Санкт-Петербургский государственный университет
Кафедра моделирования экономических систем

Боброва Александра Алексеевна

Выпускная квалификационная работа бакалавра

Сравнительный анализ методов сглаживания контурных линий на изображении

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Ковшов А. М.

Санкт-Петербург
2016

Содержание

Введение

Одной из важных характеристик любого объекта на изображении является контур этого объекта, так как зачастую именно контур содержит ключевую для распознавания информацию. В связи с этим получило развитие такое направление в области обработки и анализа изображений, как контурный анализ, который охватывает методы выделения и описания контуров, их преобразования и анализа. Благодаря такому подходу, оказывается возможной работа системы в режиме реального времени за счёт сокращения количества вычислений [2].

Обязательным этапом любой системы анализа контурных линий является, конечно же, выделение самих контуров представленных на изображении объектов. В результате применения подобного рода алгоритмов получают представление контура в виде замкнутой ломаной линии, заданной упорядоченным набором точек $C = \{(x_i, y_i)\}_{i=1}^{n}$. Лишь затем возможен переход к решению поставленной задачи. К таким задачам можно отнести задачи распознавания объектов по их форме, поиск похожих изображений и анализ полученного контура с целью получения специфичной для конкретной предметной области информации.

Зачастую, прежде чем перейти к непосредственному решению задачи необходимо специальным образом подготовить контур, например провести генерализацию или сглаживание контура. Алгоритмы сглаживания применяются в случае, когда необходимо устранить шумы, например, при анализе контуров полученных из рентгеновских снимков [3]. Также, алгоритмы сглаживания используюся для придания контуру более эстетичного вида, например в картографии [9].

Достаточно распространена следующая классификация алгоритмов сглаживания [8]:

1. Алгоритмы аппроксимации. Результатом работы такого рода алгоритмов является математическая функция, описывающая геометрический характер сглаживаемой линии. Параметры полученной функции могут быть сохранены и затем использованы для воссоздания контурной линии на произвольном количестве точек.

2. Алгоритмы, использующие различные геометрические отношения между точками. Такие алгоритмы могут убирать ненужные точки из исходного контура и генерировать дополнительные.

3. Алгоритмы на основе усреднения точек. Результатом работы подобного рода алгоритмов является набор точек, размер которого остаётся неизменным. Значения координат точек сглаженного контура получается путём усредения координат соседних точек. Такие алгоритмы сглаживания относительно легко модифицировать.

Далее в работе под алгоритмами сглаживания будут иметься ввиду алгоритмы третьего типа.

Постановка задачи

Задача данной работы заключается в том, чтобы реализовать алгоритмы сглаживания, сравнить и проанализоировать результаты работы этих алгоритмов.

Обзор литературы

Базовые алгоритмы и понятия для основных направлений в области обработки и анализа цифровых изображений представлены в [1]. Эта книга охватывает множество алгоритмов, среди которых алгоритмы улучшения, восстановления, сжатия и сегментации изображений. Несмотря на то, что контурному анализу уделено недостаточно внимания, всё же книга даёт тот фундамент, без которого невозможно дальнейшее плодотворное изучение этого направления.

Более узко специализированным научным трудом является [2]. Однако, в этой книге рассматриваются алгоритмы для комплексозначного представления контура, в котором точка задаётся комплексным числом $a + ib$, где a – это смещение вдоль оси x от фиксированной начальной точки, а b – смещение по y. В связи с этим, книга оказалась полезной лишь с точки зрения основных идей и подходов работы с контурными линиями.

В статье [8] Mansouryar и Hedayati предлагают алгоритм сглаживания, в основе которого, как и в алгоритмах, изложенных в данной работе, лежит фильтр скользящего среднего. Более, того в этой статье представлены классификация и краткий обзор различных подходов к сглаживанию.

Также, в статье [5] предлагается алгоритм устранения шумов из контуров печатных букв для дальнейшего их использования в алгоритмах распознавания. Однако, предложенный метод является специфичным и в общем случае сильно уступает другим подходам.

Hobby в статье [4] предлагает алгоритм сглаживания, похожий на описанный в параграфе 1.1 данной работы. Основное отличие алгоритма Hobby от представленного в этой работе заключается в том, что алгоритм Hobby использует представление контура в виде фиксированной точки (x_0, y_0) и отрезков $e_i = (l_i, d_i)$, где l_i – длина отрезка, d_i – направление этого отрезка.

Глава 1. Реализация алгоритмов сглаживания

Для сравнения результатов работы алгоритмов сглаживания было реализовано соответствующее программное обеспечение. В качестве языка разработки был выбран объектно-ориентированный язык программирования Java [10]. Ключевыми особенностями, повлиявшими на выбор данного языка программирования, являются:

- Широкий выбор обучающих материалов. Помимо превосходной документации [7], которой в большинстве случаев оказывается достаточно, существуют электронные ресурсы [11], официально поддерживаемые компанией Oracle, и проверенные годами учебники и справочники.
- Открытый исходный код стандартной библиотеки. При реализации алгоритмов сглаживания есть необходимость использования классов и методов из стандартной библиотеки. Возможность посмотреть реаилизацию тех или иных классов оказывается полезной, если полученный результат отличается от ожидаемого.
- Современные средства разработки. Интегрированная среда разработки IntelliJ IDEA является передовым интсрументом дла разработки программных продуктов. Данная среда разработки обаладает большим количеством эффективных инструментов, таких как: поиск дублирования кода, интеллектуальные подсказки и многое другое [6]. Такие современные продукты не только упрощают процесс написания кода, но и позволяют уменьшить количество ошибок.
- Простота создания графического интерфейса. Java обладает простой и в то же время исчерпывающей библиотекой для создания пользовательского интерфейса Swing. Современные ИСР языка Java поддерживают возможность визуального редактирования графического интерфейса.
- Кроссплатформенность. Возможность абстрагироваться от особенностей реализации программного обеспечения для различных операционных систем, также упрощает и ускоряет процесс разработки.

С помощью выбранного языка программирования была создана программа с графическим интерфейсом для демонстрации работы алгоритмов (Рис. 1).

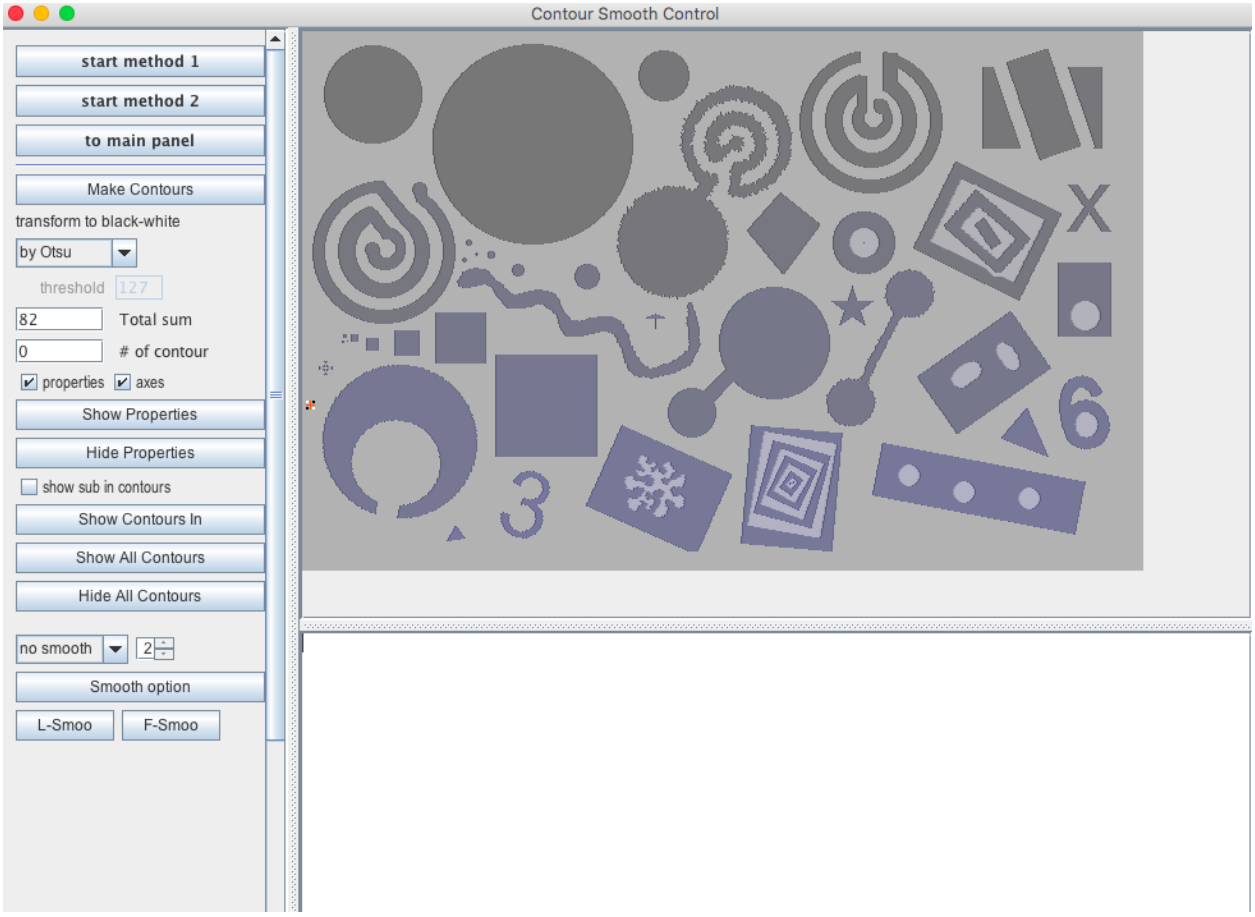


Рис. 1: Графический интерфейс разработанной программы

В общем случае для работы алгоритмов сглаживания не важен способ получения контуров, но в данной работе будут использоваться контуры полученные с помощью метода сегментации контуров.

Как уже было отмечено ранее, в основе представленных в данной работе алгоритмов сглаживания лежит контурный фильтр скользящего среднего [2]:

$$z_i = \frac{\sum_{j=i-k}^{i} z_j}{k} \qquad i = 0, 1, \ldots, n-1$$

1.1. Алгоритм сглаживания по двум близко расположенным точкам

Одним из самых простых алгоритмов сглаживания является сглаживание по двум соседним точкам. На вход алгоритм получает массив $C = \{c_0, c_1, \ldots, c_{n-1}\}$, а возвращает массив $S, |S| = |C|$. Алгоритм заключается в следующем:

1. Проинициализируем $p_l = c_{n-2}, p_c = c_{n-1}, i = 0$. Такая инициализация необходима для того, чтобы корректно обработать точки, расположенные в конце массива.

2. Если p_l и p_c образуют "ступеньку":

$$|p_l.x - p_c.x| = 1 \vee |p_l.y - p_c.y| = 1,$$

то значение координат точки нового сглаженного контура вычисляется следующим образом:

$$s_i.x = \frac{p_l.x + p_c.x}{2}, s_i.y = \frac{p_l.y + p_c.y}{2}.$$

3. Если же условие 2 не выполняется, то точка оставляется:

$$s_i = p_c$$

4. Сдвигаем указатели p_l, p_c на одну точку вправо, увеличиваем счётчик $i = i + 1$ и повторяем шаги 2, 3 пока не посчитаем значения координат для всех точек.

Описанный алгоритм вошёл в разработанную систему и может быть вызван с помощью графического интерфейса при выборе опции `smooth 2` из выпадающего списка.

1.2. Алгоритм сглаживания по трём точкам с коэффициентом

В предыдущем алгоритме на значение координат вычисляемой точки в одинаковой степени влияло как значение координат p_l, так и p_l. Для того, чтобы при сглаживании разряженного контура не произошло сильного искажения значений координат новых точек можно ввести k – вес исходной точки. Так как координаты точки не зависят от индекса в массиве (то есть нет временной зависимости) может оказаться полезным использование симметричного алгоритма.

1. Соответсвует пункту 1 из предыдущего алгоритма.

2. На i-ой итерации проинициализируем переменную $p_n = c_i$. Значение координаты $(i-1)$-ой координаты нового контура будет вычисляться по формуле:

$$s_{i-1}.x = \frac{p_l.x + kp_c.x + p_n.x}{k+2}, s_{i-1}.y = \frac{p_l.y + kp_c.y + p_n.y}{k+2}.$$

3. Увеличим счётчик i и повторим шаг 2 до тех пор, пока не будут посчитаны все точки нового контура.

Алгоритм также был реализован и может быть вызван при помощи выбора опции `smooth 3`.

1.3. Алгоритм сглаживания по семи точкам с учётом углов

Несложно заметить, что описанные ранее алгоритмы будут сглаживать точки на углах, что, в свою очередь, может привести к нежелательным изменениям формы контура, потери его особенностей. Это происходит потому, что при сглаживании внешнего контура часть точек перемещается внутрь контура, а при сглаживании внутреннего контура наоборот – наружу.

Предложенный алгоритм, помимо массива с точками исходного контура и коэффициента k, также принимает на вход значение $b \in \{-2, 1, 2\}$.

1. Проинициализируем $p_{l_3} = c_{n-6}, p_{l_2} = c_{n-5}, p_{l_1} = c_{n-4}, p_c = c_{n-3}, p_{n_1} = c_{n-2}, p_{n_2} = c_{n-1}, i = 0$
2. На i-ой итерации проинициализируем $p_{n_1} = c_i$ и вычислим значения координат по формуле:

$$x_i = \frac{p_{l_3}.x + p_{l_2}.x + p_{l_1}.x + kp_c.x + p_{n_1}.k + p_{n_2}.x + p_{n_3}.x}{k+6}$$

$$y_i = \frac{p_{l_3}.y + p_{l_2}.y + p_{l_1}.y + kp_c.y + p_{n_1}.y + p_{n_2}.y + p_{n_3}.y}{k+6}$$

3. В зависимости от значения b возможны два сценария:
   (a) Если $b = 2$ ($b = -2$), то сглаживаются только точки, которые не перемещаются внутрь (наружу) внешнего (внутреннего) контура. Поэтому если выполняется:

$$\begin{cases} b((c_i.x - c_{i-1}.x)(s_i.y - c_i.y) - (c_i.y - c_{i-1}.y)(s_i.x - c_i.x)) \le 0 \\ b((c_i.x - c_{i+1}.x)(s_i.y - c_i.y) - (c_i.y - c_{i+1}.y)(s_i.x - c_i.x)) \ge 0 \end{cases}$$

   то оставялется прежнее значение:

$$s_i = \begin{cases} c_{i-3} & i > 2 \\ c_{n-i+3} & 0 \le i \le 2 \end{cases}$$

   (b) При прочих значениях b переходим к следующему шагу.
4. Увеличим счётчик i, сдвинем указатели на одну точку вперёд. Шаги 2, 3 повторяются до тех пор пока не будут посчитаны координаты всех точек сглаженного контура.

Опция `smooth 7+` соответствует вызову описанного алгоритма при $b = 2$, если контур внешний, и $b = -2$, если контур внутренний. Соответсвенно, при выборе опции `smooth 7` вызовется реализация этого алгоритма со значением $b = 1$.

1.4. Фиксирование точек, не подлежащих сглаживанию

Другим подходом, способным обеспечить сохранение значений координат точек, расположенных на углах, является фиксирование точек, не подлежащих сглаживанию. Для этого необходимо создать массив из значений булевского типа, в котором на i-ой позиции будет стоять значение false, если $c_i \in C$ сглаживать не нужно, и true в противном случае. Полученный массив может быть использован алгоритмами, описанными выше, при принятии решения о необходимости сглаживания точки на очередной итерации. Алгоритм фиксирования точек выглядит следующим образом:

1. На подготовительном этапе находятся три последовательно идущих в контуре прямых l_p, l_c, l_n.
2. Если эти три прямые не образуют зигзаг, то возможны два сценария:
   (a) Если длины l_p, l_c, l_n больше единицы, то все точки, входящие в прямую l_c помечаются как несглаживаемые.
   (b) Если же длина хотя бы одной из этих прямых равна единице, несглаживаемой помечается точка в середине l_c (или две точки, если длина l_c является чётным числом).
3. Находится следующая прямая l и шаг 2 повторяется для $l_p = l_c, l_c = l_n, l_n = l$ до тех пор пока не достигнут конец массива.

Описанный выше алгоритм, может фиксировать близко расположенные друг к другу точки, что в свою очередь может приводить к устранению острых углов, которые могут быть важными для определённых объектов. Указанный недостаток можно устранить следующим образом:

1. Найдём две подряд идущие в массиве точки c_i, c_{i+1}, которые были помечены как несглаживаемые.
2. Если $c_{i-1}, c_i, c_{i+1}, c_{i+2}$ не образуют зигзаг, то значения c_i, c_{i+1} будет заменено на среднее c_i и c_{i+1}:

$$c_i.x = c_{i+1}.x = \frac{c_i.x + c_{i+1}.x}{2}$$

$$c_i.y = c_{i+1}.y = \frac{c_i.y + c_{i+1}.y}{2}$$

3. Шаги 1, 2 повторяются до тех пор, пока не достигнут конец массива C.

Описанные алгоритмы могут быть вызваны с помощью опций `fix` и `av fix` из графического интерфейса.

Глава 2. Анализ и сравнение

Для анализа алгоритмов сглаживания было использовано тестовое изображение (рис. 2), отображающее ключевые, для сравнения качества работы алгоритмов, особенности объектов на изображениях. Все описанные ранее алгоритмы, как будет показано ниже, обладают разной сглаживающей способностью, которая зависит от типа сглаживаемого контура. В данной работе были представлены наиболее репрезентативные результаты, позволяющие определить случаи для подходящего применения алгоритмов.

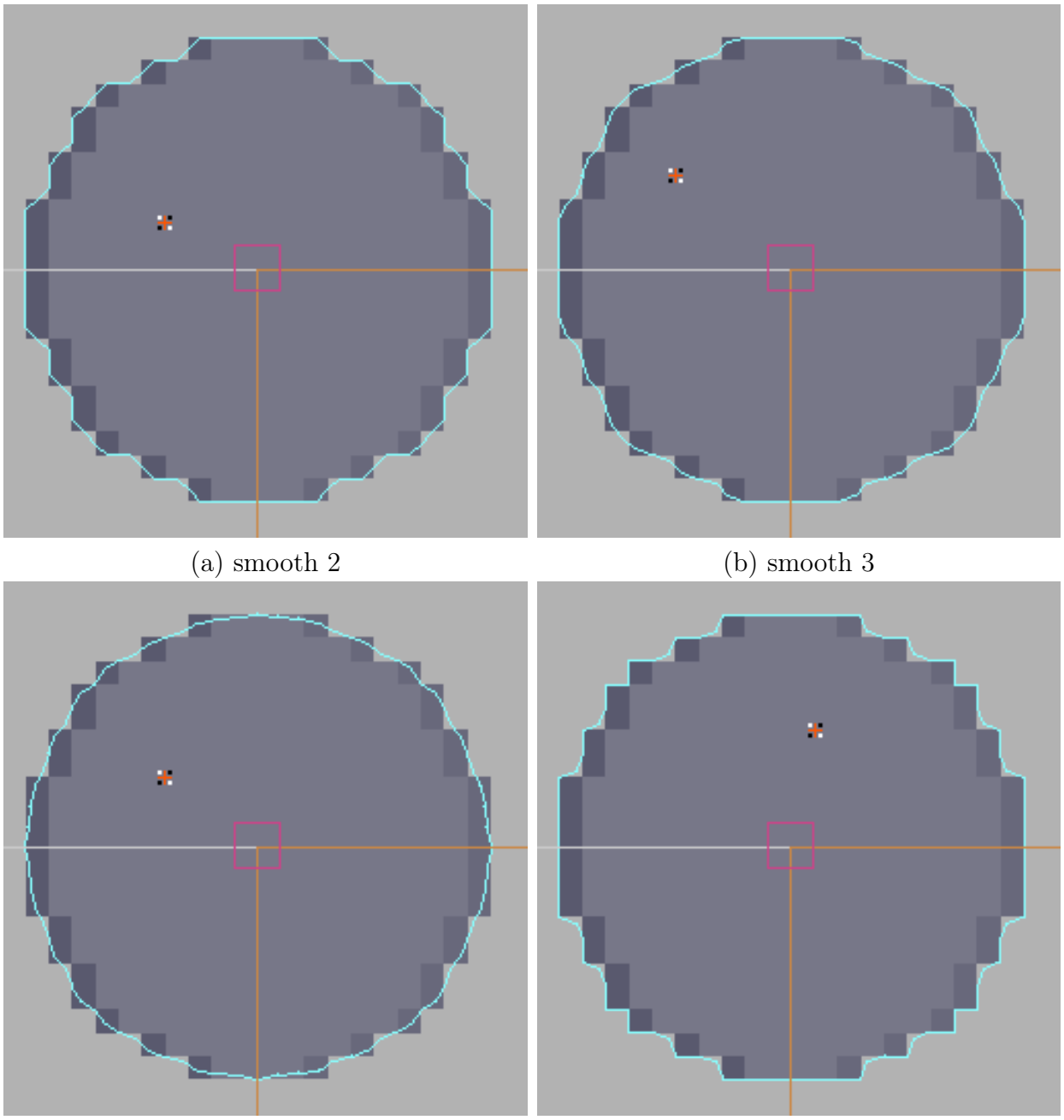
Рис. 2: Тестовое изображение

Качество сглаживания можно оценивать по следующим критериям:

1. **Устранение ступенчатости контура.** Этот критерий является ключевым, так как большинство контуров имеет ступенчатый вид в силу дискретности изображений, с помощью которых они были получены.
2. **Устранение шумов.** Использование чувствительных приборов для получения изображений приводит к шумам в контурных линиях. Алгоримт сглаживания должен уменьшить выраженность шумов.
3. **Сохранение формы углов.** Углы зачастую являются особыми признаками для контуров, поэтому желательно чтобы форма углов в результате сглаживания не изменялась сильно.

Из рис. 3 видно, что для округлых контурных линий наилучший результат показывает алгоритм smooth7, а наихудший – smooth2 и smooth7+ Однако, как видно из рис. 4, использование smooth2, smooth3, smooth7 приводит, в той или иной мере, к сглаживанию углов, что может быть

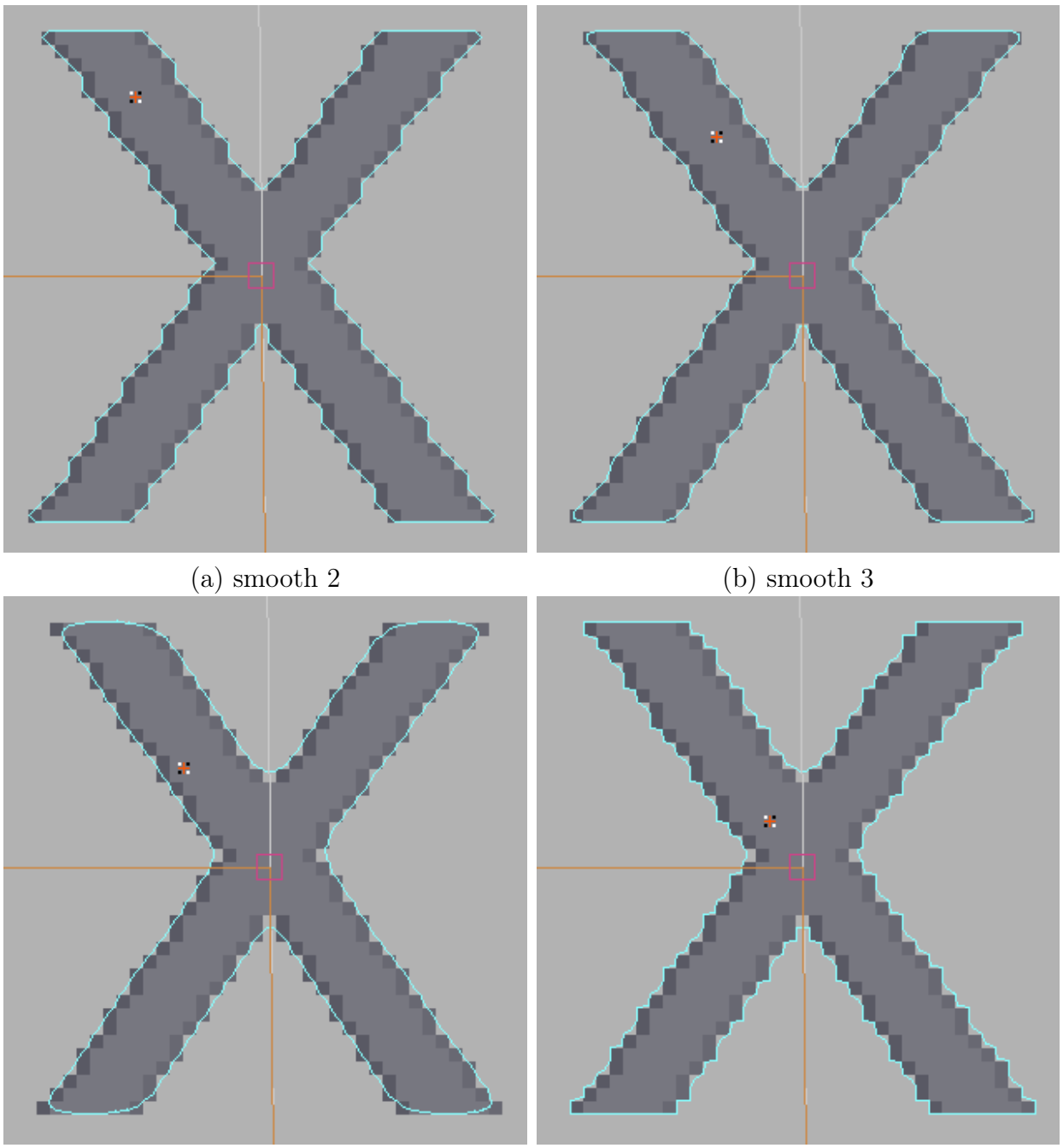
критично для некоторых предметных областей. В свою очередь, smooth7+ оказывается наиболее оптимальным алгоритмом, в случае, если углы являются важной особенностью контура.



(a) smooth 2 (b) smooth 3

(c) smooth 7 (d) smooth 7+

Рис. 3: Для сглаживания округлых контуров лучше использовать smooth 7 или smooth 3.

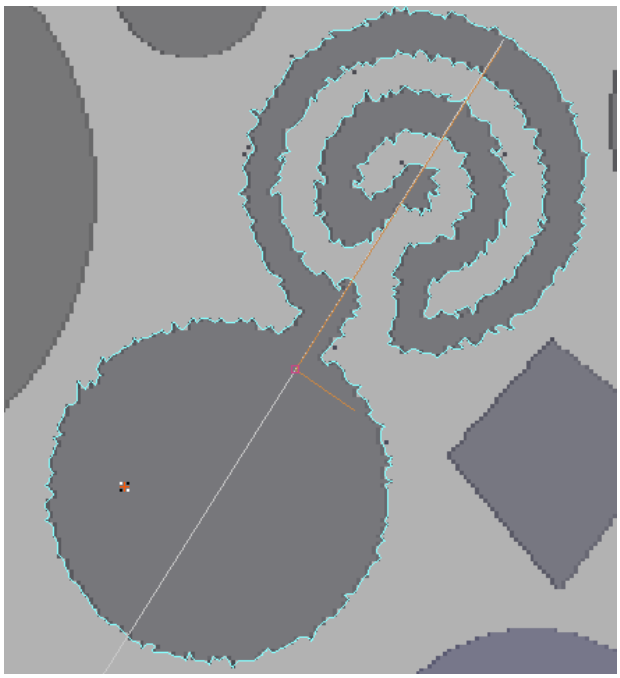


(a) smooth 2 (b) smooth 3

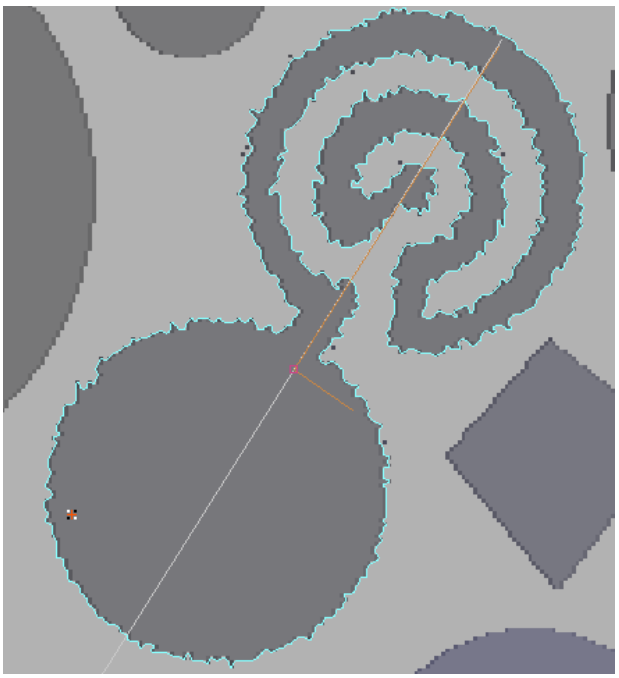
(c) smooth 7 (d) smooth 7+

Рис. 4: Для сглаживания конутров с углами лучше применять smooth 7+.

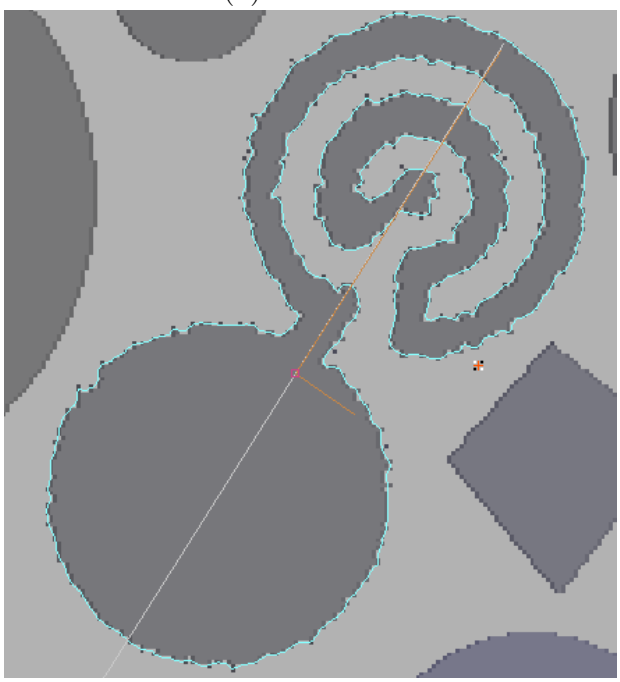
На рис. 3, 4 было приведено использование алгоритмов на незашумлённых, обладающих лишь эффектом ступенчатости, контурах. Зачастую, сглаживание применяется также для уменьшения "зашумлённости"контура. В таком случае, лучше применять алгоритмы smooth3 и smooth7 (рис.5), так как они лучше, по сравнению с smooth7+ и smooth2, справляются с устранением шума.



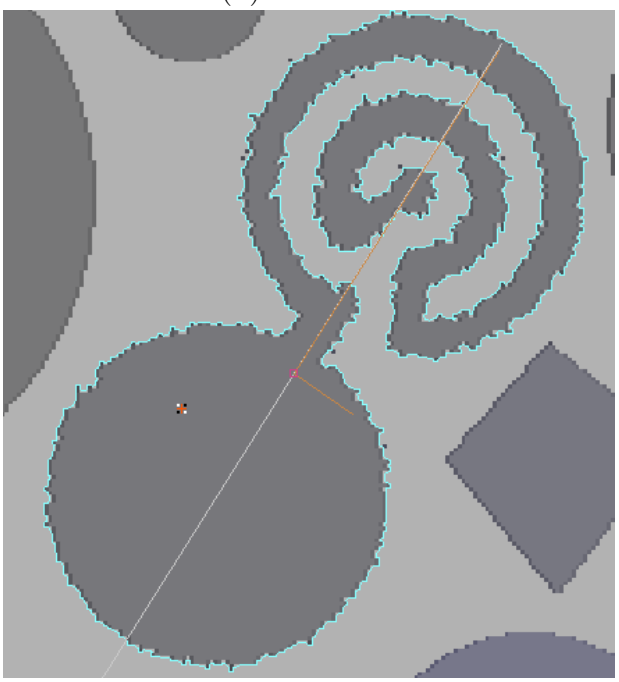
(a) smooth 2



(b) smooth 3



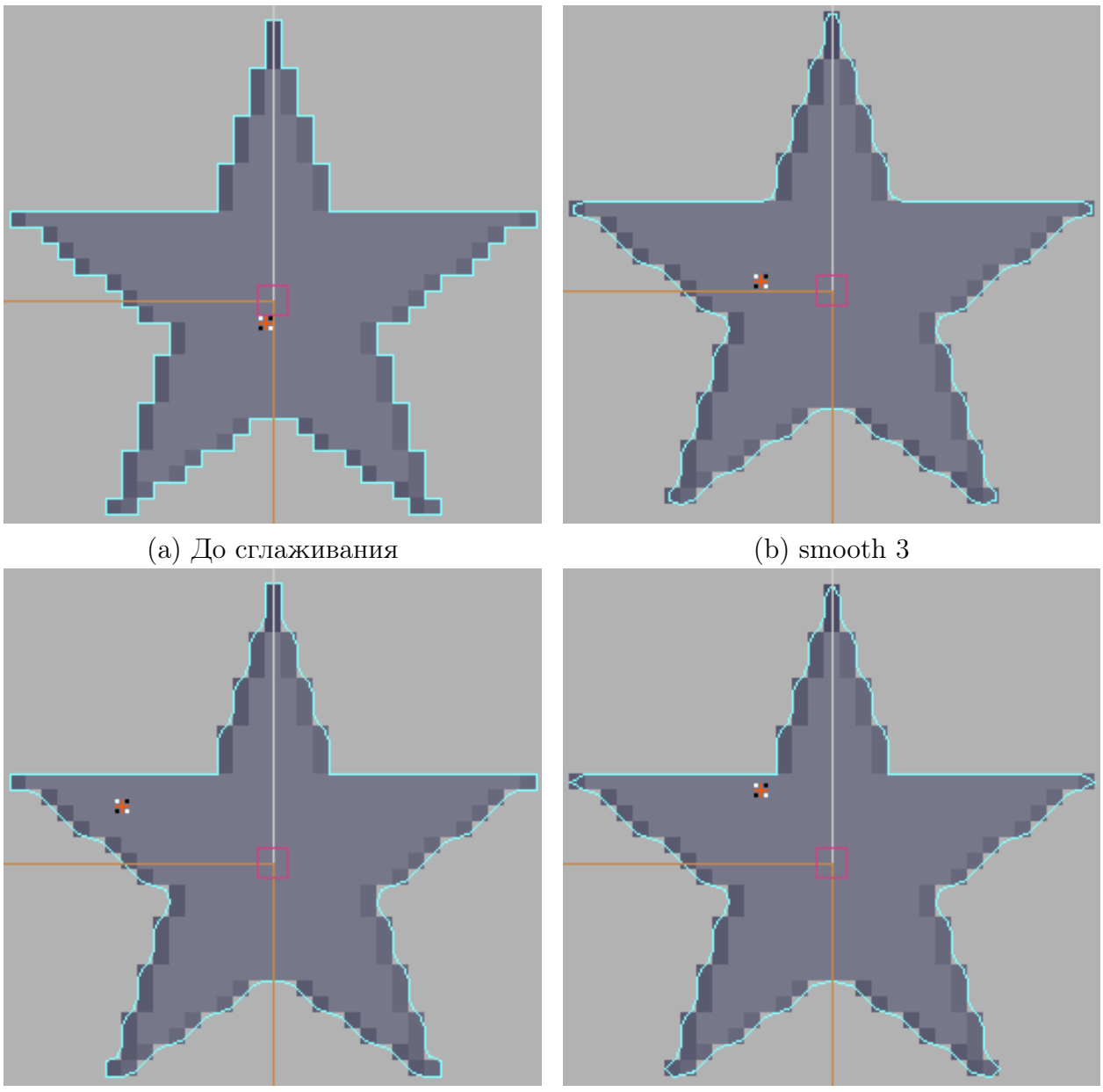
(c) smooth 7



(d) smooth 7+

Рис. 5: Лучше всего с шумом справляется smooth 7. Также, достаточно хороший результат дал smooth 3.

На рис. 6, видно что применение алгоритма smooth3 вместе с fix позволило устранить эффект ступенчатости, при достаточно сохранении углов, которые являются ключевыми особенностями для данного контура.

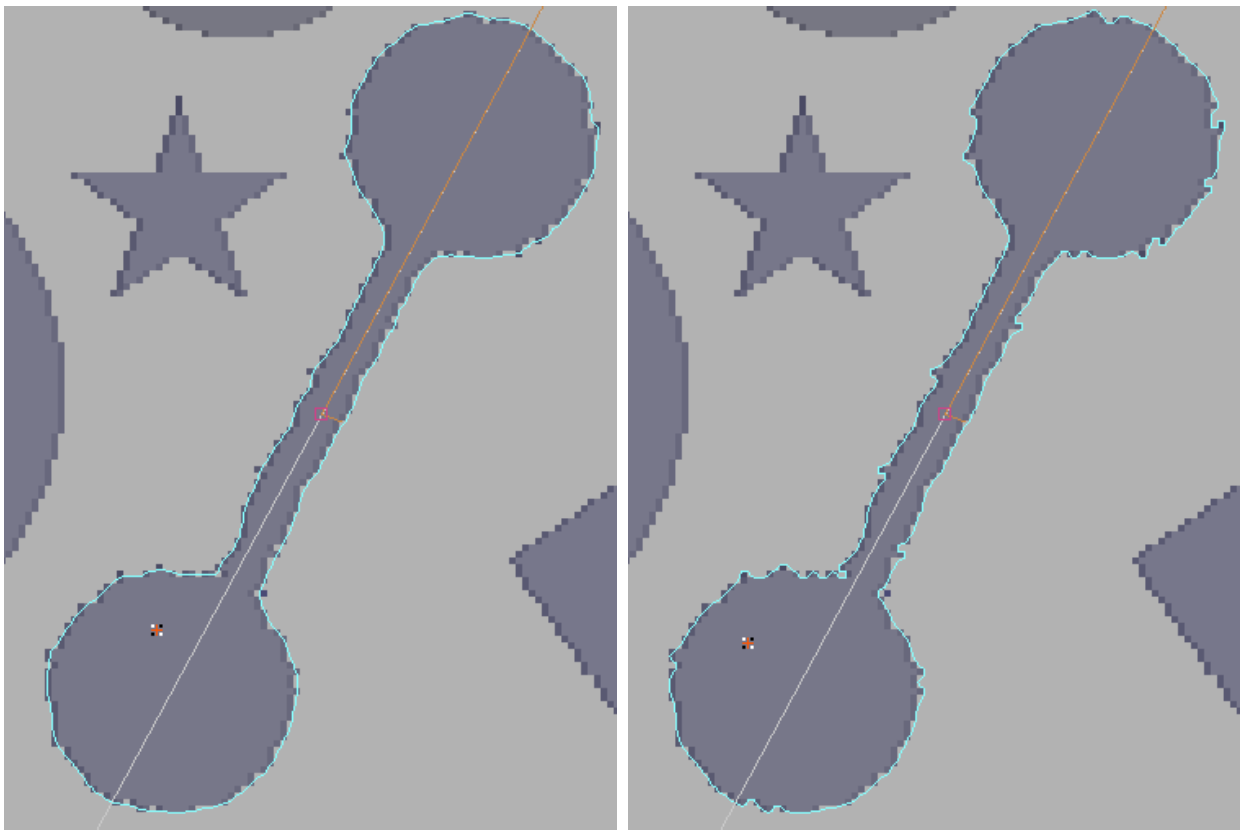


(a) До сглаживания (b) smooth 3

(c) fix, smooth 3 (d) fix, smooth 3, av fix

Рис. 6: Благодаря fix и av fix, после применения алгоритма сглаживания контур лучше сохранил основную форму.

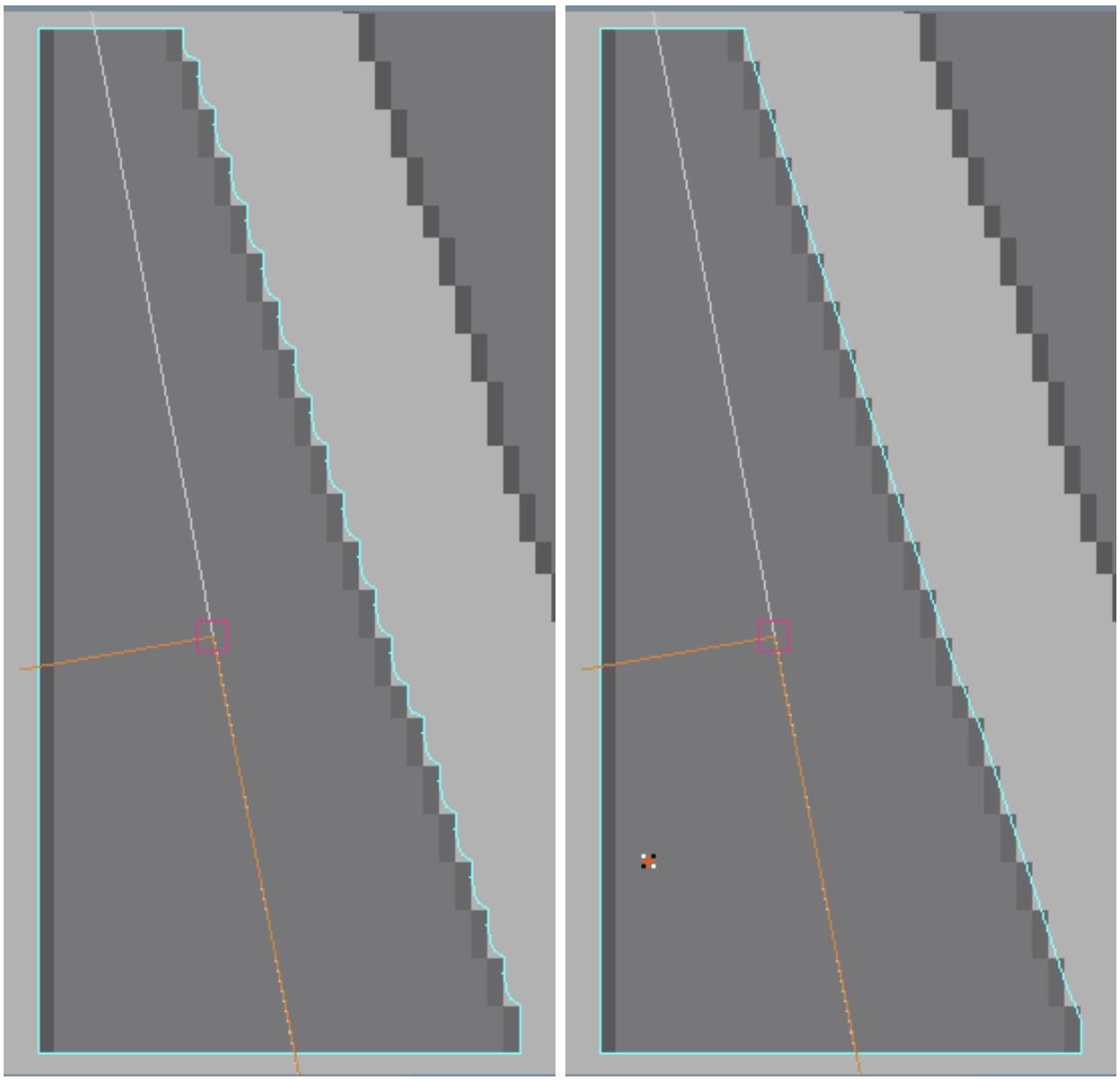
В то же время, для сильно зашумлённных контуров (рис.7) алгоритм фиксирует точки, которые желательно было бы сгладить. Таким образом, fix не стоит использовать для устранения шума.



(a) smooth 7 (b) fix, smooth 7

Рис. 7: smooth 7+ лучше справляется с шумами, чем последовательное применение fix и smooth 7.

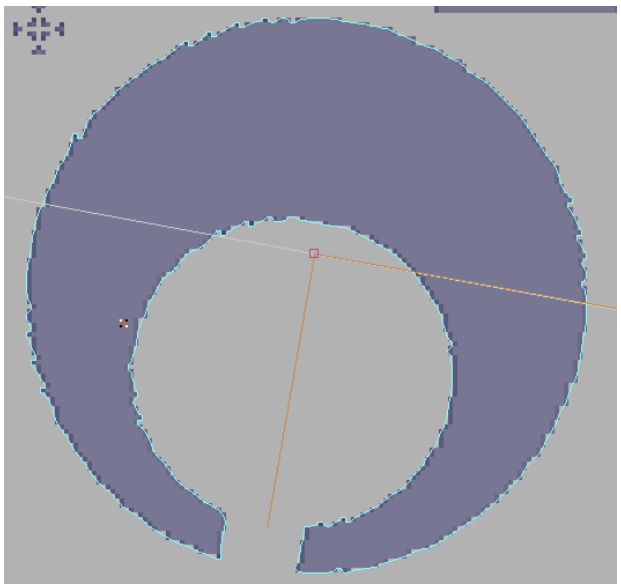
Стоит отметить, что fix7+ является алгоритмом сглаживания, позволяющим достаточно хорошо сохранять углы, поэтому логично сравнить результаты применения smooth7+ с результатами последовательного применения fix и smooth7. Так, из рис. 8 видно, что fix и smooth7 отлично справились с эффектом ступенчатости, чего, конечно, нельзя сказать о результатах работы smooth7+.



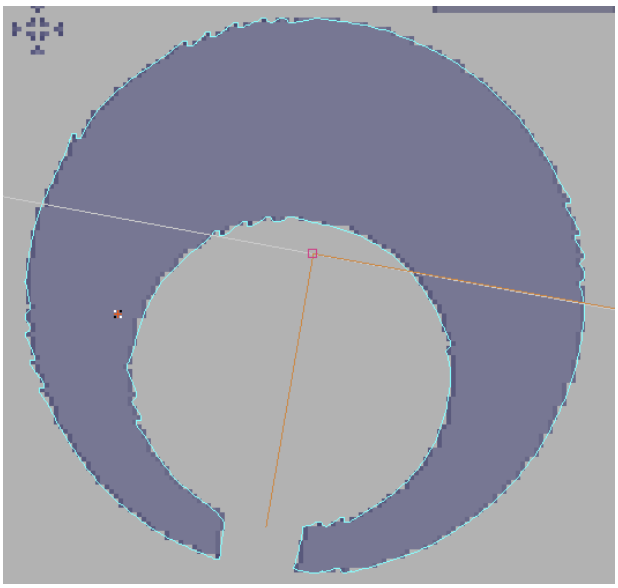
(a) smooth 7+ (b) fix, smooth 7

Рис. 8: smooth 7+ хуже устраняет эффект ступенчатости по сравнению с fix, smooth 7

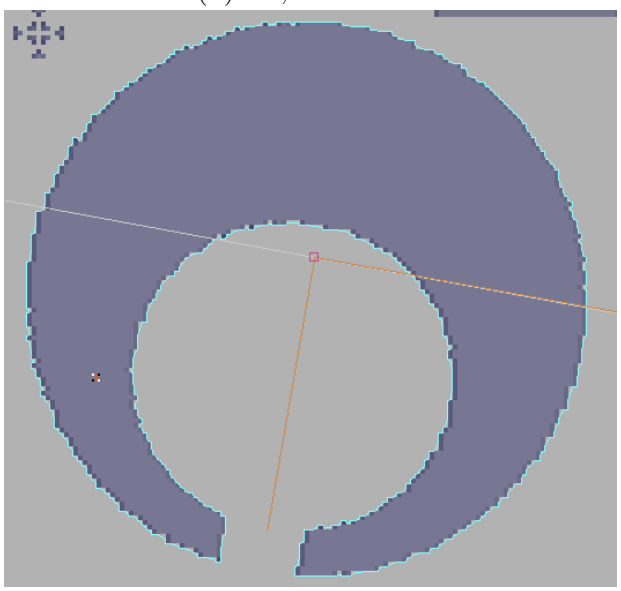
На рис. 9 приведён пример контура, для которого необходимо устранить эффект "ступенчатости а также уменьшить шум, сохранив при этом форму углов. Видно, что после однократного применения лучший результат показывает подход fix, smooth7, но после нескольких применений fix, smooth7 можно заметить, что неоторые шумы остались несглаженными, в то время как, после нескольких применений smooth7+ все заметные шумы и "ступеньки"были сглажены.



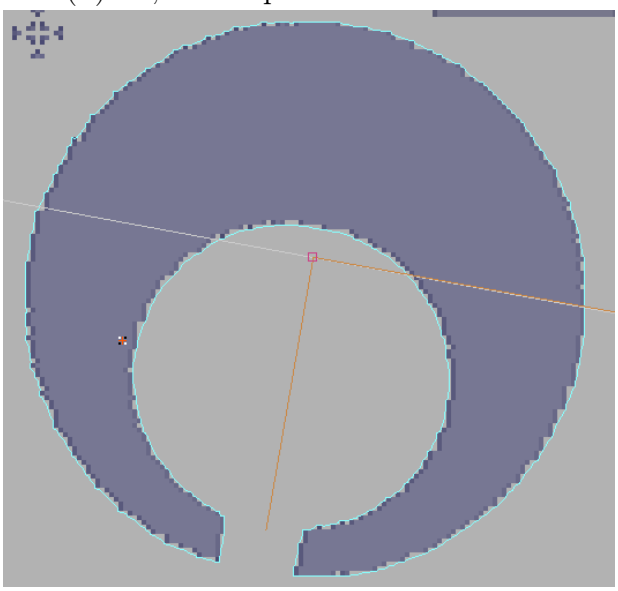
(a) fix, smooth 7



(b) fix, пять применений smooth 7



(c) smooth 7+



(d) пять применений smooth 7+

Рис. 9: Для округлых контуров с углами для устранения шумов лучше использовать smooth 7+. Однако, может понадобиться несколько раз запустить алгоритм.

Выводы

Алгоритм сглаживания по двум точкам обладет низкой сглаживающей способностью, однако хорошо справлятеся с эффектом “ступенчатости”, хотя для получения желаемого результата может потребоваться несколько раз применить данный алгоритм. Большей сглаживающей способностью, по сравнению с предыдущим алгоритмом, обладает алгоритм сглаживания по трём точкам с коэффициентом. В связи с этим, данный алгоритм достаточно хорошо справляется не только с эффектом “ступенчатости”, но и со слабыми возмущениями в контуре. Алгоритм сглаживания по семи точкам с коэффициентом (при $b = 1$) обладает ещё большей сглаживающей способностью из-за большего количества точек, используемых в вычислении. При использовании предыдущих трёх алгоритмов неминуемо сглаживается и форма углов, причём чем большей сглаживающей способностью обладает алгоритм, тем сильнее нежелательные изменения.

Сглаживание по семи точкам с учётом углов ($b = \pm 2$) уменьшает зашумлённость контура, при этом не изменяя форму углов. Однако, для устранения “ступенчатости” контура необходимо несколько раз запустить этот алгоритм. Фиксирование точек с применением любого из отмеченных в предыдущем абзаце алгоритмов сглаживания подходит для тех случаев, когда необходимо устранить эффект “ступенчатости” и сохранить форму углов. Стоит отметить, что не стоит применять такую комбинацию на контурах с сильными шумами, так как в этом случае алгоритм определения несглаживаемых точек может зафиксировать те точки, которые как раз и нуждаются в сглаживании. Усреднение фиксированных точек оказывается полезным в случае с острыми углами.

Таким образом, описанные алгоритмы обладают разной сглаживающей способностью, поэтому должен осуществляться выбор подходящего алгоритма в каждом конкретном случае.

Заключение

В ходе работы были рассмотрены основные подходы к сглаживанию, приведены условия для оптимального применения тех или иных алгоритмов сглаживания. Были усовершенствованы навыки программирования на языке Java и работы с IntelliJ IDEA. Также был получен опыт построения графических интерфейсов.

Результатом работы является программное обеспечение с графическим интерфейсом, предназначенное для применения рассмотренных алгоритмов сглаживания.

Список литературы

[1] Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005. 1072 с.

[2] Фурман Я.А., Кревецкий А. В., Передреев А. К., Роженцев А. А., Хафизов Р. Г., Егошина И. Л., Леухин А. Н. Введение в контурный анализ и его приложения к обработке изображений и сигналов. Изд. 2-е, испр. М.: Физматлит, 2003. 592 с.

[3] Feudjio C. K., Tiedeu A., Noubeg M.-L., Gordan M., Vlaicu A., Domngang S. Extracting and smoothing contours in mammograms using Fourier descriptors // Biomedical Science and Engineering, 2014. No. 7. P. 119-129.

[4] Hobby J. D. Smoothing Digitized Contours // Theoretical Foundations of Computer Graphics and CAD, Springer Berlin Heidelberg, 1988. P. 777–793.

[5] Hu J., Yu D., Yan H. Structural Boundary Feature Extraction for Printed Character Recognition // Advances in Pattern Recognition: Joint IAPR International Workshops, SSPR'98 and SPR'98, Sydney, Australia, August 11-13, 1998, Proceedings, P. 500-507

[6] IntelliJ IDEA 2016.1 Help
https://www.jetbrains.com/help/idea/2016.1/meet-intellij-idea.html

[7] Java™ Platform, Standard Edition 8 API Specification
https://docs.oracle.com/javase/8/docs/api/index.html

[8] Mansouryar M., Hedayati A. Smoothing Via Iterative Averaging (SIA) A Basic Technique for Line Smoothing // International Journal of Computer and Electrical Engineering, 2012. Vol. 4, No. 3. P. 307-311.

[9] Shea K. S., McMaster R. B. Cartographic Generalization in a Digital Environment: When and How to Generalize // Proceedings of the International Symposium on Computer-Assisted Cartography, 1989. P. 56 – 67.

[10] The Java® Language Specification.
https://docs.oracle.com/javase/specs/jls/se8/html/index.html

[11] The Java™ Tutorials
https://docs.oracle.com/javase/tutorial/tutorialLearningPaths.html

Приложение

Пример реализации алгоритма сглаживания

Ниже приведена реализация алгоритма сглаживания, описанного в . Остальные алгоритмы сглаживания были реализованы аналогичным образом.

```
private Point2D.Float[] smoothContour3(Point2D.Float[] points,
        int mixCoeff) {
    Point2D.Float[] newPoints = new Point2D.Float[points.length];
    Point2D.Float pl = points[points.length - 2];
    Point2D.Float pt = points[points.length - 1];
    int it = points.length - 1;
    Point2D.Float pn;
    for (int i = 0; i < newPoints.length; i++) {
        pn = points[i];
        if (smoothablePoints == null || smoothablePoints.length == 0
            || smoothablePoints[it]) {
            newPoints[it] = new Point2D.Float(
                    (pl.x + pn.x + mixCoeff * pt.x) / (mixCoeff + 2),
                    (pl.y + pn.y + mixCoeff * pt.y) / (mixCoeff + 2));
        } else {
            newPoints[it] = pt;
        }
        it = i;
        pl = pt;
        pt = pn;
    }
    smoothedPoints = newPoints;
    return newPoints;
}
```