*УДК 621.391*

**О.Г. Шевчук, В.Ю. Цветков**

# НОРМАЛИЗАЦИЯ КОНТУРНЫХ ЛИНИЙ ПО ТОЛЩИНЕ НА ОСНОВЕ АНАЛИЗА ЛОКАЛЬНЫХ ОРИЕНТАЦИЙ ИХ ФРАГМЕНТОВ

*Предлагается метод нормализации двухконцевых контурных линий по толщине на бинарных изображениях, основанный на анализе локальных ориентаций их фрагментов. Проводится сравнение предложенного метода с известными методами скелетизации. Показывается, что предложенный метод превосходит известные методы скелетизации по быстродействию и качеству.*

## Введение

После обработки полутоновых изображений такими операторами выделения краев, как Канни [1], Робертс [2] и др., формируются бинарные изображения, которые содержат множество линий, образованных связанными совокупностями единичных пикселов. Получаемые таким образом линии имеют толщину, как правило, в несколько пикселов, что приводит к ошибкам их параметризации и последующей идентификации. Решить данную проблему можно путем минимизации толщины линий. Для этого могут использоваться известные методы скелетизации объектов [3–7]. В результате работы данных методов происходит удаление избыточных контурных пикселов и формируются контурные линии или скелет объекта толщиной в один пиксел. Для обработки бинарных изображений широко применяются методы скелетизации Зонга – Суня [4], шаблонный [8], волновой [9], Щепина [10]. Они являются итеративными и имеют высокую вычислительную сложность. Ориентация на площадные объекты с учетом итеративного характера обработки делает данные методы неэффективными для скелетизации контурных линий, имеющих толщину в несколько пикселов. Быстрая нормализация контурных линий по толщине возможна за счет неитеративного анализа локальных ориентаций небольших фрагментов контурных линий размером три пиксела и удаления избыточных контурных пикселов в окрестности Мура.

Целью настоящей работы является разработка быстрого метода нормализации контурных линий по толщине на основе анализа локальных ориентаций их фрагментов.

## 1. Метод нормализации контурных линий по толщине на основе масок

Предлагается метод нормализации двух концевых контурных линий по толщине на основе анализа локальных ориентаций их фрагментов, образованных смежными контурными пикселами, и удаления избыточных контурных пикселов. Метод отличается от известных методов скелетизации, использующих многократную обработку пикселов, однократным анализом каждого пиксела в результате квантования по ориентации фрагментов контурной линии с помощью масок 2×2 пиксела, определением избыточных контурных пикселов в этих фрагментах и их удалением, что позволяет повысить скорость и качество контурной обработки.

Исходными данными для метода нормализации контурных линий по толщине на основе анализа локальных ориентаций их фрагментов являются координаты $\{X(n)\}_{(n=\overline{0,N_L-1})}$, $\{Y(n)\}_{(n=\overline{0,N_L-1})}$ контурных пикселов выделенных линий, координаты $X_e=\|x_1(n),x_2(n)\|_{(n=\overline{0,N_L-1})}$, $Y_e=\|y_1(n),y_2(n)\|_{(n=\overline{0,N_L-1})}$ их концевых точек и бинарная матрица $P=\|p(x,y)\|_{(y=\overline{0,Y-1},x=\overline{0,X-1})}$ образов выделенных контурных линий, где $n=\overline{0,N_L-1}$ – порядковый номер линии; $N_L$ – количество выделенных линий; $p(x,y)=\{0,1\}$ – значение пиксела в бинарной матрице образов; $y=\overline{0,Y-1}$, $x=\overline{0,X-1}$ – координаты пиксела; $X$, $Y$ – размер матрицы образов; $X(n)=\|x(n,i)\|_{(i=\overline{0,N_p(n)-1})}$, $Y(n)=\|y(n,i)\|_{(i=\overline{0,N_p(n)-1})}$ – координаты пикселов $n$-й линии;

$i=\overline{0,N_p(n)-1}$ – порядковый номер пиксела $n$-й линии; $N_p(n)$ – количество пикселов в $n$-й линии. Контурные пикселы формируются в результате сегментации бинарного изображения методом выращивания областей (Region Growing, RG) [11]. Каждому контурному пикселу присваивается номер линии, которой он принадлежит.

Алгоритм нормализации $n$-й выделенной линии состоит из следующих шагов:

*Шаг 1.* Инициализация матрицы $H(n)=\|h(n,i)\|_{(i=\overline{1,N_p(n)-1})}$ состояний контурных пикселов $n$-й линии, где $h(n,i)=\begin{cases}0, & \text{если пиксел должен быть удален;}\\ 1, & \text{если пиксел не обработан, } h(n,i)\leftarrow 1 \text{ при } i=\overline{0,N_p(n)-1};\\ 2, & \text{если пиксел обработан.}\end{cases}$

*Шаг 2.* Определение ориентации $o(n)$ линии с помощью выражения

$$o(n)=\frac{y_2(n)-y_1(n)}{x_2(n)-x_1(n)}.$$

*Шаг 3.* Формирование бинарных масок $M_1=\|m_1(g,j)\|_{(g=\overline{0,1},\,j=\overline{0,1})}$, $M_2=\|m_2(g,j)\|_{(g=\overline{0,1},\,j=\overline{0,1})}$ размером $2\times 2$ пиксела для квантования по ориентации фрагментов линии, где

$$m_1(0,0)=\begin{cases}0 \text{ при } o(n)<0,\\ 1 \text{ при } o(n)\ge 0;\end{cases}\quad m_1(0,1)=\begin{cases}1 \text{ при } o(n)<0,\\ 0 \text{ при } o(n)\ge 0;\end{cases}\quad m_1(1,0)=1,\ m_1(1,1)=1;$$

$$m_2(0,0)=\begin{cases}1 \text{ при } o(n)<0,\\ 0 \text{ при } o(n)\ge 0;\end{cases}\quad m_2(0,1)=\begin{cases}0 \text{ при } o(n)<0,\\ 1 \text{ при } o(n)\ge 0;\end{cases}\quad m_2(1,0)=1,\ m_2(1,1)=1.$$

Примерами масок являются матрицы $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}$ и $\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$.

*Шаг 4.* Инициализация счетчика цикла $i\leftarrow 0$ обработки пикселов $n$-й линии.

*Шаг 5.* Начало цикла анализа пикселов $n$-й линии.

Обработка $i$-го элемента матрицы $H(n)$ состояний согласно выражению

$$\begin{cases}\big(h(n,i)=1\big)\Rightarrow y_0\leftarrow y(n,i),\ x_0\leftarrow x(n,i), \text{ переход на шаг 6;}\\ \big(h(n,i)=2\big)\Rightarrow \text{ переход на шаг 12;}\\ \big(h(n,i)=0\big)\Rightarrow \text{ удаляются } y(n,i),\ x(n,i) \text{ и } h(n,i),\ N_p(n)=N_p(n)-1, \text{ переход на шаг 13,}\end{cases}$$

где $y_0, x_0$ – координаты опорного пиксела.

*Шаг 6.* Формирование матриц $X_R=\|x_R(q)\|_{(q=\overline{0,2})}$, $Y_R=\|y_R(q)\|_{(q=\overline{0,2})}$ координат и $I_R=\|i_R(q)\|_{(q=\overline{0,2})}$ индексов связанных пикселов с использованием маски $M_1$, элементы которых вычисляются с помощью выражений

$$\begin{cases}\exists b\Big(b\in\big[0,N_p(n)-1\big]\Big)\left(\begin{array}{l}\big(m_1(g,j)\,p(y_0+g,x_0+j)=1\big)\wedge\big((y_0+g)=y(n,b)\big)\wedge\\ \wedge\big((x_0+j)=x(n,b)\big)\wedge\big(h(n,b)=1\big)\wedge(b\ne i)\end{array}\right)\Rightarrow\\ \Rightarrow\Big(y_R(K_q)\leftarrow(y_0+g)\Big),\Big(x_R(K_q)\leftarrow(x_0+j)\Big),\Big(i_R(K_q)\leftarrow b\Big) \text{ при } o(n)<0;\\ \exists b\Big(b\in\big[0,N_p(n)-1\big]\Big)\left(\begin{array}{l}\big(m_1(g,j)\,p(y_0-g,x_0-j)=1\big)\wedge\big((y_0-g)=y(n,b)\big)\wedge\\ \wedge\big((x_0-j)=x(n,b)\big)\wedge\big(h(n,b)=1\big)\wedge(b\ne i)\end{array}\right)\Rightarrow\\ \Rightarrow\Big(y_R(K_q)\leftarrow(y_0-g)\Big),\Big(x_R(K_q)\leftarrow(x_0-j)\Big),\Big(i_R(K_q)\leftarrow b\Big) \text{ при } o(n)\ge 0;\end{cases}$$

$$\left(K_q \leftarrow K_q + 1\right)$$

при $g = \overline{0,1}$, $j = \overline{0,1}$, где $K_q$ – количество найденных связанных пикселов (при инициализации шага 6 $K_q \leftarrow 0$).

Если $K_q = 0$, осуществляется формирование матриц $X_R$, $Y_R$ и индексов $I_R$ связанных пикселов с использованием маски $M_2$, элементы которых вычисляются с помощью выражений

$$\begin{cases} \exists b\left(b \in \left[0, N_p(n)-1\right]\right)\left(\begin{matrix}\left(m_2(g,j)\, p(y_0-g, x_0-j)=1\right)\wedge\left(\left(y_0-g\right)=y(n,b)\right)\wedge \\ \wedge\left(\left(x_0-j\right)=x(n,b)\right)\wedge\left(h(n,b)=1\right)\wedge\left(b\neq i\right)\end{matrix}\right)\Rightarrow \\ \Rightarrow\left(y_R\left(K_q\right)\leftarrow\left(y_0-g\right)\right),\left(x_R\left(K_q\right)\leftarrow\left(x_0-j\right)\right),\left(i_R\left(K_q\right)\leftarrow b\right) \text{ при } o(n)<0; \\ \exists b\left(b \in \left[0, N_p(n)-1\right]\right)\left(\begin{matrix}\left(m_2(g,j)\, p(y_0+g, x_0+j)=1\right)\wedge\left(\left(y_0+g\right)=y(n,b)\right)\wedge \\ \wedge\left(\left(x_0+j\right)=x(n,b)\right)\wedge\left(h(n,b)=1\right)\wedge\left(b\neq i\right)\end{matrix}\right)\Rightarrow \\ \Rightarrow\left(y_R\left(K_q\right)\leftarrow\left(y_0+g\right)\right),\left(x_R\left(K_q\right)\leftarrow\left(x_0+j\right)\right),\left(i_R\left(K_q\right)\leftarrow b\right) \text{ при } o(n)\geq 0; \end{cases}$$

$$\left(K_q \leftarrow K_q + 1\right)$$

при $g = \overline{0,1}$, $j = \overline{0,1}$.

При $K_q = 1..3$ осуществляется переход на шаг 7, при $K_q = 0$ – переход на шаг 11.

*Шаг 7.* Формирование матрицы $O_R = \left\|o_R(q)\right\|_{q=\overline{0,K_q}}$ локальных ориентаций контурных фрагментов, образованных опорным пикселом и пикселами, связанными с ним. Локальная ориентация $o_R(q)$ каждого фрагмента с координатами $(y_0, x_0)$ и $\left(x_R(q), y_R(q)\right)$ вычисляется с помощью выражения

$$o_R(q) = \frac{y_R(q) - y_0}{x_R(q) - x_0}$$

при $q = \overline{0, K_q - 1}$.

*Шаг 8.* Определение состояний $h\left(n, i_R(q)\right)$ связанных пикселов на основе анализа их локальных ориентаций $O_R$ относительно ориентации линии $o(n)$ в соответствии с выражением

$$\begin{cases} \left(\left(o_R(q)=0\right)\vee\left(o_R(q)\to\infty\right)\right)\Rightarrow\left(h\left(n,i_R(q)\right)\leftarrow\begin{cases}1 \text{ при } \left\|o_R(q)\right|-\left|o(n)\right\|\leq 0{,}1\\ 0 \text{ при } \left\|o_R(q)\right|-\left|o(n)\right\|> 0{,}1\end{cases}\right), \\ \left(o_R(q)=1\right)\Rightarrow\left(h\left(n,i_R(q)\right)\leftarrow\begin{cases}1 \text{ при } \left(o(n)\neq 0\right)\wedge\left(\left|o(n)\right|<<\infty\right)\\ 0 \text{ при } \left(o(n)=0\right)\vee\left(\left|o(n)\right|\to\infty\right)\end{cases}\right) \end{cases}$$

при $q = \overline{0, K_q - 1}$.

*Шаг 9.* Проверка наличия разрывов в линии.

Если выполняется условие

$$\left(K_q = 2\right)\wedge\left(\sum_{q=0}^{K_q-1} h\left(n, i_R(q)\right) = 0\right),$$

то для устранения разрыва линии осуществляется переход на шаг 10, иначе – переход на шаг 11.

*Шаг 10.* Устранение разрывности.

Для устранения разрывности линии в матрицы $Y(n)$, $X(n)$ и $H(n)$ добавляются новые элементы в соответствии с выражениями

$$N_p(n) \leftarrow N_p(n)+1;$$

$$y(n, N_p(n)-1) \leftarrow \begin{cases} y_0 \text{ при } |o(n)| <= 0,5, \\ y_0+1 \text{ при } |o(n)| > 0,5; \end{cases}$$

$$x(n, N_p(n)-1) \leftarrow \begin{cases} x_0-1 \text{ при } o(n)=[-2;0], \\ x_0 \text{ при } |o(n)| > 2, \\ x_0+1 \text{ при } o(n)=(0;2]; \end{cases}$$

$$h(n, N_p(n)-1) \leftarrow 1.$$

Осуществляется переход на шаг 11.

*Шаг 11.* В соответствии с порядковым номером координат $i$-го опорного пиксела $y_0$, $x_0$ матриц $Y(n)$ и $X(n)$ устанавливается состояние данного пиксела $h(n,i) \leftarrow 2$ в стеке $H(n)$.

*Шаг 12.* Установка значения счетчика $i \leftarrow i+1$.

*Шаг 13.* Анализ значения счетчика.

При $i > (N_p(n)-1)$ осуществляется выход из алгоритма, иначе – переход на шаг 5.

В результате выполнения метода происходят поиск и удаление избыточных пикселов. Избыточными являются пикселы, исключение которых уменьшает толщину контура до одного пиксела, не разрывая его.

## 2. Оценка эффективности метода нормализации контурных линий по толщине

Разработанный алгоритм реализован на языке C++ с использованием библиотеки OpenCV 2.4.10. Для сравнительной оценки работы алгоритма реализован наиболее известный алгоритм скелетизации Зонга – Суня. Эксперимент проведен на компьютере со следующими техническими характеристиками: процессор Intel(R) Core(TM) i7-4700HQCPU @ 2,40 ГГц; ОЗУ – 6 ГБ; тип системы – 64-разрядная операционная система Windows 8.1.

Для первичного тестирования алгоритма нормализации контурных линий по толщине использованы следующие линии:

– искусственные, построенные и повернутые в графическом редакторе;

– выделенные на изображении, полученном с помощью фотокамеры, и повернутые в графическом редакторе.

В качестве критериев эффективности алгоритмов использованы дисперсия формфактора и время нормализации линии. Формфактор $F_i$ линии для $i$-го угла поворота изображения вычисляется как отношение размера линии $r_i$, определяемого по известным координатам ее конечных точек, к длине линии $s_i$, определяемой суммой образующих ее контурных точек [12]:

$$F_i = r_i / s_i .$$

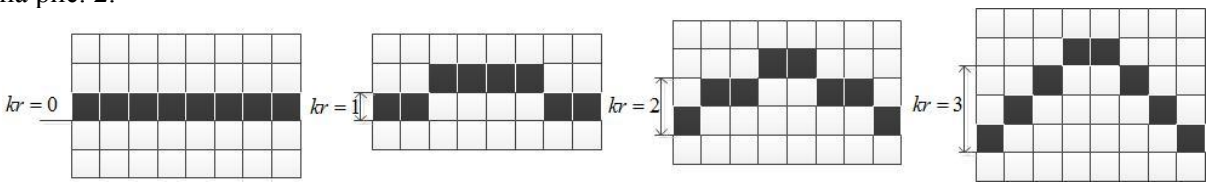
В отличие от известных методов оценки качества выделения линий (метода Прэтта [13], метрики ошибок Баддели [14], ро-коэффициента Григореску [15]) формфактор имеет существенно меньшую вычислительную сложность, что позволяет строить быстрые алгоритмы обработки изображения на его основе.

Дисперсия формфактора рассчитывается для множества изображений одной и той же линии при повороте на различные углы с помощью выражения

$$D = \frac{\sum_{i=1}^{n} \left( F_i - \overline{F} \right)^2}{n},$$

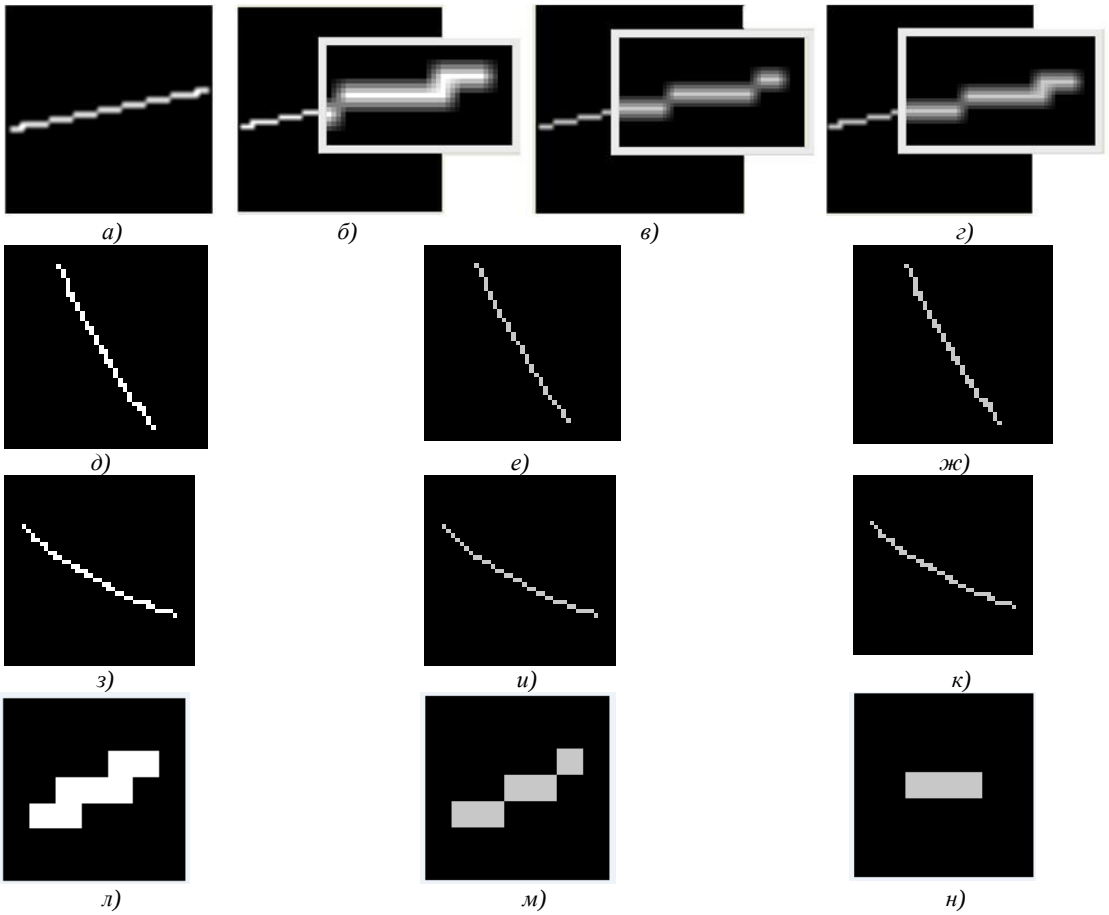
где $n$ – количество линий; $\overline{F}$ – значение формфактора линии, усредненное по углам поворота изображения.

Для искусственно созданных линий использованы размеры 5, 11, 15, 25, 41, 65 и 101 пиксел с кривизной $kr = 0..3$ (рис. 1) относительно идеальной прямой линии. Каждая линия поворачивалась на угол от 0 до 180° относительно центра изображения. Примеры исходной линии и линии, обработанной разработанным алгоритмом и алгоритмом Зонга – Суня, представлены на рис. 2.




*а)* *б)* *в)* *г)*

Рис. 1. Кривизна линии длиной восемь пикселов: *а)* $kr = 0$; *б)* $kr = 1$; *в)* $kr = 2$; *г)* $kr = 3$



*а)* *б)* *в)* *г)*

*д)* *е)* *ж)*

*з)* *и)* *к)*

*л)* *м)* *н)*

Рис. 2. Результаты работы алгоритма нормализации линии по толщине: *а)* исходная линия 1 длиной 41 пиксел при $k = 0$ и повороте на 11°; *б)* исходная линия 1 с выделенным проблемным участком; *в)* результат обработки линии 1 разработанным алгоритмом; *г)* результат обработки линии 1 алгоритмом Зонга – Суня; *д)* исходная линия 2 длиной 41 пиксел при $k = 2$ и повороте на 120°; *е)* результат обработки линии 2 разработанным алгоритмом; *ж)* результат обработки линии 2 алгоритмом Зонга – Суня; *з)* исходная линия 3 длиной 41 пиксел при $k = 3$ и повороте на 150°; *и)* результат обработки линии 3 разработанным алгоритмом; *к)* результат обработки линии 3 алгоритмом Зонга – Суня; *л)* исходная линия 4 длиной пять пикселов при повороте на 26°; *м)* результат обработки линии 4 разработанным алгоритмом; *н)* результат обработки линии 4 алгоритмом Зонга – Суня

Из рис. 2 следует, что алгоритм Зонга – Суня оставляет больше избыточных пикселов, а это влияет на вычисление формфактора. На рис. 2, *л–н* показано, что для алгоритма Зонга – Суня характерны искажения линий.

Графики зависимости дисперсии формфактора от кривизны линии различной длины приведены на рис. 3. Для линии длиной пять пикселов использована кривизна $kr = 0..1$ пиксела, для линии длиной 11 пикселов – $kr = 0..2$ пиксела, для линий длиной 15, 25, 41, 65 и 101 пиксел – $kr = 0..3$ пиксела.

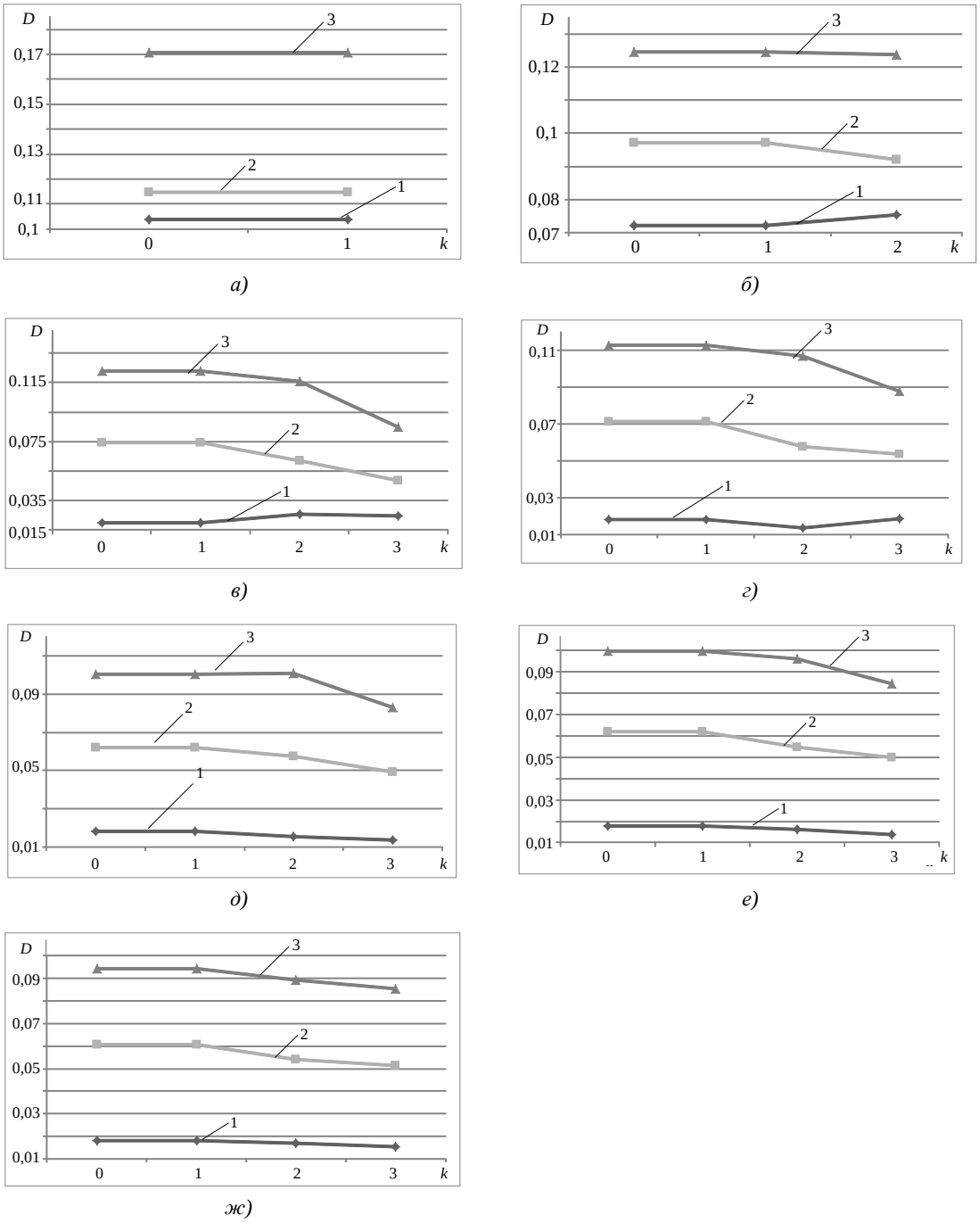
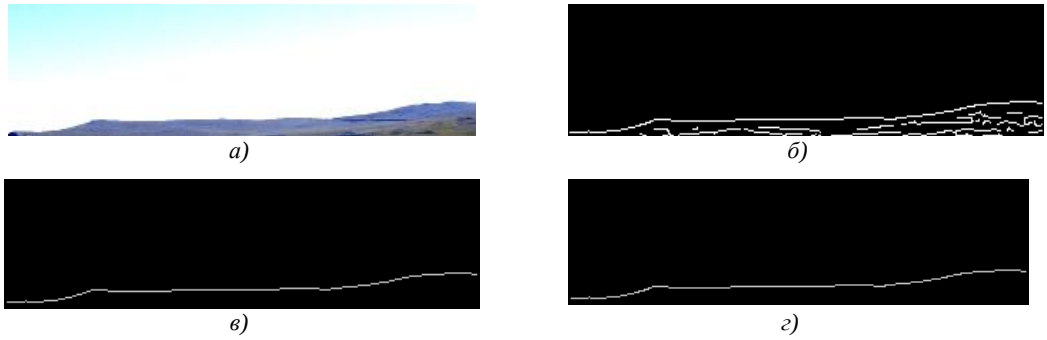



Рис. 3. Зависимости дисперсии формфактора от кривизны линии при длине линии:
*а)* 5 пикселов; *б)* 11 пикселов; *в)* 15 пикселов; *г)* 25 пикселов; *д)* 41 пиксел; *е)* 65 пикселов; *ж)* 101 пиксел;
1 – при использовании алгоритма нормализации линии по толщине;
2 – при использовании алгоритма Зонга – Суня;
3 – без применения алгоритмов скелетизации

Из рис. 3 следует, что использование методов нормализации и скелетизации уменьшает дисперсию формфактора при повороте искусственной линии. При применении разработанного алгоритма нормализации дисперсия $D$ формфактора в зависимости от длины и кривизны линии в 1,6–4,2 раза меньше по сравнению с алгоритмом скелетизации Зонга – Суня и в 3,4–7,8 раз меньше по сравнению с вариантом без использования алгоритмов нормализации или скелетизации. С увеличением кривизны $k$ линии дисперсия $D$ для алгоритма нормализации линий по толщине незначительно увеличивается или уменьшается в зависимости от длины линии, а при использовании алгоритма Зонга – Суня или варианта без использования алгоритмов нормализации и скелетизации увеличивается в среднем на 10 % для искусственных линий.

Результаты работы алгоритма нормализации контурных линий по толщине на реальном изображении представлены на рис. 4.



*а)* *б)* *в)* *г)*

Рис. 4. Результаты нормализации линии реального изображения по толщине:
*а)* тестовое изображение; *б)* тестовое изображение, обработанное алгоритмом Канни; *в)* результат обработки линии разработанным алгоритмом; *г)* результат обработки линии алгоритмом Зонга – Суня

Приведем значения дисперсии формфактора линии реального изображения (рис. 4, *а*) при повороте изображения в графическом редакторе на угол от 0 до 180° относительно центра изображения с шагом в 15°, полученные с помощью следующих методов:

алгоритма нормализации линий по толщине – 0,019 484;

алгоритма Зонга – Суня – 0,103 612;

без нормализации – 0,166 342.

Видно, что разработанный алгоритм нормализации контурных линий по толщине для реальных изображений улучшает значение формфактора в 5,3 раза по сравнению с алгоритмом Зонга – Суня и в 8,5 раза по сравнению с вариантом без использования алгоритмов нормализации и скелетизации линий.

Оценка среднего времени (усреднение по 180 углам поворота изображения) выполнения разработанного алгоритма по сравнению с алгоритмом Зонга – Суня и среднее время сегментации (усреднение по 180 углам поворота изображения) для искусственной линии приведена в табл. 1 (алгоритм нормализации контурных линий по толщине используется после сегментации, а алгоритм Зонга – Суня – до).

Таблица 1

Среднее время выполнения алгоритмов нормализации и сегментации
для искусственной линии различной длины при повороте изображения

| Длина линии, пиксел | Алгоритм нормализации линий по толщине | | Алгоритм Зонга – Суня | |
|---|---|---|---|---|
| | Время выполнения, мс | Время сегментации, мс | Время выполнения, мс | Время сегментации, мс |
| 5 | 0,057 0065 | 0,038 7545 | 0,110 769 | 0,039 522 |
| 11 | 0,101 445 | 0,062 543 67 | 0,355 2903 | 0,063 474 |
| 15 | 0,130 6015 | 0,078 612 | 0,593 886 | 0,080 837 |
| 25 | 0,195 1233 | 0,116 095 25 | 1,560 834 | 0,118 144 |
| 41 | 0,293 3203 | 0,179 705 | 4,119 6445 | 0,183 963 |
| 65 | 0,445 7818 | 0,290 381 | 10,051 333 | 0,292 878 |
| 101 | 0,668 9235 | 0,477 1175 | 24,203 958 | 0,481 981 |

Из табл. 1 видно, что среднее время сегментации для обоих алгоритмов примерно одинаково, при этом среднее время выполнения разработанного алгоритма уменьшается в 1,9–36 раз в зависимости от длины линии по сравнению с алгоритмом Зонга – Суня при обработке искусственной линии. В результате общее среднее время выполнения разработанного алгоритма уменьшается до 21,5 раза по сравнению с алгоритмом Зонга – Суня.

В табл. 2 приведено среднее время (усреднение по 12 углам поворота изображения) выполнения алгоритма сегментации и алгоритмов нормализации и скелетизации для линии изображения на рис. 4, *а*.

Таблица 2

Среднее время выполнения алгоритмов сегментации, нормализации и скелетизации линии реального изображения при его повороте

| Время, мс | Алгоритм нормализации линий по толщине | Алгоритм Зонга – Суня |
|---|---|---|
| Нормализации | 3,203 058 | 92,422 05 |
| Сегментации | 1,756 553 | 1,980 582 |

Из табл. 2 следует, что среднее время сегментации для обоих алгоритмов примерно одинаково, при этом среднее время выполнения разработанного алгоритма до 29 раз меньше по сравнению с алгоритмом Зонга – Суня для линии реального изображения, а общее среднее время выполнения до 19 раз меньше.

Для оценки работы алгоритма нормализации линий по толщине в реальных условиях использованы полутоновые изображения размером $3920 \times 2204$ пиксела, полученные с помощью поворачиваемой фотокамеры. В качестве критериев оценки применимы стабильность сегментации линий при повороте камеры (определяемая числом выделяемых линий при различных углах поворота) и время обработки изображения:

$$F(\alpha) = \left( f(\alpha_i) \cdot 100\,\% \right) / f(\alpha_1),$$

где $f(\alpha_1)$ – множество прямых линий с заданным значением формфактора на эталонном изображении при повороте камеры на угол $\alpha_1 = 0$; $f(\alpha_i)$ – множество прямых линий с заданным значением формфактора на изображении $i$ при повороте камеры на угол $\alpha_i$°. Линии на изображениях выделены по значению формфактора в пределах 0,8–1,2.

На рис. 5 показаны зависимости стабильности сегментации линий на изображении от угла поворота камеры при использовании алгоритма нормализации линий по толщине и алгоритма Зонга – Суня и без использования каких-либо алгоритмов нормализации. Из рисунка следует, что в зависимости от угла поворота камеры использование разработанного алгоритма повышает стабильность $F(\alpha)$ количества выделенных прямых линий по заданному формфактору в 1,04–1,62 раза по сравнению с алгоритмом Зонга – Суня и в 1,52–2,3 раза по сравнению с вариантом без нормализации или скелетизации линий (количество выделенных прямых линий увеличивается до 186 раз).

Оценка среднего времени (усреднение по шести-семи углам поворота изображения) выполнения алгоритма сегментации и нормализации изображений, полученных с помощью поворачивающейся фотокамеры, представлена в табл. 3.

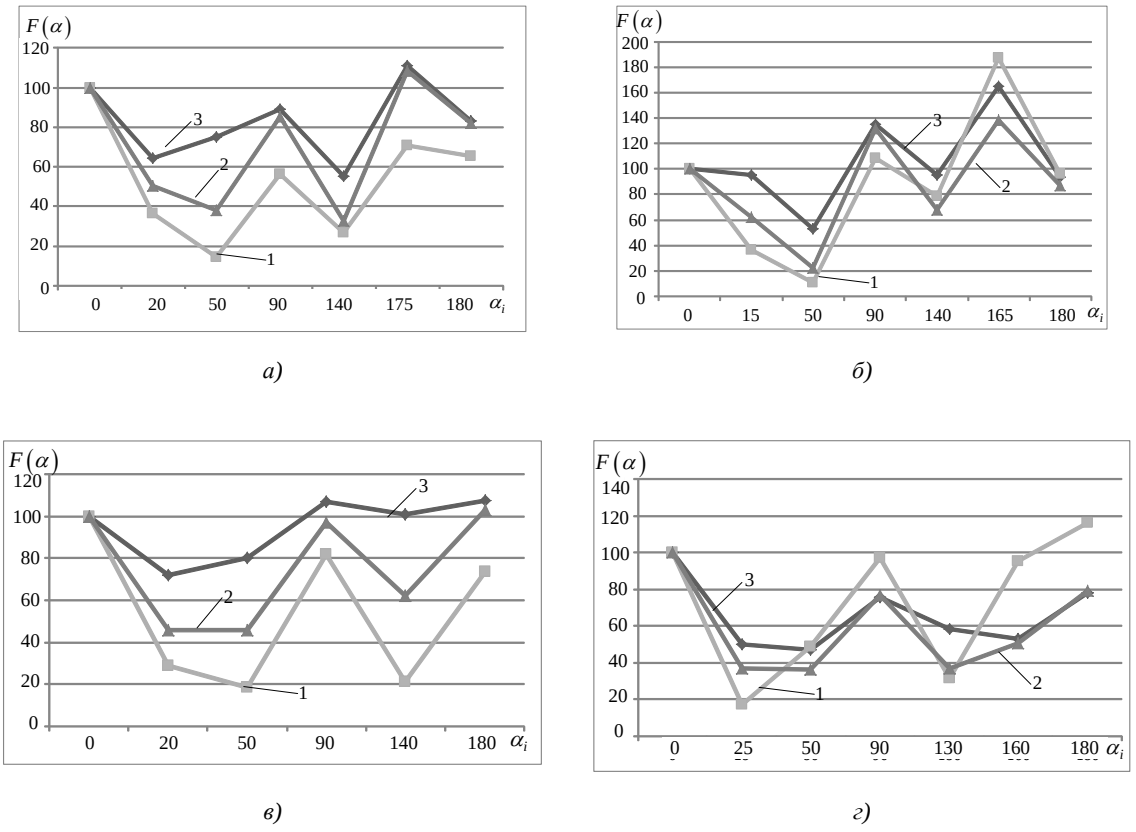



Рис. 5. Зависимости числа линий от угла поворота изображений:
*а)* для изображения 1; *б)* для изображения 2; *в)* для изображения 3; *г)* для изображения 4;
1 – без применения алгоритмов скелетизации;
2 – при использовании алгоритма Зонга – Суня;
3 – при использовании алгоритма нормализации линии по толщине

Таблица 3

Время выполнения алгоритмов сегментации, нормализации и скелетизации линий при повороте реального изображения размером $3920 \times 2204$ пиксела

| **Изображение 1** | | Время, мс | **Изображение 3** | | Время, мс |
|---|---|---|---|---|---|
| Алгоритм Зонга – Суня | Сегментация | 32,594 | Алгоритм Зонга – Суня | Сегментация | 10,320 |
| | Скелетизация | 244,094 | | Скелетизация | 233,619 |
| Алгоритм нормализации | Сегментация | 34,209 | Алгоритм нормализации | Сегментация | 10,845 |
| | Скелетизация | 8,408 | | Скелетизация | 2,300 |
| **Изображение 2** | | Время, мс | **Изображение 4** | | Время, мс |
| Алгоритм Зонга – Суня | Сегментация | 14,191 | Алгоритм Зонга – Суня | Сегментация | 39,870 |
| | Скелетизация | 160,262 | | Скелетизация | 507,012 |
| Алгоритм нормализации | Сегментация | 14,912 | Алгоритм нормализации | Сегментация | 40,953 |
| | Скелетизация | 5,336 | | Скелетизация | 6,477 |

В табл. 3 показано, что среднее время выполнения разработанного алгоритма в 29– 101 раз меньше времени выполнения алгоритма Зонга – Суня при примерно одинаковом среднем времени сегментации. В результате общее среднее время выполнения разработанного алгоритма уменьшается до 18,5 раз по сравнению с алгоритмом Зонга – Суня.

**Заключение**

Разработаны метод и алгоритм нормализации контурных линий по толщине на бинарных изображениях на основе анализа локальных ориентаций их фрагментов. Показано, что для искусственных линий, построенных и повернутых в графическом редакторе, а также линии, выделенной на изображении, которое получено с помощью фотокамеры и повернуто в графическом редакторе, предложенный метод обеспечивает уменьшение дисперсии форм-фактора линий до 4,2 и 5,3 раза по сравнению с методом скелетизации Зонга – Суня и до 7,8 и 8,5 раза по сравнению с вариантом без использования методов нормализации или скелетизации соответственно. Предложенный метод по сравнению с методом Зонга – Суня позволяет уменьшить время обработки до 36 и 29 раз для искусственных линий и линий, выделенных на реальных изображениях. Установлено, что для линий, выделенных на фотографиях с поворачиваемой камеры, предложенный метод обеспечивает повышение стабильности сегментации линий до 1,6 раза по сравнению с методом скелетизации Зонга – Суня и до 2,3 раза по сравнению с вариантом без использования методов нормализации или скелетизации. При этом предложенный метод по сравнению с методом Зонга – Суня позволяет уменьшить время обработки до 100 раз.

**Список литературы**

1. Canny, J. A Computational Approach to Edge Detection / J. Canny // IEEE Trans. Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, no. 6. – P. 679–698.

2. Roberts, L. Machine Perception of 3D Solids / L. Roberts // Optical and Electro-optical Information Processing. – 1965. – Vol. 1. – P. 159–197.

3. Lam, L. Thinning Methodologies – a Comprehensive Survey / L. Lam, S.-W. Lee, C.Y. Suen // IEEE Trans. on Pattern Analysis and Machine Intelligence. – 1992. – Vol. 14, no. 9. – P. 869–885.

4. Chen, W. Improved Zhang – Suen thinning algorithm in binary line drawing applications / W. Chen // IEEE International Conference on Systems and Informatics 2012. – Yantai, 2012. – P. 1947–1950.

5. Абламейко, С.В. Полутоновое утоньшение цветного изображения / С.В. Абламейко, А.М. Недзьведь // Цифровая обработка изображений : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1998. – № 2. – С. 41–51.

6. Nedzved, A. Thinning of the Gray-Scale and Color Images by Sequential Analysis of the Binary Layers / A. Nedzved, S. Ablameyko // Pattern Recognition and Image Analysis. – 2000. – Vol. 10, no. 2. – P. 226–235.

7. Color Thinning with Applications to Biomedical Images / A. Nedzved [et al.] // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – 2001. – Vol. 2124. – P. 256–263.

8. Молчанова, В.С. Решение задачи топологического утончения объектов бинарного растра с использованием специализированных агентов / В.С. Молчанова, И.С. Грунский // Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ–2013). – 2013. – Vol. 12. – С. 743–747.

9. Клубков, И.М. Применение волнового алгоритма для нахождения скелета растрового изображения / И.М. Клубков // Вестник ДГТУ. – 2001. – Т. 1, № 1(7). – С. 126–133.

10. Щепин, Е.В. К топологическому подходу в анализе изображений / Е.В. Щепин, Г.М. Непомнящий // Геометрия, топология и приложения : межвузовский сб. науч. тр. / М-во высшего и средн. спец. образования РСФСР. – 1990. – С. 13–25.

11. Shih, F.Y. Automatic seeded region growing for color image segmentation / F.Y. Shih, S. Cheng // Image and Vision Computing. Newark. – 2005. – No. 23. – P. 877–886.

12. Бородина, О.Г. Выделение изолированных прямых линий на изображениях с использованием форм-фактора / О.Г. Бородина, В.Ю. Цветков // Изв. СПбГЭТУ «ЛЭТИ». – 2015. – № 1. – С. 41–45.

13. Прэтт, У. Цифровая обработка изображений / У. Прэтт. – М. : Мир, 1982. – 312 с.

14. Baddeley, A.J. An error metric for binary images / A.J. Baddeley // Robust Computer Vision: Quality of Vision Algorithms. – Amsterdam : Centre for mathematics and computer science, 1992. – P. 59–78.

15. Grigorescu, C. Contour detection based on nonclassical receptive field inhibition / C. Grigorescu // IEEE Trans. on Image Processing. – 2003. – Vol. 12, no. 7. – P. 729–739.

16. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1007 с.

**Поступила 13.06.2016**

*Белорусский государственный университет информатики и радиоэлектроники,*
*Минск, ул. П. Бровки, 4*
*e-mail: vtsvet@bsuir.by*

**A.G. Shauchuk, V.Yu. Tsviatkou**

## NORMALIZATION OF CONTOUR LINES IN THICKNESS BASED ON ANALYSIS OF LOCAL ORIENTATION OF THEIR FRAGMENTS

We propose a method of normalization in thickness of the double-ended contour lines based on the analysis of local orientations of fragments for binary images. Comparison of the proposed method with known methods of thinning is held. It is shown that the proposed method is superior to the known methods of thinning on speed and quality.