АКАДЕМИЯ НАУК БССР
Институт технической кибернетики

АВТОМАТИЗАЦИЯ ТЕХНИЧЕСКОЙ ПОДГОТОВКИ ПРОИЗВОДСТВА

(направление: "Вычислительная техника в машиностроении)

научно-технический сборник

Выпуск 1У

Минск
1977

УДК 681.3.06:51

В.А. Малышев, Г.Н. Плотницкий

АЛГОРИТМ ФИЛЬТРАЦИИ ЛИНИЙ ШТРИХОВЫХ ИЗОБРАЖЕНИЙ ПО ТОЛЩИНЕ

Рассматривается одна математическая модель следующей задачи: на изображении имеется несколько линий различной толщины, требуется классифицировать эти линии по толщине, например, отфильтровать их, оставив только линии наибольшей толщины. В рассматриваемой модели указывается алгоритм решения этой задачи и доказывается его оптимальность в смысле надежности и порядка времени его работы.

Штриховые изображения представляют широкий класс объектов, возникающих при обработке информации в ядерной физике, машиностроения, картографии и т.д.

Рассмотрим одну математическую модель следующей задачи: на изображении имеется нескол[illegible] линий р[illegible] толщины, требуется классифицировать эти [illegible], отфильтровать их, оставив только линии [illegible] толщины.

В формулируемой ниже модели указываем алгоритм решения этой задачи и доказываем его оптимальность в смысле надежности и порядка времени работы.

1. Описание модели. Пусть $\tilde{\Omega}=\{x=(x_1, x_2): 1 \leqslant x_i \leqslant N, i=1,2\}$ – квадрат, а Ω – множество его целочисленных точек (N^2 штук). Изображение на Ω есть просто функция $\mathfrak{I}(x)$, принимающая два значения – 0 и 1. Будем интересоваться классом изображений следующего вида:

$$\mathfrak{I}(x) = (\mathfrak{I}_1(x) \vee \ldots \vee \mathfrak{I}_n(x)) \oplus \mathfrak{I}_0(x),$$

где $\vee$ – дизъюнкция;

$\oplus$ – сумма по $mod\, 2$;

$\mathfrak{I}_i(x)$ – i -я линия;

$\mathfrak{I}_0(x)$ – случайная помеха.

Точнее, $\mathfrak{I}_i(x)$, $i=1,\ldots,n$ получается следующим образом. В квадрате Ω проводится отрезок прямой или дуга окружности L_i. $\mathfrak{I}_i(x)$ полагается равным 1, если находиться от L_i на расстоянии не большем δ_i и 0 – в противном случае. Пусть $\delta_1 > \delta_2 > \cdots > \delta_n$.

Выберем здесь один из простейших вариантов помехи $\mathfrak{N}_0(X)$. Будем считать ее случайной величиной со значением 0 и 1 в каждой точке $X \in Q$, а независимой – в разных точках. Обозначим $p(X)$ вероятность того, что $\mathfrak{N}_0(X)=1$, т.е. что точка X искажается. Уточним $p(X)$ позже.

2. <u>Постановка задачи</u>. Фиксируем некоторое число $\delta_k > \delta > \delta_{k+1} > 0$. Требуется найти алгоритм, строящий по $\mathfrak{N}(X)$ изображение $(\mathfrak{N}_1(X) \vee \ldots \vee \mathfrak{N}_k(X)) \oplus \mathfrak{N}'(X)$, где $\delta_k > \delta > \delta_{k+1}$, т.е. отфильтровать все линии толщины меньшей δ. $\mathfrak{N}'(X)$ есть некоторое допустимое отклонение, равное 1 не более чем в $N_1 < N^2$ точках.

Для того, чтобы иметь возможность сформулировать точные математические результаты об описываемом ниже алгоритме, необходима асимптотическая постановка задачи:

1) общая площадь, занимаемая линиями, мала в сравнении с площадью квадрата. Точнее $N \to \infty$, $\frac{n}{N} \to 0$, $\delta_i = Const$;

2) длина l_{ij} каждого куска линии L_i, на котором нет пересечений с этой или другими линиями, сравнима с N, т.е. существует такая константа $c > 0$, что $\frac{l_{ij}}{N} > c$. Отсюда следует, что радиус кривизны кривой ограничен снизу числом $\frac{cN}{2\pi}$ (предполагается, что радиус кривизны каждой линии примерно постоянен). Наш алгоритм работает также для произвольных кривых с ограниченным таким образом радиусом кривизны;

3) $p(X) \equiv p$. От этого предположения можно частично отказаться. При этом $N^2 p^{\delta_n} \to 0$;

4) каждая точка линии находится либо на расстоянии меньше 2δ от пересечения этой линии с другой, либо на расстоянии больше 2δ от любой другой линии.

Алгоритм, описываемый ниже, дает [illegible] перебора точек из Q: $X^1, \ldots, X^M$, и попутно совершает некоторые другие операции. Под сложностью алгоритма будем понимать обычную алгоритмическую сложность (например [1]). Как будет видно ниже, сложность определяется числом M. Будет доказано, что для нашего алгоритма $M \asymp N$ и что M не может быть меньше, т.е. приводимый ниже алгоритм оптимален. $M \asymp N$ означает, что $0 < \varliminf_{N \to \infty} \frac{M}{N} \leq \varlimsup_{N \to \infty} \frac{M}{N} < \infty$

3. <u>Описание алгоритма</u>. Рассмотрим прямые в Q вида

$$X_1 = k\left[\frac{cN}{\pi\alpha_0}\right], \; 1 \leq k \leq \frac{\pi\alpha_0}{c} \quad и \quad X_2 = k\left[\frac{cN}{\pi\alpha_0}\right], \; 1 \leq K \leq \frac{\pi\alpha_0}{c}$$

и занумеруем их точки в произвольной удобной для программирования последовательности: $X^1, \ldots, X^{\frac{2\pi}{c}\alpha_0}$, $\alpha_0 = Const > 1$, $\alpha_0 > 1$. Алгоритм последовательно "попадает" в эти точки. При попадании в точку X^i алгоритм проверяет наличие линии около точки X^i по подпрограмме 1. Если линии нет, то он переходит к точке X^{i+1}. Если линия есть, то алгоритм оценивает ширину линии по подпрограмме 2. Если ширина больше δ или имеет место пересечение линий, то алгоритм переходит к точке X^{i+1}.

В противном случае алгоритм начинает отслеживать и стирать линию по подпрограмме 3. После окончания работы подпрограммы 3 алгоритм переходит в точку X^{i+1}, где все начинается сначала. Алгоритм останавливается при попадании в точку $X^{\frac{2\pi}{c}\alpha_0}$.

Подпрограмма 1. Подсчитывает число g единиц в квадратном "окне" со стороной d_1 и с центром в точке X^i. Если $g<\alpha_1\delta_n d_1$, $\alpha_1<1$, то алгоритм переходит в точку X^{i+1}. В противном случае алгоритм начинает работать по подпрограмме 2.

Подпрограмма 2. Выберем ширину d_2 второго окна так, чтобы внутри него линия могла считаться прямой, т.е. d_2 много меньше нижней границы радиуса кривизны.

Замечание. Для окружностей нижняя граница равна $\frac{cN}{2\pi}$. Оценка ширины линии производится статистической оценкой отрезков a и b (рис. 1)

$$a=\frac{1}{d_2}\sum_{i=1}^{d_2}a_i ,$$

где a_i - число единиц в i-й строке окна. Аналогично

$$b=\frac{1}{d_2}\sum_{i=1}^{d_1}b_i ,$$

где b_i - число единиц в i-м столбце окна. Отсюда ширина линии

$$\mathcal{Y}=\frac{ab}{\sqrt{a^2+b^2}} .$$

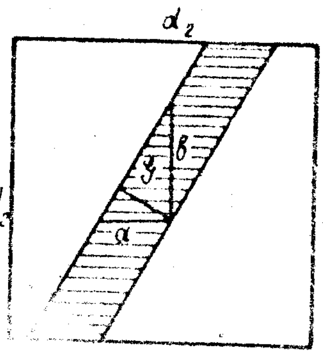


Рис. 1

Для установления наличия пересечения в каждой строке подсчитываются те же величины. Если линия из набора ширин $\delta_1,\dots,\delta_k$ проходит через окно, то эмпирическое $\mathcal{Y}$ будет не меньше δ_k. Если $\mathcal{Y}>\delta_{k+1}+\frac{\delta_{k+1}-\delta_k}{2}$, то алгоритм переходит к точке X^{i+1}. В противном случае начинает работать подпрограмма 3.

Дополнительно тем же способом вычисляется $\mathcal{Y}$ для областей A_i, $i=1,\dots,4$, указанных на рис. 2. Если где-либо окажется, что $\mathcal{Y}<\delta_n$, то алгоритм переходит в точку X^{i+1}. Этим устраняются случаи, указанные на рис. 3.

Замечание. В важном для практики случае, когда $n-k=1$, т.е. имеются тонкие линии только одного типа, подпрограмма 2 отсортировывает все пересечения вообще.

Подпрограмма 3. В нижней строке ищется 1. Пусть она в точке X. От нее на трех отрезках (рис. 4) длины $l=max(a,b)$ измеряется число единиц. Следующая точка берется в конце того отрезка, где число единиц максимально.

Далее процедура повторяется с той разницей, что из внутренней точки измерения производится по четырем взаимно перпендикулярным отрезкам той же длины (рис. 3). Когда дойдем по другой границе в точку y_1, то строим окно ширины d_2 с центром в этой точке, после чего все повторяется сначала, предварительно стирая все в области $A - B$, где A – старое окно, B – новое окно.

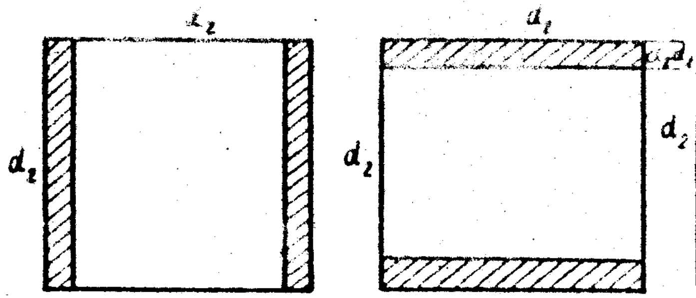


Рис. 2

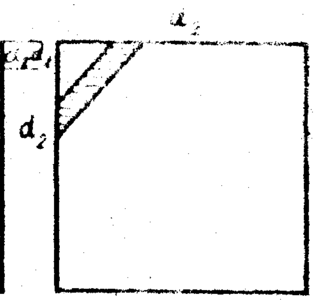
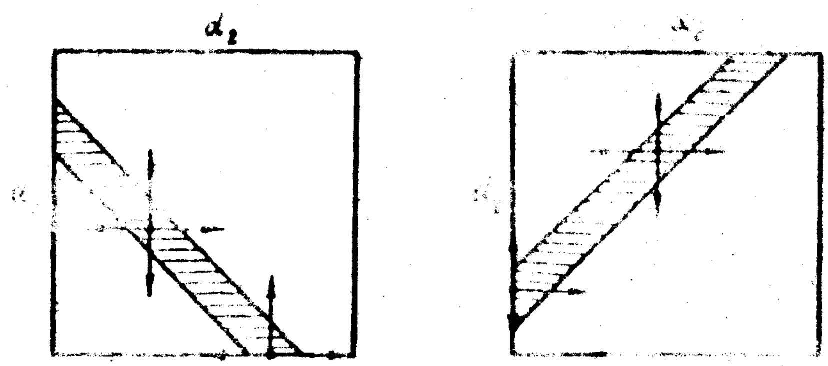


Рис. 3

Далее опять применяем подпрограммы 2 и 3. Если же подпрограмма 2 обнаружит пересечение или линию большей ширины или концевой участок линии, то работает подпрограмма 4. После чего алгоритм переходит к точке x^{i+1}.



П о д п р о г р а м м а 4. Стирает часть тонкой линии, большей ширины d_2. Это несущественная часть и здесь не будет описана.

4. Результаты.

1. Сложность данного алгоритма $T \simeq n$.

2. Вероятность того, что $|X'| \leq |X_0| + Const$, где X – число единиц в изображении $Л$, стремится к 1.

3. Не существует алгоритма сложности L, такой, что ..., обладающего свойством 2.

Иначе говоря, построенный нами алгоритм оптимален в указанном точном смысле этого слова.

Скажем несколько слов о доказательствах. Предположим, что выбрали $\mathfrak{Z}$ точек в Ω для обследования. Так как линия имеет случайное направление, то вероятность того, что хотя бы одна из точек $\mathfrak{Z}$ попадает на эту линию, стремится к нулю при $N \to \infty$. Отсюда нетрудно получается утверждение 3.

Утверждение 1 получается стандартным подсчетом сложности и выполнением всех предельных переходов.

Утверждение 2 доказывается сложнее и здесь его доказывать не будем.

Параметрами алгоритма, подлежащими выбору, являются: N - длина стороны рисунка; C - априорная нижняя граница длины связных компонент изображения; $d_0 \approx 2$ - константа, определяющая число сечений; d_1 - ширина окна в подпрограмме 1; $\alpha_1 \approx \frac{1}{2}$ - константа в подпрограмме 1; d_2 ширина окна в подпрограмме 2; $\alpha_2 \approx \frac{1}{2}$ - константа в подпрограмме 3. Предполагается, что данный алгоритм может быть эффективно применен в задачах ввода фильмовой информации с автоматов на ЭЛТ широкого класса штриховых изображений.

Литература

1. Клосс Б.М., Малышев Б.А. Оценка сложности некоторых классов функций.-"Вестник МГУ", 1965, вып. 4, с. 44-51.

Москва

Алгоритм фильтрации линий по толщине

В.А. Малышев, Уточкин Б.А., Аникеев В.Б., Кумин Ю.М., Плотницкий Г.Н.

Мы рассмотрим одну математическую модель следующей задачи: на изображении имеется несколько линий различной толщины, требуется классифицировать ~~а~~ эти линии по толщине, например, отфильтровать их, оставив только линии наибольшей толщины.

~~Мы~~ В формулируемой ниже модели мы указываем алгоритм решения этой задачи, ~~дог~~ и доказываем его оптимальность в смысле ~~надежности~~ надежности и ~~против помех и~~ порядка времени его работы.

1. ~~Формулировка~~ Описание модели.

Пусть $\tilde{\Omega}$ ~~— целочисленный~~ $= \{ x = (x_1, x_2) : 1 \le x_i \le N \}$ (а Ω — множество ~~по целочисленных точек~~) — ~~целочисленный~~ квадрат. Изображение на Ω есть просто функция $\eta(x)$ на Ω, принимающая два значения, 0 и 1.

Мы будем интересоваться классом изображений следующего вида:

$$\eta(x) = \left(\eta_1(x) \wedge \dots \wedge \eta_n(x) \right) \oplus \eta_0(x)$$

где $\wedge$ — дизъюнкция, $\oplus$ — сумма по mod 2, $\eta_i(x)$ есть i-тая линия, а $\eta_0(x)$ — случайная помеха.

Точнее, $\eta_i(x)$, $i = 1, \dots, n$, получаются следующим образом. В ~~К~~ ~~от~~ квадрате $\tilde{\Omega}$ проводится отрезок прямой или дуга окружности L_i.

$\xi_i(x)$ полагается равным 1, если x отстоит от L_i на расстоянии не большем δ_i, и 0 — в противном случае. Пусть $\delta_1 > \delta_2 > \ldots > \delta_n$.

~~[illegible]~~

Мы выберем здесь один из простейших вариантов помехи $\xi_0(x)$. Мы предположим её независимой в разных точках, ~~но неодинаково распределенной~~. Обозначим $p(x)$, ~~$q(x)=1$~~ — вероятность того, что $\xi_0(x)=1$, т.е. что точка x искажается. Мы уточним $p(x)$ позже.

2. Постановка задачи.

Фиксируем некоторое число $\delta > 0$. Требуется найти алгоритм, строящий по изображению $\xi(x)$ изображение $(\xi_1(x) \wedge \ldots \wedge \xi_k(x)) \oplus \xi'(x)$, где $\delta_k > \delta > \delta_{k+1}$, т.е. отфильтровать все линии толщины меньшей δ. $\xi'(x)$ есть некоторое допустимое отклонение, равное 1 не более чем в N_1 точках.

Для того, чтобы иметь возможность сформулировать точные математические результаты об описываемом ниже алгоритме, необходима асимптотическая постановка. Поэтому мы будем предполагать, что (заметив, что алгоритм работает в любых предположениях, а утверждение о его оптимальности ~~действует~~ доказано только в формулируемых ниже допущениях): (3)

1. Общая площадь, занимаемая линиями, мала в сравнении с площадью $\frac{1}{k}$ квадрата. Точнее

$N \to \infty$, $\frac{n}{N} \to 0$ $\delta_i = \mathrm{const}$.

2. Длина ~~[illegible]~~ (каждого) куска линии L_i, на котором нет пересечения с другими линиями, (l_i, ~~каждого~~) сравнима с ~~[illegible]~~ N, т.е. ~~[illegible]~~ Существует такая константа c, что $\frac{l_i}{N} > c$

(вставка на обороте **)

3. ~~[illegible]~~. ~~[illegible]~~ $p(x) \equiv p$, т.е. помеха максимально случайна. От этого предположения можно частично отказаться. При этом $N^{2}p^{\delta n} \to 0$

На обороте *) вставка

Алгоритм, описываемый ниже, дает правило частичного перебора точек из Ω: $x^1, \dots, x^M$ и попутно совершает некоторые другие операции. Под сложностью алгоритма мы будем понимать обычную ~~сложность~~ алгоритмическую сложность (см., например, /1/). ~~[illegible]~~ Как будет видно ниже, ~~[illegible]~~ ~~порядок~~ сложность примерно определяется ~~[illegible]~~ числом M. ~~[illegible]~~ ~~в смысле точное асимптотическое значение сложности~~ ~~[illegible] асимптотики)~~ Точнее, будет доказано, что $M \asymp^{*} N$ (для нашего алгоритма) и что M не может быть меньше, т.е. приводимый ниже алгоритм оптимален.

*) $M \asymp N$ означает что $0 < \underline{\lim} \frac{M}{N} \le \overline{\lim} \frac{M}{N} < \infty$

*) 4. Каждая точка линии находится либо на расстоянии $\leq 2\delta_1$ от пересечения этой линии с другой либо на расстоянии $> 2\delta_1$ от любой другой линии

**) Отсюда следует, что радиус кривизны кривой ограничен снизу числом $\frac{c}{2\pi}$. Наш алгоритм работает также для произвольных кривых с ограниченным таким образом радиусом кривизны.

3. Описание алгоритма

Рассмотрим прямые в Ω вида:

$x_1 = k\left[\frac{cN}{\pi\alpha_0}\right]$, $1 \le k \le \frac{\pi}{c}$ и $x_2 = k\left[\frac{c}{\pi\alpha_0}\right]$, $1 \le k \le \frac{\pi}{c}$

и занумеруем их точки в ~~некоторой~~ произвольной удобной для программирования ~~форме~~ последовательности:

$$x^1, \dots, x^{\frac{2\pi}{c}\alpha_0}, \quad \alpha_0 = const.$$

Алгоритм последовательно „попадает" в эти точки. При попадании в точку x^i алгоритм проверяет по подпрограмме 1 наличие линии около точки x^i. Если линии нет, то он переходит к точке x^{i+1}. Если линия есть, то алгоритм оценивает ширину линии по подпрограмме 2. Если ширина больше δ или имеет место пересечение линий, то алгоритм переходит к точке x^{i+1}. ~~Если же ширина меньше~~ В противном случае алгоритм начинает отслеживать и стирать линию по подпрограмме 3. После окончания работы подпрограммы 3 алгоритм переходит в точку x^{i+1}, где все начинается сначала.

Алгоритм ~~...~~ останавливается при попадании в точку $x^{\frac{2\pi}{c}\alpha_0}$.

<u>Подпрограмма 1.</u>

Подсчитывает число единиц в квадратном „окне" со стороной d_1 и с центром в точке x^i. ~~...~~ Если $g < \alpha_1 \delta d_1^2$, $\alpha_1 < 1$, то алгоритм переходит в точку x^{i+1}. В противном случае алгоритм начинает работать по подпрограмме 2. О выборе α_1 и d_1 мы скажем ниже.

Подпрограмма 2.

Выберем ширину d_2 второго окна так, чтобы внутри него линия могла считаться прямой, т.е. $d_2 \ll$ ~~радиуса кривизны~~ ширины границы радиуса кривизны { ~~Для ос~~

Замечание. Для окружностей эта ширина границы равна $\frac{cN}{2\pi}$. ~~(*) вставка на обороте~~

~~Пусть~~ ~~Начальная~~ Оценка ширины линии производится статистической оценкой отрезков a и b

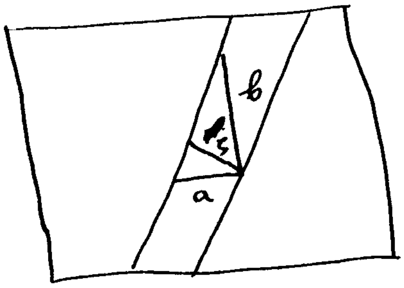


Рис. 1

~~Отрезков~~

$$a = \frac{1}{d_2}\sum a_i$$

где a_i – число единиц в i-той строке окна.

Аналогично $b = \frac{1}{d_2}\sum b_i$, где b_i – число единиц в i-том столбце окна.

Откуда ширина линии

$$\delta = \frac{ab}{\sqrt{a^2+b^2}}$$

~~Также~~ Для устанавливания наличия пересечения в каждой строке подсчитываются те же величины.

~~Рассмотрим, что будет в случае, когда $n=2$,~~
~~причем $\delta_1 = 3\delta_2$ и расположение такое~~

Если линии ширины $\delta_1, \dots, \delta_k$ проходят через окно, то эмпирическое δ будет не меньше δ_k.

Если $\delta > \delta_{k+1} + \frac{\delta_{k+1}-\delta_k}{2}$, то алгоритм переходит к ~~той~~ x^{i+1}. В противном случае начинает работать подпрограмма 3. (*) вставка на обороте x/. Дополнительно тем же способом вычисляется ~~[illegible]~~ c_4 для областей A_i, $i=1,\dots,4$, указанных на рис. 2. Если где либо окажется, что $c_4 < б_4$, то алгоритм переходит в точку x^{i+1}. Этим устраняются случаи, указанные на рис. 5.

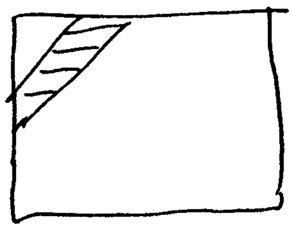


Рис. 5.

~~Подпрограмма 3.~~

Замечание. В важном для практики случае, когда $n-k=1$, т.е. имеются только линии одного типа, подпрограмма 2 ~~отбрасывает~~ отсортировывает все пересечения вообще.

~~Подпрограмма 3.~~

~~Подсчитывает число единиц в полосах ширины~~ в ~~областях~~ A_1, A_2, A_3, A_4. ~~(см. рис. 2)~~

~~длины d_2 и ширины $\alpha_2 d_2$~~, $\alpha_2 d_2$

~~где $\alpha_2 < \frac{1}{2}$ задается отдельно.~~

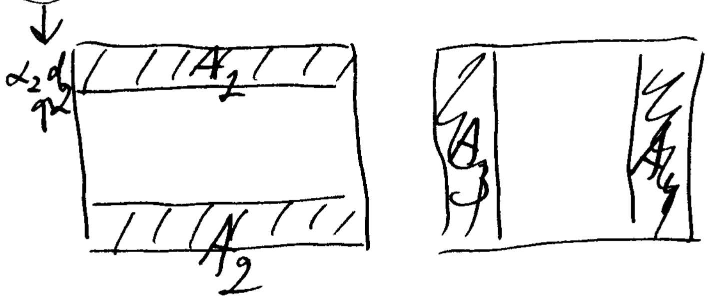


Рис. 2.

Подпрограмма 3 ~~отсортировывания и стирания линии имеет свои особенности для линий, где δ_n маленькая величина~~

В нижней стороне ищется 1 (пусть она в точке x). От нее по трех отрезках (рис. 3) ~~шириной~~ длины $l=\max(a, b)$ измеряется число единиц.

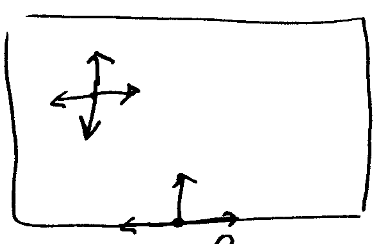


Рис. 3.

Следующая точка берется в конце того отрезка, где число единиц максимально.

Далее процедура повторяется с той разницей, что ~~из~~ из внутренней точки измерения производятся по четырём взаимно перпендикулярным отрезкам той же длины (рис. 3). Когда дойдем до другой границы l_2 в точке y_1, то строим окно ширины d_2 с центром в этой точке, ~~где~~ после чего все повторяется сначала, предварительно стирая ~~тут, их старое окно~~ все в области $A-B$, где A – старое окно, B – новое окно.

~~Применяем~~ Далее опять применяем ~~п~~ подпрограмму 2 и 3. Если же подпрограмма 2 обнаружит пересечение или ~~линию~~ линию большой ширины) или концевой участок линии, то работает подпрограмма 4. После чего алгоритм переходит к точке X^{i+1}.

Подпрограмма 4 стирает ~~всю~~ часть толстой линии, попавшую в окно ширины d_2. Это несущественная часть ~~будет~~ и мы здесь не будем её описывать.

4. Результаты. ~~и доказательства~~ исследования

1. Сложность данного алгоритма
$$\mathcal{L} \asymp N.$$

2. Вероятность того, что ~~удаляемый~~ $\mathcal{K}(A)$ содержит ~~небольшое единиц, чем~~ ~~часть единицы~~
$$|\psi'| \le |\psi_0| + \mathrm{const},$$
где $|\psi|$ — число единиц в изображении ψ, стремится к 1.

3. Не существует алгоритма сложности $\mathcal{L}$ такой, что $\frac{\mathcal{L}}{N} \to 0$, обладающего свойством 2.
Иначе говоря, построенный нами алгоритм оптимален в указанном точном смысле этого слова.

Скажем несколько слов о доказательствах.

Предположим, что мы выбрали $\mathcal{L}$ точек в Ω для обследования. Так как линия имеет случайное направление, то вероятность того, что хотя бы одна из точек $\mathcal{L}$ попадет на эту линию, стремится к нулю при $N \to \infty$. Отсюда нетрудно получается утверждение 3.
Утверждение 1 получается стандартным подсчетом сложности и выполнением всех предельных переходов.

~~Ко~~ Утверждение 2 доказывается сложнее и мы здесь его доказывать не будем.

5. Некоторые практические замечания.

В конкретных случаях алгоритм может быть улучшен за счет оптимального выбора параметров, перечисленных ниже. Общую рецептуру их выбора дать трудно. Поэтому мы рассмотрим следующий численный пример.

Приложение 1

Параметры алгоритма.

N – длина стороны рисунка,

C – априорная нижняя граница длины связных компонент изображения

α_0 – константа, определяющая число сечений

d_1 – ширина окна в подпрограмме 1

α_1 – константа в подпрограмме 1.

d_2 – ширина окна в подпрограмме 2

α_2 – константа в подпрограмме 3

Литература.

1. Клосс Б.М., Малышев В.А. Оценка сложности некоторых классов функций. Вестник МГУ, 1965, вып. 4, стр. 44-51.