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:
«Разработка системы векторизации растровых 2D-чертежей»

Выполнил:
студент группы 381507м Дёгтев А.С.
_____
подпись

Научный руководитель:
к.т.н., в.н.с. ИИТММ ННГУ Васин Д.Ю.
_____
подпись

Нижний Новгород
2017

Аннотация

В работе содержится описание информационного, алгоритмического и программного обеспечения системы формирования векторной модели с узловыми точками растрового 2D-изображения машиностроительных чертежей. Описываются алгоритмы подготовки изображения к векторизации и самой векторизации, программная реализация, условия функционирования комплекса, а также результаты, полученные с её помощью.

Содержание

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[7].

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.)[7].

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[7].

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 системы формирования векторной модели с узловыми точками растрового 2D-изображения 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[8, 13] – 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 узловых точек) и распространяется на платной основе.

1.1.2. Spotlight

Spotlight[8, 14] – 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 узловых точек) и распространяется на платной основе.

Из всех рассмотренных программных продуктов, наиболее удовлетворяющим заданным требованиям является Spotlight. Однако отсутствует сохранение узловых точек 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

Задан растровый машиностроительный чертёж, сканированный с бумажной копии формата А4 со значением DPI равным 300, представляющий собой 2D проекции 3D модели машиностроительной детали, геометрическими размерами не более 5000x5000 пикселей, глубиной цвета 8, 16 или 32 бита, представленный одном из стандартных распространённых форматов (BMP, JPG, PNG или TIFF).

Необходимо разработать информационное, алгоритмическое и программное обеспечение автоматической системы формирования векторной моделей с сохранением узловых точек 2D-чертежей на растровом изображении в формате BMP, JPG, PNG или TIFF.

На рис. 1 приведён пример растрового 2D-чертежа.

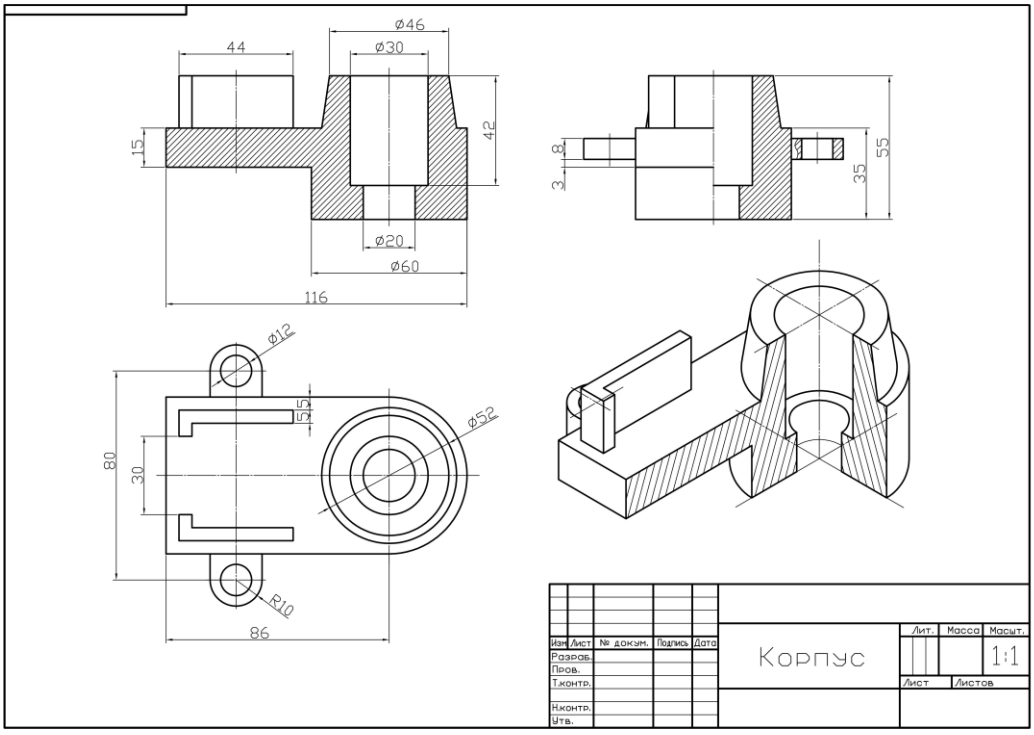

Рисунок 1. Пример чертежа

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.

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[12], которая представлена тремя компонентами Y – яркостная составляющая, I – синфазный сигнал, Q – квадратурный сигнал.

Для перевода пространства RGB в YIQ используется следующая формула:

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

Из формулы (1) необходимо использовать уравнение (2):

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

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 (3):

$$brightness = 255 - brightness, \quad (3)$$

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 Оцу.

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 (4):

$$brightness = \begin{cases} 0, brightness < k \\ 255, brightness \geq k \end{cases}, \quad (4)$$

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[5, 6] для нахождения порогового значения.

Общий алгоритм метода Оцу состоит из следующих этапов:

1. Вычисление нормализованной гистограммы изображения (5);

$$p_i = n_i/N, \quad (5)$$

где N – общее число пикселей изображения, n_i – число пикселей с уровнем яркости i, i = 0..L.

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:

$$\sigma_b^2(t) = \sigma^2(t) - \sigma_w^2(t) = P_1(t)P_2(t)[\mu_1(t) - \mu_2(t)]^2, \quad (6)$$

где

$$\sigma_w^2(t) = P_1(t)\sigma_1^2(t) + P_2(t)\sigma_2^2(t), \quad (7)$$

$$P_1(t) = \sum_{i=0}^{t} p(i), \quad (8)$$

$$P_2(t) = \sum_{i=t+1}^{L-1} p(i) = 1 - P_1, \quad (9)$$

$$\mu_2(t) = (\sum_{i=0}^{t} ip(i))/P_1, \quad (10)$$

$$\mu_2(t) = (\sum_{i=t+1}^{L-1} ip(i))/P_2, \quad (11)$$

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

Формулы представляют собой вероятности классов.

А формулы служат для вычисления среднего значения класса.

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 (рис. 2, рис. 3).

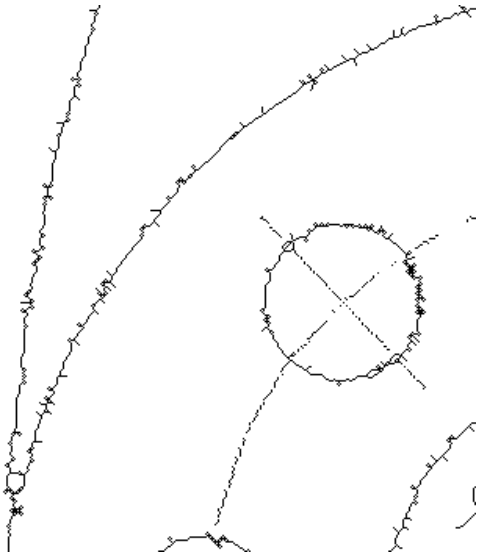

Рисунок 2. Результат векторизации без предварительного удаления «шума»

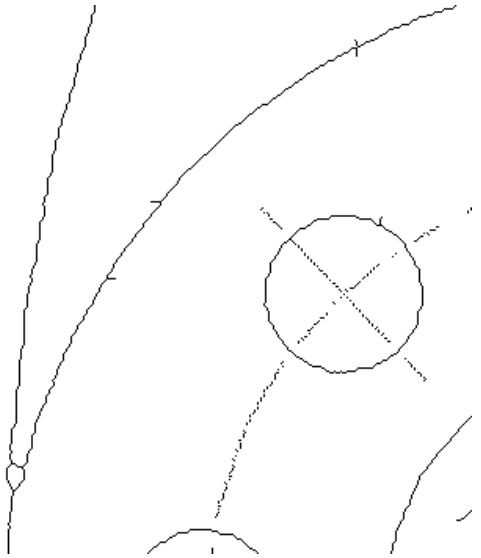

Рисунок 3. Результат векторизации с предварительным удалением «шума»

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 (рис. 4, рис. 5).

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

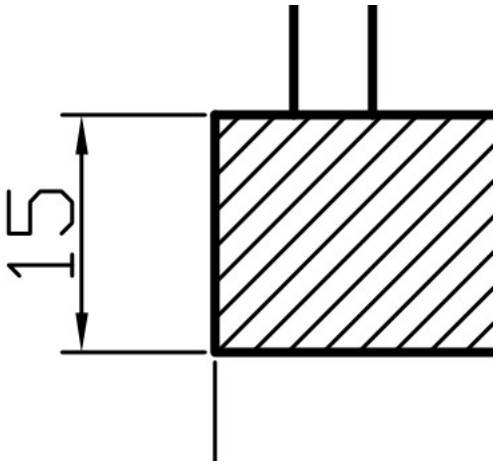

Рисунок 4. Изображение до применения скелетизации

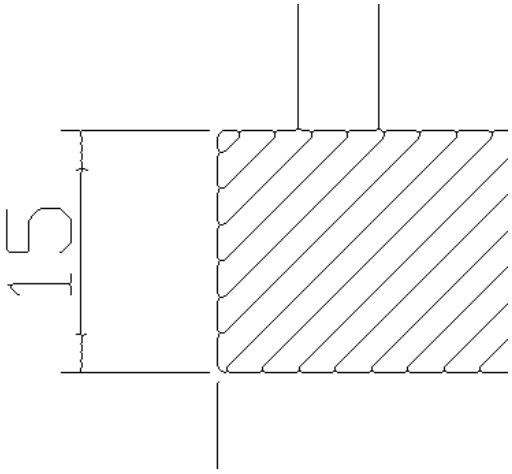

Рисунок 5. Изображение после применения скелетизации

3.5.1. Алгоритм Зонга-Суня (Zhang-Suen)

Алгоритм Зонга-Суня[10] является итеративным. На каждой итерации удаляются граничные пиксели. Итерации повторяются пока хотя бы один пиксель был удалён.

Вводится матрица, вида (рис. 6):

P9	P2	P3
P8	P1	P4
P7	P6	P5

Рисунок 6. Матрица скелетизации

На каждой итерации выполняется наложение матрицы скелетизации (рис. 6) 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 (12):

$$\begin{cases} 2 \leq \sum_{i=2}^{9} P_i \leq 6 \\ S(P_1) = 1 \\ P_2 P_4 P_6 = 0 \\ P_4 P_6 P_8 = 0 \end{cases}, \quad (12)$$

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 (13):

$$\begin{cases} 2 \leq \sum_{i=2}^{9} P_i \leq 6 \\ S(P_1) = 1 \\ P_2 P_4 P_8 = 0 \\ P_2 P_6 P_8 = 0 \end{cases}. \quad (13)$$

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[11] подобен алгоритму Зонга-Суня, однако имеются отличия в условиях удаления пикселей в подитерациях.

Пусть заданы следующие уравнения (14), (15), (16), (17):

$$C(P_1) = !P_2 \& (P_3 | P_4) + !P_4 \& (P_5 | P_6) + !P_6 \& (P_7 | P_8) + !P_8 \& (P_1 | P_2) \quad (14)$$

$$N_1(P_1) = (P_9 | P_2) + (P_3 | P_4) + (P_5 | P_6) + (P_7 | P_8) \quad (15)$$

$$N_2(P_1) = (P_2 | P_3) + (P_4 | P_5) + (P_6 | P_7) + (P_8 | P_9) \quad (16)$$

$$N(P_1) = \min[N_1(P_1), N_2(P_1)] \quad (17)$$

На первой подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия (18):

$$\begin{cases} C(P_1) = 1 \\ 2 \le N(P_1) \le 3 \\ (P_2|P_3|!\,P_5)\ \&\ P_4 = 0 \end{cases} \quad (18)$$

На второй подитерации пиксель, являющийся элементом P1, удаляется, если выполняются условия (19):

$$\begin{cases} C(P_1) = 1 \\ 2 \le N(P_1) \le 3 \\ (P_6|P_7|!\,P_9)\ \&\ P_8 = 0 \end{cases} \quad (19)$$

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 (рис. 7, рис. 8).

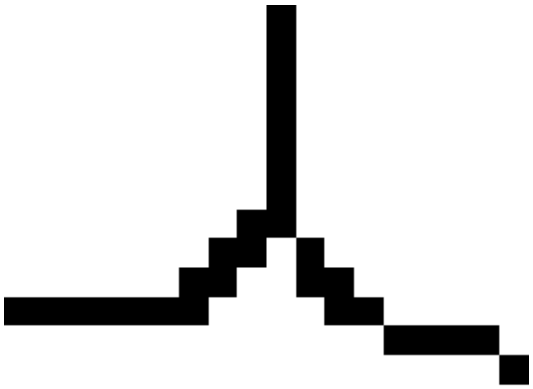

Рисунок 7. Пример выполнения алгоритма Зонга-Суня

Рисунок 8. Пример выполнения алгоритма Гуо-Хелла

3.6. Векторизация

Были рассмотрены несколько алгоритмов векторизации[1-4]. 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 узловых точек 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).

3.6.1.1. 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.

3.6.2. Штриховой алгоритм векторизации

Недостатком предыдущего метода векторизации является зависимость от результатов скелетизации, что может вносить искажения в результат векторизации.

Штриховой алгоритм векторизации лишён данного недостатка, так как на вход поступает бинаризованное изображение.

Данный алгоритм основан на проведении вертикальных штрихов вдоль всего изображения (рис. 9). При попадании штриха на линию объекта, точки переходов цветов с белого на чёрный и с чёрного на белый помечаются, как граничные точки линии.

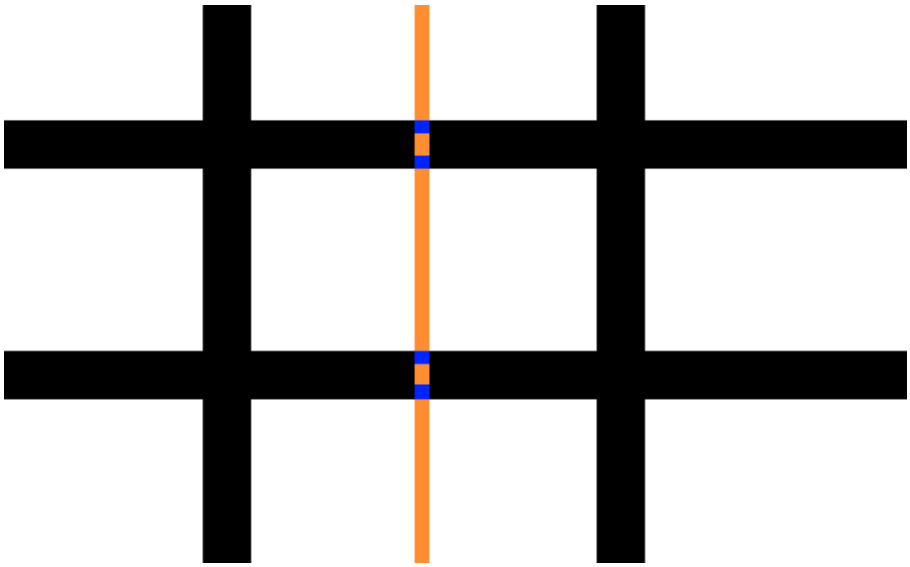

Рисунок 9. Проведение штриха по изображению

По их координатам вычисляется средняя точка, которая переносится на отдельный слой.

Далее выполняются те же действия для горизонтальных штрихов.

Полученное множество точек на новом слое объединяется в линии (рис. 10) используя метод наименьших квадратов. Уравнение прямой, объединяющей точки (20):

$$y = kx + b, \tag{20}$$

где

$$k = \frac{N\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} x_i^2 - (\sum_{i=1}^{N} x_i)^2}, \tag{21}$$

$$b = \frac{1}{N}(\sum_{i=1}^{N} y_i - k\sum_{i=1}^{N} x_i) \tag{22}$$

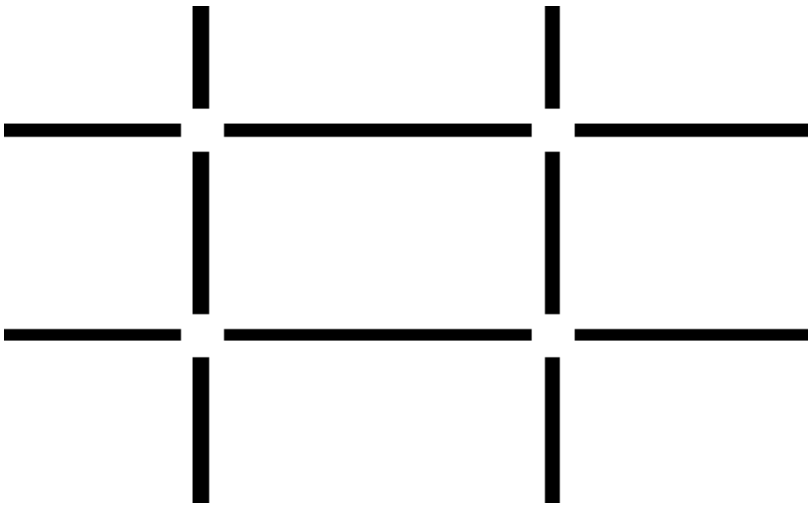

Рисунок 10. Множество точек, объединённых в линии

Полученные отрезки имеют пробелы. Для их устранения анализируется область в радиусе 2-3 ширины линии и если в этой области найдены другие линии и их направление совпадает, то линии объединяются. Если при этом образуется точка пересечения, то она запоминается. Так же запоминаются точки начала и окончания отрезков.

В результате получаем векторную модель бинаризованного изображения (рис. 11).

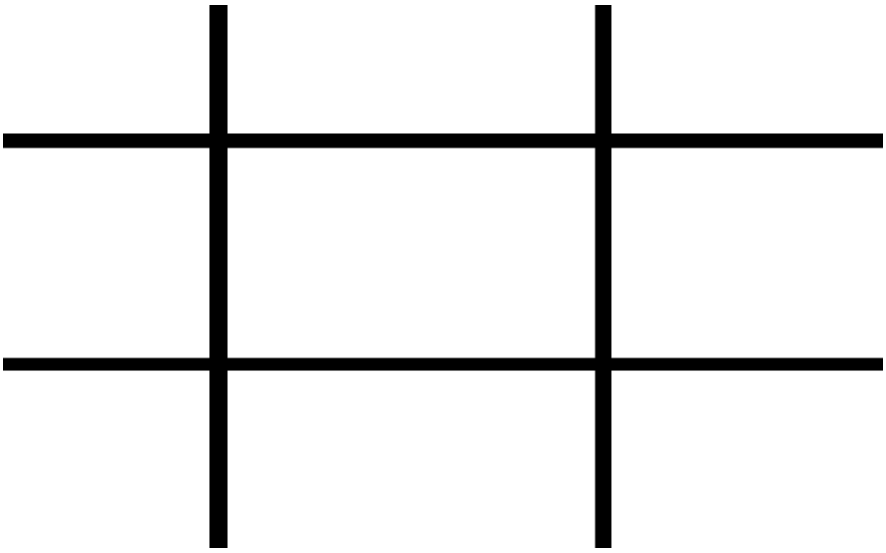

Рисунок 11. Итоговое векторизованное изображение

4. Информационное обеспечение

4.1. Информационное обеспечение системы

Информационное обеспечение системы составляют форматы открываемых и сохраняемых файлов.

4.1.1. Растровые графические форматы

В процессе работы системы могут быть открыты и сохранены растровые изображения в одном их следующих графических форматов:

- BMP;
- JPG;
- PNG;
- TIFF.

Их поддержка обеспечивается средствами OpenCV[9]. Текст лицензионного соглашения приведён в приложении А.

4.1.2. Векторные графические форматы

Полученные в результате работы программы векторные данные могут быть сохранены в один из следующих форматов:

- SVG;
- XML.

4.1.2.1. SVG

Выходной SVG-формат предназначен для визуального просмотра построенной векторной модели.

Основной используемой структурой данных является:

struct LINE {

int x1;	Координата x точки начала отрезка.
int y1;	Координата y точки начала отрезка.
int x2;	Координата x точки окончания отрезка.
int y2;	Координата y точки окончания отрезка.

int color; Цвет отрезка.

};

Пример SVG-файла приведён на рис. 12:

```
<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg"
    width="1892" height="1421">
    <line x1="0" y1="0" x2="10" y2="10" stroke="rgb(0,0,0)" />
    <line x1="10" y1="0" x2="0" y2="10" stroke="rgb(0,0,0)" />
</svg>
```

Рисунок 12. Пример выходного SVG-файла

4.1.2.2. XML

Выходной XML-формат предназначен для хранения векторной модели, а также для хранения точек пересечения объектов.

Пример выходного XML-файла представлен на рис. 13:

```
<?xml version="1.0"?>
<Model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <proections>
    <proection type="front">
      <lines>
        <line type="visible" id="1">
          <points>
            <point x="0" y="0" z="0" id="1" />
            <point x="10" y="10" z="0" id="2" />
          </points>
          <intersertion_points>
            <point id="1" />
          </intersertion_points>
        </line>
        <line type="visible" id="2">
          <points>
            <point x="10" y="0" z="0" id="3" />
            <point x="0" y="10" z="0" id="4" />
          </points>
          <intersertion_points>
            <intersertion_point id="1" />
          </intersertion_points>
        </line>
      </lines>
    </proection>
    ...
  </proections>
  <intersections>
    <intersertion_point x="5" y="5" z="0" id="1">
      <objects>
        <line id="1" />
        <line id="2" />
      </objects>
    </intersertion_point>
  </intersections>
</Model>
```

Рисунок 13. Пример выходного XML-файла

5. Разработанное программное обеспечение

5.1. Условия функционирования

Для использования созданного ПО необходимы:

- операционная система: OS Windows 7 64bit;
- процессор 64-разрядный (x64) процессор (с 1 или более ядер) с тактовой частотой 1 ГГц или выше;
- 2 ГБ оперативной памяти (ОЗУ);
- монитор с пространственным разрешением не ниже 1024x768px;
- видеопроцессор Intel GMA HD или лучше;
- 104-клавишная клавиатура,
- 3-клавишный манипулятор «мышь»;
- установленные библиотеки OpenGL, версии не ниже 2.0;
- .NET Framework 4.5.

Созданное ПО было разработано на языках C++ и C# в среде Microsoft Visual Studio 2015. ПО интегрально содержит 38 классов, объём программного кода составляет ~ 3.5 тыс. операторов.

5.2. Диаграммы связей разработанных классов

Общая диаграмма классов (рис. 14):

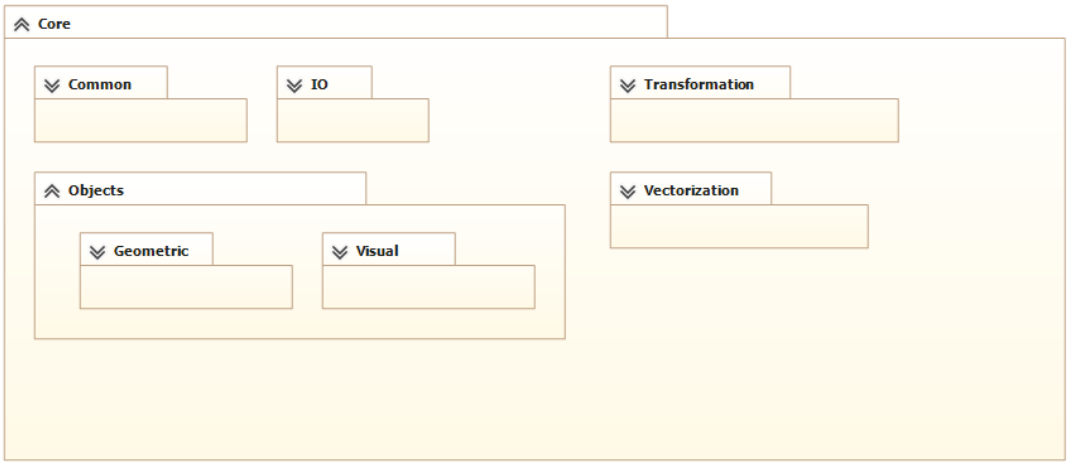

Рисунок 14. Общая диаграмма классов

Диаграмма связей компонента Vectorization (рис. 15):

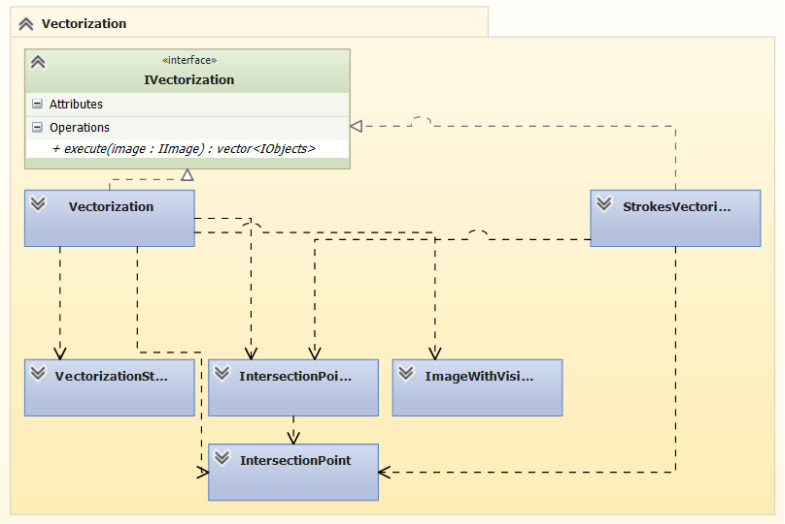

Рисунок 15. Диаграмма связей компонента Vectorization

Диаграмма связей компонента Transformation (рис. 16):

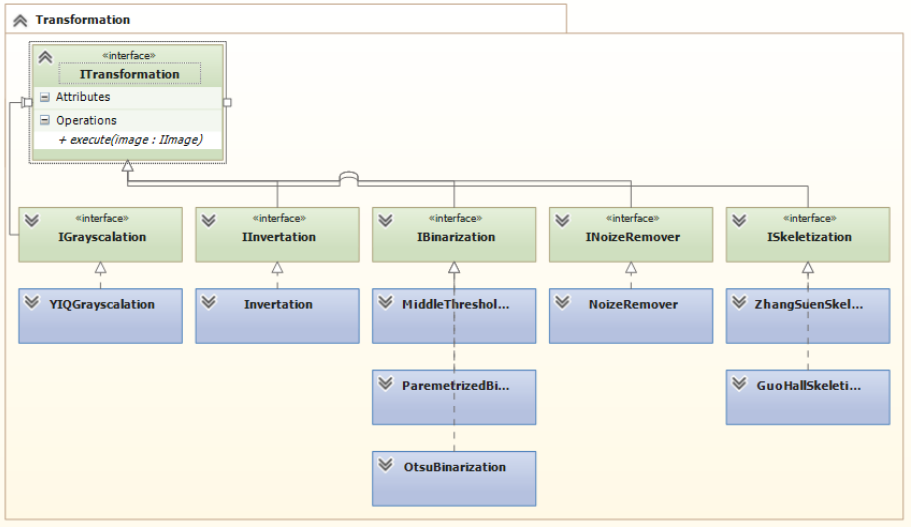

Рисунок 16. Диаграмма связей компонента Transformation

Диаграмма связей компонента Objects (рис. 17):

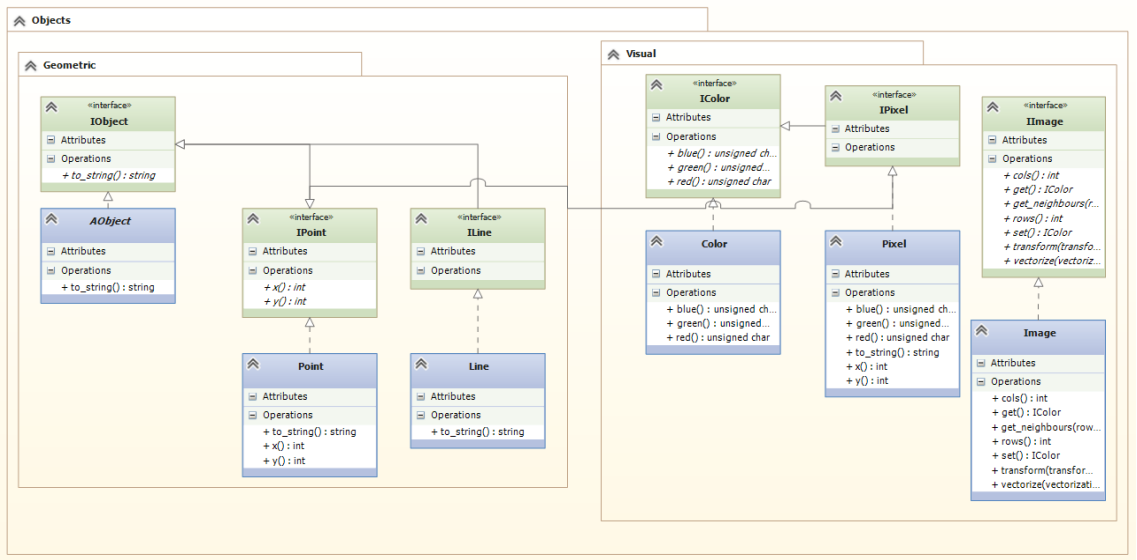

Рисунок 17. Диаграмма связей компонента Objects

Диаграмма связей компонента IO (рис. 18):

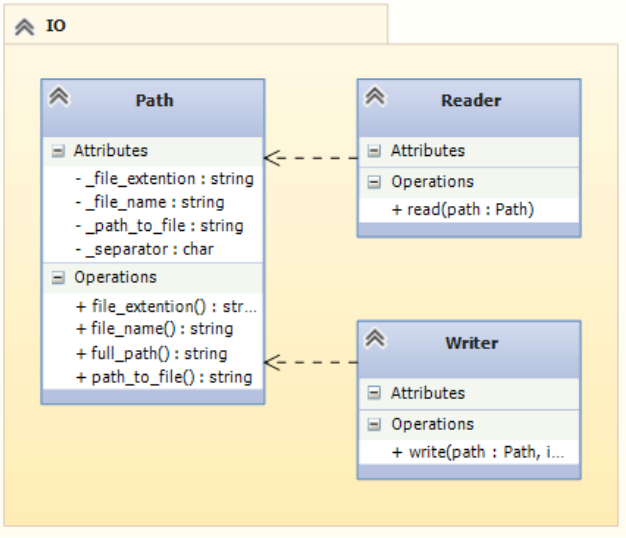

Рисунок 18. Диаграмма связей компонента IO

Диаграмма связей компонента Common (рис. 19):

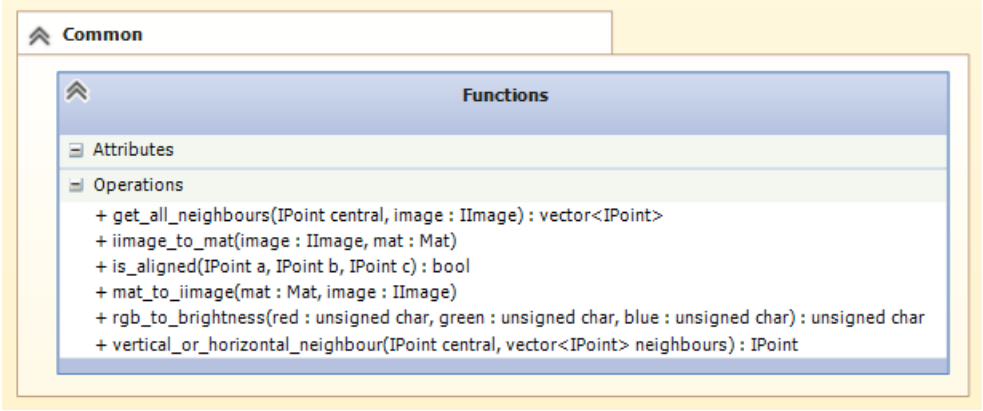

Рисунок 19. Диаграмма связей компонента Common

6. Руководство пользователя к разработанному программному продукту

6.1. Интерфейс

Система представляет собой консольную программу Windows (рис. 20).

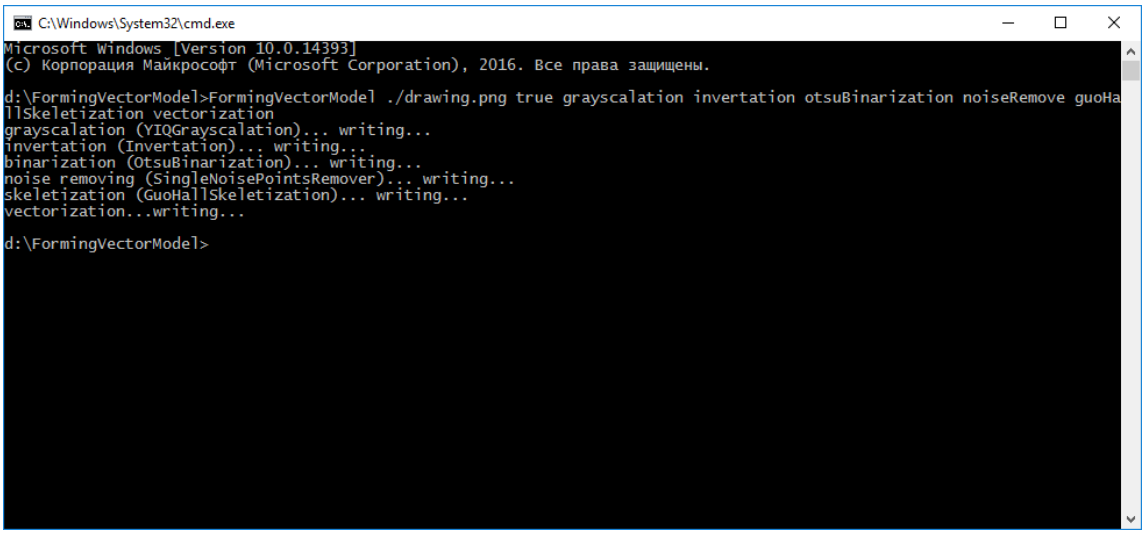

Рисунок 20. Интерфейс системы

6.2. Выполнение программы

Для выполнения программы необходимо запустить exe-файл с определённой последовательностью агрументов.

Список возможных аргументов:

1. Полный или относительный путь к растровому изображению;
2. «true» - если необходимо сохранять промежуточные результаты во время выполнения программы, иначе – «false». Промежуточные результаты будут сохранены в папке исходного файла;
3. «grayscalation» – команда для преобразования изображения в оттенки серого;
4. «invertation» – команда для инвертации изображения;
5. «middleThresholdBinarization» - бинаризация будет выполнена по среднему пороговому значению, «otsuBinarization» - бинаризация будет выполнена с применением метода Оцу, «paremetrizedBinarization» и следующая за ним цифра - бинаризация будет выполнена с заданным пороговым значением;
6. «noiseRemove» - команда для удаления «шумов» на изображении;

7. «zhangSuenSkeletization» - команда для скелетизации изображения с применением алгоритма Зонга-Суня, «guoHallSkeletization» - команда для скелетизации изображения с применением алгоритма Гуо-Хелла;
8. «vectorization» - команда для векторизации изображения.

Если какой-либо из аргументов не указан, то соответствующее действие выполняться не будет.

В процессе выполнения программы будут отображаться выполняемые в данный момент действия.

После завершения выполнения программы результирующий файл будет сохранён в папку с исходным файлом.

Заключение

По результатам выполнения выпускной квалификационной работы сформулируем основные выводы:

- в результате критического анализа литературных данных установлено, что задача формирования векторной модели с сохранением узловых точек на 2D-машиностроительных чертежах является актуальной, сложной, многоаспектной проблемой в технологиях создания электронных моделей изделий;
- изучены существующие средства векторизации растровых изображений, форматы хранения данных, а также существующие аналоги;
- разработаны и реализованы следующие алгоритмы:
  - преобразования изображения в оттенки серого;
  - инвертации изображения;
  - бинаризации по пороговому значению
  - бинаризации с применением метода Оцу;
  - удаления «шумов» на изображении;
  - алгоритм скелетизации Зонга-Суня;
  - алгоритм скелетизации Гуо-Хелла;
  - алгоритм векторизации;
  - штриховой алгоритм векторизации.
- разработаны компоненты работы с растровыми файлами BMP, JPG, PNG и TIFF; а также SVG и XML-файлами;
- на базе созданных алгоритмов и библиотек создано ПО формирования векторной модели с сохранением узловых точек 2D-машиностроительных чертежей;
- проведена практическая апробация программного комплекса и выявлена его работоспособность на реальных данных.

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. Болотова, Ю.А. Распознавание автомобильных номеров на основе метода связных компонент и иерархической временной сети / Ю.А. Болотова, В.Г. Спицын, М.Н. Рудометкина // Компьютерная оптика – 2015 – том 39, №2
6. Bryan S. Morse, Lecture 4: Thresholding. Brigham Young University, 2000, pp. 1-5
7. Щербаков, В.С. Автоматизация проектирования устройств управления положением платформы строительной машины / В.С. Щербаков, М.С. Корытов, М.Г. Григорьев / Омск: СибАДИ, 2011. – 119 с.
8. Racter Arts – Режим доступа: http://rasterarts.ru/ – Загл. с экрана.
9. OpenCV library – Режим доступа: http://opencv.org/ – Загл. с экрана.
10. Implementation of thinning algorithm in OpenCV – Режим доступа: http://opencv-code.com/quick-tips/implementation-of-thinning-algorithm-in-opencv/ – Загл. с экрана.
11. Implementation of Guo-Hall thinning algorithm – Режим доступа: http://opencv-code.com/quick-tips/implementation-of-guo-hall-thinning-algorithm/ – Загл. с экрана.
12. The YIQ Color Model – Режим доступа: http://www.mat.univie.ac.at/~kriegl/Skripten/CG/node14.html – Загл. с экрана.
13. RasterDesk Pro 11.0 – Режим доступа: http://www.csoft.ru/catalog/soft/rasterdesk-pro/rasterdesk-pro-11.html – Загл. с экрана.
14. Spotlight 11.0 – Режим доступа: http://www.csoft.ru/catalog/soft/spotlight/spotlight-11.html – Загл. с экрана.

Приложения

Приложение А. Лицензионное соглашение OpenCV

License Agreement

For Open Source Computer Vision Library

(3-clause BSD License)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the copyright holders nor the names of the contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall copyright holders or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.