

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 Васин Д.Ю.

подпись

Нижний Новгород
2017

Аннотация

В работе содержится описание информационного, алгоритмического и программного обеспечения системы формирования векторной и топологической модели 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	6
Введение	7
1. Теоретическая часть	9
1.1. Существующие системы векторизации растровых изображений	9
1.1.1. RasterDesk	9
1.1.2. Spotlight	10
2. Постановка задачи	11
3. Алгоритмическое обеспечение	12
3.1. Преобразование изображения в оттенки серого	12
3.2. Инвертация изображения	13
3.3. Бинаризация изображения	13
3.3.1. Бинаризация по пороговому значению	14
3.3.2. Бинаризация с применением метода Оцу	14
3.4. Удаление «шумов» на изображении	15
3.5. Скелетизация	16
3.5.1. Алгоритм Зонга-Суня (Zhang-Suen)	16
3.5.2. Алгоритм Гуо-Хелла (Guo-Hall)	17
3.6. Векторизация	18
3.6.1. Алгоритм векторизации	19
3.6.2. Штриховой алгоритм векторизации	22
4. Информационное обеспечение	24
4.1. Информационное обеспечение системы	24
4.1.1. Растровые графические форматы	24
4.1.2. Векторные графические форматы	24
5. Разработанное программное обеспечение	26
5.1. Условия функционирования	26
5.2. Диаграммы связей разработанных классов	26

5.3. Описание разработанных компонентов	29
5.3.1. Описание класса IObject	29
5.3.2. Описание класса IPoint	29
5.3.3. Описание класса Point.....	30
5.3.4. Описание класса ILine.....	32
5.3.5. Описание класса Line	33
5.3.6. Описание класса IColor.....	34
5.3.7. Описание класса Color	36
5.3.8. Описание класса IPixel.....	39
5.3.9. Описание класса Pixel.....	39
5.3.10. Описание класса Image	40
5.3.11. Описание класса Image	42
5.3.12. Описание класса Drawing	45
5.3.13. Описание класса ITransformation	46
5.3.14. Описание класса IGrayscale	47
5.3.15. Описание класса YIQGrayscale	48
5.3.16. Описание класса Invertation	48
5.3.17. Описание класса Invertation.....	49
5.3.18. Описание класса IBinarization	49
5.3.19. Описание класса ParametrizedBinarization	49
5.3.20. Описание класса MiddleThresholdBinarization.....	50
5.3.21. Описание класса OtsuBinarization	50
5.3.22. Описание класса INoiseRemover	51
5.3.23. Описание класса SingleNoisePointsRemover	52
5.3.24. Описание класса ISkeletization	52
5.3.25. Описание класса ZhangSuenSkeletization	53
5.3.26. Описание класса GuoHallSkeletization.....	54
5.3.27. Описание класса IVectorization	54

5.3.28. Описание класса Vectorization	55
5.3.29. Описание класса StrokesVectorization	55
5.3.30. Описание класса VectorizationState	56
5.3.31. Описание класса IntersectionPoint	58
5.3.32. Описание класса IntersectionPointsSet	60
5.3.33. Описание класса ImageWithVisitedMarks	61
5.3.34. Описание класса Path	64
5.3.35. Описание класса Reader	65
5.3.36. Описание класса Writer	66
5.3.37. Описание класса Functions	68
6. Руководство пользователя к разработанному программному продукту	70
6.1. Интерфейс	70
6.2. Выполнение программы	70
7. Результаты работы программного продукта	72
Заключение.....	74
Литература	75
Приложения	76
Приложение А. Лицензионное соглашение OpenCV	76

Глоссарий

Растровое изображение – изображение, представляющее собой матрицу пикселей.

Векторное изображение – изображение, использующее математическое описание изображённых геометрических объектов.

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

Скелетизация изображения – преобразование всех линий изображения таким образом, чтобы их толщина была не более 1 пикселя.

Топологическая модель – установление местоположения растровых объектов в геометрическом пространстве и их взаимодействия друг с другом.

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 приводя к искажениям, 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 и топологической модели 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 топологической модели) 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 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

Задан растровый машиностроительный чертёж, сканированный с бумажной копии формата A4 со значением DPI равным 50 - 300, представляющий собой 2D проекции 3D модели машиностроительной детали, геометрическими размерами не более 5000x5000 пикселей, глубиной цвета 8, 16 или 32 бита, представленный одним из стандартных распространённых форматов (BMP, JPG, PNG или TIFF).

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

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

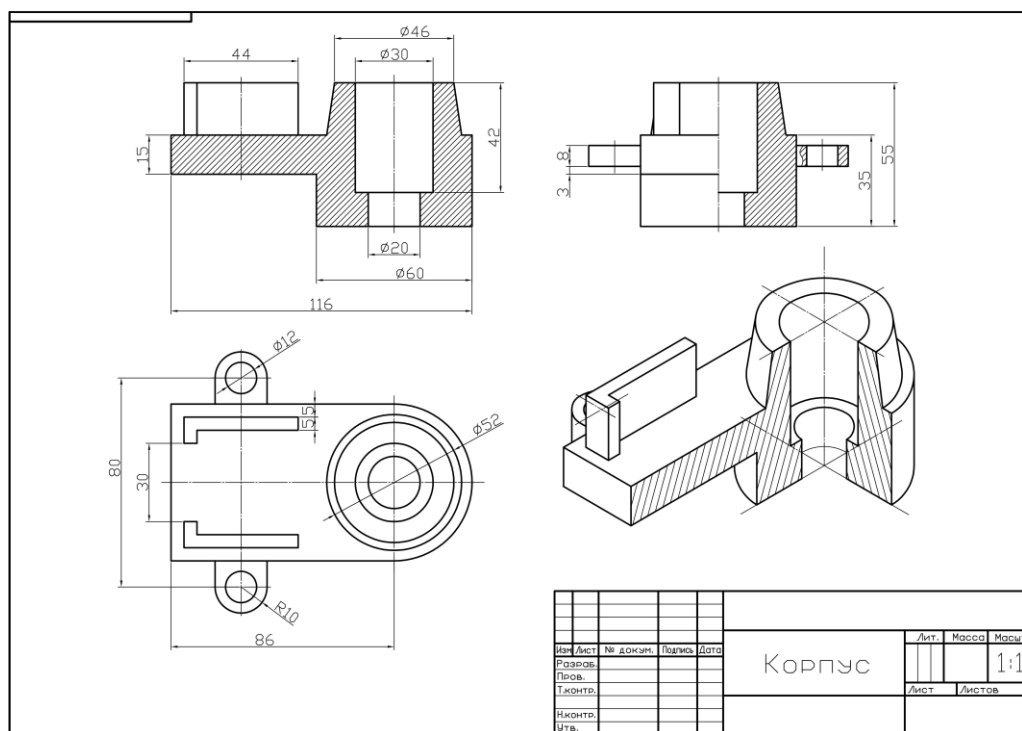


Рисунок 1. Пример чертежа

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»

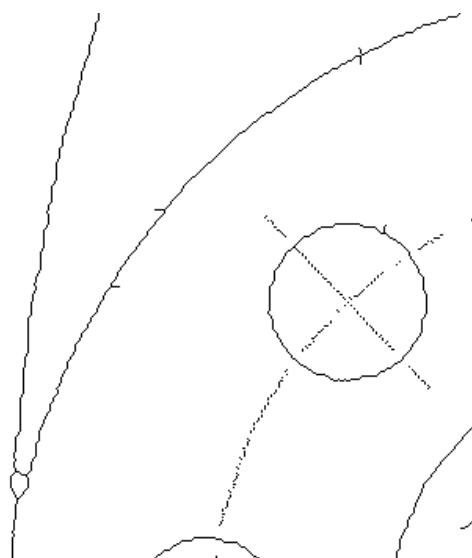


Рисунок 3. Результат векторизации с предварительным удалением «шума»

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.

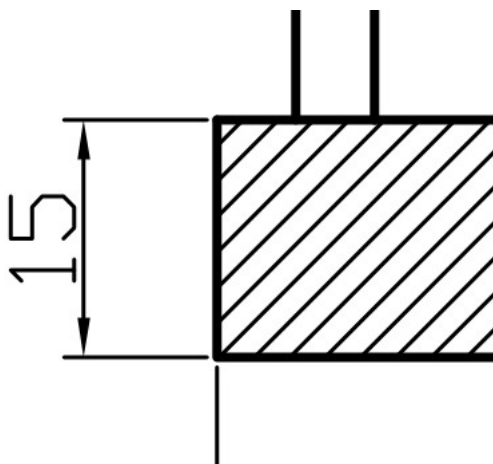


Рисунок 4. Изображение до применения скелетизации

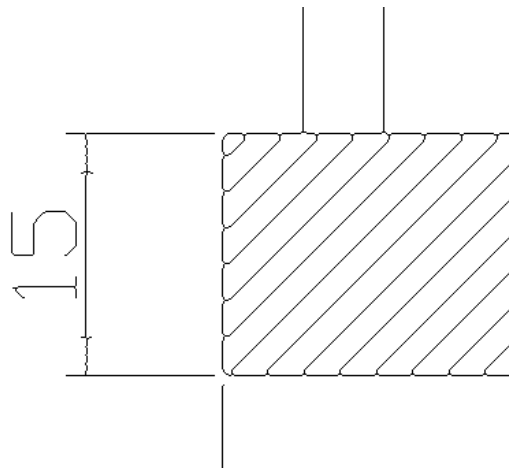


Рисунок 5. Изображение после применения скелетизации

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):

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

На второй подитерации пиксель, являющийся элементом P_1 , удаляется, если выполняются условия (19):

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

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).

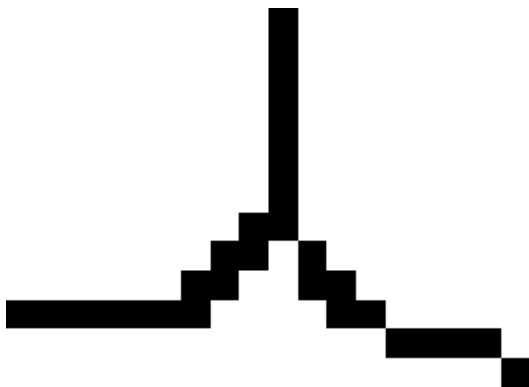


Рисунок 7. Пример выполнения алгоритма Зонга-Суня

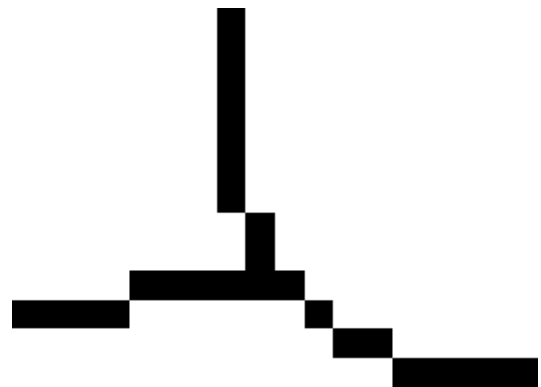


Рисунок 8. Пример выполнения алгоритма Гуо-Хелла

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 топологической модели 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 3×3 , 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.

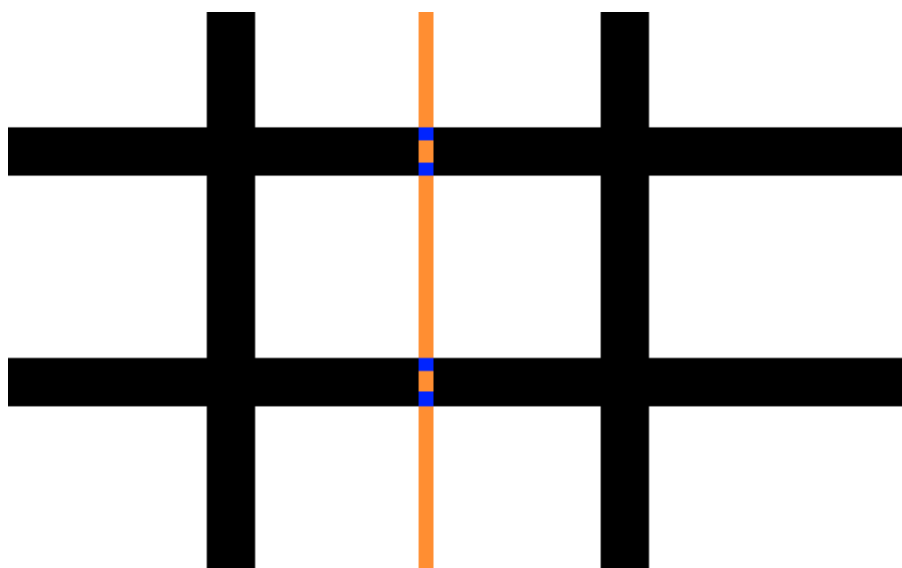


Рисунок 9. Проведение штриха по изображению

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):

$$y = kx + b, \quad (20)$$

где

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

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

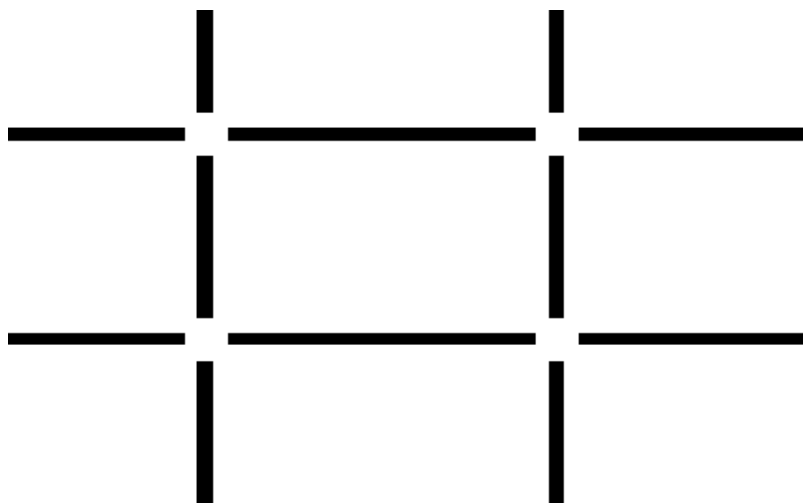


Рисунок 10. Множество точек, объединённых в линии

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).

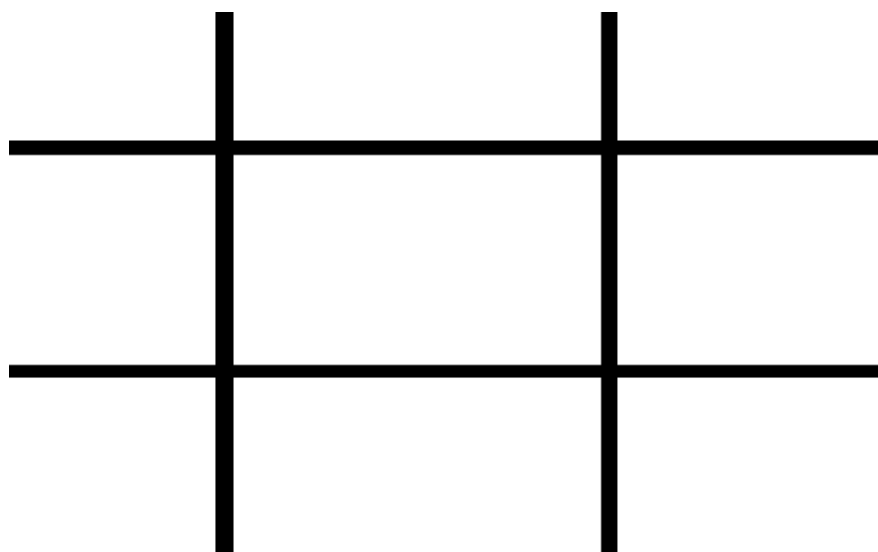


Рисунок 11. Итоговое векторизованное изображение

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 из 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>
          <intersersion_points>
            <point id="1" />
          </intersersion_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>
          <intersersion_points>
            <intersersion_point id="1" />
          </intersersion_points>
        </line>
      </lines>
    </proection>
    ...
  </proections>
  <intersections>
    <intersersion_point x="5" y="5" z="0" id="1">
      <objects>
        <line id="1" />
        <line id="2" />
      </objects>
    </intersersion_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 37 классов, объём программного кода составляет ~ 3.5 тыс. операторов.

5.2. Диаграммы связей разработанных классов

Общая диаграмма классов (рис. 14):

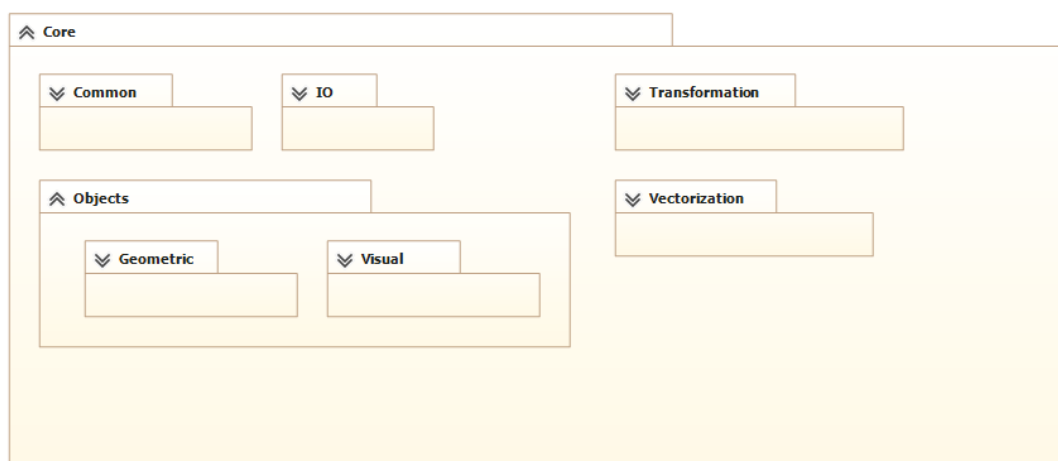


Рисунок 14. Общая диаграмма классов

Диаграмма связей компонента Vectorization (рис. 15):

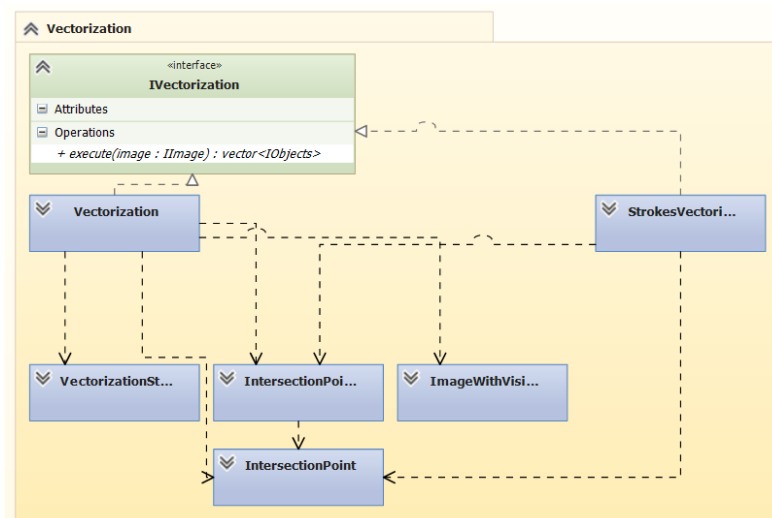


Рисунок 15. Диаграмма связей компонента Vectorization

Диаграмма связей компонента Transformation (рис. 16):

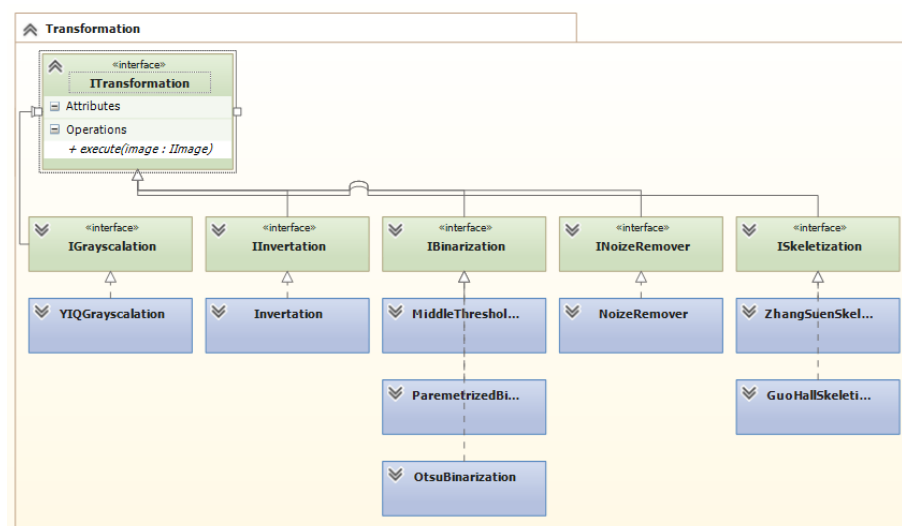


Рисунок 16. Диаграмма связей компонента Transformation

Диаграмма связей компонента Objects (рис. 17):

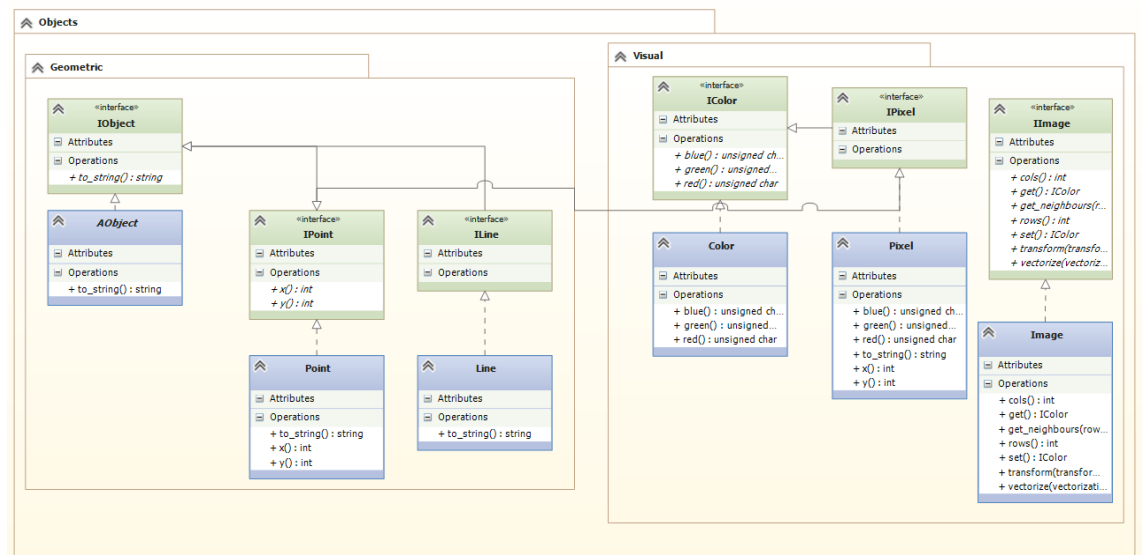


Рисунок 17. Диаграмма связей компонента Objects

Диаграмма связей компонента IO (рис. 18):

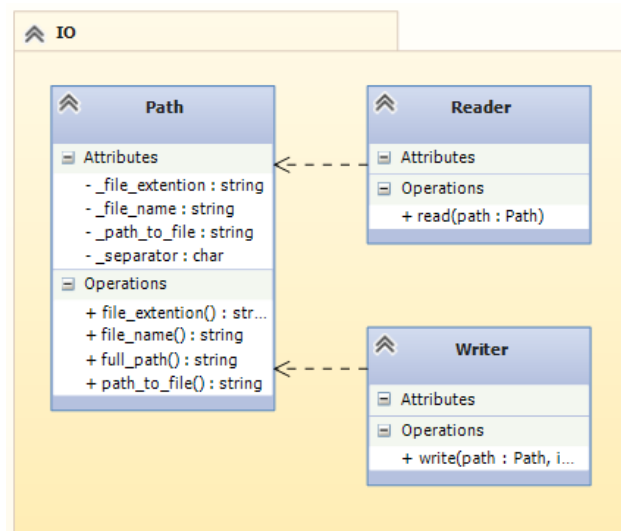


Рисунок 18. Диаграмма связей компонента IO

Диаграмма связей компонента Common (рис. 19):

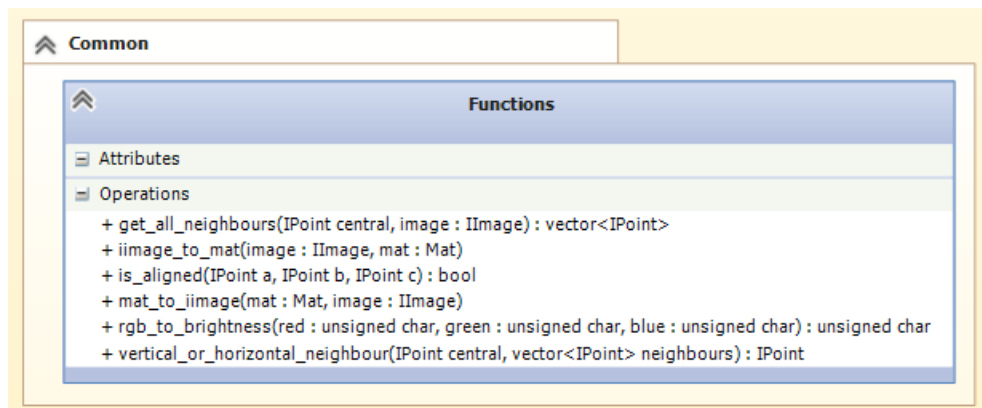


Рисунок 19. Диаграмма связей компонента Common

5.3. Описание разработанных компонентов

5.3.1. Описание класса IObject

5.3.1.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) всех геометрических объектов.

5.3.1.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: virtual ~IObject().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод преобразования объекта в строковое представление

Сигнатура: virtual std::string to_string() = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа std::string содержит строковое представление объекта.

Описание: метод преобразования объекта в строковое представление по его содержимому.

5.3.2. Описание класса IPoint

5.3.2.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) объекта точки.

5.3.2.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: virtual ~IPoint().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к координате X

Сигнатура: virtual int x() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является X-координата точки.

Описание: возвращает значение X-координаты точки.

Метод доступа к координате Y

Сигнатура: virtual int y() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является Y-координата точки.

Описание: возвращает значение Y-координаты точки.

5.3.3. Описание класса Point

5.3.3.1. Назначение

Класс предназначен для реализации интерфейса точки IPoint.

5.3.3.2. Публичные методы

Конструктор

Сигнатура: Point().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских координат

Сигнатура: Point(int x, int y).

Входные и выходные параметры:

Входные параметры:

— x, y – координаты точкию

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор копирования с передачей интерфейса

Сигнатура: Point(IPoint* other).

Входные и выходные параметры:

Входные параметры:

- other – объект, реализующий интерфейс IPoint.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор копирования с передачей объекта Point

Сигнатура: Point(const Point& other).

Входные и выходные параметры:

Входные параметры:

- other – объект Point.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к координате X

Сигнатура: int x() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является X-координата точки.

Описание: возвращает значение X-координаты точки.

Метод доступа к координате Y

Сигнатура: int y() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является Y-координата точки.

Описание: возвращает значение Y-координаты точки.

Метод преобразования объекта в строковое представление

Сигнатура: std::string to_string() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа std::string содержит строковое представление объекта.

Описание: метод преобразования объекта в строковое представление по его содержимому.

5.3.4. Описание класса ILine

5.3.4.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) объекта отрезка.

5.3.4.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: virtual ~ILine().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к начальной точке

Сигнатура: virtual IPoint* begin() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является начальная точка отрезка.

Описание: возвращает начальную точку отрезка.

Метод доступа к конечной точке

Сигнатура: virtual IPoint* end () const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является конечная точка отрезка.

Описание: возвращает конечную точку отрезка.

Метод доступа к типу отрезка

Сигнатура: virtual Type type() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является тип отрезка.

Описание: возвращает тип отрезка.

Метод доступа задания типа отрезка

Сигнатура: virtual void type(Type value) = 0.

Входные и выходные параметры: Входным параметром является тип отрезка.

Выходные параметры отсутствуют.

Описание: задаёт тип отрезка.

5.3.5. Описание класса Line

5.3.5.1. Назначение

Класс предназначен для реализации интерфейса отрезка ILine.

5.3.5.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: Line().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских точек начала и окончания

Сигнатура: Line(IPoint* begin, IPoint* end).

Входные и выходные параметры:

Входные параметры:

- begin – начальная точка;
- end – конечная точка.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских точек начала и окончания и точек пересечения

Сигнатура: Line(IPoint* begin, IPoint* end, std::set<Vectorization::IntersectionPoint*> intersections).

Входные и выходные параметры:

Входные параметры:

- begin – начальная точка;
- end – конечная точка;
- intersections – множество точек пересечения.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Деструктор

Сигнатура: virtual ~Line().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к начальной точке

Сигнатура: `IPoint* begin() const override`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является начальная точка отрезка.

Описание: возвращает начальную точку отрезка.

Метод доступа к конечной точке

Сигнатура: `IPoint* end() const override`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является конечная точка отрезка.

Описание: возвращает конечную точку отрезка.

Метод доступа к типу отрезка

Сигнатура: `Type type() const override`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является тип отрезка.

Описание: возвращает тип отрезка.

Метод доступа задания типа отрезка

Сигнатура: `void type(Type value) override`.

Входные и выходные параметры: Входным параметром является тип отрезка.

Выходные параметры отсутствуют.

Описание: задаёт тип отрезка.

Метод преобразования объекта в строковое представление

Сигнатура: `std::string to_string() override`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа `std::string` содержит строковое представление объекта.

Описание: метод преобразования объекта в строковое представление по его содержимому.

5.3.6. Описание класса IColor

5.3.6.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) цвета в пространстве RGB.

5.3.6.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: virtual ~IColor().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к значению красного цвета

Сигнатура: virtual unsigned char red() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала R.

Описание: возвращает яркостную характеристику канала R.

Метод изменения значения красного цвета

Сигнатура: virtual void red(unsigned char value) = 0.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала R.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала R.

Метод доступа к значению зелёного цвета

Сигнатура: virtual unsigned char green() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала G.

Описание: возвращает яркостную характеристику канала G.

Метод изменения значения зелёного цвета

Сигнатура: virtual void green(unsigned char value) = 0.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала G.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала G.

Метод доступа к значению синего цвета

Сигнатура: virtual unsigned char blue() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала В.

Описание: возвращает яркостную характеристику канала В.

Метод изменения значения синего цвета

Сигнатура: virtual void blue(unsigned char value) = 0.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала В.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала В.

Метод доступа к значению общей яркости цвета

Сигнатура: virtual unsigned char brightness() const = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является общей яркостной характеристикой цвета.

Описание: возвращает общую яркостную характеристику цвета.

Метод изменения значения общей яркости цвета

Сигнатура: virtual void brightness(unsigned char value) = 0.

Входные и выходные параметры:

Входные параметры:

- value – общая яркостная характеристика цвета.

Выходные параметры отсутствуют.

Описание: изменяет общую яркостную характеристику цвета.

5.3.7. Описание класса Color

5.3.7.1. Назначение

Класс предназначен для реализации интерфейса IColor.

5.3.7.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: Color().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием общей яркости цвета

Сигнатура: Color(unsigned char brightness).

Входные и выходные параметры:

Входные параметры:

- brightness – общая яркость.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием яркостей каналов RGB

Сигнатура: Color(unsigned char red, unsigned char green, unsigned char blue).

Входные и выходные параметры:

Входные параметры:

- red – яркость канала R;
- green – яркость канала G;
- blue – яркость канала B.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор копирования

Сигнатура: Color(IColor* c)

Входные и выходные параметры:

Входные параметры:

- c – объект, реализующий интерфейс IColor.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к значению красного цвета

Сигнатура: unsigned char red() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала R.

Описание: возвращает яркостную характеристику канала R.

Метод изменения значения красного цвета

Сигнатура: void red(unsigned char value) override.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала R.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала R.

Метод доступа к значению зелёного цвета

Сигнатура: unsigned char green() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала G.

Описание: возвращает яркостную характеристику канала G.

Метод изменения значения зелёного цвета

Сигнатура: void green(unsigned char value) override.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала G.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала G.

Метод доступа к значению синего цвета

Сигнатура: unsigned char blue() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является яркостной характеристикой канала B.

Описание: возвращает яркостную характеристику канала B.

Метод изменения значения синего цвета

Сигнатура: void blue(unsigned char value) override.

Входные и выходные параметры:

Входные параметры:

- value – яркостная характеристика канала B.

Выходные параметры отсутствуют.

Описание: изменяет яркостную характеристику канала B.

Метод доступа к значению общей яркости цвета

Сигнатура: unsigned char brightness() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа unsigned char является общей яркостной характеристикой цвета.

Описание: возвращает общую яркостную характеристику цвета.

Метод изменения значения общей яркости цвета

Сигнатура: void brightness(unsigned char value) override.

Входные и выходные параметры:

Входные параметры:

- value – общая яркостная характеристика цвета.

Выходные параметры отсутствуют.

Описание: изменяет общую яркостную характеристику цвета.

5.3.8. Описание класса IPixel

5.3.8.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) пикселя.

5.3.8.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~IPixel ().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.9. Описание класса Pixel

5.3.9.1. Назначение

Класс предназначен для реализации интерфейса пикселя (IPixel).

5.3.9.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: Pixel().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских объектов точки и цвета

Сигнатура: Pixel(IPoint* point, IColor* color).

Входные и выходные параметры:

Входные параметры:

- point – объект точки (содержит координаты);
- color – объект цвета.

Описание: метод инициализации объекта.

Конструктор копирования

Сигнатура: Pixel(IPixel* other).

Входные и выходные параметры:

Входные параметры:

- other – объект, реализующий интерфейс IPixel.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Деструктор

Сигнатура: ~Pixel().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Также содержит методы, аналогичные методам в разделах 5.3.3.2 и 5.3.7.2.

5.3.10. Описание класса Pimage

5.3.10.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) изображения.

5.3.10.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~Pimage().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к количеству строк изображения

Сигнатура: virtual int rows() = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством строк изображения.

Описание: возвращает количество строк изображения.

Метод доступа к количеству столбцов изображения

Сигнатура: virtual int cols() = 0.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством столбцов изображения.

Описание: возвращает количество столбцов изображения.

Метод доступа к элементу изображения

Сигнатура: virtual IColor* get(int row, int col) = 0.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца.

Выходной параметр типа IColor является цветом в заданной ячейке.

Описание: возвращает цвет в заданной координатами ячейке.

Метод изменения элемента изображения

Сигнатура: virtual void set(int row, int col, IColor* value) = 0.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца;
- value – объект цвета, который будет записан.

Выходные параметры отсутствуют.

Описание: изменяет цвет в заданной координатами ячейке.

Метод преобразования изображения

Сигнатура: virtual void transform(Transformation::ITransformation* transforamtion) = 0.

Входные и выходные параметры:

Входные параметры:

- transforamtion – объект преобразования.

Выходные параметры отсутствуют.

Описание: выполняет преобразование изображения.

Метод векторизации изображения

Сигнатура: virtual std::vector<Geometric::IObject*> vectorize (Vectorization::IVectorization* vectorization) = 0.

Входные и выходные параметры:

Входные параметры:

- vectorization – объект векторизации.

Выходной параметр типа std::vector<Geometric::IObject*> содержит полученные в результате векторизации объекты.

Описание: выполняет векторизацию изображения.

Метод получения соседних элементов заданной ячейки

Сигнатура: virtual std::vector<IColor*> get_neighbours(int row, int col, unsigned char brightness = IColor::binary_color::turn_on) = 0.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца;
- brightness – цвет выбираемых соседних ячеек.

Выходной параметр типа std::vector<IColor*> является соседними ячейками заданного цвета.

Описание: возвращает соседние элементы заданной ячейки изображения.

5.3.11. Описание класса Image

5.3.11.1. Назначение

Класс предназначен для реализации интерфейса изображения (Image).

5.3.11.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: Image().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских размеров изображения

Сигнатура: Image(int rows, int cols).

Входные и выходные параметры:

Входные параметры:

- rows – число строк;
- cols – число столбцов.

Описание: метод инициализации объекта.

Деструктор

Сигнатура: ~ Image().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к количеству строк изображения

Сигнатура: int rows() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством строк изображения.

Описание: возвращает количество строк изображения.

Метод доступа к количеству столбцов изображения

Сигнатура: int cols() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством столбцов изображения.

Описание: возвращает количество столбцов изображения.

Метод доступа к элементу изображения

Сигнатура: IColor* get(int row, int col) override.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца.

Выходной параметр типа IColor является цветом в заданной ячейке.

Описание: возвращает цвет в заданной координатами ячейке.

Метод изменения элемента изображения

Сигнатура: void set(int row, int col, IColor* value) override.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;

- col – номер столбца;
- value – объект цвета, который будет записан.

Выходные параметры отсутствуют.

Описание: изменяет цвет в заданной координатами ячейке.

Метод преобразования изображения

Сигнатура: void transform(Transformation::ITransformation* transformtion) override.

Входные и выходные параметры:

Входные параметры:

- transformtion – объект преобразования.

Выходные параметры отсутствуют.

Описание: выполняет преобразование изображения.

Метод векторизации изображения

Сигнатура: std::vector<Geometric::IObject*> vectorize(Vectorization::IVectorization* vectorization) override.

Входные и выходные параметры:

Входные параметры:

- vectorization – объект векторизации.

Выходной параметр типа std::vector<Geometric::IObject*> содержит полученные в результате векторизации объекты.

Описание: выполняет векторизацию изображения.

Метод получения соседних элементов заданной ячейки

Сигнатура: std::vector<IColor*> get_neighbours(int row, int col, unsigned char brightness = IColor::binary_color::turn_on) override.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца;
- brightness – цвет выбираемых соседних ячеек.

Выходной параметр типа std::vector<IColor*> соседние ячейки заданного цвета.

Описание: соседние элементы заданной ячейки изображения.

5.3.12. Описание класса Drawing

5.3.12.1. Назначение

Класс предназначен для задания чертежа в трёх проекциях.

5.3.12.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: Drawing().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор

Сигнатура: Drawing(std::vector<Geometric::ILine*> front_view, std::vector<Geometric::ILine*> top_view, std::vector<Geometric::ILine*> side_view).

Входные и выходные параметры:

Входные параметры:

- front_view – массив отрезков фронтального вида;
- top_view – массив отрезков вида сверху;
- side_view – массив отрезков вида сбоку.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к объектам фронтальной проекции

Сигнатура: std::vector<Geometric::ILine*> front_view() const

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является объект типа std::vector<Geometric::ILine*>, содержащий массив отрезков проекции.

Описание: возвращает массив отрезков фронтальной проекции.

Метод изменения объектов фронтальной проекции

Сигнатура: void front_view(std::vector<Geometric::ILine*> value)

Входные и выходные параметры:

Входные параметры:

- value - массив отрезков проекции.

Выходные параметры отсутствуют.

Описание: изменяет массив отрезков фронтальной проекции.

Метод доступа к объектам проекции сверху

Сигнатура: `std::vector<Geometric::ILine*> top_view() const`

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является объект типа `std::vector<Geometric::ILine*>`, содержащий массив отрезков проекции.

Описание: возвращает массив отрезков проекции сверху.

Метод изменения объектов проекции сверху

Сигнатура: `void top_view(std::vector<Geometric::ILine*> value)`

Входные и выходные параметры:

Входные параметры:

- value - массив отрезков проекции.

Выходные параметры отсутствуют.

Описание: изменяет массив отрезков проекции сверху.

Метод доступа к объектам проекции сбоку

Сигнатура: `std::vector<Geometric::ILine*> side_view () const`

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является объект типа `std::vector<Geometric::ILine*>`, содержащий массив отрезков проекции.

Описание: возвращает массив отрезков проекции сбоку.

Метод изменения объектов проекции сбоку

Сигнатура: `void side_view(std::vector<Geometric::ILine*> value)`

Входные и выходные параметры:

Входные параметры:

- value - массив отрезков проекции.

Выходные параметры отсутствуют.

Описание: изменяет массив отрезков проекции сбоку.

5.3.13. Описание класса ITransformation

5.3.13.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) преобразования изображения.

5.3.13.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~ITransformation().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод преобразования изображения

Сигнатура: virtual void execute(Objects::Visual::Image* image) = 0.

Входные и выходные параметры:

Входные параметры:

- image – изображение, подлежащее преобразованию

Выходные параметры отсутствуют.

Описание: выполняет преобразование изображения.

5.3.14. Описание класса IGrayscale

5.3.14.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) преобразования изображения в оттенки серого.

5.3.14.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~IGrayscale().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.15. Описание класса YIQGrayscale

5.3.15.1. Назначение

Класс предназначен для реализации интерфейса IGrayscale. Преобразование в оттенки серого выполняется при помощи YIQ-модели.

Алгоритм описан в разделе 3.1.

5.3.15.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Метод преобразования изображения

Сигнатура: virtual void execute(Objects::Visual::Image* image) = 0.

Входные и выходные параметры:

Входные параметры:

- image – изображение, подлежащее преобразованию

Выходные параметры отсутствуют.

Описание: выполняет преобразование изображения.

5.3.16. Описание класса Invertation

5.3.16.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) инвертирования цветов изображения.

5.3.16.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~Invertation().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.17. Описание класса Invertation

5.3.17.1. Назначение

Класс предназначен для реализации интерфейса Invertation.

Алгоритм описан в разделе 3.2.

5.3.17.2. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.18. Описание класса IBinarization

5.3.18.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) бинаризации изображения.

5.3.18.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~IBinarization().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.19. Описание класса ParemetrizedBinarization

5.3.19.1. Назначение

Класс предназначен для реализации интерфейса IBinarization. Бинаризация выполняется по заданному пороговому значению.

Алгоритм описан в разделе 3.3.1.

5.3.19.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: ParametrizedBinarization(int threshold).

Входные и выходные параметры:

Входные параметры:

- threshold – пороговое значение.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод преобразования изображения

Сигнатура: virtual void execute(Objects::Visual::Image* image) = 0.

Входные и выходные параметры:

Входные параметры:

- image – изображение, подлежащее преобразованию

Выходные параметры отсутствуют.

Описание: выполняет преобразование изображения.

5.3.20. Описание класса MiddleThresholdBinarization

5.3.20.1. Назначение

Класс предназначен для реализации интерфейса IBinarization. Бинаризация выполняется по среднему пороговому значению.

5.3.20.2. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.21. Описание класса OtsuBinarization

5.3.21.1. Назначение

Класс предназначен для реализации интерфейса IBinarization. Бинаризация выполняется по пороговому значению, определяемому при помощи метода Оцу.

Алгоритм описан в разделе 3.3.2.

5.3.21.2. Приватные методы

Методы, описанные в этом разделе, являются приватными, так как не входят в интерфейс работы с рассматриваемым классом и не предполагают непосредственной работы пользователя с ними.

Метод вычисления порогового значения бинаризации по гистограмме

Сигнатура: unsigned char threshold().

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа unsigned char, представляющее пороговое значение бинаризации.

Описание: Метод вычисления порогового значения бинаризации.

Метод вычисления гистограмме

Сигнатура: std::vector<int> get_histogram(int *min, int *max).

Входные и выходные параметры: Входные параметры отсутствуют.

Выходные параметры:

- возвращаемое функцией значение типа unsigned char, представляющее пороговое значение бинаризации;
- min — указатель на переменную для записи минимального значения гистограммы;
- max — указатель на переменную для записи максимального значения гистограммы.

Описание: Метод вычисления гистограммы изображения.

5.3.21.3. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.22. Описание класса INoiseRemover

5.3.22.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) удаления «шумов» на изображении.

5.3.22.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~INoiseRemover().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.23. Описание класса SingleNoisePointsRemover

5.3.23.1. Назначение

Класс предназначен для реализации интерфейса INoiseRemover. Удаляются одиночные пиксели, все соседние пиксели которого имеют противоположный цвет.

Алгоритм описан в разделе 3.4.

5.3.23.2. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.24. Описание класса ISkeletization

5.3.24.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) скелетизации изображения.

5.3.24.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~ISkeletization().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

5.3.25. Описание класса ZhangSuenSkeletization

5.3.25.1. Назначение

Класс предназначен для реализации интерфейса ISkeletization. Скелетизация выполняется с применением алгоритма Зонга-Суня.

Алгоритм описан в разделе 3.5.1.

5.3.25.2. Приватные методы

Методы, описанные в этом разделе, являются приватными, так как не входят в интерфейс работы с рассматриваемым классом и не предполагают непосредственной работы пользователя с ними.

Метод утоньшения объектов изображения.

Сигнатура: void thinning(cv::Mat& im).

Входные и выходные параметры:

Входные параметры:

- im – изображение, представленное в виде матрицы (объект OpenCV) пикселей.

Выходные параметры отсутствуют.

Описание: метод выполняет итеративное утоньшение объектов изображения.

Метод выполнения итерации утоньшения.

Сигнатура: void thinning_iteration(cv::Mat& im, int iter).

Входные и выходные параметры:

Входные параметры:

- im – изображение, представленное в виде матрицы (объект OpenCV) пикселей;
- iter – номер подитерации.

Выходные параметры отсутствуют.

Описание: метод выполняет удаление пикселей в зависимости от номера подитерации.

5.3.25.3. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.26. Описание класса GuoHallSkeletization

5.3.26.1. Назначение

Класс предназначен для реализации интерфейса ISkeletization. Скелетизация выполняется с применением алгоритма Гуо-Хелла.

Алгоритм описан в разделе 3.5.2.

5.3.26.2. Приватные методы

Приватные методы аналогичны описанным в разделе 5.3.25.2.

5.3.26.3. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.15.2

5.3.27. Описание класса IVectorization

5.3.27.1. Назначение

Класс предназначен для задания интерфейса (полностью абстрактный класс) векторизации изображения.

5.3.27.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Деструктор

Сигнатура: ~IVectorization().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод выполнения векторизации изображения

Сигнатура: virtual std::vector<Objects::Geometric::IObject*> execute(Objects::Visual::Image* image) = 0.

Входные и выходные параметры:

Входные параметры:

- image – изображение, подлежащее векторизации.

Выходные параметры представлены объектом типа `std::vector<Objects::Geometric::IObject*>`, являющимся набором объектов, полученных в результате скелетизации.

Описание: метод выполнения векторизации изображения.

5.3.28. Описание класса Vectorization

5.3.28.1. Назначение

Класс предназначен для реализации интерфейса `IVectorization`. Векторизация выполняется поиском пикселей объектов и оценкой их окрестности.

Алгоритм описан в разделе 3.6.1.

5.3.28.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Метод выполнения векторизации изображения

Сигнатура: `virtual std::vector<Objects::Geometric::IObject*> execute(Objects::Visual::Image* image) = 0.`

Входные и выходные параметры:

Входные параметры:

- image – изображение, подлежащее векторизации.

Выходные параметры представлены объектом типа `std::vector<Objects::Geometric::IObject*>`, являющимся набором объектов, полученных в результате скелетизации.

Описание: метод выполнения векторизации изображения.

5.3.29. Описание класса StrokesVectorization

5.3.29.1. Назначение

Класс предназначен для реализации интерфейса `IVectorization`. Векторизация выполняется штриховым методом.

Алгоритм описан в разделе 3.6.2.

5.3.29.2. Публичные методы

Публичные методы аналогичны описанным в разделе 5.3.28.2.

5.3.30. Описание класса VectorizationState

5.3.30.1. Назначение

Класс предназначен для реализации состояния векторизации, состоящего из начальной, предыдущей, текущей, соседних и следующих точек.

5.3.30.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: VectorizationState(ImageWithVisitedMarks* image).

Входные и выходные параметры:

Входные параметры:

- image – изображение, для которого задаётся состояние векторизации.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к значению предыдущей точки

Сигнатура: ImageWithVisitedMarks::Pixel* previous() const.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа ImageWithVisitedMarks::Pixel, представляющее собой предыдущую точку.

Описание: метод доступа к значению предыдущей точки.

Метод задания значения предыдущей точки

Сигнатура: void previous(ImageWithVisitedMarks::Pixel* value).

Входные и выходные параметры:

Входные параметры:

- value – новое значение предыдущей точки.

Выходные параметры отсутствуют.

Описание: метод задания значения предыдущей точки.

Метод доступа к значению текущей точки

Сигнатура: ImageWithVisitedMarks::Pixel* current() const.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа ImageWithVisitedMarks::Pixel, представляющее собой текущую точку.

Описание: метод доступа к значению текущей точки.

Метод задания значения текущей точки

Сигнатура: void current(ImageWithVisitedMarks::Pixel* value).

Входные и выходные параметры:

Входные параметры:

- value – новое значение текущей точки.

Выходные параметры отсутствуют.

Описание: метод задания значения текущей точки.

Метод доступа к значению стартовой точки

Сигнатура: ImageWithVisitedMarks::Pixel* start() const.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа ImageWithVisitedMarks::Pixel, представляющее собой стартовую точку.

Описание: метод доступа к значению стартовой точки.

Метод задания значения стартовой точки

Сигнатура: void start(ImageWithVisitedMarks::Pixel* value).

Входные и выходные параметры:

Входные параметры:

- value – новое значение стартовой точки.

Выходные параметры отсутствуют.

Описание: метод задания значения стартовой точки.

Метод доступа к значениям соседних точек точки

Сигнатура: std::vector<ImageWithVisitedMarks::Pixel*> neighbours() const.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа std::vector<ImageWithVisitedMarks::Pixel*>, представляющее собой множество соседних точек.

Описание: метод доступа к значениям соседних точек.

Метод задания значения текущей точки

Сигнатура: void neighbours(std::vector<ImageWithVisitedMarks::Pixel*> value).

Входные и выходные параметры:

Входные параметры:

- value – новые значения соседних точек.

Выходные параметры отсутствуют.

Описание: метод задания значений соседних точек.

Метод доступа к значениям следующих точек точки

Сигнатура: std::vector<ImageWithVisitedMarks::Pixel*> next().

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа std::vector<ImageWithVisitedMarks::Pixel*>, представляющее собой множество следующих точек.

Описание: метод доступа к значениям следующих точек.

Метод сброса состояния векторизации

Сигнатура: void reset().

Входные и выходные параметры: отсутствуют.

Описание: выполняет сброс состояния векторизации обнулением начальной, предыдущей, текущей, соседних и следующих точек.

5.3.31. Описание класса IntersectionPoint

5.3.31.1. Назначение

Класс предназначен для реализации точки пересечения нескольких объектов.

5.3.31.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: IntersectionPoint(IPoint *point).

Входные и выходные параметры:

Входные параметры:

- point – объект с координатами точки пересечения нескольких объектов.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор

Сигнатура: IntersectionPoint(IPoint *point).

Входные и выходные параметры:

Входные параметры:

- point – объект с координатами точки пересечения нескольких объектов.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к координате X

Сигнатура: int x() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является X-координата точки.

Описание: возвращает значение X-координаты точки.

Метод доступа к координате Y

Сигнатура: int y() const override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является Y-координата точки.

Описание: возвращает значение Y-координаты точки.

Метод преобразования объекта в строковое представление

Сигнатура: std::string to_string() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа std::string содержит строковое представление объекта.

Описание: метод преобразования объекта в строковое представление по его содержимому.

Метод доступа к пересекающимся объектам

Сигнатура: std::vector<IObject*> intersected_objects() const.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходным параметром является значение типа std::vector<IObject*>, которое содержит пересекающиеся объекты.

Описание: возвращает пересекающиеся объекты.

Метод добавления нового объекта

Сигнатура: void add_object(IObject *object).

Входные и выходные параметры:

Входные параметры:

- object – новый объекты, пересекающийся с другими объектами в заданной точке.

Выходные параметры отсутствуют.

Описание: метод добавления нового объекта.

5.3.32. Описание класса IntersectionPointsSet

5.3.32.1. Назначение

Класс предназначен для реализации множества точек пересечения нескольких объектов.

5.3.32.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: IntersectionPointsSet().

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Деструктор

Сигнатура: ~IntersectionPointsSet().

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к точке пересечения по заданным координатам

Сигнатура: IntersectionPoint* get(int row, int col).

Входные и выходные параметры:

Входные параметры:

- row, col – координаты точки пересечения.

Выходной параметр является значением типа IntersectionPoint, представляющим точку пересечения.

Описание: возвращает точку пересечения по заданным координатам.

Метод проверки существования точки пересечения по заданным координатам

Сигнатура: bool is_exist(int row, int col).

Входные и выходные параметры:

Входные параметры:

- row, col – координаты точки пересечения.

Выходной параметр является значением типа `bool`, равным `true` (в случае существования) или `false` (в случае отсутствия точки).

Описание: выполняет проверку существования точки пересечения по заданным координатам.

5.3.33. Описание класса `ImageWithVisitedMarks`

5.3.33.1. Назначение

Класс предназначен для реализации изображения с возможностью помечать пиксели метками «просмотрено», «не просмотрено», «исключено».

5.3.33.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор

Сигнатура: `ImageWithVisitedMarks()`.

Входные и выходные параметры: отсутствуют.

Описание: метод инициализации объекта.

Конструктор с сохранением пользовательского изображения

Сигнатура: `ImageWithVisitedMarks(Objects::Visual::Image* image)`.

Входные и выходные параметры:

Входные параметры:

- `image` – изображение, экземпляр `Image`.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Деструктор

Сигнатура: `~ ImageWithVisitedMarks ()`.

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Деструктор

Сигнатура: `~ ImageWithVisitedMarks ()`.

Входные и выходные параметры: отсутствуют.

Описание: метод деинициализации объекта.

Метод доступа к количеству строк изображения

Сигнатура: int rows() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством строк изображения.

Описание: возвращает количество строк изображения.

Метод доступа к количеству столбцов изображения

Сигнатура: int cols() override.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа int является количеством столбцов изображения.

Описание: возвращает количество столбцов изображения.

Метод доступа к элементу изображения

Сигнатура: Pixel* get(int row, int col) override.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца.

Выходной параметр типа Pixel является пикселем в заданной ячейке.

Описание: возвращает цвет в заданной координатами ячейке.

Метод изменения элемента изображения

Сигнатура: void set(int row, int col, Pixel* value) override.

Входные и выходные параметры:

Входные параметры:

- row – номер строки;
- col – номер столбца;
- value – объект пикселя, который будет записан.

Выходные параметры отсутствуют.

Описание: изменяет пиксель в заданной координатами ячейке.

Метод преобразования всех исключённых пикселей в не посещённые

Сигнатура: void convert_all_excluded_to_unvisited().

Входные и выходные параметры: отсутствуют.

Описание: преобразует все исключённые пиксели в не посещённые.

Метод нахождения первого не посещённого пикселя

Сигнатура: Pixel* get_first_unvisited().

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа Pixel является первым не посещённым пикселем.

Описание: возвращает первый не посещённый пиксель.

Метод получения соседних элементов заданной ячейки

Сигнатура: `std::vector<Pixel*> get_neighbours(Pixel* central).`

Входные и выходные параметры:

Входные параметры:

- `central` – центральный пиксель.

Выходной параметр типа `std::vector<Pixel*>` является множеством соседних элементов

Описание: возвращает соседние элементы заданной ячейки изображения.

Метод получения соседних элементов заданной ячейки, учитывая исключённые

Сигнатура: `std::vector<Pixel*> get_neighbours_with_excluded(Pixel* central).`

Входные и выходные параметры:

Входные параметры:

- `central` – центральный пиксель.

Выходной параметр типа `std::vector<Pixel*>` является множеством соседних элементов

Описание: возвращает соседние элементы заданной ячейки изображения, учитывая исключённые точки.

Метод определения наличия не посещённых пикселей

Сигнатура: `bool has_unvisited_pixels() const.`

Входные и выходные параметры: Входные параметры отсутствуют.

Выходной параметр типа `bool` является значением `true`, если не посещённые точки присутствуют, иначе – `false`.

Описание: определяет наличие не посещённых пикселей.

Метод нахождения горизонтальной или вертикальной соседней точки

Сигнатура: `static Pixel *vertical_or_horizontal_neighbour(Pixel *central, std::vector<Pixel*> neighbours).`

Входные и выходные параметры: Входные параметры:

- `central` – центральна точка;
- `neighbours` – множество соседних точек.

Выходной параметр типа `Pixel` является первой попавшейся вертикальной или горизонтальной соседней точкой.

Описание: возвращает горизонтальную или вертикальную соседнюю точку.

Метод проверяет лежат ли точки на одной прямой

Сигнатура: `static bool is_aligned(Pixel *a, Pixel *b, Pixel *c).`

Входные и выходные параметры: Входные параметры:

- `a, b, c` – точки.

Выходной параметр типа `bool` является значением `true`, если точки лежат на одной прямой, иначе - `false`.

Описание: метод проверяет лежат ли точки на одной прямой.

5.3.34. Описание класса `Path`

5.3.34.1. Назначение

Класс предназначен для упрощения работы из программы с путями к файлам.

5.3.34.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Конструктор с полным путём к файлу

Сигнатура: `Path(std::string full_path)`.

Входные и выходные параметры:

Входные параметры:

- `full_path` – полный путь к файлу.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Конструктор с заданием пользовательских параметров

Сигнатура: `Path(std::string path_to_file, std::string file_name, std::string file_extention)`.

Входные и выходные параметры:

Входные параметры:

- `path_to_file` – путь к файлу;
- `file_name` – имя файла;
- `file_extention` – расширение файла.

Выходные параметры отсутствуют.

Описание: метод инициализации объекта.

Метод доступа к пути к файлу

Сигнатура: `std::string path_to_file() const`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходные параметры являются объектом типа `std::string` и представляют собой путь к файлу.

Описание: возвращает путь к файлу.

Метод доступа к имени файла

Сигнатура: `std::string file_name() const`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходные параметры являются объектом типа `std::string` и представляют собой имя файла.

Описание: возвращает имя файла.

Метод доступа к расширению файла

Сигнатура: `std::string file_extention() const`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходные параметры являются объектом типа `std::string` и представляют собой расширение файла.

Описание: возвращает расширение файла.

Метод доступа к полному пути к файлу

Сигнатура: `std::string full_path() const`.

Входные и выходные параметры: Входные параметры отсутствуют.

Выходные параметры являются объектом типа `std::string` и представляют собой полный путь к файлу.

Описание: возвращает полный путь к файлу.

5.3.35. Описание класса Reader

5.3.35.1. Назначение

Класс предназначен для чтения растровых файлов форматов BMP, JPG, PNG, TIFF.

5.3.35.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Метод чтения растрового изображения

Сигнатура: `static Objects::Visual::Image* read(Path path)`.

Входные и выходные параметры:

Входные параметры:

- `path` – путь к файлу.

Выходной параметр является объектом типа `Image` и представляет собой считанное изображение.

Описание: считывает растровое изображение.

5.3.36. Описание класса `Writer`

5.3.36.1. Назначение

Класс предназначен для записи растровых и векторных данных.

5.3.36.2. Приватные методы

Методы, описанные в этом разделе, являются приватными, так как не входят в интерфейс работы с рассматриваемым классом и не предполагают непосредственной работы пользователя с ними.

Метод записи растровых данных

Сигнатура: `static void write_raster(std::string path, Objects::Visual::Image* image).`

Входные и выходные параметры:

Входные параметры:

- `path` – путь к файлу;
- `image` – изображение.

Выходные параметры отсутствуют.

Описание: записывает растровые данные.

Метод записи SVG-файлов

Сигнатура: `static void write_svg(std::string path, std::vector<Objects::Geometric::IObject*> objects, Objects::Visual::Image* image).`

Входные и выходные параметры:

Входные параметры:

- `path` – путь к файлу;
- `objects` – объекты, полученные в результате векторизации изображения;
- `image` – изображение.

Выходные параметры отсутствуют.

Описание: записывает SVG-файлы.

Метод записи XML-файлов

Сигнатура: static void write_xml(std::string path, std::vector<Objects::Geometric::IObject*> objects).

Входные и выходные параметры:

Входные параметры:

- path – путь к файлу;
- objects – объекты, полученные в результате векторизации изображения.

Выходные параметры отсутствуют.

Описание: записывает XML-файлы.

5.3.36.3. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Метод записи изображения

Сигнатура: static void write(Path path, Objects::Visual::Image* image).

Входные и выходные параметры:

Входные параметры:

- path – путь к файлу;
- image – изображение.

Выходные параметры отсутствуют

Описание: записывает изображение.

Метод записи изображения с полученными объектами

Сигнатура: static void write(Path path, std::vector<Objects::Geometric::IObject*> objects, Objects::Visual::Image* image).

Входные и выходные параметры:

Входные параметры:

- path – путь к файлу;
- objects – объекты, полученные в результате векторизации изображения;
- image – изображение.

Выходные параметры отсутствуют.

Описание: записывает изображение с векторными объектами.

5.3.37. Описание класса Functions

5.3.37.1. Назначение

Класс вспомогательных функций.

5.3.37.2. Публичные методы

Методы, перечисленные в этом разделе, объявлены публичными, так как составляют интерфейс для работы пользователя с рассматриваемым классом.

Метод получения яркостной характеристики RGB-цвета

Сигнатура: static unsigned char rgb_to_brightness(unsigned char red, unsigned char green, unsigned char blue).

Входные и выходные параметры:

Входные параметры:

- red, green, blue – значения каналов RGB-цвета.

Выходные параметры отсутствуют.

Описание: возвращает яркостную характеристику RGB-цвета.

Метод конвертации объекта QImage в объект cv::Mat

Сигнатура: static void QImage_to_mat(Objects::Visual::QImage* image, cv::Mat* mat).

Входные и выходные параметры:

Входные параметры:

- image - изображение;
- mat – матрица.

Выходные параметры отсутствуют.

Описание: выполняет конвертацию объекта QImage в объект cv::Mat.

Метод конвертации объекта cv::Mat в объект QImage

Сигнатура: static void mat_to_QImage(cv::Mat* mat, Objects::Visual::QImage* image).

Входные и выходные параметры:

Входные параметры:

- mat – матрица;
- image – изображение.

Выходные параметры отсутствуют.

Описание: выполняет конвертацию объекта cv::Mat в объект QImage.

Метод определяет лежат ли точки на одной прямой

Сигнатура: static bool is_aligned(Objects::Geometric::IPoint* a, Objects::Geometric::IPoint* b, Objects::Geometric::IPoint* c).

Входные и выходные параметры:

Входные параметры:

– a, b, c – точки.

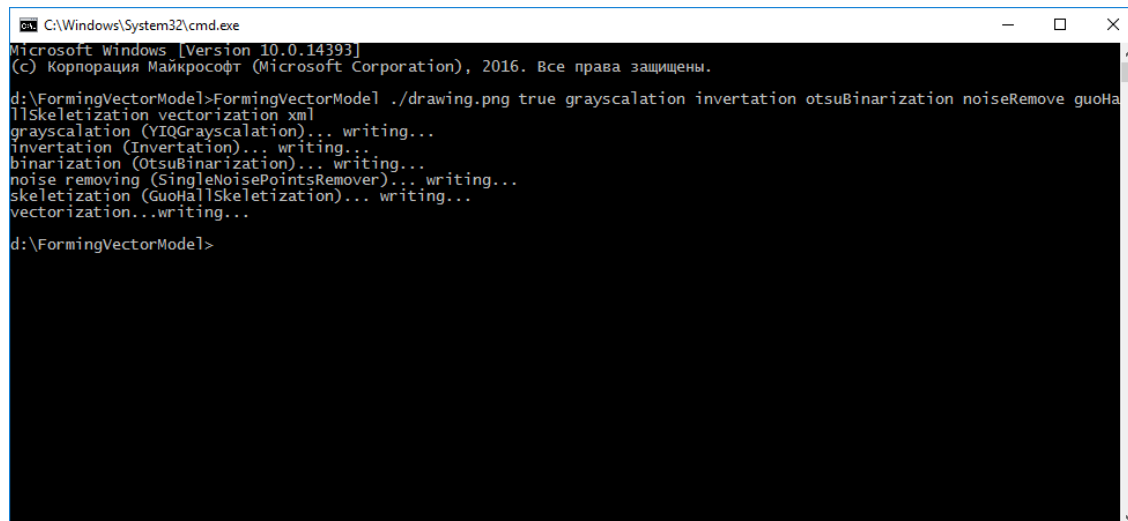
Выходные параметры являются значением типа bool и равны true, если точки лежат на одной прямой, иначе – false.

Описание: определяет лежат ли точки на одной прямой.

6. Руководство пользователя к разработанному программному продукту

6.1. Интерфейс

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



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) Корпорация Майкрософт (Microsoft Corporation), 2016. Все права защищены.

d:\FormingVectorModel>FormingVectorModel ./drawing.png true grayscaleation invertation otsuBinarization noiseRemove guoHallSkeletization vectorization xml
grayscaleation (YIQGrayscaleation)... writing...
invertation (Invertation)... writing...
binarization (OtsuBinarization)... writing...
noise removing (SingleNoisePointsRemover)... writing...
skeletization (GuoHallSkeletization)... writing...
vectorization...writing...

d:\FormingVectorModel>
```

Рисунок 20. Интерфейс системы

6.2. Выполнение программы

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

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. «grayscaleation» – 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;
9. «xml» - команда для сохранения векторной модели в формате XML; «svg» - команда для сохранения векторной модели в формате SVG.

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.

7. Результаты работы программного продукта

Был задан сканированный растровый чертёж (рис. 21) со следующими характеристиками:

- Геометрические размеры: 1892x1421;
- DPI: 72;
- Глубина цвета: 24bit;
- Формат: PNG.

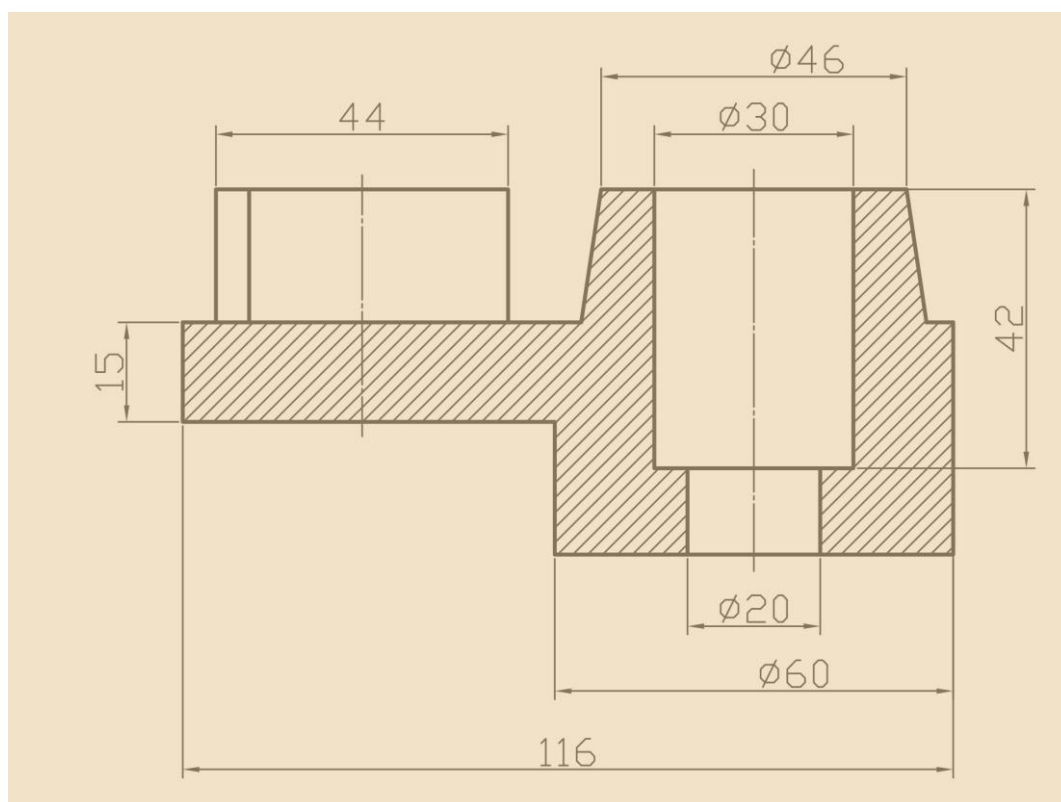


Рисунок 21. Сканированный растровый чертёж

Был выполнен запуск разработанной программы в командной строке Windows со следующими параметрами: `FormingVectorModel ./drawing.png true grayscale inversion otsuBinarization noiseRemove guoHallSkeletization vectorization xml`.

Время работы программы составило 1 минуту 13 секунд.

Были получены следующие выходные файлы:

- Изображение в оттенках серого;
- Инвертированное изображение;
- Бинаризованное изображение;
- Изображение с удалённым «шумом»;
- Скелетизированное изображение;

– Векторная модель в формате XML.

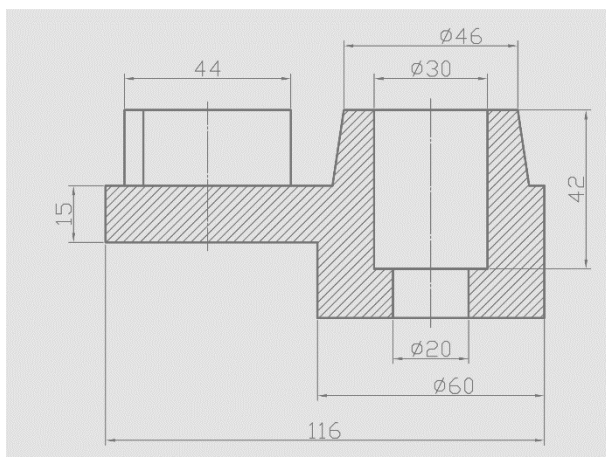


Рисунок 22. Изображение в оттенках серого

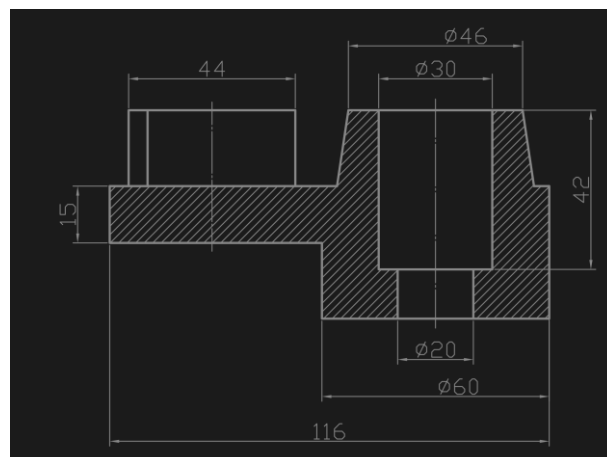


Рисунок 23. Инвертированное изображение

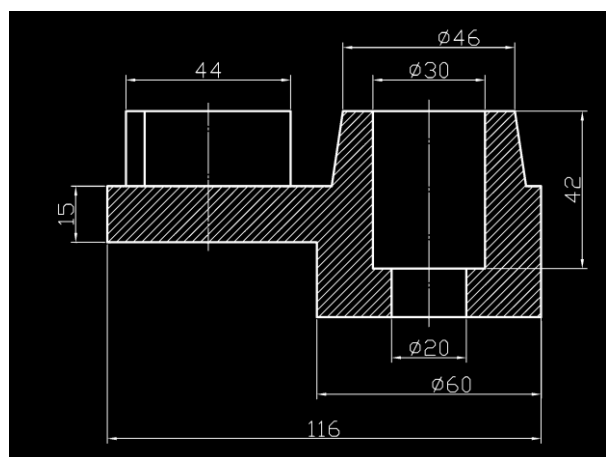


Рисунок 24. Бинаризованное изображение

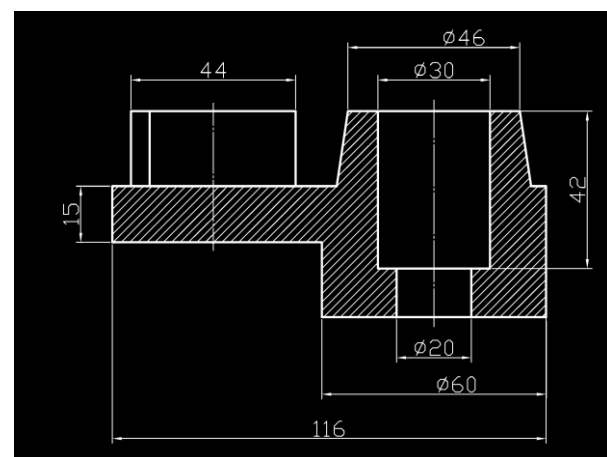


Рисунок 25. Изображение с удалённым «шумом»

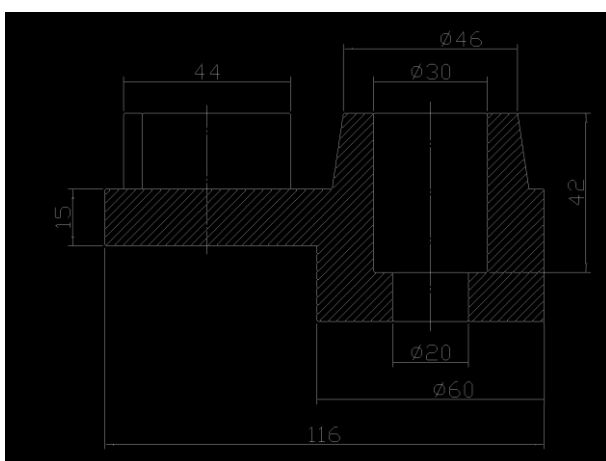


Рисунок 26. Скелетизированное изображение

В векторной модели находилось 2164 объекта и 2257 точек пересечения объектов.

Заключение

По результатам выполнения выпускной квалификационной работы сформулируем основные выводы:

- в результате критического анализа литературных данных установлено, что задача формирования векторной и топологической модели 2D-машиностроительных чертежей 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 и топологической моделей 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. Ключев // 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 отиска печати в изображении документа / И.С. Сай // Вестник ТОГУ «Электроника и информационно-измерительные приборы» - 2009 - №4
3. Москаленко, С.В. Разработка методов и алгоритмов векторизации растровых 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 связанных 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. Raster 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.