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: Название кафедры**

Направление подготовки: «Название направления»

Магистерская программа: «Название магистерской программы»

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

**Тема:**

**«Формирование топологической векторной модели машиностроительного чертежа»**

Допущена к защите Выполнил:

Заведующий кафедрой: студент группы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая. степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись подпись

Научный руководитель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание, ф.и.о.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Рецензент:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ученая степень, ученое звание,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

сь

Нижний Новгород

2017

**Содержание**

[Введение 3](#_Toc483431061)

[Алгоритмическое обеспечение 4](#_Toc483431062)

[Преобразование изображения в оттенки серого 4](#_Toc483431063)

[Инвертация изображения 5](#_Toc483431064)

[Бинаризация изображения 5](#_Toc483431065)

[Бинаризация по пороговому значению 6](#_Toc483431066)

[Бинаризация с применением метода Оцу 6](#_Toc483431067)

[Удаление «шумов» на изображении 7](#_Toc483431068)

[Скелетизация 7](#_Toc483431069)

[Алгоритм Зонга-Суня 8](#_Toc483431070)

[Алгоритм Гуо-Хелла 9](#_Toc483431071)

[Литература 11](#_Toc483431072)

# Введение

# Алгоритмическое обеспечение

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

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

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

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

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

Далее приведённые выше шаги будут описаны.

## Преобразование изображения в оттенки серого

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

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

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

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

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

Для перевода пространства RGB в YIQ используется следующая формула:

Из формулы (1) необходимо использовать уравнение

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.

## 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:

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

## Бинаризация изображения

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

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

Были рассмотрены два алгоритма бинаризации:

* Бинаризация по пороговому значению;
* Бинаризация методом Отсу.

### 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:

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.

### Бинаризация с применением метода Оцу

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

Общий алгоритм метода Оцу состоит из следующих этапов:

1. Вычисление гистограммы изображения;
2. Для каждого из значений гистограммы:
   1. Вычисление ;
   2. Если больше имеющегося, то сохранение и порога t.

вычисляется по формуле:

где

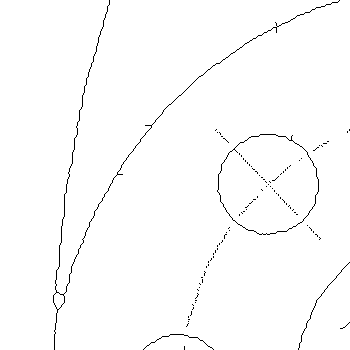
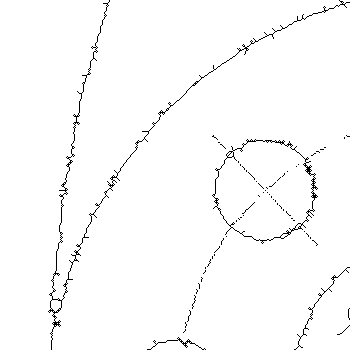
что представляет собой сумму различий двух классов. Классы являются множествами пикселей изображения, разделённых относительно порогового значения t.

Веса – вероятности классов, – дисперсии классов.

## Удаление «шумов» на изображении

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

При последующей векторизации такие пиксели могут вносить существенные искажения.



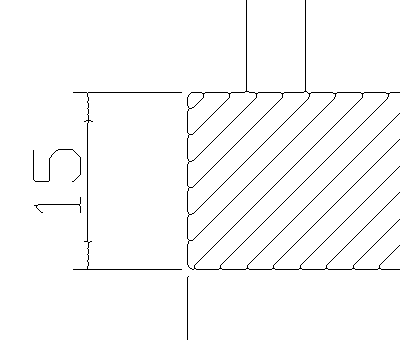
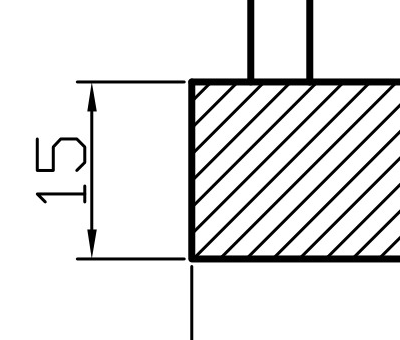
Для предотвращения появления таких эффектов производится удаление пикселей, являющихся «шумом», следующим образом:

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

## Скелетизация

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

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



### Алгоритм Зонга-Суня (Zhang-Suen)

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

Вводится матрица, вида:

|  |  |  |
| --- | --- | --- |
| P8 | P2 | P3 |
| P8 | P1 | P4 |
| P7 | P6 | P5 |

Матрица 1

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

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

где – количество найденных переходов от 0 к 1 в последовательности , , , , , , , , .

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

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.

### Алгоритм Гуо-Хелла (Guo-Hall)

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

Пусть заданы следующие уравнения:

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

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

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.

|  |  |
| --- | --- |
| C:\Users\alex\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ZhangSuenSkeletization_sample_big.png  Пример выполнения алгоритма Зонга-Суня. | C:\Users\alex\AppData\Local\Microsoft\Windows\INetCache\Content.Word\GuoHallSkeletization_sample_big.png  Пример выполнения алгоритма Гуо-Хелла. |

## Векторизация

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

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

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

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

### Алгоритм векторизации

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

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

### 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:
   1. Выполнить процедуру итерации.

Где процедура итерации имеет следующий вид:

1. Если состояние векторизации не инициализировано:
   1. Очистить множество особых точек;
   2. Преобразовать все исключённые точки изображения в непосещённые;
   3. Если на изображении есть непосещённые точки:
      1. Сохранить непосещённую точку в состояние векторизации, как текущую;
   4. Иначе:
      1. Завершить алгоритм;
2. Если у текущей точки состояния векторизации отсутствуют соседние точки:
   1. Пометить текущую точку векторизации, как посещённую;
   2. Если предыдущая точка состояния векторизации инициализирована:
      1. Добавить текущую точку состояния векторизации во множество особых точек;
      2. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
      3. Очистить множество особых точек;
      4. Присвоить значение текущей точки стартовой точке состояния векторизации;
      5. Завершить итерацию;
3. Если предыдущая точка состояния векторизации не инициализирована:
   1. Добавить текущую точку состояния векторизации во множество особых точек;
   2. Выбрать точки для продолжения векторизации;
   3. Если среди выбранных точек для продолжения векторизации имеется горизонтальная или вертикальная соседняя точка:
      1. Запомнить её, как следующую точку;
   4. Иначе:
      1. Запомнить первую точку из точек для продолжения векторизации, как следующую точку;
   5. Пометить остальные точки для продолжения векторизации, как исключённые из векторизации;
   6. Если точки были помечены, как исключённые:
      1. Пометить текущую точку векторизации, как исключённую;
   7. Иначе:
      1. Пометить текущую точку векторизации, как посещённую;
   8. Присвоить значение текущей точки стартовой точке состояния векторизации;
4. Иначе:
   1. Выбрать точки для продолжения векторизации;
   2. Если таковых не имеется:
      1. Добавить текущую точку состояния векторизации во множество особых точек;
      2. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
      3. Завершить итерацию;
   3. Если среди точек для продолжения векторизации найдена точка, которая лежит на одной прямой с предыдущей и текущей точками состояния векторизации:
      1. Запомнить её, как следующую точку;
   4. Иначе:
      1. Запомнить первую точку из точек для продолжения векторизации, как следующую точку;
   5. Пометить остальные точки для продолжения векторизации, как исключённые из векторизации;
   6. Если точки были помечены, как исключённые:
      1. Пометить текущую точку векторизации, как исключённую;
   7. Иначе:
      1. Пометить текущую точку векторизации, как посещённую;
   8. Если текущая точка состояния векторизации была исключена из векторизации или у текущей точки состояния векторизации имеются соседние точки в горизонтальных направлениях:
      1. Добавить текущую точку состояния векторизации во множество особых точек;
   9. Если следующая точка состояния векторизации не находится на одной прямой с предыдущей и текущей точками состояния векторизации:
      1. Добавить текущую точку состояния векторизации во множество особых точек;
      2. Сохранить новый векторный объект (начальная точка – стартовая точка состояния векторизации, конечная точка – текущая точка состояния векторизации, множество особых точек);
      3. Очистить множество особых точек;
      4. 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.

# Литература

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

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

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

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