МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Кафедра: Информатики и автоматизации научных исследований

Направление подготовки: 09.04.03 «Прикладная информатика»
Магистерская программа: «Прикладная информатика в области принятия решений»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема:

«Автоматизированный комплекс формирования 3d-модели машиностроительных чертежей. Подсистема ввода векторных данных»

Выполнил:
студент группы 381507м Дёгтев А.С.

_____
подпись

Научный руководитель:
к.т.н., доц. каф. ИАНИ Васин Д.Ю.

_____
подпись

Нижний Новгород
2017

Аннотация

Содержание

1. Теоретическая часть 7
1.1. Существующие системы векторизации растровых изображений 7
1.1.1. RasterDesk 7
1.1.2. Spotlight 8
2. Постановка задачи 9
3. Алгоритмическое обеспечение 10
3.1. Преобразование изображения в оттенки серого 10
3.2. Инвертация изображения 11
3.3. Бинаризация изображения 11
3.3.1. Бинаризация по пороговому значению 12
3.3.2. Бинаризация с применением метода Оцу 12
3.4. Удаление «шумов» на изображении 13
3.5. Скелетизация 13
3.5.1. Алгоритм Зонга-Суня (Zhang-Suen) 14
3.5.2. Алгоритм Гуо-Хелла (Guo-Hall) 15
3.6. Векторизация 16
3.6.1. Алгоритм векторизации 16
3.6.2. Описание алгоритма 17
4. Информационное обеспечение 20
5. Разработанное программное обеспечение 21
5.1. Условия функционирования 21
5.2. Диаграмма связей разработанных классов 21
6. Руководство пользователя к разработанному программному продукту 22
7. Заключение 23
8. Литература 24

Глоссарий

Бинарное изображение – изображение, состоящее из пикселей двух типов: фоновых точек и активных точек.

Введение

Различные возможности и границы применения вычислительной техники для автоматизации проектирования определяются уровнем формализации научно-технических знаний в конкретной отрасли. Чем глубже разработана теория того или иного класса технических систем, тем большие возможности объективно существуют для автоматизации процесса их проектирования.

Применение ЭВМ при проектно-конструкторских работах в своем развитии прошло несколько стадий и претерпело значительные изменения. С появлением вычислительной техники был сделан акцент на автоматизацию проектных задач, имеющих четко выраженный расчетный характер, когда реализовывались методики, ориентированные на ручное проектирование. Затем, по мере накопления опыта, стали создавать программы автоматизированных расчетов на основе методов вычислительной математики (параметрическая оптимизация, метод конечных элементов и т. п.). С внедрением специализированных терминальных устройств появляются универсальные программы для ЭВМ для решения как расчетных, так и некоторых рутинных проектных задач (изготовление чертежей, спецификаций, текстовых документов и т. п.).

Решение проблем автоматизации проектирования с помощью ЭВМ основывается на системном подходе, т. е. на создании и внедрении САПР – систем автоматизированного проектирования технических объектов, которые решают весь комплекс задач от анализа задания до разработки полного объема конструкторской и технологической документации. Это достигается за счет объединения современных технических средств и математического обеспечения, параметры и характеристики которых выбираются с максимальным учетом особенностей задач проектно-конструкторского процесса. Примерами таких систем могут служить AutoCAD, КОМПАС-3D, CorelCAD. Однако следует заметить, что основной целью указанных САПР является создание конструкторских документов, но важной частью разработки является и анализ готовых машиностроительных чертежей, в том числе и векторизация существующих чертежей в растровом формате. Указанные САПР не имеют возможности анализировать чертеж на предмет геометрических объектов, что является существенным недостатком этих систем, так как при отсутствии возможности автоматического обнаружения объектов приходится воссоздавать чертеж заново, что может привести к ошибкам и, следовательно, к неверно спроектированному объекту. Не стоит забывать о том, что многие конструкторские документы, в том числе и чертежи, создаются вручную с помощью кульмана. Проанализировать данный чертеж на предмет наличия

геометрических ошибок, не привнося искажений, достаточно проблематично. Важной областью разработки машиностроительной документации является хранение.

В последнее время наметилась тенденция на переход к электронным архивам хранения. Следовательно, проблема обнаружения геометрических объектов на растровых сканированных чертежах, так же, как и занесения в электронный архив, является актуальной.

Цель работы – разработка информационного, алгоритмического и программного обеспечения подсистемы формирование топологической векторной модели растрового 2D-изображениях машиностроительных чертежей.

1. Теоретическая часть

Существует два способа векторизации чертежей: помощью специальных программ и ручной.

При векторизации вручную растровый чертёж помещается на нижний слой, над ним создаётся ещё один слой. Далее выполняется сопоставление масштаба и обводка элементов чертежа.

Данный способ несовершенен по причине возможных ошибок, совершаемых человеком, и длительности ручной векторизации.

При векторизации при помощи специальных программ процесс может быть автоматическим или полуавтоматическим, что существенно повышает скорость работы с чертежами. Однако, как и в предыдущем методе, возможно появление различных ошибок, что требует контроля результата человеком.

1.1. Существующие системы векторизации растровых изображений

1.1.1. RasterDesk

RasterDesk – графический редактор, дополняющий функции AutoCAD профессиональными инструментами для работы с растровыми изображениями.

RasterDesk имеет две модификации: базовую (RasterDesk) и профессиональную (RasterDesk Pro). Базовая версия программы позволяет повысить качество отсканированного изображения, редактировать растровую графику с использованием инструментов AutoCAD и векторизовать растровое изображение в полуавтоматическом режиме. Профессиональная версия программы дополнена инструментами автоматической векторизации, автоматической векторной коррекции и модулем распознавания текстов Abbyy FineReader 10.

Данная программа платная, но имеются демонстрационные версии и временные версии.

Данный аналог частично удовлетворяет поставленным требованиям (отсутствует сохранение топологической модели) и распространяется на платной основе.

1.1.2. Spotlight

Spotlight – профессиональный гибридный графический редактор, позволяющий осуществить полный комплекс работ с растровыми монохромными, полутоновыми и цветными изображениями: отсканированными чертежами, картами, схемами и другими графическими материалами.

Spotlight имеет две модификации: базовую (Spotlight) и профессиональную (Spotlight Pro). Базовая версия программы позволяет повысить качество отсканированного изображения, редактировать растровую и векторную графику, векторизовать растровое изображение в полуавтоматическом режиме. Профессиональная версия программы дополнена инструментами автоматической векторизации, автоматической векторной коррекции, модулем распознавания текстов Abbyy FineReader 10 и встроенным редактором кода, поддерживающим JavaScript или VB Script.

Данная программа платная, но имеются демонстрационные версии и временные версии.

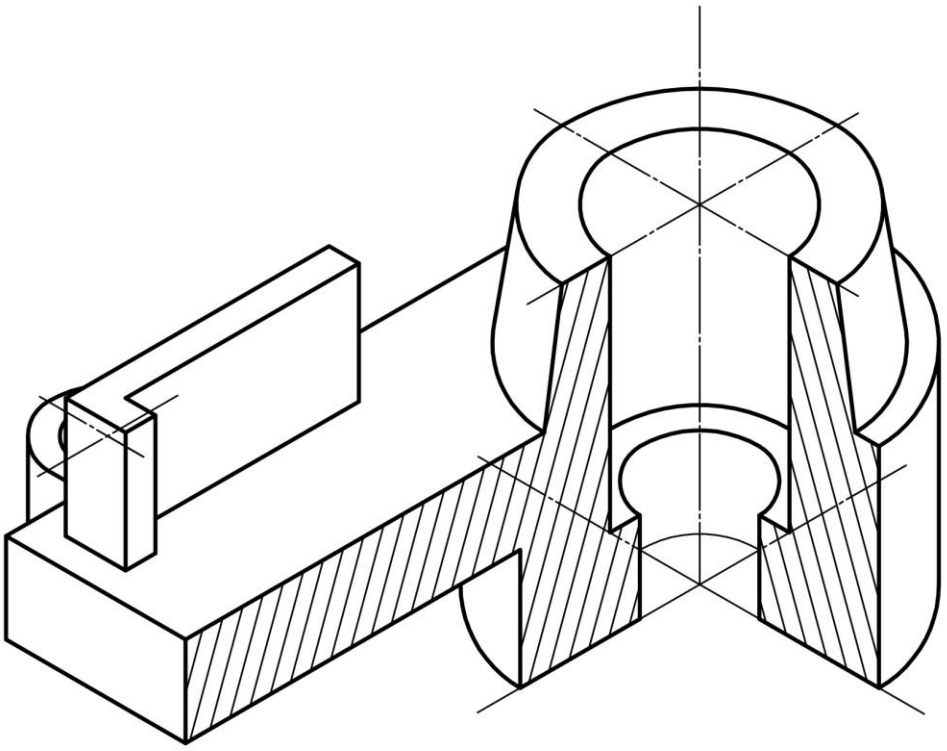
Данный аналог частично удовлетворяет поставленным требованиям (отсутствует сохранение топологической модели) и распространяется на платной основе.

Из всех рассмотренных программных продуктов, наиболее удовлетворяющим заданным требованиям является Spotlight. Однако отсутствует сохранение топологической модели и данный продукт распространяется на платной основе. Таким образом, рассматриваемая проблема распознавания объектов на растровых изображениях на данный момент и в будущем будет актуальна в связи с всё большим распространением электронных средств проектирования и хранения чертежей.

2. Постановка задачи

Задано сканированное с листа формата А4 со значением DPI равным 300 и разрешением не более 5000x5000 пикселей растровое изображение в одном из форматов bmp, jpg, png или tiff.

Необходимо реализовать информационное, алгоритмическое и программное обеспечение автоматизированной подсистемы формирования топологической векторной модели 2D-чертежей на растровом изображении в формате форматов bmp, jpg, png или tiff.



3. Алгоритмическое обеспечение

Для того, чтобы получить векторное представление растрового изображения необходимо выполнить два этапа.

Первый – предобработка растрового изображения. На этом этапе изменяется глубина цвета, удаляются «шумы» и наименее значимые элементы на изображении. Результатом этого этапа является изображение упрощающее последующие действия с изображением.

Второй этап – векторизация изображения, т.е. преобразование растровой матрицы в набор векторов, что существенно расширяет область применения имеющихся данных.

Выполняемые в работе преобразования растровых данных в векторные состоят из следующих шагов:

1. Преобразование изображения в оттенки серого;
2. Инвертация изображения;
3. Бинаризация изображения;
4. Удаление «шумов» на изображении;
5. Скелетизация изображения;
6. Векторизация.

Далее приведённые выше шаги будут описаны.

3.1. Преобразование изображения в оттенки серого

Учитывая, что источником данных для векторизации являются машиностроительные чертежи, представленные в растровом формате (цветовая модель RGB), можно сделать вывод, что основными цветами будут являться белый (цвет фона) и чёрный (цвет объектов). Значит остальными цветами можно пренебречь.

Однако простой отбор белого и чёрного цветов и игнорирование всех остальных может привести к существенной потере данных, т.к. довольно малое количество информации представлено чистыми чёрным и белым цветами.

Чтобы этого избежать требуется постепенное уменьшение глубины цвета, путём преобразования цвета всех пикселей изображения к требуемому диапазону.

Изначально каждый пиксель изображения представлен 24-битным TrueColor-цветом, в котором на каждый цветовой канал (красный, зелёный, синий) отводится 8 бит, т.е. 256 оттенков. Всего получаем 16777216 оттенков. Это число можно уменьшить, оставив вместо трёх цветовых каналов один 8-битный, используемый для отображения яркости пикселя.

Для этого необходимо воспользоваться цветовой моделью YIQ, которая представлена тремя компонентами Y – яркостная составляющая, I – синфазный сигнал, Q – квадратурный сигнал.

Для перевода пространства RGB в YIQ используется следующая формула:

$$\begin{cases} Y = 0.299\,R + 0.587\,G + 0.144\,B \\ I = 0.596\,R - 0.274\,G - 0.321\,B \\ Q = 0.211\,R - 0.526\,G + 0.311\,B \end{cases}. (1)$$

Из формулы (1) необходимо использовать уравнение

$$Y = 0.299\,R + 0.587\,G + 0.144\,B, (2)$$

в котором R, G, B – красный, зелёный и синий цвета соответственно, а Y – полученная яркость.

Таким образом применяя формулу (2) к каждому пикселю изображения получим яркостную матрицу или изображение в оттенках серого.

3.2. Инвертация изображения

На исходном изображении фоновые цвета отражены оттенками, близкими к белым (значение яркости близко к 255), а объекты чертежа оттенками, близкими к чёрным (значение яркости близко к 0), что не так удобно с точки зрения обработки чертежа. Поэтому над изображением выполняется операция инвертирования путём изменения яркости каждого пикселя по формуле:

$$brightness = 255 - brightness,$$

где brightness – яркостная составляющая пикселя, 255 – максимальное значение яркости.

3.3. Бинаризация изображения

После выполненных ранее преобразований изображение имеет большой цветовой диапазон (256 оттенков), что для векторизации является избыточным, а также учитывая, что для векторизации достаточно глубины цвета, равной одному биту, следует, что можно выполнить бинаризацию изображения.

Бинаризация представляет собой сведение всего цветового пространства изображения к двум цветам: цвету, обозначающему фоновые пиксели, и цвету, обозначающему пиксели интереса (изображённых объектов).

Были рассмотрены два алгоритма бинаризации:

- Бинаризация по пороговому значению;

– Бинаризация методом Отсу.

3.3.1. Бинаризация по пороговому значению

Данный алгоритм основан на том, что есть заранее заданное пороговое значение и всем пикселям изображения присваивается яркость в соответствии с формулой:

$$brightness = \begin{cases} 0, brightness < k \\ 255, brightness \geq k \end{cases},$$

где k является пороговым значением.

Недостатком данного алгоритма является потребность подбора параметра k человеком вручную, что может быть затратным по времени при подборе наиболее оптимального параметра.

3.3.2. Бинаризация с применением метода Оцу

Поскольку бинаризация по пороговому значению не предоставляет возможности автоматической обработки изображений, была рассмотрена бинаризация с применением метода Оцу для нахождения порогового значения.

Общий алгоритм метода Оцу состоит из следующих этапов:

1. Вычисление гистограммы изображения;
2. Для каждого из значений гистограммы:

   2.1. Вычисление $\sigma_b^2(t)$;

   2.2. Если $\sigma_b^2(t)$ больше имеющегося, то сохранение $\sigma_b^2(t)$ и порога t.

$\sigma_b^2(t)$ вычисляется по формуле:

$$\sigma_b^2(t) = \sigma^2(t) - \sigma_w^2(t) = P_1(t)P_2(t)[\mu_1(t) - \mu_2(t)]^2,$$

где

$$\sigma_w^2(t) = P_1(t)\sigma_1^2(t) + P_2(t)\sigma_2^2(t),$$

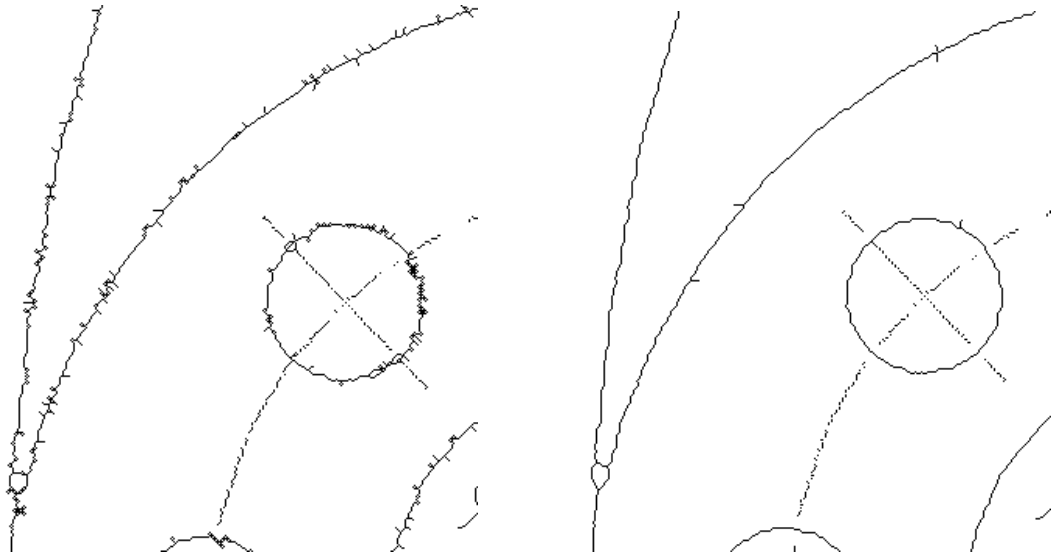
что представляет собой сумму различий двух классов. Классы являются множествами пикселей изображения, разделённых относительно порогового значения t.

Веса P_i – вероятности классов, σ_i^2 – дисперсии классов.

3.4. Удаление «шумов» на изображении

Полученное изображение может содержать пиксели, которые являются «шумом» - одиночными пикселями, соседние пиксели которых имеют цвет противоположный центральному.

При последующей векторизации такие пиксели могут вносить существенные искажения.



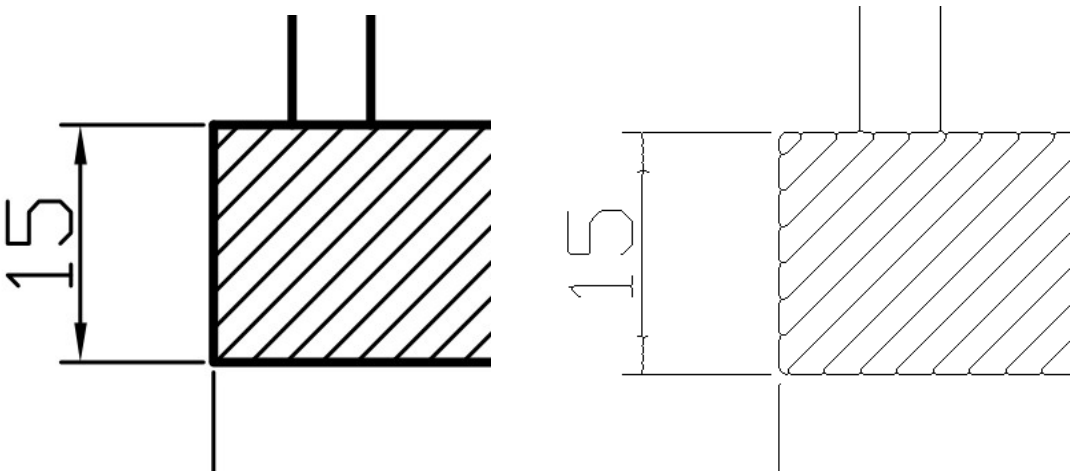
Для предотвращения появления таких эффектов производится удаление пикселей, являющихся «шумом», следующим образом:

1. Для каждого пикселя:
   1.1. Получение цветов соседних пикселей;
   1.2. Если все соседние пиксели одного цвета и этот цвет противоположен цвету центрального пикселя
      1.2.1. Центральному пикселю присваивается цвет соседних пикселей;
   1.3. В противном случае
      1.3.1. Оставить старый цвет центрального пикселя.

3.5. Скелетизация

Алгоритмы скелетизации предназначены для максимального утоньшения линий на бинарном изображении, что приводит к получению скелета изображённых объектов, в котором все линии имеют толщину не более одного пикселя.

Так же, далее учитывается, что фоновые пиксели имеют значение яркости 0, а пиксели интереса (изображённых объектов) – 1.



3.5.1. Алгоритм Зонга-Суня (Zhang-Suen)

Алгоритм Зонга-Суня является итеративным. На каждой итерации удаляются граничные пиксели. Итерации повторяются пока хотя бы один пиксель был удалён.

Вводится матрица, вида:

P8	P2	P3
P8	P1	P4
P7	P6	P5

Матрица 1

На каждой итерации выполняется наложение матрицы 1 на изображение таким образом, чтобы центральный элемент (P1) был наложен на все пиксели изображения поочерёдно.

Итерация разделяется на две подитерации. На первой подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия:

$$\begin{cases} 2 \le \sum_{i=2}^{9} P_i \le 6 \\ S(P_1) = 1 \\ P_2 P_4 P_6 = 0 \\ P_4 P_6 P_8 = 0 \end{cases},$$

где $S(P_1)$ – количество найденных переходов от 0 к 1 в последовательности P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2.

На второй подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия:

$$\begin{cases} 2 \le \sum_{i=2}^{9} P_i \le 6 \\ S(P_1) = 1 \\ P_2 P_4 P_8 = 0 \\ P_2 P_6 P_8 = 0 \end{cases}.$$

Информация об удалённых пикселях сохраняется в отдельный массив, который, после выполнения обеих подитераций, накладывается на изображение.

Если после выполнения итерации были удалены какие-либо пиксели, то итерация повторяется. Выполнение алгоритма останавливается, когда не был удалён ни один пиксель.

Стоит заметить, что рассмотренный алгоритм относится к классу параллельных алгоритмов, за счёт сохранения информации об удаляемых пикселях в отдельном массиве.

3.5.2. Алгоритм Гуо-Хелла (Guo-Hall)

Данный алгоритм подобен алгоритму Зонга-Суня, однако имеются отличия в условиях удаления пикселей в подитерациях.

Пусть заданы следующие уравнения:

$$C(P_1) = !P_2\&(P_3|P_4) + !P_4\&(P_5|P_6) + !P_6\&(P_7|P_8) + !P_8\&(P_1|P_2)$$

$$N_1(P_1) = (P_9|P_2) + (P_3|P_4) + (P_5|P_6) + (P_7|P_8)$$

$$N_2(P_1) = (P_2|P_3) + (P_4|P_5) + (P_6|P_7) + (P_8|P_9)$$

$$N(P_1) = \min[N_1(P_1), N_2(P_1)]$$

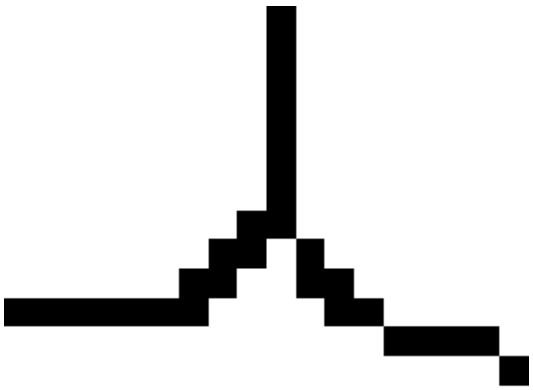
На первой подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия:

$$\begin{cases} C(P_1) = 1 \\ 2 \le N(P_1) \le 3 \\ (P_2|P_3|!P_5)\ \&\ P_4 = 0 \end{cases}$$

На второй подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия:

$$\begin{cases} C(P_1) = 1 \\ 2 \le N(P_1) \le 3 \\ (P_6|P_7|!P_9)\ \&\ P_8 = 0 \end{cases}$$

Алгоритм Гуо-Хелла является более предпочтительным для векторизации, так как алгоритм Зонга-Суня имеет недостаток в виде ступенчатости скелетизируемых объектов, что является излишним и, как следствие, усложняет векторизацию.



Пример выполнения алгоритма Зонга-Суня.

Пример выполнения алгоритма Гуо-Хелла.

3.6. Векторизация

Были рассмотрены несколько алгоритмов векторизации. Однако их применение не было возможно по нескольким причинам:

- Алгоритмы не предполагают обнаружения и сохранения топологии изображённых объектов;
- Некоторые алгоритмы предлагают сохранение обнаруженных объектов в виде полигональных объектов, что мало применимо к машиностроительным чертежам.

По причине отсутствия подходящего алгоритма векторизации было решено разработать новый алгоритм, который удовлетворял бы следующим требованиям:

- Извлечение объектов изображения в виде линейных объектов;
- Получение топологии изображённых объектов.

3.6.1. Алгоритм векторизации

В результате был разработан итеративный алгоритм, удовлетворяющий приведённым выше требованиям.

Суть алгоритма сводится к перебору всех пикселей изображения, изучения соседних пикселей (лежащих в матрице размерностью 3x3, где центральная ячейка – текущий пиксель) и принятия решения о дальнейших действиях: начале извлечения нового объекта, сохранения извлечённого объекта, переходов в следующие точки, а также сохранении топологических точек объектов (точек начала, окончания и пересечения объектов).

3.6.2. Описание алгоритма

Пусть имеются следующие объекты:

- Изображение, представляющее собой пиксельную матрицу;
- Множество для хранения особых точек;
- Состояние векторизации – объект хранящий стартовую, предыдущую и текущую точки векторизации.

Процедура векторизации:

1. Пока на изображении есть непосещённые точки:
    a. Выполнить процедуру итерации.

Где процедура итерации имеет следующий вид:

1. Если состояние векторизации не инициализировано:
    a. Очистить множество особых точек;
    b. Преобразовать все исключённые точки изображения в непосещённые;
    c. Если на изображении есть непосещённые точки:
        i. Сохранить непосещённую точку в состояние векторизации, как текущую;
    d. Иначе:
        i. Завершить алгоритм;
2. Если у текущей точки состояния векторизации отсутствуют соседние точки:
    a. Пометить текущую точку векторизации, как посещённую;
    b. Если предыдущая точка состояния векторизации инициализирована:
        i. Добавить текущую точку состояния векторизации во множество особых точек;
        ii. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
        iii. Очистить множество особых точек;
        iv. Присвоить значение текущей точки стартовой точке состояния векторизации;
        v. Завершить итерацию;
3. Если предыдущая точка состояния векторизации не инициализирована:
    a. Добавить текущую точку состояния векторизации во множество особых точек;
    b. Выбрать точки для продолжения векторизации;

c. Если среди выбранных точек для продолжения векторизации имеется горизонтальная или вертикальная соседняя точка:
    i. Запомнить её, как следующую точку;
d. Иначе:
    i. Запомнить первую точку из точек для продолжения векторизации, как следующую точку;
e. Пометить остальные точки для продолжения векторизации, как исключённые из векторизации;
f. Если точки были помечены, как исключённые:
    i. Пометить текущую точку векторизации, как исключённую;
g. Иначе:
    i. Пометить текущую точку векторизации, как посещённую;
h. Присвоить значение текущей точки стартовой точке состояния векторизации;

4. Иначе:
    a. Выбрать точки для продолжения векторизации;
    b. Если таковых не имеется:
        i. Добавить текущую точку состояния векторизации во множество особых точек;
        ii. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
        iii. Завершить итерацию;
    c. Если среди точек для продолжения векторизации найдена точка, которая лежит на одной прямой с предыдущей и текущей точками состояния векторизации:
        i. Запомнить её, как следующую точку;
    d. Иначе:
        i. Запомнить первую точку из точек для продолжения векторизации, как следующую точку;
    e. Пометить остальные точки для продолжения векторизации, как исключённые из векторизации;
    f. Если точки были помечены, как исключённые:
        i. Пометить текущую точку векторизации, как исключённую;
    g. Иначе:

i. Пометить текущую точку векторизации, как посещённую;

h. Если текущая точка состояния векторизации была исключена из векторизации или у текущей точки состояния векторизации имеются соседние точки в горизонтальных направлениях:
    i. Добавить текущую точку состояния векторизации во множество особых точек;

i. Если следующая точка состояния векторизации не находится на одной прямой с предыдущей и текущей точками состояния векторизации:
    i. Добавить текущую точку состояния векторизации во множество особых точек;
    ii. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
    iii. Очистить множество особых точек;
    iv. Присвоить значение текущей точки стартовой точке состояния векторизации;

5. Присвоить значение текущей точки предыдущей точке состояния векторизации;
6. Присвоить значение следующей точки текущей точке состояния векторизации.

4. Информационное обеспечение

5. Разработанное программное обеспечение

5.1. Условия функционирования

5.2. Диаграмма связей разработанных классов

6. Руководство пользователя к разработанному программному продукту

7. Заключение

8. Литература

1. Гудков, В.Ю. Скелетизация бинарных изображений и выделение особых точек для распознавания отпечатков пальцев / В.Ю. Гудков, Д.А. Клюев // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2015. – Т. 15, № 3. – С. 11 –17. DOI: 10.14529/ctcr150302
2. Сай, И.С. Эффективность алгоритмов поиска оттиска печати в изображении документа / И.С. Сай // Вестник ТОГУ «Электроника и информационно-измерительные приборы» - 2009 - №4
3. Москаленко, С.В. Разработка методов и алгоритмов векторизациирастровых изображений в САПР / С.В. Москаленко / 2011 – 18с.
4. Стержанов, М. Алгоритм векторизации штриховых изображений отрезками прямых / М. Стержанов / БГУИР – 4с.
5. Racter Arts – Режим доступа: http://rasterarts.ru/ – Загл. с экрана.

https://habrahabr.ru/post/181580/

https://ru.wikipedia.org/wiki/YIQ

http://opencv-code.com/quick-tips/implementation-of-thinning-algorithm-in-opencv/

http://opencv-code.com/quick-tips/implementation-of-guo-hall-thinning-algorithm/