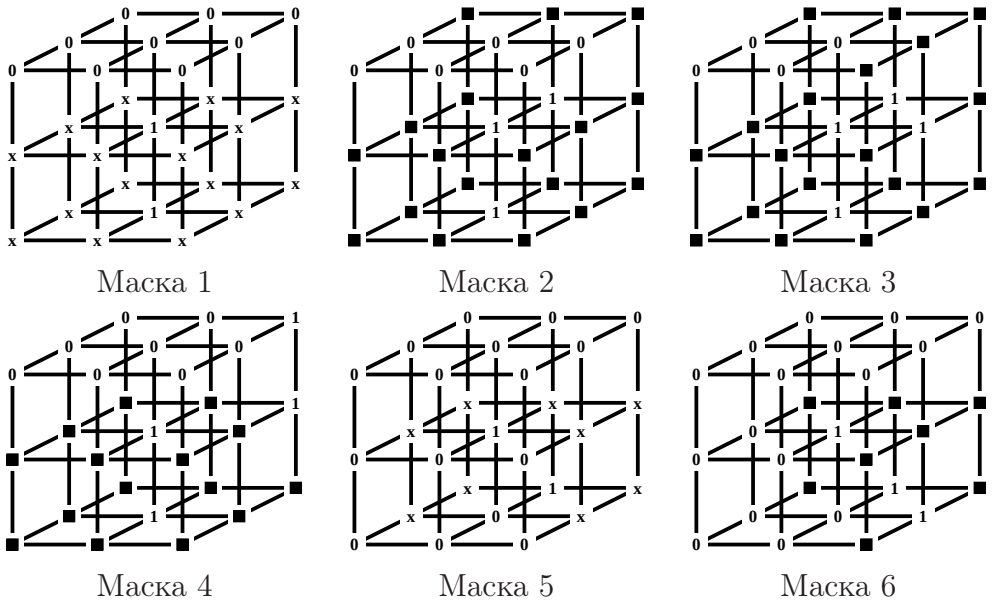


Рис. 2: Базовый набор шаблонов для просмотра изображения в направлении сверху вниз. Шаблон соответствует комбинации соседних точек, если на месте единиц находятся точки принадлежащие объекту, каждая позиция помеченная нулём содержит воксел фона и как минимум одна позиция отмеченная 'x' содержит точку принадлежащую скелетизируемому объекту. В остальных позициях тип точки не имеет значения.

сок), а также их повороты вокруг оси просмотра изображения на 90, 180 и 270 градусов (рис.2). Шаблон может содержать три вида точек: принадлежащие объекту, точки фона, произвольные точки — они могут быть как точками фона, так и точками объекта.

Фактически, описанное множество шаблонов будет состоять из $1+4+4+4+4+4=21$ элемента , так как первый шаблон рисунка 2 (Маска 1) симметричен – его повороты на 90, 180 и 270 градусов не порождают новых шаблонов. Второй шаблон (Маска 2) породит следующие шаблоны (рис.3). Аналогично и повороты оставшихся шаблонов породят по три новых каждый.

Таким образом, при просмотре изображения в направлении сверху вниз (UD) кандидатами на удаление являются только U-граничные точки – множество шаблонов заданно таким образом (рис.2), что центральная точка всегда будет U-граничной. Множество шаблонов для других направлений строится аналогично – кандидатами на удаление будут соответствующие граничные точки. На рисунке 4 показан пример того, как будет выглядеть первый шаблон (Маска 1) рисунка 2 для оставшихся на-

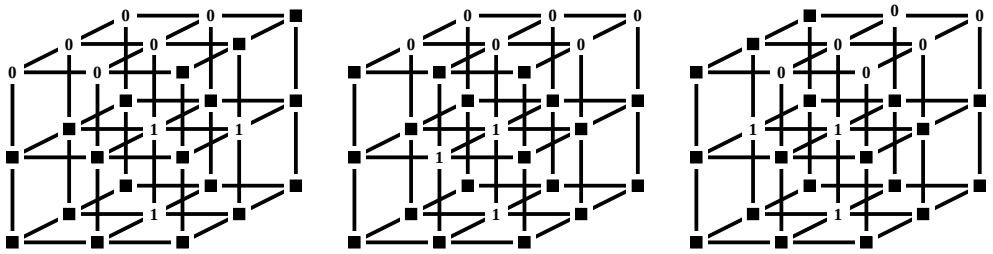


Рис. 3: Маска 2 рисунка 2 при повороте на 90, 180 и 270 градусов относительно вертикальной оси породит следующие три шаблона (маски).

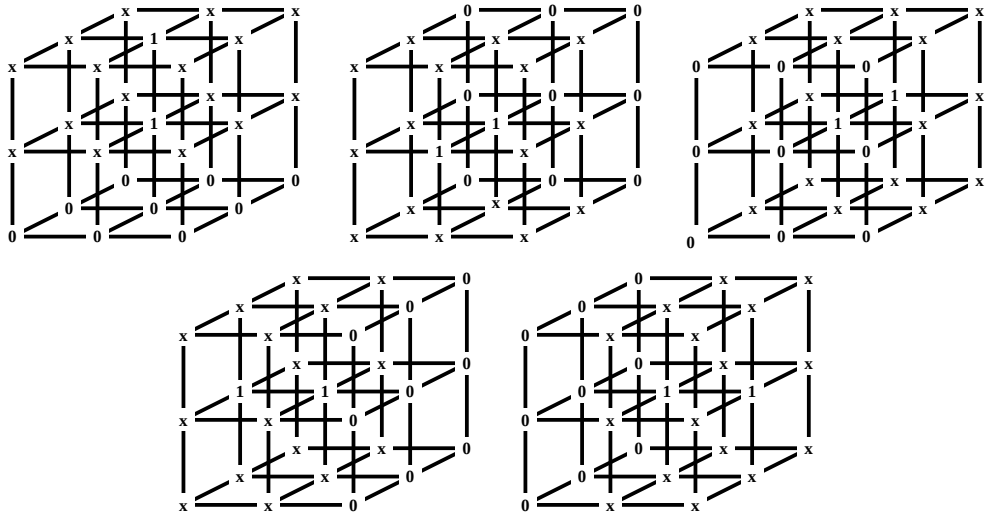


Рис. 4: Первый шаблон (Маска 1 рисунка 2) будет выглядеть так соответственно для направлений снизу вверх (DU), с севера на юг (NS), с юга на север (SN), с востока на запад (EW) и с запада на восток (WE).

правлений. Так будут повёрнуты все 21 шаблон в соответствии со своим направлением просмотра изображения.

Самой частой операцией в алгоритме является процедура проверки соседних точек на соответствие одному из 21 шаблону. Таким образом, для каждой точки необходимо произвести 21×26 проверок (21 шаблонов, 26 соседних точек). Для того чтобы избежать такого большого количества проверок можно использовать *таблицу поиска (Look Up Table)*.

Занумеруем соседей текущей точки в соответствии с рисунком 5. Последовательно обходя всех соседей рассматриваемой точки, сформируем такое число, что i-тый бит этого числа будет равен единице, если на i-том месте в шаблоне (i-тый сосед) находится точка принадлежащая объекту.

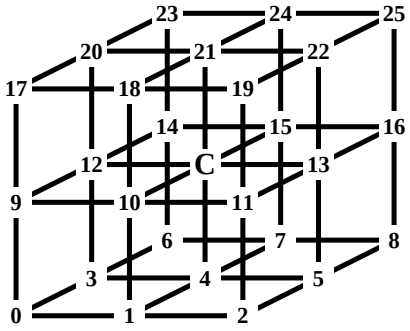


Рис. 5: Нумерация соседних точек в направлении сверху вниз (UD).

Если на i-том месте в шаблоне находится точка фона, то i-тый бит числа устанавливается в нуль. Таким образом, каждой рассматриваемой точке ставится в соответствие число N. Это число формируется во время обхода соседей текущей точки. Затем за константное время проверяется, какой бит находится на N-том месте в таблице поиска. Если найденный бит – единица, то текущая точка удаляется (перекрашивается в цвет фона), если найденный бит – нуль, то алгоритм переходит к следующей точке. При просмотре изображения в других направлениях число N формируется путём обхода всех соседей текущей точки в другом порядке (соседи нумеруются по другому) в соответствии с таблицей 1).

3 Построение таблицы поиска

Каждая точка имеет 26 соседей, и каждый сосед может быть либо фоном, либо точкой объекта. Тогда существует всего 2^{26} всевозможных комбинаций. Занумеровав каждую комбинацию как на странице 4, сформируем число N. Каждому числу в таблице поиска соответствует бит, находящийся на N-том месте. Следовательно, размер таблицы поиска составляет $2^{26-3} = 2^{23} = 8$ Мегабайт. Данная таблица является частью алгоритма и не зависит от изображения. Поэтому, такую таблицу построить достаточно один раз.

Алгоритм заполнения таблицы поиска заключается в переборе всех возможных комбинаций. Как было показано выше, каждой такой комбинации можно поставить в соответствие число N. Тогда, последовательно проверяя все числа от 0 до $2^{26} - 1$, можно перебрать все комбинации положения соседей.

Число j, полученное на j-том шаге перебора проверяется на соответствие одному из элементов множества шаблонов. Если такое соответ-

Таблица 1: Порядок обхода соседних точек для разных направлений просмотра изображения. Сосед 1 направления UD будет 24 соседом для направления DU и 11 для направления WE.

UD	**DU**	**NS**	**SN**	**EW**	**WE**	**UD**	**DU**	**NS**	**SN**	**EW**	**WE**
0	23	17	6	17	2	**1**	24	18	7	9	11
2	25	19	8	0	19	**3**	20	9	14	20	5
4	21	10	15	12	13	**5**	22	11	16	3	22
6	17	0	23	23	8	**7**	18	1	24	14	16
8	19	2	25	6	25	**9**	14	20	3	18	1
10	15	21	4	10	10	**11**	16	22	5	1	18
12	12	12	12	21	4	**13**	13	13	13	4	21
14	9	3	20	24	7	**15**	10	4	21	15	15
16	11	5	22	7	24	**17**	6	23	0	19	0
18	7	24	1	11	9	**19**	8	25	2	2	17
20	3	14	9	22	3	**21**	4	15	10	13	12
22	5	16	11	5	20	**23**	0	6	17	25	6
24	1	7	18	16	14	**25**	2	8	19	8	23

ствие найдено, то в таблице j-тый бит устанавливается в единицу, иначе – в нуль. Для проверки на соответствие используются следующие булевы функции:

$$
\begin{aligned}
m1(x) &= (x_0|x_1|x_2|x_3|x_5|x_6|x_7|x_8|x_9|x_{10}|x_{11}|x_{12}|x_{13}|x_{14}| \\
&\quad |x_{15}|x_{16})\ x_4\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{19}\ \overline{x}_{20}\ \overline{x}_{21}\ \overline{x}_{22}\ \overline{x}_{23}\ \overline{x}_{24}\ \overline{x}_{25}; \\
m2(x) &= x_4\ x_{15}\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{19}\ \overline{x}_{20}\ \overline{x}_{21}\ \overline{x}_{22}; \\
m3(x) &= x_4\ x_{13}\ x_{15}\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{20}\ \overline{x}_{21}; \\
m4(x) &= x_4\ x_{16}\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{19}\ \overline{x}_{20}\ \overline{x}_{21}\ \overline{x}_{23}\ \overline{x}_{24}\ x_{25}; \\
m5(x) &= \overline{x}_0\ \overline{x}_1\ \overline{x}_2\ (x_3|x_5|x_6|x_8|x_{12}|x_{13}|x_{14}|x_{15}|x_{16})\ \overline{x}_4\ x_7 \\
&\quad \overline{x}_9\ \overline{x}_{10}\ \overline{x}_{11}\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{19}\ \overline{x}_{20}\ \overline{x}_{21}\ \overline{x}_{23}\ \overline{x}_{24}\ \overline{x}_{25}; \\
m6(x) &= \overline{x}_0\ \overline{x}_1\ \overline{x}_2\ \overline{x}_3\ \overline{x}_4\ x_5\ x_7\ \overline{x}_9\ \overline{x}_{10}\ \overline{x}_{12}\ \overline{x}_{17}\ \overline{x}_{18}\ \overline{x}_{19}\ \overline{x}_{20} \\
&\quad \overline{x}_{21}\ \overline{x}_{23}\ \overline{x}_{24}\ \overline{x}_{25};
\end{aligned}
$$

Эти булевы функции соответствуют шаблонам, приведённым на рисунке 2. Для проверки всего множества шаблонов (стр. 3) – 21 элемент – осуществляются повороты шаблонов на 90, 180 и 270 градусов. Таким образом проверяется, соответствует ли числу j некий шаблон (выполняется ли одна из вышеприведённых функций), либо его поворот. Поворот

Таблица 2: Таблица перевода бит при моделировании поворота шаблона на 90°

Позиция	**Биты**												
старая	0	1	2	3	4	5	6	7	8	9	10	11	12
новая	2	5	8	1	4	7	0	3	6	11	13	16	10
старая	13	14	15	16	17	18	19	20	21	22	23	24	25
новая	15	9	12	14	19	22	25	18	21	24	17	20	23

осуществляется последовательно на 90 градусов против часовой стрелки. Моделирование поворота происходит посредством обмена значения бит в соответствии с таблицей 2.

Таким образом, формируется новое число, второй бит которого будет соответствовать нулевому биту исходного числа, пятый бит будет равен первому биту исходного числа и так далее. Полученное число, аналогично числу j, проверяется на соответствие одному из шаблонов. Если такое соответствие не найдено, осуществляется ещё один поворот. Всего выполняется 3 поворота.

Построенная таблица поиска сохраняется. Во время работы алгоритма скелетизации эта таблица загружается. Тем самым достигается быстрый доступ к элементам таблицы поиска.

4 Улучшенный алгоритм скелетизации

Во время работы алгоритма просматриваются не только точки объекта, но и точки фона. На их чтение и идентификацию тратится процессорное время. Чтобы уменьшить количество операций чтения и сравнения предварительно вычисляется параллелепипед, в который вписан скелетизируемый объект (рис. 6). Тогда просмотр точек в слое будет осуществляться в пределах прямоугольника, а слои не содержащие точек объекта вообще обрабатываться не будут. Далее, точки объекта в слое могут образовывать одну либо несколько связных компонент. Чтобы уменьшить количество проверок, для каждой компоненты каждого слоя вычисляется прямоугольная область, задающая границу компоненты. Таким образом, для каждого слоя хранится список областей с точками объекта, и скелетизация выполняется только в границах этих областей (рис. 6).

Толщина скелетизируемого объекта в разных слоях может сильно отличаться. В этом случае, скелет изображения (рис. 7) для слоёв 20–30 будет построен раньше, чем для слоёв 5–15. Чтобы избежать просмотр

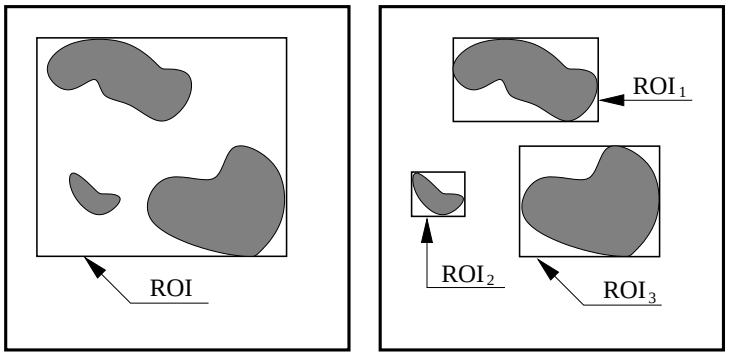


Рис. 6: Слева: прямоугольник задаёт границу точек объекта в слое; Справа: для каждой связной компоненты в слое вычисляется её прямоугольная граница.

слоёв, где скелет уже построен, используется менеджер слоёв. Он отслеживает удаление точек и указывает какой слой проверять следующим.

Каждый слой изображения находится в одном из трёх состояний: активном, пассивном и полупассивном. Состояния слоёв обновляется после каждой подитерации и итерации алгоритма. Просматриваются только те слои, которые не находятся в пассивном состоянии. Слой становится пассивным, если слой ниже и слой выше не являются активными. Такое обновление делается после каждой итерации. Слой становится полупассивным, если в предыдущих подитерациях данной итерации не удалялись точки объекта. Такое обновление делается после каждой подитерации. Слой становится активным, если в нём удалена хотя бы одна точка объекта. Это состояние обновляется после каждой подитерации (алг. 2 и алг. 3). Таким образом, изначально, все слои полупассивны. Если на какой-то подитерации удаляется точка, то слой содержащий эту точку становится активным. По завершению итерации (после шести проходов изображения) полупассивный слой становится пассивным, если его окру-

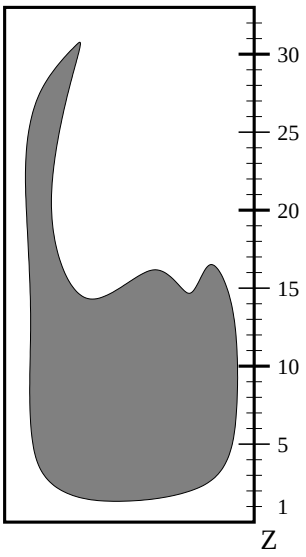


Рис. 7: Скелет для слоёв 20–30 будет построен раньше, чем для слоёв 5–15.

жают полупассивные слои. Это связанно с тем, что значение точки во время просмотра изображения зависит от значений точек текущего, а так же верхнего и нижнего слоёв. Далее, на следующей итерации просматриваются слои только в активном и полупассивном состояниях. При этом, после каждой подитерации пассивный слой становится полупассивным, если хотя бы один из соседних слоёв находится в активном состоянии.

```
1: for all слоёв i do
2:   if state[i] = active then
3:     newState := active;{активное состояние}
4:   else
5:     if state[i − 1] = active OR state[i + 1] = active then
6:       newState[i] := undefined;{полупассивное состояние}
7:     else
8:       newState[i] := passive;{пассивное состояние}
9:     end if
10:  end if
11: end for
12: state := newState;
```

Алгоритм 2: Обновление состояния менеджера слоёв после каждой итерации.

```
1: for all слоёв i do
2:   if state[i] = active then
3:     newState := active;{активное состояние}
4:   else
5:     if state[i − 1] = active OR state[i + 1] = active then
6:       newState[i] := undefined;{полупассивное состояние}
7:     end if
8:   end if
9: end for
10: state := newState;
```

Алгоритм 3: Обновление состояния менеджера слоёв после каждой подитерации.

Описанные приёмы позволяют уменьшить количество операций просмотра точек изображения. Эффективность таких подходов зависит от формы скелетизируемого объекта.

5 Скелетизация дерева сосудистой системы головного мозга

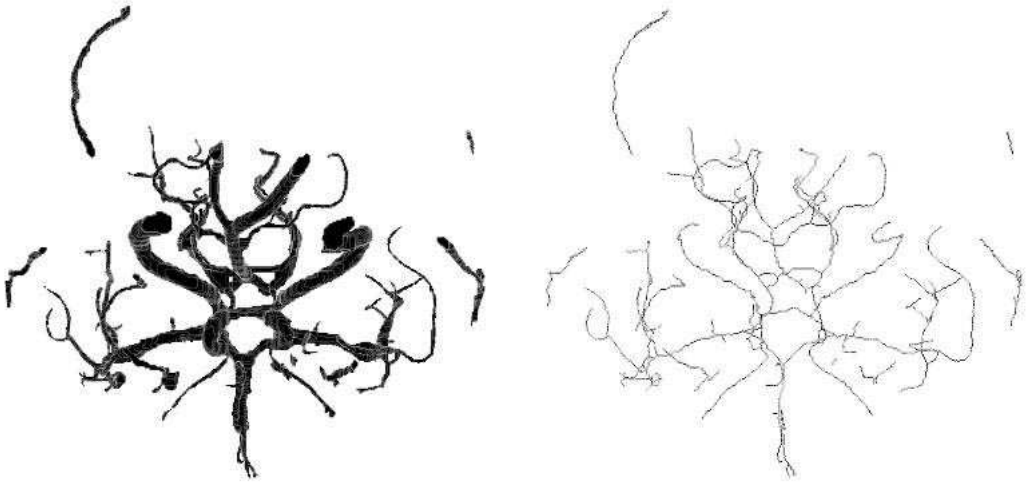
Для вычисления скелета (средней линии) сосудистой системы головного мозга был применён итерационный алгоритм скелетизации (раздел 2, стр. 2). На вход алгоритма подаётся трёхмерное изображение сосудистой системы головного мозга. Вокселы нулевой интенсивности считаются точками фона, все остальные вокселы – точки скелетизируемого объекта. Изображение может содержать несколько несвязных компонент. Тогда каждая компонента входного изображения имеет свою яркость. Поэтому реализованный алгоритм скелетизации на вход принемает не бинарное изображение (точки фона – точки объекта), а псевдобинарное (точки фона – точки объекта 1,. . .,точки объекта m) и имеет фиксированное количество яркостей. Результатом работы алгоритма является трёхмерное изображение, состоящее из трёх типов вокселов: точки фона (нулевая яркость), точки объекта (но не скелет) и точки скелета. Вокселы объекта и вокселы скелета имеют свои яркости, которые кодируют тип точки и её принадлежность к компоненте. Таким образом, точки изображения перекрашиваются в соответствии с таблицей 3.

Вокселы	До	После
Фон	0	0
Максимальная яркость	$m = max\{v_i\}$	$2 \times m$
i-тый объект	v_i	$v_i + m$
Скелет i-ого объекта	- -	v_i

Таблица 3: Интенсивность вокселов изображения до и после скелетизации.

Во время работы итерационного алгоритма, когда граничная точка удаляется, она перекрашивается не в цвет фона, а в цвет, кодирующий номер компоненты, к которой эта точка принадлежала. Этот способ накладывает ограничение на количество компонент во входном изображении, так как единица данных (байт, слово и т.д.) кодирует фиксированное количество яркостей. Точки скелета, так как они не удаляются, сохраняют яркость компоненты исходного изображения. Таким образом, выходное изображение не утрачивает данные входного изображения и добавляет информацию о точках скелета объектов.

Кроме того, реализованный алгоритм может работать в двух режимах: не различая цвета компонент и различая их. В первом случая, любая точка с ненулевой яркостью считается точкой скелетизируемого объекта. Во втором случае, точками скелетизируемого объекта (при обходе соседних точкек) считаются только вокселы текущей яркости. Эта осо-



а) б)

Рис. 8: Скелетизация дерева сосудистой системы головного мозга: а) входное изображения размером $512 \times 512 \times 120$; б) скелет системы.

бенность позволяет эффективно скелетизировать даже связные объекты (но отличающиеся по яркости).

Пример работы реализованного алгоритма скелетизации показан на рисунке 8. Входное изображение размером $512 \times 512 \times 120$ после сегментации и удаления пустот имело 7 несвязных компонент. Точки скелета расположены в центре сосудов.

5.1 Библиографическая справка

Скелетом $SK(X)$ непрерывного изображения $X \subseteq R^m$ называется множество центров максимальных шаров, вписанных в X. Шар называется максимальным, если он не содержится полностью в другом шаре, содержащемся в X. Поэтому, максимальный шар должен касаться границы объекта как минимум в двух различных точках.

Свойства скелета: неединственность – один и тот же скелет может соответствовать различным объектам; неустойчивость – возмущения границы объекта влияют на форму скелета; скелет содержит информацию о локальной симметричности объекта и о его топологическом строении; топологические свойства объекта и скелета идентичны; скелет инвариантен относительно основных геометрических преобразований, включая смещение, вращение и растяжение.

Выделяют три основных метода скелетизации: дистанционное преоб-

разование, скелетизация диаграммой Вороного и скелетизация утоньшением.

В статье [2] представлены два алгоритма утоньшения трёхмерных изображений с сохранением топологических свойств объектов. Первый алгоритм строит скелет объекта, а второй извлекает тонкий скелет – древовидную структуру без поверхностей. Алгоритмы являются итерационными и могут быть частично распараллелены.

Предварительно все точки фона перекрашиваются в цвет $-maxint$, а точки обьекта перекрашиваются в 0. Вокселы со значением меньше нуля считаются точками фона, вокселы со значением ноль считаются непомеченными, а со значением больше нуля – помеченными. Алгоритм извлечения скелета состоит из двух этапов.

На первом этапе итерация выполняется за три просмотра изображения. Каждый просмотр проверяет непомеченные *s-open*, *e-open* и *v-open* точки соответственно. Соседние вокселы текущей точки определяют её тип и задаются в виде шаблонов (масок). Непомеченная точка может быть оставлена без изменений, удалена либо помечена. При удалении воксел перекрашивается в цвет $-maxint + i$. Итерации выполняются до тех пор, пока существуют точки, которые можно удалить. На втором этапе удаляются оставшиеся артефакты. Итерация удаляет точки соответствующие дополнительному шаблону.

Алгоритм извлечения тонкого скелета также состоит из двух этапов: два прохода изображения за итерацию первого этапа и один просмотр за итерацию второго этапа. Для каждого просмотра задаются свои шаблоны (маски). Недостатком преставленного метода является то, что задаваемые шаблоны очень громоздки, некоторые из них включают соседей соседних точек, что существенно увеличивает количество проверок и сложность реализации алгоритма.

В статьях [1, 5, 3] описывается итерационный алгоритм утоньшения для извлечения тонкого скелета объектов трёхмерных цифровых изображений. Алгоритм заключается в последовательном удалении простых точек. Точка называется простой, если после её удаления топология объекта не изменяется. Алгоритм является итерационным и продолжается до тех пор, пока все простые точки не будут удалены. Каждая итерация состоит из трёх проходов изображения. Во время первого прохода удаляются верхние и нижние простые точки объектов, во втором проходе – передние и задние простые точки, в третьем – правые и левые. Точки, которые удаляются, являются граничными точками, то есть хотя бы один их ближайший соседей есть точка фона. Удаляются только те граничные точки, которые удовлетворяют определённым условиям.

Для каждого прохода (направления) существуют свои условия, однако все они изоморфны. Условия задаются с помощью множества шаблонов, соответствующих различным комбинациям соседей текущей точки. Таким образом, текущая точка удаляется, если набор из 26 соседних ей точек соответствует одному из шаблонов. Шаблоны разных направлений могут быть получены друг из друга с помощью соответствующих поворотов. В статье приводятся доказательства того, что алгоритм конечен и что сохраняется топология объектов. Представленный алгоритм применён при оценки сужений гортанно-трахеального тракта [6], оценки артериальных аневризмов сосудов кровеносной системы, оценки толстой кишки. Для вычисления оценок на основе извлечённого скелета производится подсчёт площадей поперечных сечений исследуемых объектов.

Статья [4] исследует изменения топологии объектов при 6-подитерационной операции сокращения (6-subiteration reduction operation). При итерационном процессе удаляются только те точки изображения, которые удовлетворяют определённым условиям. Эти условия могут быть заданы по-разному, в зависимости от того, что необходимо получить в результате: скелет (medial surface), тонкий скелет (medial line) или ядро объекта (kernel). В статье описаны условия, при выполнении которых, топология объектов не изменится. Приводятся доказательства корректности выдвинутых ограничений и их достаточности.

В статье [7] представлен параллельный итерационный алгоритм утоньшения. Каждая итерация состоит из двух этапов. На первом этапе изображение просматривается в шести ортогональных направлениях и удаляются граничные точки удовлетворяющие определённым условиям. На втором этапе просматривается изображение ещё раз, и удаляются оставшиеся точки. Преимуществом алгоритма является то, что он позволяет частично распараллелить итерационный процесс – просмотр изображения в шести ортогональных направлениях может происходить независимо, так как условия удаления граничных точек при таком просмотре не влияют друг на друга. Заключительный этап итерации ещё раз просматривает изображение и удаляет те точки, которые при параллельном просмотре не были удалены. Так как представленный алгоритм чувствителен к различным шумам и артефактам изображения, то применяется предобработка изображения. В результате перекрашиваются вокселы: точка объекта становится точкой фона, если у неё более 24 фоновые соседних точки, и точка фона становится объектом, если у неё более трёх 6-смежных соседних точек объекта.

Список литературы

[1] K. Palágyi and A. Kuba, A 3D 6-subiteration thinning algorithm for extracting medial lines, *Pattern Recognition Letters*, 1998, vol.19, 613-627

[2] P.K. Saha and B.B. Chaudhuri and D.D. Majumder,A new shape preserving parallel thinning algorithm for 3D digital images, *Pattern Recoghition*, 1997, Vol. 30, Num 12, pages 1939-1955

[3] K. Palágyi and E. Sorantin and E. Balogh and A. Kuba and C. Halmai and B. Erdöhelyi and K. Hausegger, A Sequential 3D Thinning Algorithm and Its Medical Application, *In proceeding IPMI 2001*, 2001, pages 409-415

[4] C.-M. Ma, Topology preservation of template-based 6-subiteretion reduction operations, *Pattern Recognition*, 2003, Vol. 36, pages 1775-1782

[5] K. Palágyi, A 3D 3-subiteration thinning algorithm for medial surfaces, *Springer*, 2000, pages 406-417

[6] E. Sorantin and C. Halmai and B. Erdöhelyi and K. Palágyi and L.G. Nyúl and K. Ollé and B. Geiger and F. Lindbichler and G. Friedrich and K. Kiesler, Spiral-CT-Based Assessment of Tracheal Stenoses Using 3-D-Skeletonization, *IEEE Transactions On Medical Imaging*, 2002, Vol. 21, Num. 3, pages 263-273

[7] W. Xie and R. P. Thompson and R. Perucchio, A topology-preserving parallel 3D thinning algorithm for extracting the curve skeleton, *Pattern Recognition*, 2003, Vol. 36, pages 1529-1544