Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

Факультет Информационных технологий и радиоэлектроники

Кафедра Информационных систем

УТВЕРЖДАЮ

Зав. кафедрой

Д.Е. Андрианов

(подпись)

"   " 2022г.

БАКАЛАВРСКАЯ РАБОТА

Тема Разработка и исследование алгоритма для комплексирования векторных данных с сохранением топологий

МИВУ.09.03.02-00.000 БР

Руководитель

Еремеев С. В.

(фамилия, инициалы)

(подпись) (дата)

Студент ИС-118

(группа)

Кашин Н. П.

(фамилия, инициалы)

(подпись) (дата)

Муром 2022

В данной бакалаврской работе выполнены реализация и иследование алгоритма комплексирования векторных данных с сохранением топологий

Табл. 1. Ил. 55. Библ. 19.

Table. 1. Il.55. Bibl. 19.

Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

*ПОЯСНИТЕЛЬНАЯ*

*ЗАПИСКА*

Тема Разработка и исследование алгоритма для комплексирования векторных данных с сохранением топологий

МИВУ.09.03.02 - 00.000 ПЗ

Муром 2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc104883674)

[1 Обзор существующих аналогов распознавания аномального зрачка глаза человека 9](#_Toc104883675)

[2 Выбор технологии разработки приложения 28](#_Toc104883676)

[3 Реализация алгоритма выделения аномального зрачка глаза человека 29](#_Toc104883677)

[4 Исследование алгоритма выделения аномального зрачка глаза человека 41](#_Toc104883678)

[ЗАКЛЮЧЕНИЕ 63](#_Toc104883679)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 64](#_Toc104883680)

[ПРИЛОЖЕНИЕ А 66](#_Toc104883681)

### ВВЕДЕНИЕ

# 1 Анализ технического задания

В данной бакалаврской работе была поставлена задача разработать и исследовать алгоритм комплексирования векторных данных.

Исходными данными для работоспособности алгоритма являются две карты одной и той же местности, но сделанные в разный период времени и/или с разным масштабом.

Под комплексированием понимается объединение, сочетание, создание комплекса или комплексов.

Алгоритм, разработанный в ходе исследовательской работы, будет объединять некоторый набор векторных данных на одной карте с другой картой. То есть копировать набор данных с одной карты на другую. После комплексирования векторных данных алгоритм проведёт анализ топологических отношений.

В качестве среды разработки будет использоваться географическая информационная система QGIS.

1.1 Географическая информационная система QGIS

QGIS — это бесплатная кроссплатформенная географическая информационная система (ГИС) с открытым исходным кодом, которая поддерживает просмотр, редактирование, печать и анализ геопространственных данных.

QGIS позволяет пользователям анализировать и редактировать пространственную информацию, а также составлять и экспортировать графические карты. QGIS поддерживает растровые, векторные и сетчатые слои. Векторные данные хранятся в виде точечных, линейных или полигональных объектов. Поддерживаются несколько форматов растровых изображений.

QGIS поддерживает шейп-файлы, персональные базы геоданных, dxf, MapInfo, PostGIS и другие стандартные отраслевые форматы. Веб-службы, в том числе служба веб-карт и служба веб-объектов, также поддерживаются, позволяя использовать данные из внешних источников.

QGIS интегрируется с другими ГИС-пакетами с открытым исходным кодом, включая PostGIS, GRASS GIS и MapServer. Плагины, написанные на Python или C++, расширяют возможности QGIS. Плагины могут выполнять геокодирование с помощью Google Geocoding API, выполнять функции геообработки, аналогичные функциям стандартных инструментов ArcGIS, и взаимодействовать с базами данных PostgreSQL/PostGIS, SpatiaLite и MySQL.

QGIS может отображать несколько слоев, содержащих различные источники или их изображения.

Как бесплатное программное приложение под GNU GPLv2, QGIS можно свободно модифицировать для выполнения других или более специализированных задач. Двумя примерами являются приложения QGIS Browser и QGIS Server, которые используют один и тот же код для доступа к данным и визуализации, но имеют разные внешние интерфейсы.

Для реализации алгоритма будет использоваться язык программирования PyQGIS. Данный язык представляет собой язык программирования Python c доступом к API функциям QGIS. Он позволяет запускать созданный скрипт в консоли системы QGIS, что ускоряет и упрощает процесс реализации алгоритма.

# 2 Первый этап разработки.

На первом этапе разработки алгоритма был создан файл barycentric\_coor.py представляющий собой скрипт, написанный на языке PyQGIS.

2.1 Скрипт barycentric\_coor.py

2.1.1 Словесное описание алгоритма.

Алгоритм основывается на принципе вычисления барицентрических координат.

Рассмотрим частный случай барицентрических координат с использованием треугольников. Допустим, имеется определенная точка *P* с координатами *x* и *y.* А также два треугольника с вершинами и соответственно. Треугольники разные по размеру, но вершины треугольников соответствуют друг другу. Точка *P* расположена внутри первого треугольника. Необходимо перенести точку из одного треугольника в другой с учетом её расположения относительно вершин и сторон исходного треугольника. Алгоритм в данном случает будет следующим:

1. Найдём площадь S треугольника с точкой P
2. Проведем от точки P линий к вершинам треугольника.
3. Найдем площади получившихся треугольников.
4. Произведем вычисления по формулам 1, 2 и 3

(1)

(2)

(3)

где *u, v, w,* - барицентрические координаты, причем *u + v + w* = 1.

1. Вычислим расположение точки по формуле 2

(4)

1. Готово. С помощью барицентрических координат точка была скопирована.

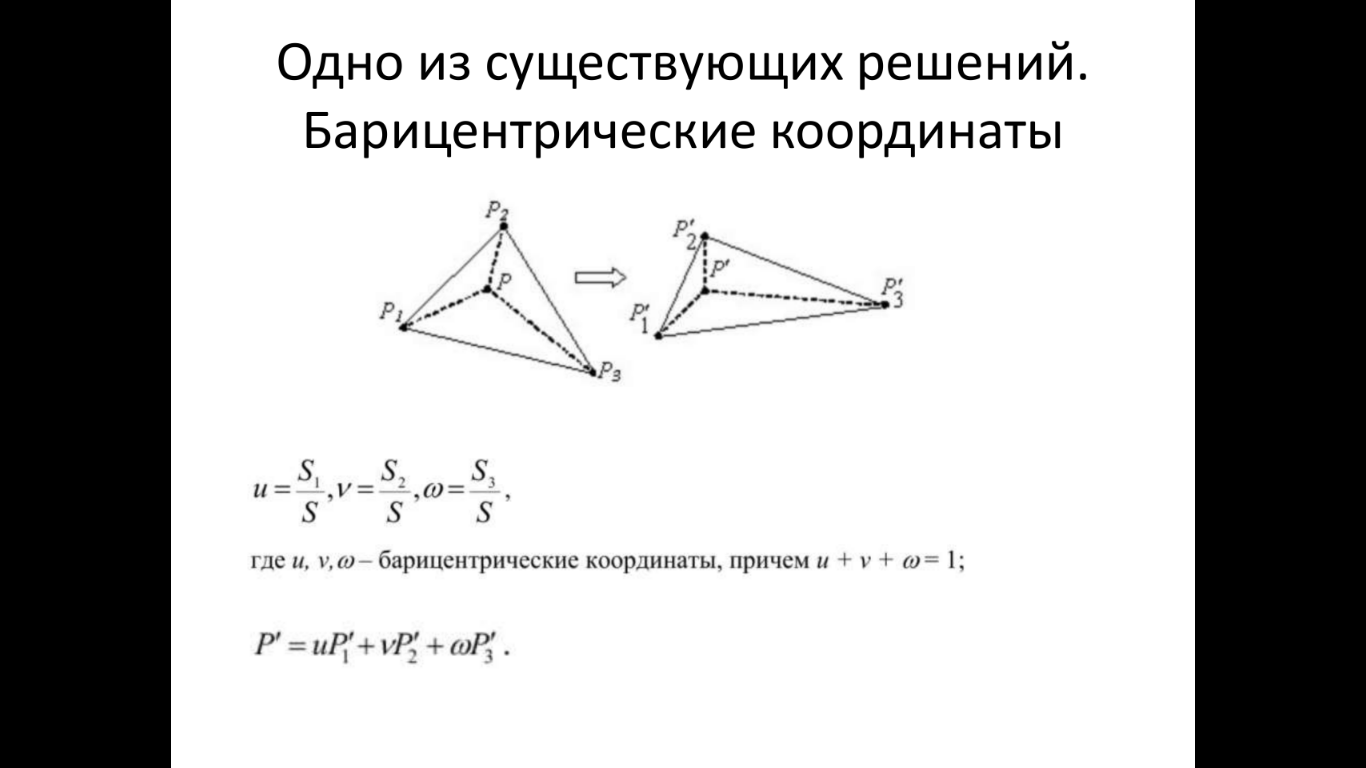


Рисунок 1 – Пример работы барицентрических координат на одном треугольнике

Следовательно, барицентрический координаты - три числа, в сумме равные единице, определяющие положение точки в треугольнике, равные массам, которые следует поместить в вершинах треугольника так, чтобы определяемая точка сделалась центром тяжести этих масс.

Алгоритм так же должен работать на больших объемах данных с большим количеством точек, а для этого требуется больше одного треугольника.

Пример работы барицентрических координат на больших данных представлен на рисунках 2-3. На рисунке 2 представлены две исходные карты с базовыми точками для построения треугольников. На рисунке 3 представлены эти же две карты после преобразования.

Для автоматического построения треугольников на базовых точках была использована триангуляция Делоне. На множестве точек на плоскости задана триангуляция, если некоторые пары точек соединены ребром, любая конечная грань в получившемся графе образует треугольник, ребра не пересекаются, и граф максимален по количеству ребер. Триангуляцией Делоне называется такая триангуляция, в которой для любого треугольника верно, что внутри описанной около него окружности не находится точек из исходного множества.



Рисунок 2 – Исходные данные



Рисунок 3 – После преобразования

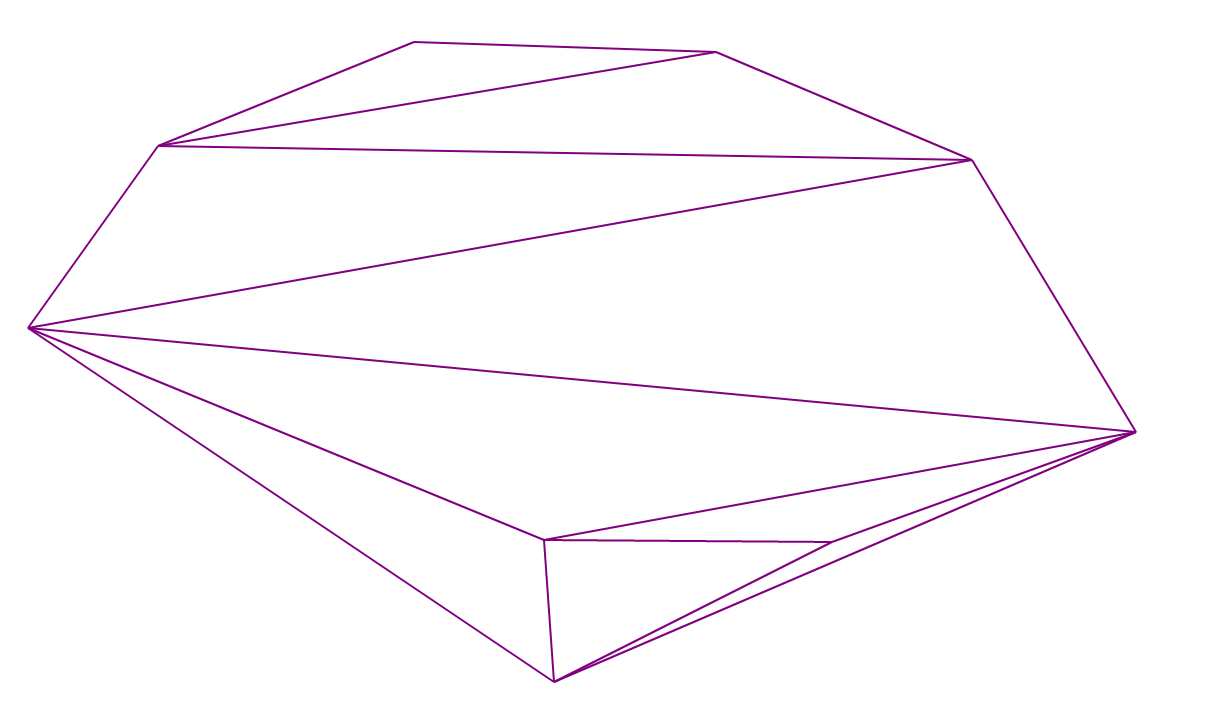


Рисунок 4 – Триангуляция

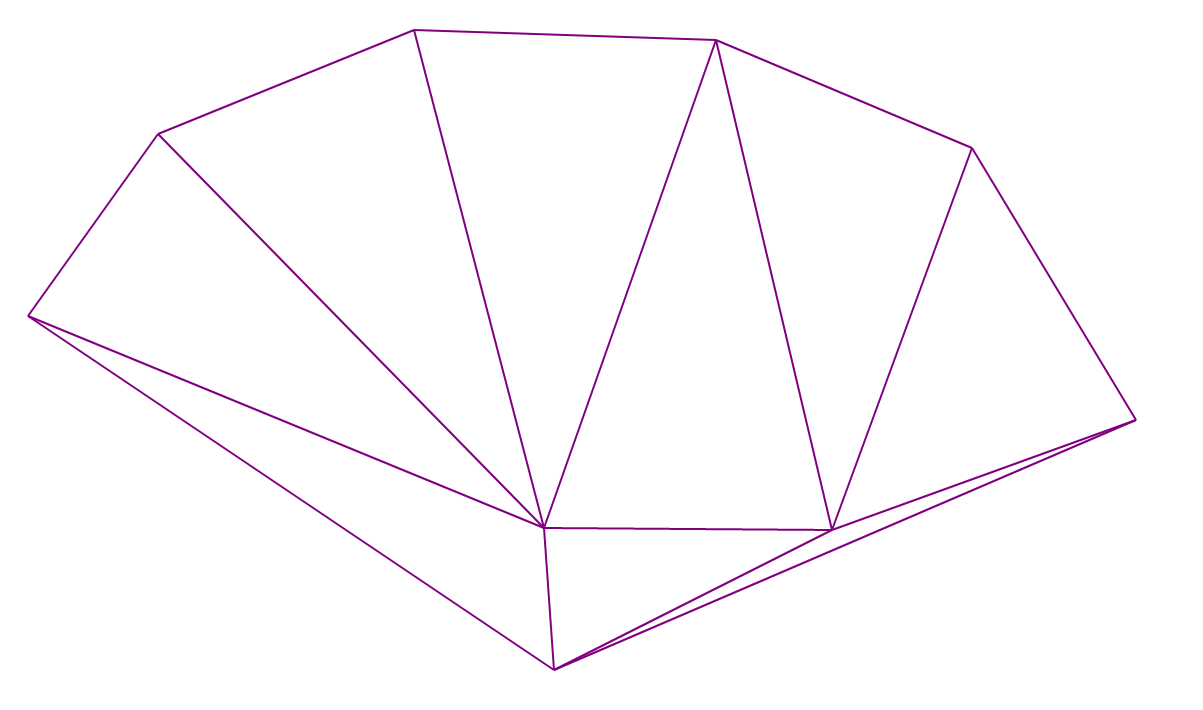


Рисунок 5 – Триангуляция Делоне

Алгоритм триангуляции Делоне основан на стандартной для многих алгоритмов методике сведения сложной задачи к более простым, в которых решение очевидно. Сам алгоритм состоит из 2 шагов.

Шаг №1. Разбиение исходного множества на более мелкие множества. Для этого мы проводим вертикальные или горизонтальные прямые в середине множества и уже относительно этих прямых разделяем точки на две части. Далее для каждой группы точек рекурсивно запускаем процесс деления.

Шаг №2. Объединение оптимальных триангуляций. Сначала находятся две пары точек, отрезки которых образуют в совокупности с построенными триангуляциями выпуклую фигуру. Они соединяются отрезками, и один из полученных отрезков выбирается как начало для последующего обхода. Обход заключается в следующем: на этом отрезке мы как будто «надуваем пузырь» внутрь до первой точки, которую достигнет раздувающаяся окружность «пузыря». С найденной точкой соединяется та точка отрезка, которая не была с ней соединена. Полученный отрезок проверяется на пересечение с уже существующими отрезками триангуляции, и в случае пересечения они удаляются из триангуляции. После этого новый отрезок принимается за начало для нового «пузыря». Цикл повторяется до тех пор, пока начало не совпадёт со вторым отрезком выпуклой оболочки.

Для реализации алгоритма был создан скрипт barycentric\_coor.py, написанный на языке программирования Python и использующий возможности QGIS Python API.

2.1.2 Структура файла

Скрипт состоит из класса Moved, создания объекта данного класса и запуска методов преобразования.

В свою очередь класс Moved состоит из нескольких функций.

Функция \_\_init\_\_ - функция инициализаций, запускается при создании объекта класса Moved. В данной функции задаются константы, использующиеся в дальнейшем во всех функциях класса, а именно:

* vertex\_point\_in – имя слоя с базовыми точками первой карты;
* vertex\_point\_out – имя слоя с базовыми точками второй карты;
* move\_layer – имя слоя с объектами для переноса;
* type\_of\_geom – тип геометрии исходного слоя;
* coordinate\_system – используемая координатная система в текущем проекте.

Функция is\_in\_triangle выполняет проверку находится ли точка внутри треугольника. Для того чтобы определить лежит ли точка P внутри треугольника ABC необходимо вычислить 3 векторных произведения: ABxAP, BCxBP и CAxCP. Получив результаты по трем векторным произведениям, остается их проанализировать, чтобы понять лежит ли точка внутри треугольника. Если мы имеем и положительные и отрицательные результаты, точка лежит вне треугольника, если результаты только положительные или только отрицательные, точка - внутри.

Функция draw\_triangles выполняет отрисовку треугольников на карте. Происходит это следующим образом:

* получаем базовые точки двух карты;
* выполняем триангуляцию Делоне на базовых точках первой карты;
* на основе треугольников первой карты создаем треугольники второй карты;
* с помощью средств QGIS обрисовываем треугольники.

Функция barycentric\_out вычисляет барицентрические координаты точки относительно треугольника.

Функция barycentric\_in вычисляет координат точки относительно ее барицентрических координат.

Функция run основная функция класса, запускающая другие функции. А также проводит проверку входных данных, типов геометрии и отрисовывает комплексированный слой.

2.1.3 Порядок работы данного алгоритма.

Загружаем в QGIS исходные данные в виде векторных слоёв. На рисунках 6 и 7 представлены исходные карты. Так как исходные карты загружены в виде слоёв можно рассмотреть сходства и различия между ними. На рисунке 8 представлены сходства и различия исходных карт.

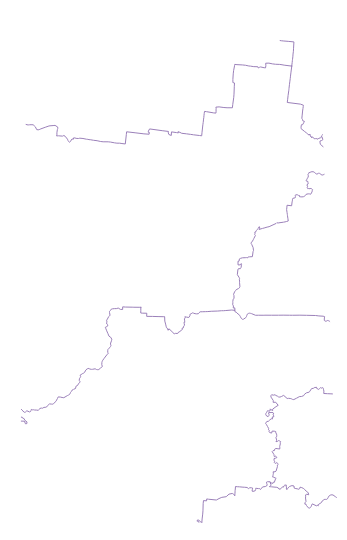


Рисунок 6 – Исходная карта 1

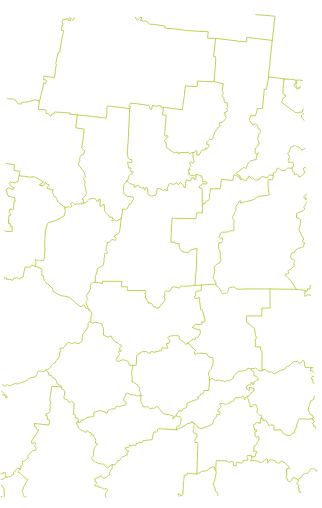


Рисунок 7 – Исходная карта 2

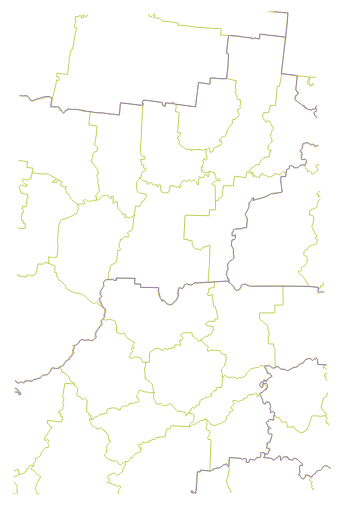


Рисунок 8 – Сходства и различия

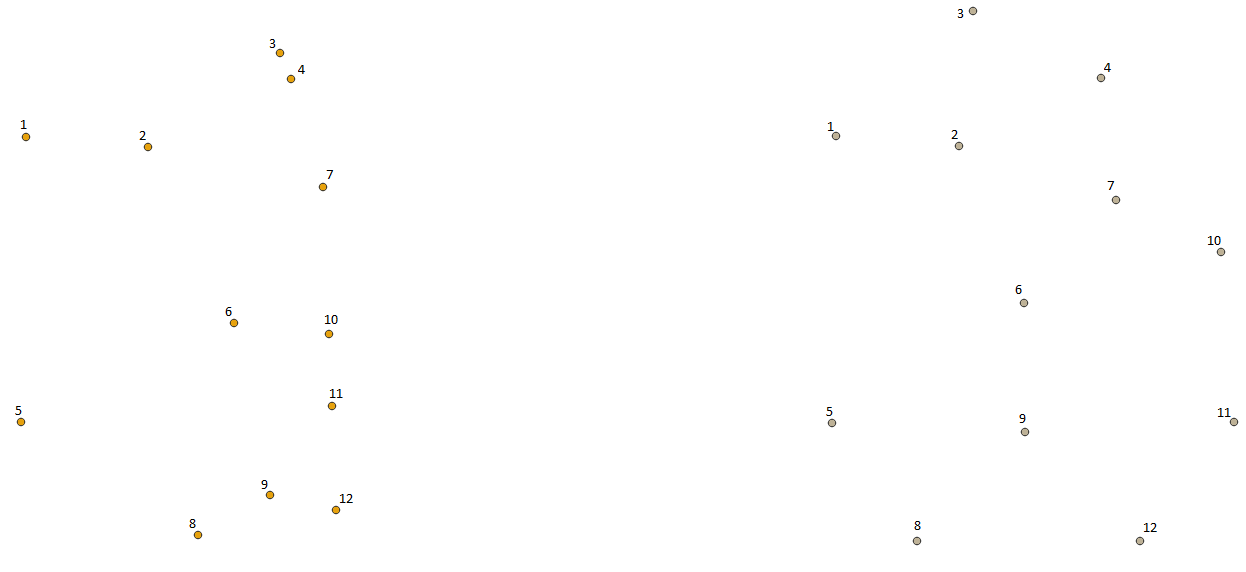
Расставим базовые точки, которые будут являться вершинами будущих треугольников. Для наглядности можно немного перенести базовые точки второй карты. Базовые точки двух карт представлены на рисунке 9.



а) б)

Рисунок 9 – Базовые точки. a – точки первой карте; б – точки второй карты

Необходимо сделать так, чтобы определённая точка первой карты соответствовала определённой точке второй карты. Для этого назначим им идентификаторы. Идентификаторы базовых точек представлены на рисунке 10.



а) б)

Рисунок 10 – Идентификаторы. a – точки первой карты; б – точки второй карты

Добавим на первую карту некоторые объекты, которые будем переносить на вторую. Добавленные объекты представлены на рисунке 11.

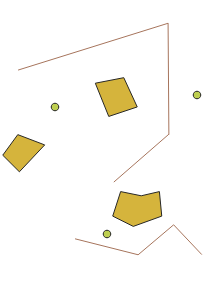


Рисунок 11 – Объекты

Произведем триангуляцию Делоне на первой карте. Результат триангуляции Делоне представлен на рисунке 12.

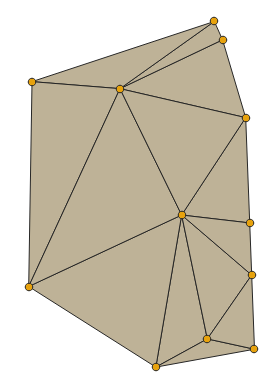


Рисунок 12 – Триангуляция

Так как точки на второй карте расположены немного по-другому относительно точек на второй карте, триангуляцию Делоне использовать нельзя, потому что могут появится нежелательные треугольники. На основе идентификаторов точек, расставленных в пункте 3, построим треугольники на второй карте, так чтобы треугольники обеих карт соответствовали друг другу. Треугольники второй карты представлены на рисунке 13.

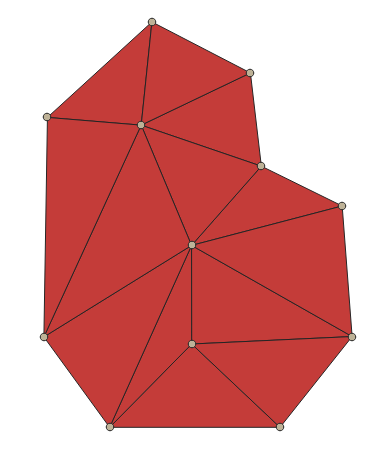
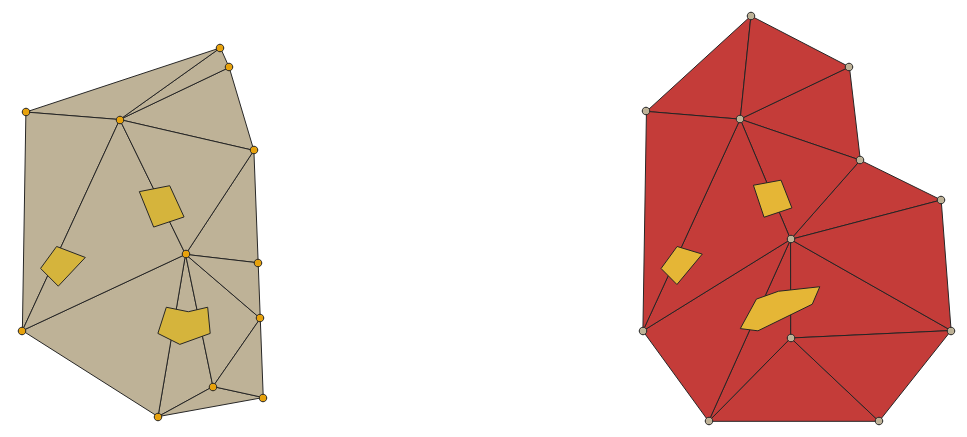


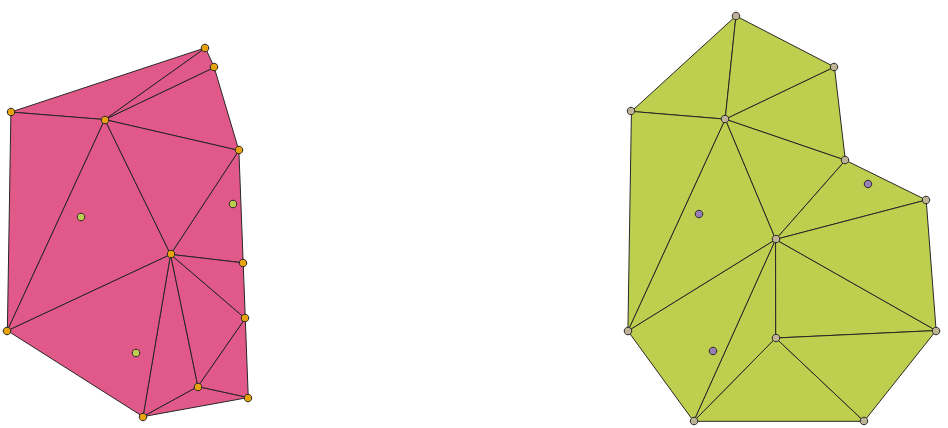
Рисунок 13 – Триангуляция на второй карте

С помощью барицентрических координат перенесём объекты с первой карты на вторую. Результаты комплексирования объектов представлены на рисунках 14 – 16.



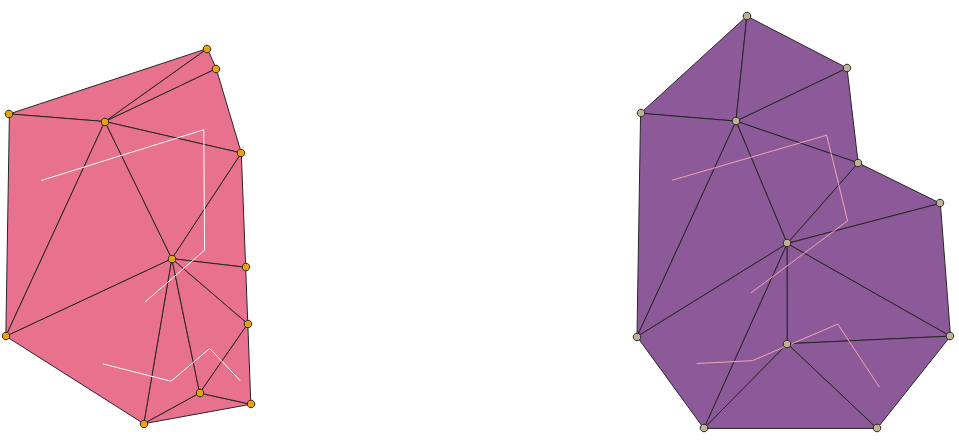
а) б)

Рисунок 14 – Перенос полигональных объектов. а – исходные объекты; б – комплексированные объекты



а) б)

Рисунок 15 – Перенос точечных объектов. а – исходные объекты; б – комплексированные объекты



а) б)

Рисунок 16 – Перенос линейных объектов. а – исходные объекты; б – комплексированные объекты

На рисунке 16 можно заметить, что на второй карте нижний линейный объект проходит через базовую точку. Это происходит, потому что в QGIS для построения линейных объектов требуется указать только массив точек, между которыми автоматически построятся отрезки. Каждая точка этого массива находится в своём треугольнике и привязана к нему. Поэтому при переносе учитывается только расположение конкретной точки внутри треугольника. Решение данной проблемы будет описано в разделе 4 «Второй этап разработки».

# 3 Исследовательская часть

В ходе бакалаврской работы был протестирован данный алгоритм на реальных данных.

Разработанный алгоритм работает с любой системой координат. В качестве данных для тестирования была выбрана карта города Устюг, имеющая систему координат Гаусса-Крюгера (EPSG:20008). Она является случаем специального применения проекции Гаусса-Крюгера и используется в Евразии, включая Россию и Китай. Данная система координат делит мир на зоны шириной в шесть градусов. В каждой зоне коэффициент масштаба равен 1, а смещение по долготе равно 500 000 метрам. Центральным меридианом зоны 1 является 3° восточной долготы.

Загрузим слой с линиями и полигонами. Загруженные слои представлены на рисунке 17.

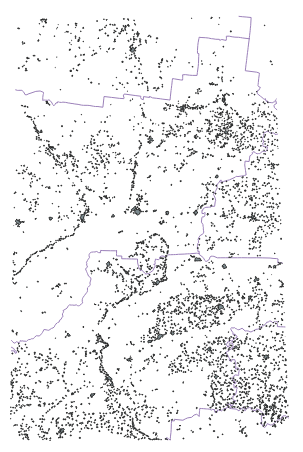


Рисунок 17 – Карта города Устюг

Добавим базовые точки для будущих треугольников. Для второй карты расположение базовых точек специально задано с различиями относительно базовых точек первой карты. Всего вершин для треугольников получилось 58 штук. Расставленные базовые точки представлены на рисунках 18 и 19. На рисунке 20 представлено сравнение расположения базовых точек.

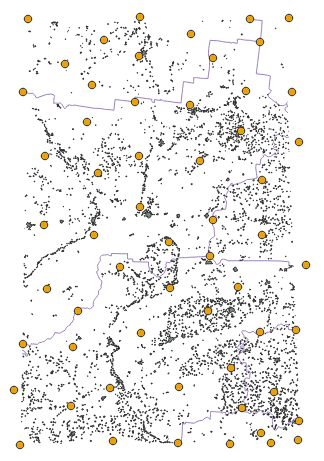


Рисунок 18 – Базовые точки первой карты

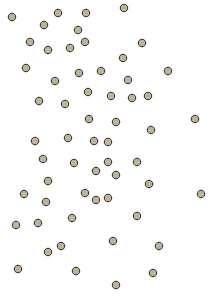
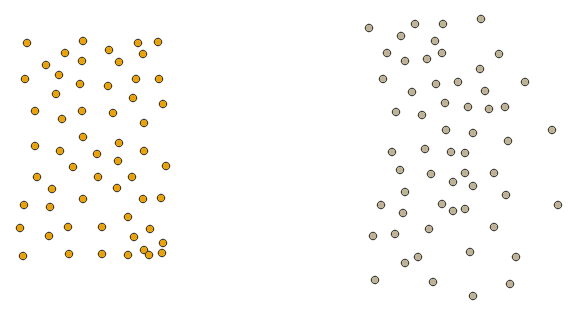


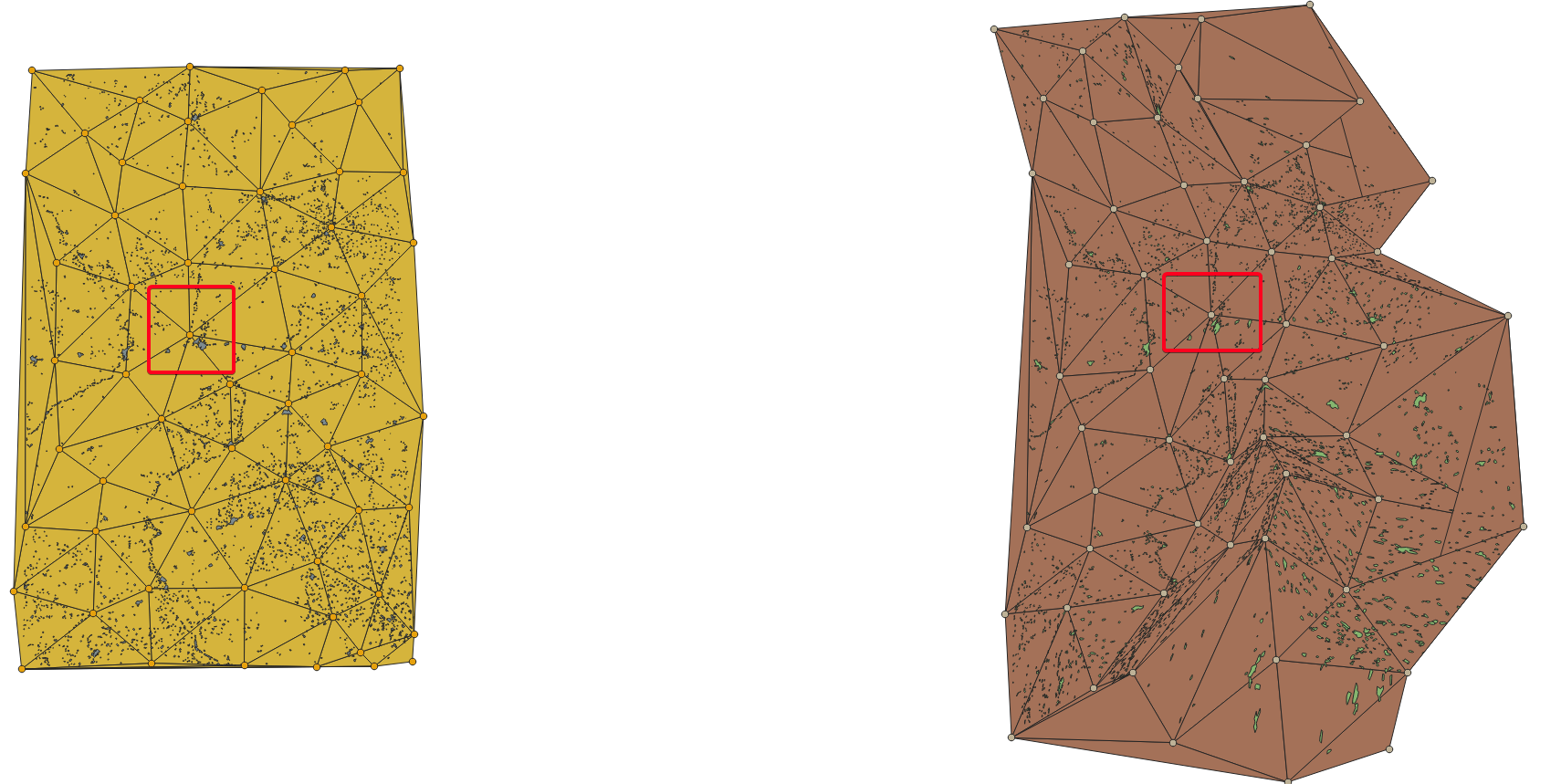
Рисунок 19 – Базовые точки второй карты



а) б)

Рисунок 20 - Сравнение базовых точек двух карт. а – базовые точки первой карты; б – базовые точки второй карты

Проведем триангуляцию и перенесём объекты с первой карты на вторую. Результат переноса представлен на рисунке 21.



а) б)

Рисунок 21 – Перенос объектов. а – первая карта с треугольниками; б – вторая карты с треугольниками

Рассмотри поближе области, выделенные красным на рисунке 21. Области представлены на рисунках 22 и 23.

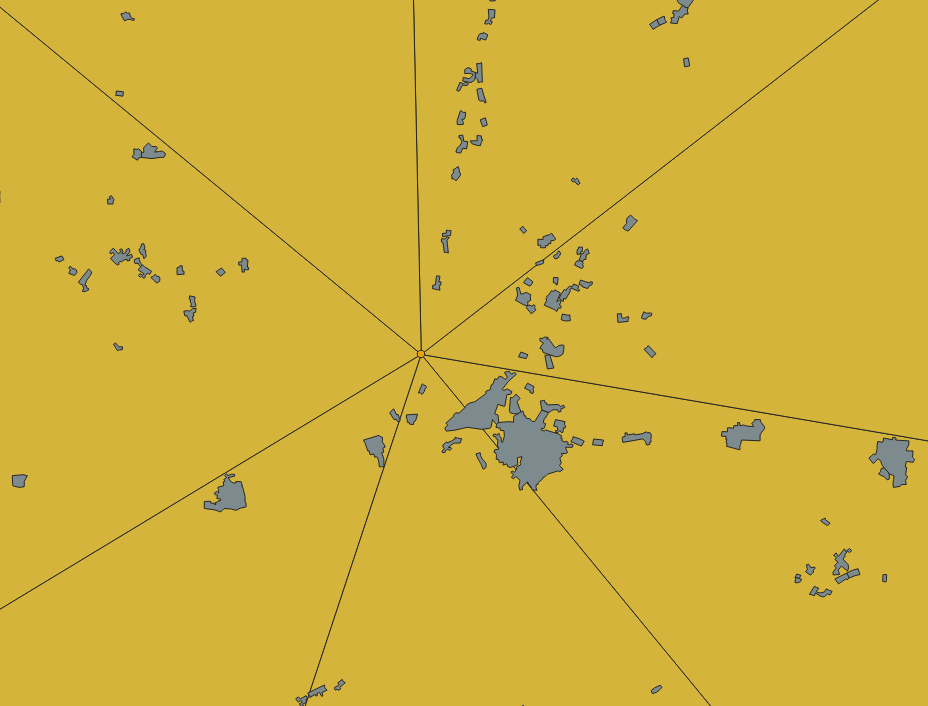


Рисунок 22 – Красная область на первой карте

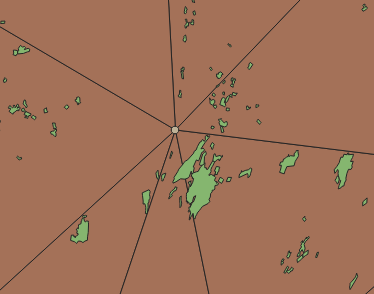


Рисунок 23 – Красная область на второй карте

По рисункам видно, что перемещенные объекты деформировались. Следовательно барицентрические координаты работают как положено, и у объектов есть привязка к треугольнику, в котором они находятся.

# 4 Второй этап разработки

Первым делом необходимо было улучшить перенос линейных и полигональных объектов. Полигональный объект состоит из нескольких зацикленных линейных объектов. Поэтому было принято решение что каждый линейный объект будет делить на несколько промежуточных объектов – отрезков. Таким образом у каждого линейного объекта появляется большее количество точек, что улучшает точность переноса объекта на другую карту.

На рисунке 24 представлена исходная линия. На рисунке 25 представлен результат переноса линии без деления ее на несколько отрезков. На рисунке 26 представлен результат переноса линии с делением.

Такой же эксперимент был проведён и над полигональными объектами. На рисунке 27 представлены исходные полигоны. На рисунке 28 представлены полигоны без деления линий, из которых состоят эти полигоны. На рисунке 29 представлены полигоны с делением линий. На рисунке 29 представлены полигоны с делением линий.

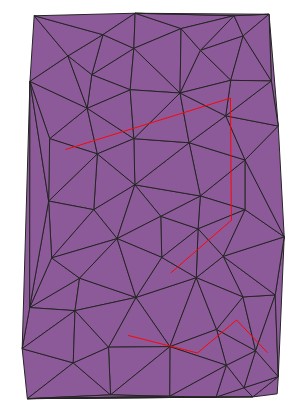
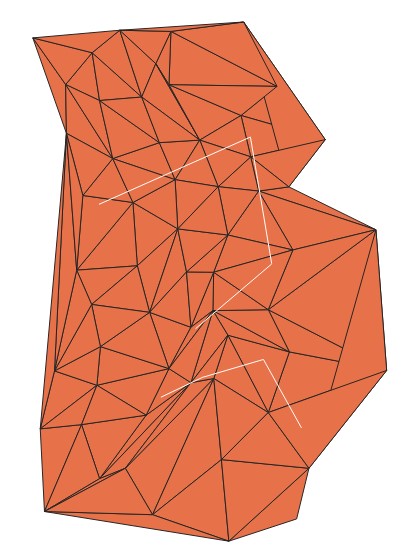


Рисунок 24 – Исходная линия

  
Рисунок 25 – Линия без деления

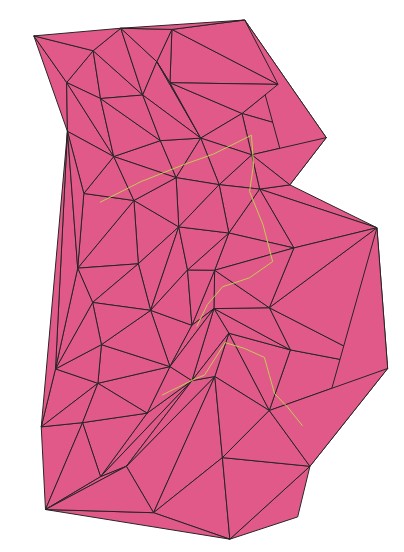


Рисунок 26 – Линия с делением

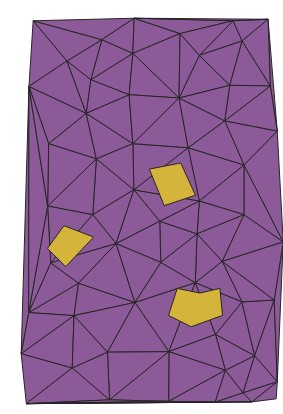


Рисунок 27 – Исходные полигоны

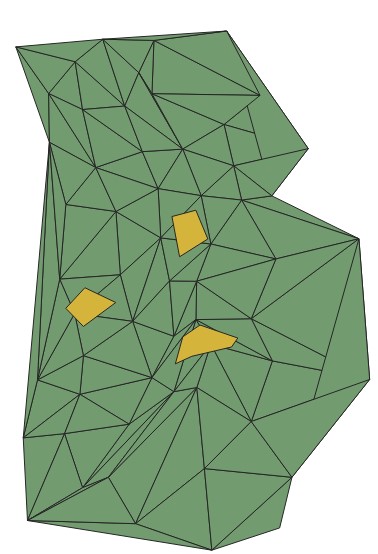


Рисунок 28 – Полигоны без деления

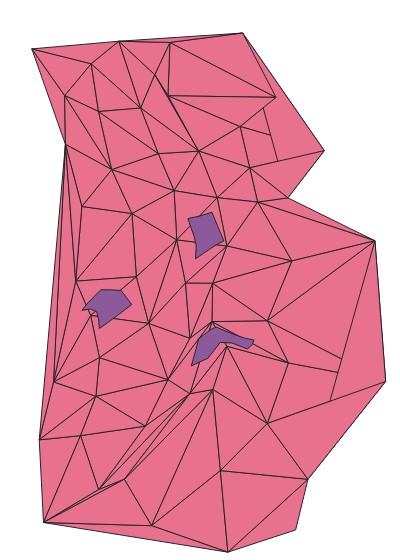


Рисунок 29 – Полигоны с делением

Одной из проблем появившейся в процессе первого этапа разработки, является необходимость расставлять базовые точки вручную. Решение нашлось в использовании алгоритма SIFT. Данный алгоритм способен находить одинаковые точки на двух растровых изображениях. Чтобы данный алгоритм работал вместе с разрабатываемым, необходимо преобразовать исходные векторные карты в растровый формат, то есть растеризация. Данный процесс можно выполнить с помощью встроенного средства QGIS.

На рисунке 30 представлен результат работы алгоритма SIFT

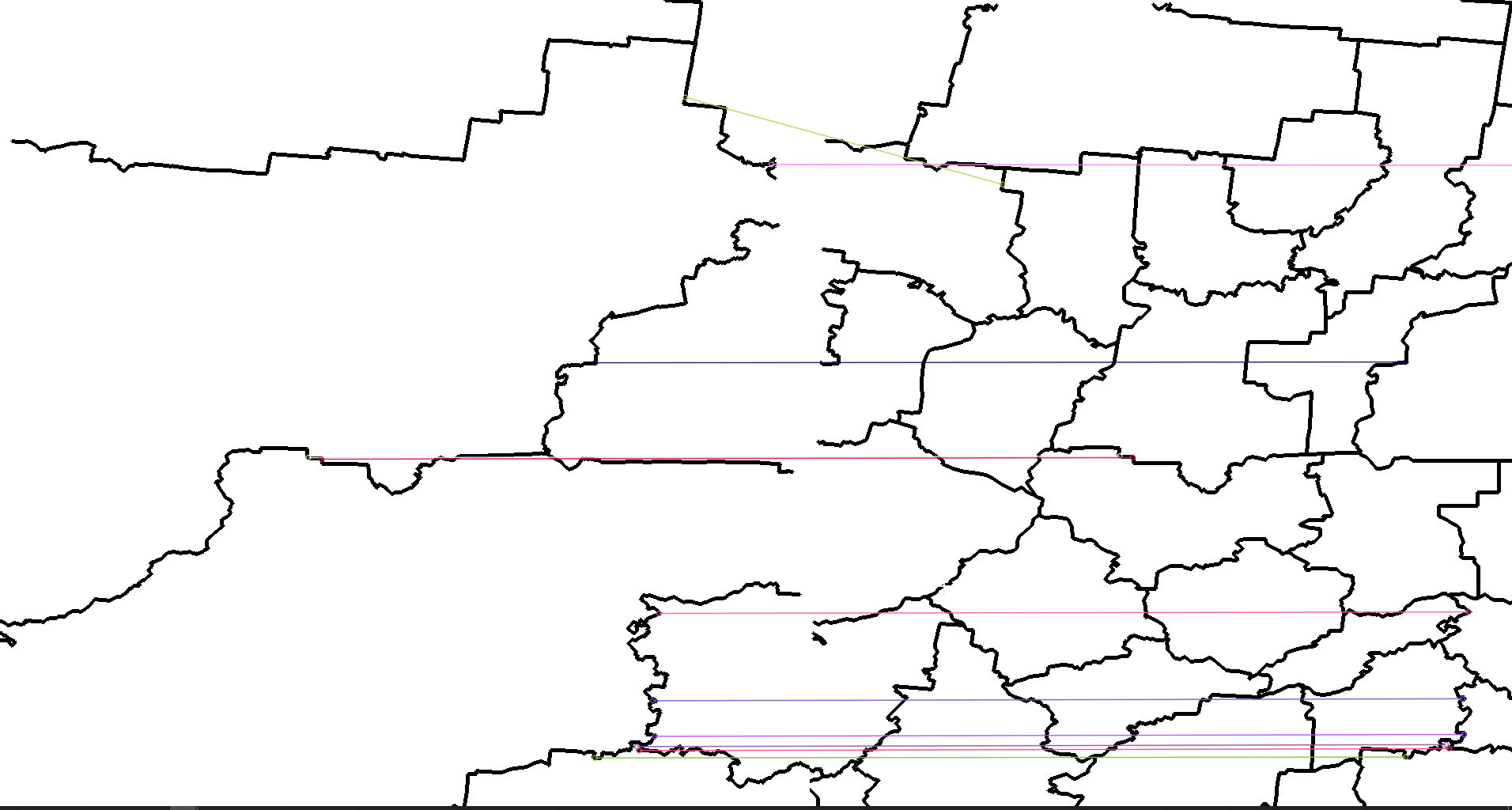


Рисунок 30 - Результат работы алгоритма SIFT.

Полученные базовые точки были перенесены в векторный формат и уже на их основе проводилась дальнейшая работа алгоритма.

На рисунке 31 представлена исходная карта с тестовыми данными и, построенными на основе полученных базовых точек, треугольниками. На рисунке 32 представлена уже преобразованная карта.

Как можно заметить переместились объекты, расположенные только внутри треугольников, поэтому было решение добавить на каждый угол карты еще по одной точке. На рисунке 33 представлены треугольники первой карты, покрывающие всю область карты. На рисунке 34 представлена преобразованная карта с добавленными по краям базовыми точками.

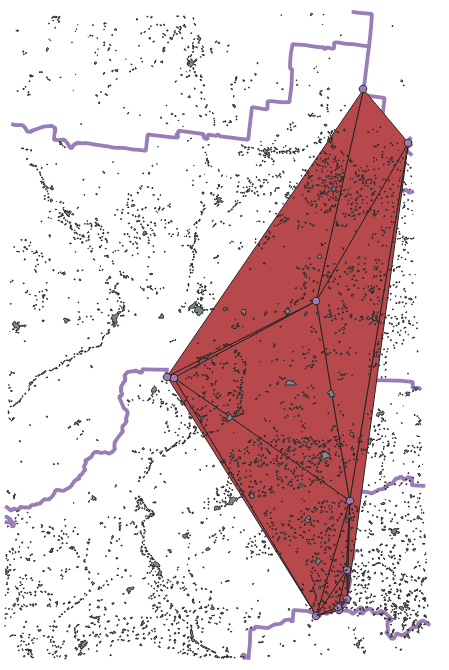


Рисунок 31 – Исходная карта

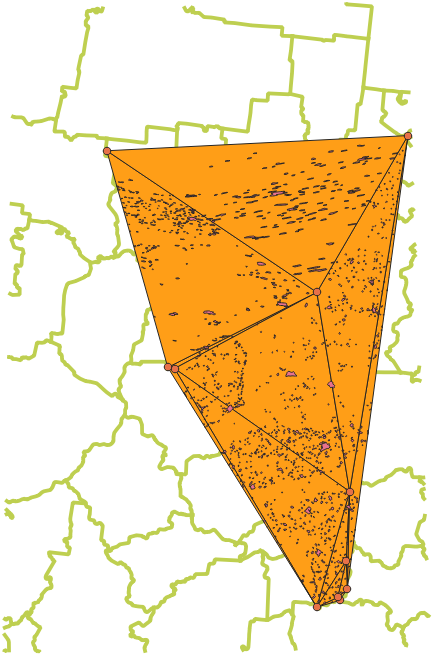


Рисунок 32 – Преобразованная карта

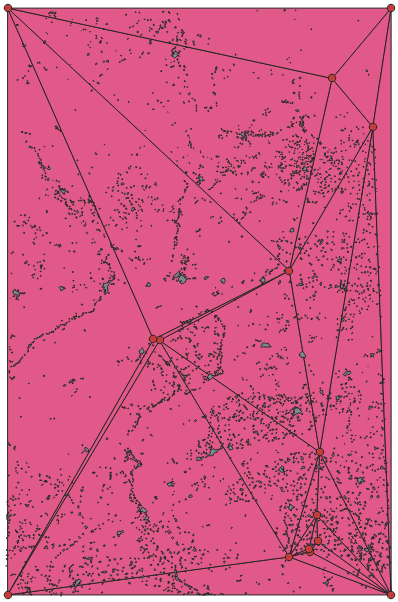


Рисунок 33 – Добавленные точки по углам карты

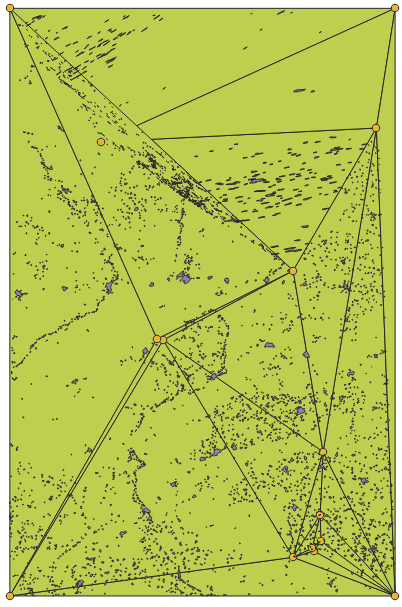


Рисунок 34 – Преобразованная карта

Основной концепцией разрабатываемого алгоритма является перемещение объектов с сохранением топологических связей. Для этого было принято решение использовать алгоритм построения пространственного индекса. С помощью него мы можем отслеживать расположение каждого перемещаемого объекта на слое и его топологические связи. Это было реализовано в файле indexing\_of\_elements.py. Скрипт был написан на языке программирования PyQGIS.

4.1 Файл indexing\_of\_elements.py

4.1.1 Словесное описание алгоритма

Файл включает в себя класс Index\_of\_elements, включающий в себя 11 основных функций. Теперь подробно разберем алгоритм индексации.

После запуска программы инициализируются 3 глобальные переменные для хранения: объекта класса QgsProject, списка названий всех слоев по выбранному проекту и пустого списка для индексирования временных слоев (прямоугольников). Кроме этого, запускается функция удаления, которая при наличии временных слоев удаляет их. Причем только те слои, в названиях которых содержится имя «rectangle», чтобы избежать удаления важных слоев.

Запускается функция для нахождения крайних точек карты. Экстремумы формируются после сравнения координат всех точек объектов в каждом слое. По этим точкам строится основной прямоугольник белого цвета в виде временного слоя. Данный слой и все созданные в дальнейшем прямоугольники будут иметь тип MultiPolygon.

Далее выполняется деление главного прямоугольника на 4 основных. На данном этапе сначала математически находятся точки, по которым будут строиться временные слои. Для этого определяются середины каждых сторон главного прямоугольника и его центр. По найденным точкам строятся 4 прямоугольника белого цвета. Белый цвет будет указывать на отсутствие в нём объектов. По завершению построения главный слой удаляется, так как в дальнейшем он будет мешать отображению внутренних прямоугольников.

После этого запускается рекурсия по 4 основным слоям. Сущность рекурсии заключается в следующем:

* если в прямоугольнике находится более 1 объекта вызвать функцию разделения и выполнить повтор, на рисунке 35 представлена данная ситуация;
* если в прямоугольнике находится 1 объект, то не выполнять разделение и перейти к следующему прямоугольнику, на рисунке 36 представлена данная ситуация;
* если в прямоугольнике несколько объектов, но они имеют общую точку пересечения, то не выполнять разделение и перейти к следующему прямоугольнику, на рисунке 37 представлена данная ситуация;
* если в каждом прямоугольнике по 1 объекту или несколько объектов с общей точкой, то остановить рекурсию, на рисунке 38 представлена данная ситуация.

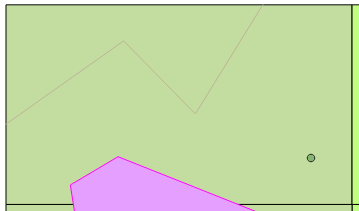


Рисунок 35 – Прямоугольник с 3-мя объектами, в котором необходимо сделать разделение



Рисунок 36 – Прямоугольник с 1-м объектом, в котором не нужно выполнять разделение

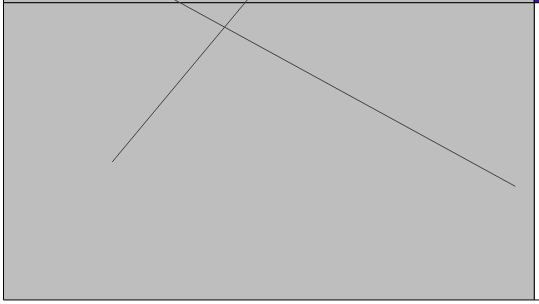


Рисунок 37 – Прямоугольник с общей точкой пересечения, в котором не нужно выполнять разделение

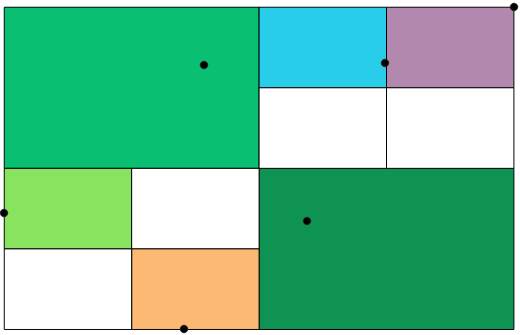


Рисунок 38 – Пример завершения рекурсии

В результате получается карта, разделенная на прямоугольники, включающие в себя один или несколько объектов. На рисунках 39 – 42 приведён пример итерационного процесса формирования прямоугольников.

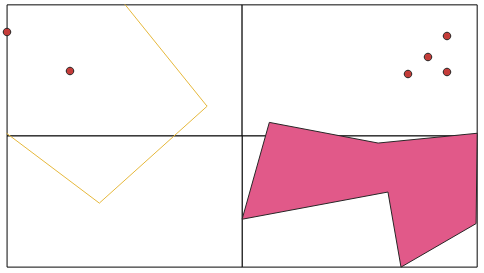


Рисунок 39 – Итерационный процесс формирования прямоугольников

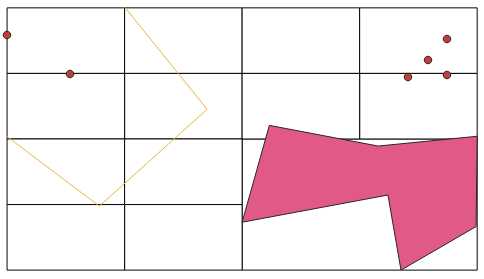


Рисунок 40 – Итерационный процесс формирования прямоугольников

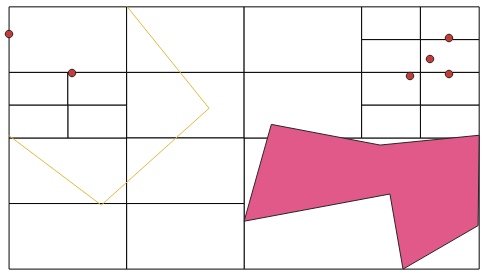


Рисунок 41 – Итерационный процесс формирования прямоугольников

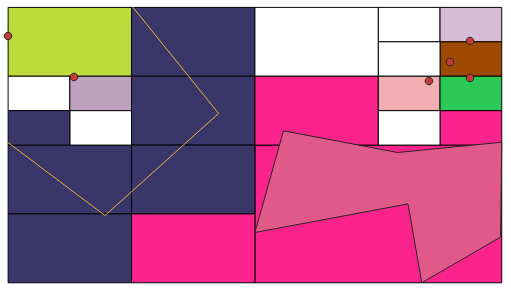


Рисунок 42 – Итерационный процесс формирования прямоугольников

Вызывается функция индексации. Запускается итерация по всем объектам и прямоугольникам. Затем проверяется условие на наличие пересечения объекта с прямоугольником. Если имеется пересечение, то сохраняется его индекс. Примерный вид индекса – «2.1.3.4». Данная запись означает, что объект пересекает 2-ой прямоугольник после 1-го разделения, 1-ый прямоугольник после 2-го разделения, 3-ий прямоугольник 3-го разделения и 4-ый прямоугольник 4-го разделения. Иными словами – это путь разделения до выбранного прямоугольника.

После прохождения цикла по прямоугольникам в список заносится название слоя объекта, id объекта и строка индексов, где эти элементы разделены символом «\_», а каждый индекс разделен символом «;». Далее цикл повторяется и проходит по всем объектам.

На рисунке 43 представлена визуальная индексация прямоугольников

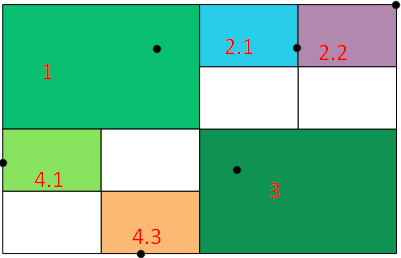


Рисунок 43 – Визуальная индексация прямоугольников

На последнем этапе вызывается функция установки цвета прямоугольников. Сущность ее заключается в следующем:

* запускаем цикл по всем объектам:
* находим его по названию слоя и id в списке;
* обрабатываем строку индексов;
* у всех прямоугольников с этими индексами случайно задаем цвет в формате rgb;
* прямоугольникам с общими точками (одинаковыми индексами) задается серый цвет.

На рисунке 44 представлена установка цветов прямоугольникам.

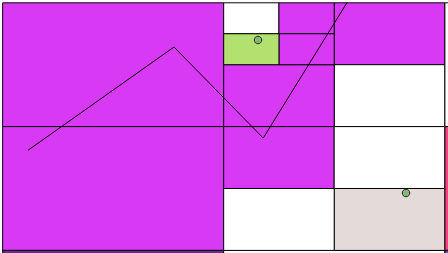


Рисунок 44 – Одноцветное выделение прямоугольников одного объекта

Следующим этапом разработки является нахождение пересекающихся объектов, расположенных на разных слоях. Можно заметить, что если объекты пересекаются, то они находятся в одном индексе, но в некоторых ситуациях объекты, расположенные на одном слое, так же находятся в одном индексе, что для нашей ситуации некорректно. Для этого необходимо правильно обработать список индексов. На рисунке 45 представлен как примерно может выглядеть список индексов.



Рисунок 45 – Список индексов

Рассмотрим один элемент из этого списка поподробнее. На рисунке 46 представлен элемент списка индексов.



Рисунок 46 – Элемент списка индексов.

Разделим данный элемент по знаку нижнего подчеркивания («\_»). Первая часть, расположенная до первого нижнего подчеркивания, означает имя слоя. Вторая часть – между двумя нижними подчеркиваниями – id объекта в слое, указанного в первой части. Третья часть, расположенная после нижнего подчеркивания, является списком идентификаторов индексов, в которых расположен объект, id которого указан во второй части элемента.



Рисунок – Пространственный индекс