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 ИТР

Кафедра информационных систем

КУРСОВОЙ ПРОЕКТ

по. Геоинформационные технологии

тема: Анализ расположения абстрактных пространственных фигур

Руководитель

к. т. н., доцент, каф. ИС

(уч. степень, звание)

Еремеев С.В.

(фамилия, инициалы)

(подпись) (дата)

Студент ИСм-121

(группа)

Толмачев А.С.

(Ф.И.О.) (фамилия, инициалы)

(Ф.И.О.) (подпись) (дата)

Муром 2022

Лист задания.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc105192404)

[1 Анализ технического задания 6](#_Toc105192405)

[1.1 Обзор предметной области 6](#_Toc105192406)

[1.2 Исходные данные к проекту 7](#_Toc105192407)

[1.3 Требования к разрабатываемому проекту 7](#_Toc105192408)

[2 Разработка алгоритма 8](#_Toc105192409)

[3 Реализаия геоинформационных технологий 13](#_Toc105192410)

[3.1 Построение графа точек 13](#_Toc105192411)

[3.2 Построение персистентной диаграммы 17](#_Toc105192412)

[3.3 Объединение точек в абстрактную фигуру. 18](#_Toc105192413)

[3.4 Определение топологических отношений абстрактных фигур. 22](#_Toc105192414)

[4 Проведение экспериментальных исследований 25](#_Toc105192415)

[ЗАКЛЮЧЕНИЕ 39](#_Toc105192416)

[СПИСОК ЛИТЕРАТУРЫ 40](#_Toc105192417)

# ВВЕДЕНИЕ

Системы ГИС постоянно развиваются. Одна из основных функций данной системы – хранение информации о топологических структурах на карте. Данные структуры могут представлять собой практически любые объекты на карте. Структуры могут быть разных видов: точки, линии, многоугольники и прочее. В данной работе будут рассматриваться абстрактные структуры. Абстрактные структуры представляют собой множество точек, которые объединяются в замкнутый многоугольник. Данные структуры могут иметь различные топологические отношения друг к другу. Существуют следующие виды топологических отношений: отношения вложенности, отношения пересечения и отсутсвие отношений.

Цель данной работы - разработка алгоритма поиска расположения абстрактных пространственных структур. Алгоритм направлен на построение фигур и анализ расположения их относительно друг друга.

Целью работы является построение абстрактных пространственных структур, представленных в виде точек, линий и полигонов и нахождения взаимосвязей между ними.

Для достижения этой цели были поставлены следующие задачи:

a) проведение анализа предметной области, предъявление требований к разрабатываемому алгоритму;

b) построение алгоритма создания абстрактных пространственных фигур, основанных на векторных картах, представленных в виде точек, линий и полигонов;

c) построение алгоритма для нахождения топологических отношений между пространственными объектами.

d) проведение экспериментов и анализ результатов моделирования

Таким образом, структура данного курсового проекта состоит из четырех глав и выглядит следующим образом:

a) в первой главе производится анализ технического задания, который включает в себя:

- обзор предметной области;

- определение исходных данных к проекту;

- установление требований к проекту;

b) во второй главе описывается процесс разработки алгоритма поиска топологических отношений на основе персистентной гомологии;

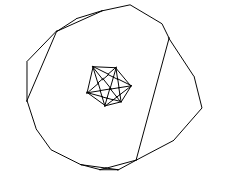
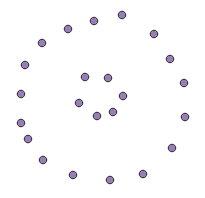
с) в третьей главе идет реализация геоинформационной технологии, разработка которой описана в главе 2.

c) в четвертой главе описывается процесс тестирования работы разработанного в предыдущей главе алгоритма и производится анализ полученных результатов.

1. **Анализ технического задания**

**1.1 Обзор предметной области**

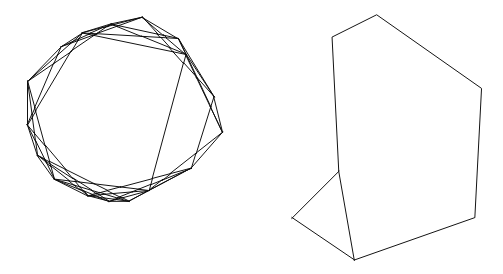
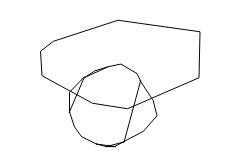
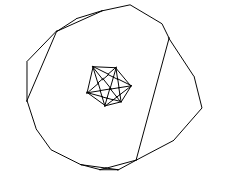
Необходимо разработать программу, которая будет выделять абстрактные структуры на карте, а также определять топологические отношение между структурами. Будет разработан алгоритм для нахождения абстрактных структур на карте. Прежде чем будут определяться абстрактные структуры, необходимо определить множество точек, по которым будут строиться абстрактные структуры на карте. Абстрактные структуры буду определяться путем соединения структурных точек, линий и полигонов в замкнутый контур.



а) б)

1. Пример нахождения абстрактных структур: а – структурные точки; б – обнаруженные структуры

Топологические отношения между структурами в разрабатываемом проекте будут следующих видов: отсутствие каких-либо отношений, пересечение структур, отношения вложенности, когда одна структура располагается внути или вокруг другой структуры. Необходимо будет найти такие отношения между всеми объектами.



1. Пример нахождения абстрактных структур: а – вхождение; б – пересечение; в – нет связей

# 1.2 Исходные данные к проекту

В качестве исходных данных к проекту были определены следующие:

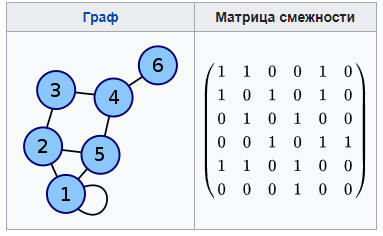
1. n-ое количество векторных карт, на которых будет тестироваться разработанный алгоритм.
2. среда разработки Microsoft Visual Studio Code.
3. алгоритм разрабатывается как плагин для программы ГИС QGIS.
   1. **Требования к разрабатываемому проекту**

Исходя из задач, поставленных перед началом работы, к разрабатываемому проекту необходимо выставить следующие требования:

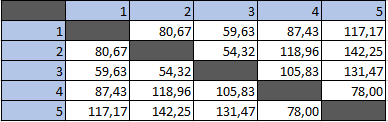
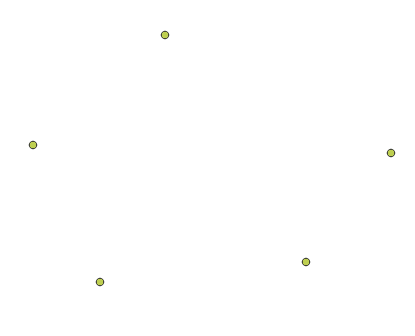
1. построение абстрактных объектов на основе данных векторных карт;
2. определение топологических отношений абстрактных фигур, созданных ранее
3. **Разработка алгоритма**

В программе будет реализован алгоритм для нахождения абстрактных структур. Это автоматическое преобразование точек на карте, по которым будут строится абстрактные структуры, алгоритм объединения структурных точек в целые абстрактные структуры и алгоритм поиска топологических отношений между ними.

Первым этапом является получение списка точек из слоев точек, линий и полигонов и на их основе построение матрицы смежности. Матрица смежности является одним из способов представления графа в виде матрицы. В данной работе вершиной матрицы смежности будет выступать точка, а ребром – расстояние между 2 точек.

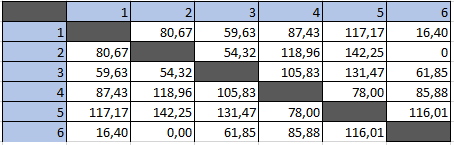
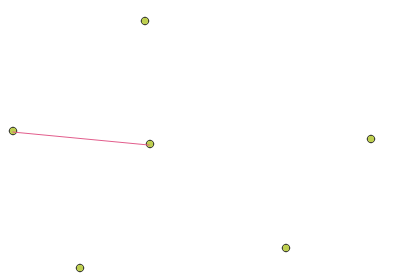


1. Матрица смежности (пример)



1. Матрица смежности, построенная на точках, где вершиной является точка, а ребром – ее растояние до другой точки

В случае, если точки находятся в одном полигоне или на одной линии, то расстояние между ними будет равно 0 на матрице смежности.



1. Пример матрица смежности, где точки 2 и 6 соединены линией

После построения матрицы смежности приступаем к анализу персистентности точек. Для анализа используем фильтрацию Вьеториса-Рипса.

Комплекс Вьеториса-Рипса, также называемый комплексом Вьеториса или комплексом Рипса , представляет собой способ формирования [топологического пространства](https://www.hmong.press/wiki/Topological_space) из расстояний в наборе точек. Это [абстрактный симплициальный комплекс,](https://www.hmong.press/wiki/Abstract_simplicial_complex) который можно определить из любого [метрического пространства](https://www.hmong.press/wiki/Metric_space)M и расстояния δ, образуя [симплекс](https://www.hmong.press/wiki/Simplex) для каждого [конечного набора](https://www.hmong.press/wiki/Finite_set) точек, [диаметр которого](https://www.hmong.press/wiki/Diameter#Generalizations) не превышает δ. То есть это семейство конечных подмножеств M , в котором представляется подмножество из k точек как об образующих a ( k - 1) -мерный симплекс (ребро для двух точек, треугольник для трех точек, тетраэдр для четырех точек и т. Д.); если конечное множество S обладает тем свойством, что расстояние между каждой парой точек в S не превосходит δ, то S описывается как симплекс в комплекс.

Топологический анализ матрицы смежности через фильтр Вьеториса-Рипса позволит создать персистентную диаграмму, на которой будет обозначены «времена жизни» симплициальных комплексов – расстояния δ, на которым они образуются и «умирают».

Персистентная диаграмма – один из способов описания облака точек с топологической точки зрения, отражающий информацию об интервалах постоянства чисел Бетти, возникающих в процессе фильтрации.

Персистентная диаграмма – один из способов описания облака точек с топологической точки зрения. это набор интервалов (b, d), где границы интервалов соответствуют значениям параметра, при которых значение числа Бетти рождается и умирает.

Числа Бетти — последовательность инвариантов топологического пространства. Каждому пространству XX соответствует некая последовательность чисел Бетти β0(X),β1(X),…β0(X),β1(X),….

Нулевое число Бетти β0(X)β0(X) совпадает с числом связных компонент;

Первое число Бетти β1(X)β1(X) интуитивно представляет собой максимальное число разрезов этого пространства, которые можно сделать без увеличения числа компонент связности.

Число Бетти может принимать неотрицательные целые значения или бесконечность. Для разумно устроенного конечномерного пространства (например, компактного многообразия или конечного симплициального комплекса), все числа Бетти конечны и, начиная с некоторого номера, равны нулю.

В случае фильтрации Вьеториса-Рипса в качестве границ интервала принимаются максимальное по длине ребро симплекса и максимальное ребро симплекса более высокой размерности, гранью которого является рассматриваемый.

Стандартный алгоритм построения персистентной диаграммы:

- Построение фильтрации – те получение упорядоченного набора симплексов (симплекс в фильтрации идет после своих граней и симплексы одной размерности упорядочены по увеличению ”времени” появления)

- Составление матрицы граничного оператора

- Редуцирование матрицы (вариация алгоритма Гаусса над конечным полем) В случае, когда вычисления проводятся над Z2, стандартный алгоритм редуцирования сводится к произведению операции xor над столбцами матрицы ”справо налево”.

- Считывание персистентной диаграммы через персистентные пары – те пары индексов (i, j), где j – номер столбца редуцированной матрицы, i – индекс нижней единицы в нем.

Матрица смежности проходит через фильтр Вьеториса-Рипса, получая персистентную диаграмму с числами Бетти.

Для алгоритма поиска структур из персистентной диаграммы достаточно получения нулевых чисел Бетти. В зависимости от необходимого нам числа объектов будет получено пороговое значение, по которому будут соединены все точки, расстояние которых меньше порогового значения.

После соединения по пороговому значению, соединенные точки можно представить как полигональные объекты, с которые можно сравнить друг с другом.

Программа QGIS предоставляет необходимый инструментарий для работы с полигонами. Так функция «Пересечение» позволит получить новый слой, в котором будет обозначено пересечение 2 полигонов. У данной функции есть 3 результата и в зависимости от него можно сделать и последующие выводы:

1. Программа создала пустой слой. Наличие пустого слоя значит отсутствие пересечений объектов и, следовательно, отсутствие связей между ними.
2. Программа создала слой с пересечением объектов. Наличие объектов в слое пересечения(qgis всегда называет так слой после одноименной функии) означает наличие связи типа «пересечение» или «вхождение». Для большей подробности определяется площадь родительских объектов и слоя с пересечением и если площадь нового объекта меньшн родительской, то связь между родителями определяется как «пересечение»
3. Программа создала слой с вхождением одного объекта в другой. Как описано выше связь между объектами также можно посчитать как «пересечение» или «вхождение», для чего также будет необходимо получение подробностей о ее площади. Для связи «вхождении» площадь нового объекта должна быть равна одному из родительских объектов

В результате был разработан алгоритм, который способен сформировывать абстрактные пространственные структуры на основе Persistent Homology и вычислять пространственные расположения объектов

1. **Реализаия геоинформационных технологий**

Для реализации предлагаемого алгоритма было разработан плагин для программы ГИС QGIS на языке python. Применение разработанного алгоритма позволит создавать абстрактные топологические структуры и позволит находить их отношения.

Для возможного использования программы необходимы быть установлены следующие библиотеки:

- imp (отвечает за загрузку обновлений модулей, необязательная библиотека)

- numpy (мощный математический инструмент python-на)

- spicy

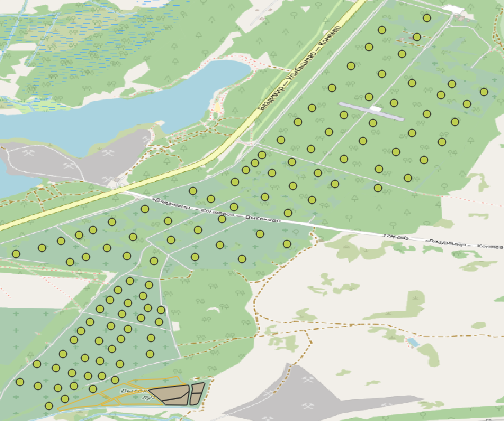
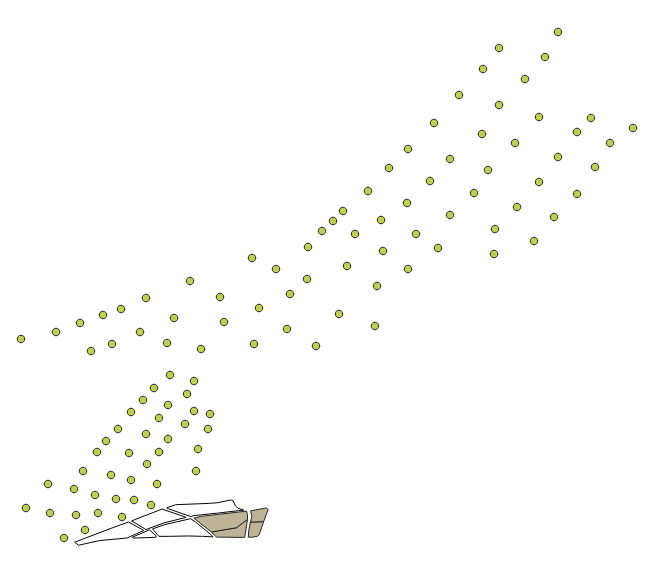
- gtda (высокопроизводительный топологический набор инструментов машинного обучения)

3.1 Построение графа точек

Первым этапом программы является определение слоев и количество объектов, которые вы желаете видеть.

layer\_names=["cemetery\_sector1","cemetery\_sector2","cemetery\_sector3"]

count\_object= int(3)

1. Демонстрация слоев кладбища на векторной карте

Для классификации точек создается класс точек, хранящий координаты точки, соседей, с которыми эта точка связана, имя точки и группа, в которую вписана данная точка. На основе координат и соседей и будет строится персистеная диаграмма.

class Point: #класс хранит координаты точки

def \_\_init\_\_(self, x,y):

self.x=x

self.y=y

class Node:

def \_\_init\_\_(self, x,y,name=None):

"""\_summary\_

Args:

x (\_type\_): координаты

y (\_type\_): координаты

name (\_type\_, optional): имя переменной. Defaults to None.

"""

self.point=Point(x,y)

self.connect=[]

self.name=name

self.name\_grup=0

def new\_connection(self,node):

""" связывает точки друг с другом

Args:

node (\_type\_): Модуль, с которым будет построено соединение

"""

self.connect.append(node)

node.connect.append(self)

def whether\_connected(self,node):

""" проверяет, связаны ли точки

Args:

node (Node): Узел, с которым идет проверка соединения

Returns:

Bool: Соединена ли точка

"""

for i in self.connect:

if node==i:

return True

return False

def rename(self,name):

"""\_summary\_ Меняет название группы, при условии, что его не было, на name.

Args:

name (str or int): Имя, на которое будут переименовываться все объекты

"""

useful=False

if self.name\_grup==0:

self.name\_grup=name

useful=True

for i in self.connect:

i.rename(name)

return useful

Для ввода в класс данных используются следующие функции:

def new\_line(coordinats):

cor=coordinats.pop(0)

nodes=[Node(cor[0],cor[1],cor[2])]

for i in coordinats:

nodes.append(Node(i[0],i[1],i[2]))

nodes[len(nodes)-1].new\_connection(nodes[len(nodes)-2])

return nodes

def new\_poligon(coordinats):

cor=coordinats.pop(0)

nodes=[Node(cor[0],cor[1],cor[2])]

for i in coordinats:

nodes.append(Node(i[0],i[1],i[2]))

nodes[len(nodes)-1].new\_connection(nodes[len(nodes)-2])

nodes[0].new\_connection(nodes[-1])

return nodes

def new\_point(cor):

return Node(cor[0],cor[1],cor[2])

Из каждого введенного слоя будут вытаскиваться все точки для построения графа точек и помещены в класс Node с помощью функий, описанных выше. В зависимости от типа слоя (линия или полигон) некоторые точки могут быть соединены между собой, поэтому они будут считаться соседями. Это необходимо, чтобы персистентная диаграмма не учитывала их при топологическом анализе.

project = QgsProject.instance()

point\_list=[]

to\_print=False

for layer\_name in layer\_names:

"""

находит все точки полей на карте в соответствии с полем ввода layer\_names

"""

layer = project.mapLayersByName(layer\_name)

my\_layer = layer[0]

features = my\_layer.getFeatures()

for feature in features:

# retrieve every feature with its geometry and attributes

if to\_print:

print("Feature ID: ", feature.id())

# fetch geometry

# show some information about the feature geometry

geom = feature.geometry()

object\_chec=0

geomSingleType = QgsWkbTypes.isSingleType(geom.wkbType())

if geom.type() == QgsWkbTypes.PointGeometry:

# the geometry type can be of single or multi type

if geomSingleType:

x = geom.asPoint()

points=[]

points.append([x.x(),x.y(),"P-"+str(feature.id())+"\_"+str(object\_chec)])

object\_chec+=1

point\_list+=ps.new\_points(points)

if to\_print:

print("Point: ", x)

else: #доработать

x = geom.asMultiPoint()

for point in x:

points=[]

points.append([point.x(),point.y(),"P-"+str(feature.id())+"\_"+str(object\_chec)])

object\_chec+=1

point\_list+=ps.new\_points(points)

if to\_print:

print("MultiPoint: ", x)

elif geom.type() == QgsWkbTypes.LineGeometry:

if geomSingleType:

x = geom.asPolyline()

line\_list=[]

for point in x:

point\_of\_line=0

line\_list.append([point.x(),point.y(),"MPL-"+str(feature.id())+"\_"+str(object\_chec)])

object\_chec+=1

point\_list+=ps.new\_poligon(line\_list)

if to\_print:

print("Line: ", x, "length: ", geom.length())

else:

x = geom.asMultiPolyline()

old\_x,old\_y=None,None

for line in x:

point\_of\_line=0

line\_list=[]

for point in line:

old\_x,old\_y=point.x(),point.y()

line\_list.append([point.x(),point.y(),"MPL-"+str(feature.id())+"\_"+str(object\_chec)])

object\_chec+=1

point\_list+=ps.new\_poligon(line\_list)

if to\_print:

print("MultiLine: ", x, "length: ", geom.length())

elif geom.type() == QgsWkbTypes.PolygonGeometry:

if geomSingleType: #доработать

x = geom.asPolygon()

if to\_print:

print("Polygon: ", x, "Area: ", geom.area())

else:

x = geom.asMultiPolygon()

if to\_print:

print("new\_object")

old\_x,old\_y=None,None

for line in x:

point\_of\_line=0

line\_list=[]

line=line[0]

for point in line:

if to\_print:

print(point)

print("Point: x: ", point.x())

print(" y: ", point.y())

# if old\_x!=None:

# print("line length:",((point.x()-old\_x)\*\*2+(point.y()-old\_y)\*\*2)\*\*(0.5))

old\_x,old\_y=point.x(),point.y()

line\_list.append([point.x(),point.y(),"MPlg-"+str(feature.id())+"\_"+str(object\_chec)])

object\_chec+=1

point\_list+=ps.new\_poligon(line\_list)

if to\_print:

print("MultiPolygon: ", x, "Area: ", geom.area())

else:

if to\_print:

print("Unknown or invalid geometry")

# fetch attributes

attrs = feature.attributes()

# attrs is a list. It contains all the attribute values of this feature

3.2 Построение персистентной диаграммы

После сбора всех точек в класс Note, с соответствующим предписанием связей точки запускается функция персистентного анализа. В качестве функции анализа используется фильтрация Виториса-Рипса (из библиотеки gtda), который принимает на вход матрицу смежности. Веса матрицы смежности если 2 точки не являются соседями, рассматриваются как расстояния между двумя точками, иначе веса будут равны 0.

В конечном итоге создается граф, в котором все точки являются вершинами графа, а все ребра – расстоянием между этими точками (у соседей расстояние не расчитывается и равно 0).

def diagram(points):

print("start persistent diagram")

new\_points=np.copy(points)

row = []

col = []

data = []

x=-1

while len(new\_points):

x+=1

point=new\_points[0]

new\_points=new\_points[1:]

y=x

for i in range(len(new\_points)):

y+=1

row.append(x)

col.append(y)

if point.whether\_connected(new\_points[i]):

data.append(0)

else:

data.append(

((point.point.x - new\_points[i].point.x)\*\*2+

(point.point.y - new\_points[i].point.y)\*\*2)\*\*0.5)

coo = coo\_matrix((data, (row, col)), shape=(len(points), len(points)))

persistent\_diogram = VietorisRipsPersistence ( metric = "precomputed" ).fit\_transform\_plot([coo])

print(persistent\_diogram)

print("The end of creat persistent diagram")

return persistent\_diogram

Функция выводит многомерный список, хранящий линии жизни всех компонентов анализируемого графа. Следующего вида:

[[[0.00000000e+00 7.18304218e-05 0.00000000e+00]

[0.00000000e+00 7.38346789e-05 0.00000000e+00]

[0.00000000e+00 8.39350323e-05 0.00000000e+00]

[0.00000000e+00 8.72674646e-05 0.00000000e+00]

...

...

[8.91389442e-04 1.04242237e-03 1.00000000e+00]

[8.61357083e-04 1.04745466e-03 1.00000000e+00]

[8.52665864e-04 8.75719881e-04 1.00000000e+00]

[8.42820446e-04 8.54698010e-04 1.00000000e+00]

[8.25317635e-04 1.12689671e-03 1.00000000e+00]

[0.00000000e+00 7.78647081e-04 1.00000000e+00]

[0.00000000e+00 4.42555145e-04 1.00000000e+00]

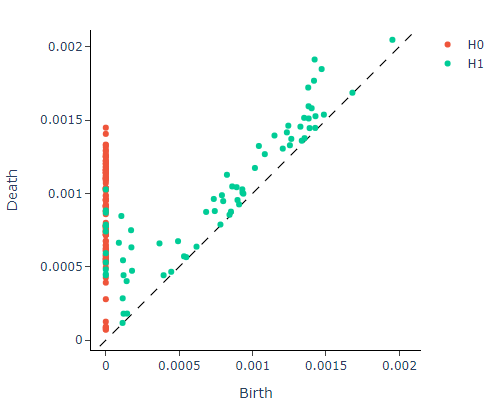
[0.00000000e+00 1.02702260e-03 1.00000000e+00]

[0.00000000e+00 4.49993095e-04 1.00000000e+00]

[0.00000000e+00 5.30066318e-04 1.00000000e+00]

[0.00000000e+00 5.93161385e-04 1.00000000e+00]]]

Помимо линий жизни программа выведит персистентную диаграмму в окне браузера:



1. Персистентная диаграмма кладбища

3.3 Объединение точек в абстрактную фигуру.

На основе полученной персистентной диаграммы идет расчет расстояния, по которому будут объединяться точки.

diagram\_per=diagram[diagram[:,2]==0]

porog=0

if len(diagram\_per)<count\_object:

print("Ошибка, количество объектов ")

count\_object= len(diagram\_per)+1

print(len(diagram\_per),count\_object)

if count\_object==len(diagram\_per)+1:

porog=diagram\_per[0][1] - 0.00001 \* diagram\_per[0][1]

else: porog=diagram\_per[len(diagram\_per)-count\_object][1] + 0.00000001 \* diagram\_per[len(diagram\_per)-count\_object][1]

print("porog",porog)

После получения порогового расстояния идет объединение точек в группы и их последующая отрисовка.

connect\_by\_treshold(point\_list,porog)

rename\_group\_of\_points\_lines(point\_list)

draw\_points(point\_list)

draw\_lines(point\_list)

Функция connect\_by\_treshold делает соседями те точки, чье расстояние ниже, чем пороговое значение:

def connect\_by\_treshold(point\_list,treshold=0):

"""Связывание точек по трешхолду

Args:

point\_list ([Note,...]): Список точек

treshold (int, optional): трешхолд. Defaults to 0.

"""

print("start draw\_lines")

new\_points=np.copy(point\_list)

while len(new\_points):

point=new\_points[0]

new\_points=new\_points[1:]

for i in range(len(new\_points)):

if point.whether\_connected(new\_points[i]):

continue

elif ((point.point.x - new\_points[i].point.x)\*\*2+

(point.point.y - new\_points[i].point.y)\*\*2)\*\*0.5<=treshold:

point.new\_connection(new\_points[i])

Функция rename\_group\_of\_points\_lines переименовывает точки для последующей группировки:

def rename\_group\_of\_points\_lines(points):

"""Переименование группы

Args:

points ([Note]): список точек

"""

name=1

for i in points:

if i.rename(name):

name+=1

Функции draw\_points, draw\_point и draw\_line отрисовывают точки и линии соответственно. Причем линиям дается имя группы для последующего объединения уже на карте QGIS.

def draw\_points(point\_list, name\_point=False, layer\_name="persistent\_points"):

"""\_summary\_

Args:

point\_list ([Note],...): Список точек

name\_point (bool, optional): Чек поинт, будет ли использоваться групповое имя

или каждой точке дадут новое

"""

if name\_point:

for i in point\_list:

if to\_print:

i.print\_connect()

draw\_point(point\_check, project, i.point.x, i.point.y, layer\_name=layer\_name)

else:

point\_check=0

for i in point\_list:

if to\_print:

i.print\_connect()

draw\_point(point\_check,project, i.point.x, i.point.y, layer\_name=layer\_name)

point\_check+=1

def draw\_point(name,project,x=1723,y=-1456, layer\_name ="persistent\_points"):

"""Рисует точки

Args:

name (\_type\_): первый атрибут - id объекта

project (\_type\_): \_description\_

x (int, optional): х-координата. Defaults to 1723.

y (int, optional): y-координата. Defaults to -1456.

layer\_name (str, optional): Название поля, в которое вносится изменение. Defaults to "persistent\_points".

"""

layer = project.mapLayersByName(layer\_name)[0]

caps = layer.dataProvider().capabilities()

if caps & QgsVectorDataProvider.AddFeatures:

feat = QgsFeature(layer.fields())

#feat.setAttributes([0, 'hello'])

# Or set a single attribute by key or by index:

#feat.setAttribute('name', 'hello')

feat.setAttribute(0, name)

feat.setGeometry(QgsGeometry.fromPointXY(QgsPointXY(x, y)))

(res, outFeats) = layer.dataProvider().addFeatures([feat])

def draw\_line(name,project,x0=1723,y0=-1456,x1=1223,y1=-956,layer\_name="persistent\_lines"):

"""\_summary\_

Args:

name ("str"): id - объекта, является атрибутом

project (\_type\_): \_description\_

x0(int, optional):x-координата первого объекта. Defaults to 1723.

y0(int, optional):у-координата первого объекта.Defaults to -1456.

x1(int, optional):x-координата второго объекта. Defaults to 1223.

y1(int, optional):у-координата второго объекта. Defaults to -956.

layer\_name (str, optional): поле, в которое заносятся данные. Defaults to "persistent\_lines".

"""

layer = project.mapLayersByName(layer\_name)[0]

caps = layer.dataProvider().capabilities()

if caps & QgsVectorDataProvider.AddFeatures:

#print(layer.fields())

feat = QgsFeature(layer.fields())

feat.setAttribute(0, name)

feat.setGeometry(QgsGeometry.fromPolylineXY([QgsPointXY(x0, y0),

QgsPointXY(x1, y1)]))

(res, outFeats) = layer.dataProvider().addFeatures([feat])

def draw\_lines(points,layer\_name="persistent\_lines"):

"""Рисует все линии

Args:

points ([Note],...): список точек

threshold (int, optional): ограничение связи. Defaults to 500.

layer\_name (str, optional): имя записывающего слоя. Defaults to "persistent\_lines".

"""

print("start draw\_lines")

new\_points=np.copy(points)

x=-1

while len(new\_points):

x+=1

point=new\_points[0]

new\_points=new\_points[1:]

y=x

for i in range(len(new\_points)):

y+=1

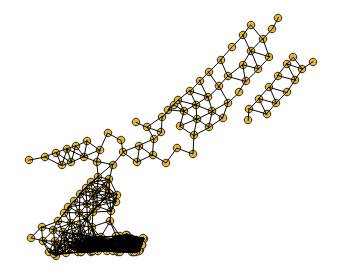
if point.whether\_connected(new\_points[i]):

draw\_line(name= point.name\_grup,project=project,x0= point.point.x,y0= point.point.y,x1=new\_points[i].point.x,y1= new\_points[i].point.y, layer\_name=layer\_name)

После отрисовки всех линий на карте идет объединение этих линий в абстрактные объекты:

myresult=processing.run("native:dissolve", { 'FIELD' : ['id'], 'INPUT' : QgsProcessingFeatureSourceDefinition('C:/Users/Ne\_ma/Documents/persistent\_lines.shp', selectedFeaturesOnly=False, featureLimit=-1, flags=QgsProcessingFeatureSourceDefinition.FlagOverrideDefaultGeometryCheck, geometryCheck=QgsFeatureRequest.GeometrySkipInvalid), 'OUTPUT' : 'TEMPORARY\_OUTPUT' })

QgsProject.instance().addMapLayer(myresult['OUTPUT'])



1. Объединенные линии и точки пересечения линий взятых из слоев «кладбища» – результат объединения точек по пороговому значению

3.4 Определение топологических отношений абстрактных фигур.

После определения абстрактной фигуры идет ее полигонизация. После создания слоя полигонов, каждому объекту данного слоя присуждается групповое имя, на основании которого оно будет объединено с соседними маленькими полигонами в 1 общий полигон.

list\_of\_poligons=[]

delet = True

for i in range(1,layer.featureCount()+1):

        layer.selectByExpression(f"""ID='{i}'""")

        poligon=processing.run("native:polygonize", {

                'INPUT' : QgsProcessingFeatureSourceDefinition(

                        layer.source(),

                        selectedFeaturesOnly=True,

                        featureLimit=-1,

                        #flags=QgsProcessingFeatureSourceDefinition.FlagCreateIndividualOutputPerInputFeature,

                        geometryCheck=QgsFeatureRequest.GeometryAbortOnInvalid),

                'KEEP\_FIELDS' : True, 'OUTPUT' : 'TEMPORARY\_OUTPUT' })

        QgsProject.instance().addMapLayer(poligon['OUTPUT'])

        for j in range(1,poligon['OUTPUT'].featureCount()+1):

                poligon['OUTPUT'].dataProvider().changeAttributeValues({ j :  { 0 : i } })

                # poligon['OUTPUT'].attributeValueChanged(j,0,i)

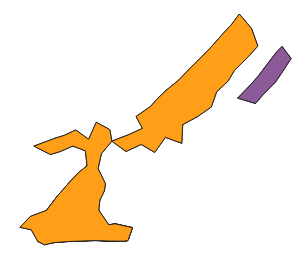
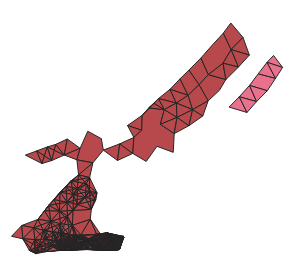
        list\_of\_poligons.append(processing.run("native:dissolve", {'INPUT':poligon['OUTPUT'].source(),

                                        'FIELD':[],

                                        'OUTPUT':'TEMPORARY\_OUTPUT'}))

        if delet:

            QgsProject.instance().removeMapLayer( poligon['OUTPUT'].id() )

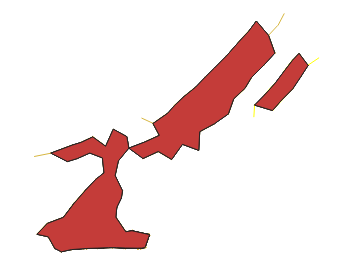


1. Полигонизация(слева) и объединение полигонов(справа) слоя объединенных ранее линий

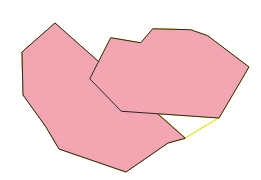
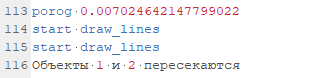
Построеный слой полигонов необходим для определения связей между объектами посредством функции пересечения, создающая новый элемент в случае пересечения или вхождения 2-х объектов.

В случае, если объекты пересекаются или один из них входит в другого, производится расчет площади нового пересеченного полигона и двух его родителей. Если площадь нового полигона меньше площади обоих родителей отдельно, то можно утверждать, что полигоны пересекаются, однако, если площадь нового полигона равна размеру одного из родителей, то можно утверждать, что 1 родительский полигон вложен во второй.

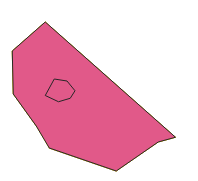
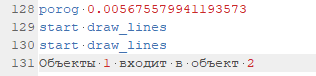
Результат плагина выведет взаимоотношения абстрактных полигонов в консоль и создаст 1 слой, на котором будут объединены все объединения полигонов. Лишние слои будут удалены.



1. Результат программы анализа расположения абстрактных пространственных структур (на основе векторных слоев кладбища)

1. Пример результата программы анализа расположения абстрактных пространственных структур

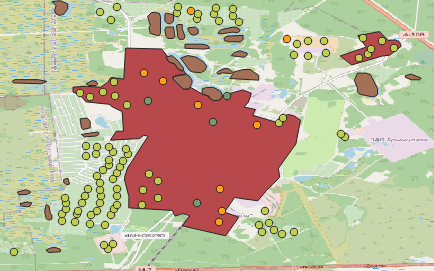
 

1. Пример результата программы анализа расположения абстрактных пространственных структур
2. **Проведение экспериментальных исследований**

**Эксперимент №1**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом.

Исходные данные: 5 векторных слоев, 2 из которых являются полигонами, а 3 из них – точками. Предполагаемое количество объектов: 40



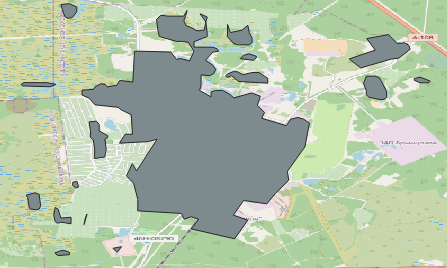
1. Исходные данные

Этапы работы:

1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 40 структур. В ходе данного этапа было выделено 17 объектов структуры. Такое резкое сокращение ожидаемого и действительного связано с наличием точек, которые не смогли примкнуть к другим объектам или точкам, всязи с чем они остались висеть мертвым грузом.



1. Абстрактные структуры
2. Анализ расположения абстрактных структур. В ходе данного этапа было построено 14 полигональных абстрактных объектов структуры. Уменьшение количества связано с наличием абстрактных линий, из которых невозможно построить полигоны. В результате анализа взаимоотношений не было выделено ни одного отношения типо «Входящий» или «Пересекающий». Даже не смотря на наличие видимого пересечения 2 разных слоев полигонов, программа не защитала его как «Пересекающийся» из-за малого количества выбранных объектов. Это в свою очередь привело к высокому диапозону поиска точек и поглощению всех 3 полигонов в 1 большой.

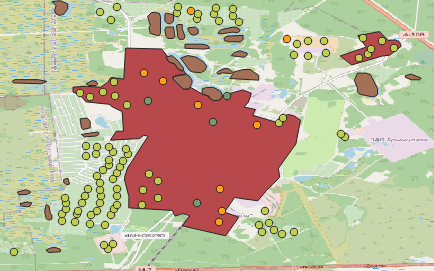


1. Результат алгоритма

**Эксперимент №2**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом. Сравнение результатов в зависимости от изменения предполагаемого количества объектов.

Исходные данные: 5 векторных слоев, 2 из которых являются полигонами, а 3 из них – точками. Предполагаемое количество объектов: 60



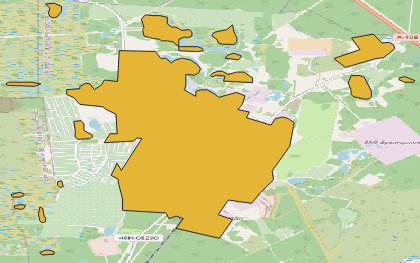
1. Исходные данные

Этапы работы:

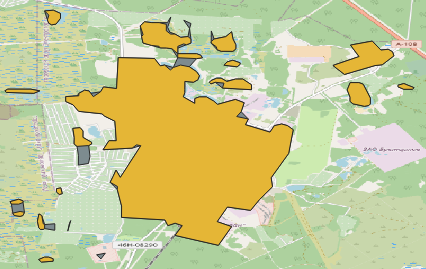
1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 60 структур. В ходе данного этапа было выделено 27 объектов структуры. При увеличении ожидаемого количества структур на 20, количество найденных объектов структур относительно прошлого исследования увеличилось на 10.



1. Абстрактные структуры
2. Анализ расположения абстрактных структур. В ходе данного этапа было также построено 14 полигональных абстрактных объектов структуры. В результате анализа взаимоотношений не было выделено ни одного отношения типо «Входящий» или «Пересекающий».



1. Результат алгоритма

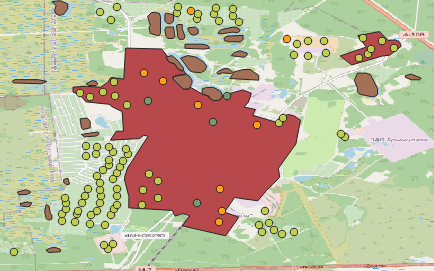


1. Сравнение алгоритма с результатами прошлого исследования

**Эксперимент №3**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом.

Исходные данные: 5 векторных слоев, 2 из которых являются полигонами, а 3 из них – точками. Предполагаемое количество объектов: 80



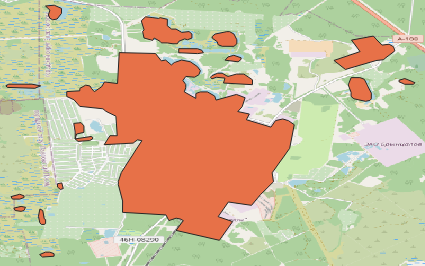
1. Исходные данные

Этапы работы:

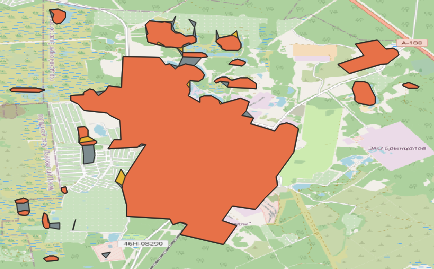
1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 80 структур. В ходе данного этапа было выделено 31 объектов структуры. При увеличении ожидаемого количества структур на 20, количество найденных объектов структур относительно прошлого исследования увеличилось лишь на 4.



1. Абстрактные структуры
2. Анализ расположения абстрактных структур. В ходе данного этапа было построено 16 полигональных абстрактных объектов структуры. В результате анализа взаимоотношений не было выделено ни одного отношения типо «Входящий» или «Пересекающий».



1. Результат алгоритма

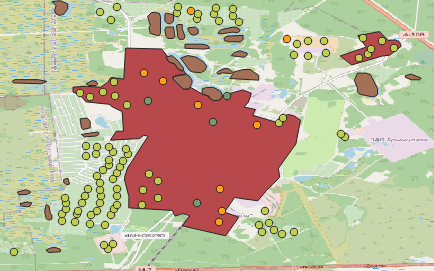


1. Сравнение алгоритма с результатами прошлых исследования

**Эксперимент №4**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом.

Исходные данные: 5 векторных слоев, 2 из которых являются полигонами, а 3 из них – точками. Предполагаемое количество объектов: 20



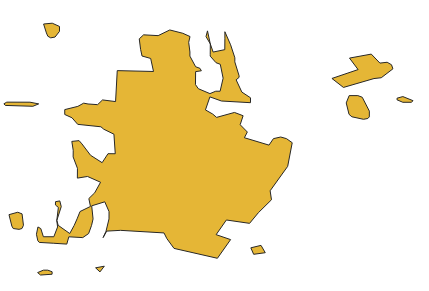
1. Исходные данные

Этапы работы:

1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 20 структур. В ходе данного этапа было выделено 16 объектов структуры. При уменьшении ожидаемого количества структур на 20, количество найденных объектов структур относительно первого исследования уменьшилось на 6.



1. Абстрактные структуры
2. Анализ расположения абстрактных структур. В ходе данного этапа было построено 8 полигональных абстрактных объектов структуры. В результате анализа взаимоотношений не было выделено ни одного отношения типо «Входящий» или «Пересекающий». В результате уменьшения предпологаемого количества объектов алгоритм поглощает ранее найденные объекты.

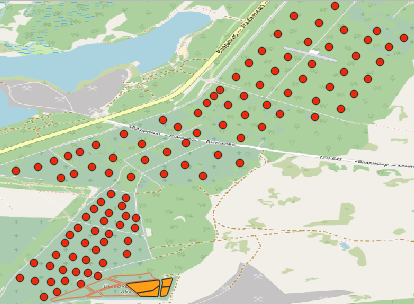


1. Результат алгоритма

**Эксперимент №5**

Цель: поиск и построение абстрактных структур.

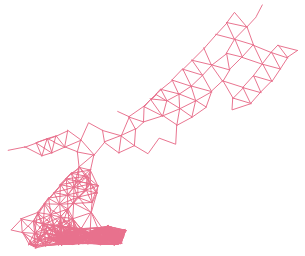
Исходные данные: 3 векторных слоев: слой точек, слой линий и слой полигонов. Предполагаемое количество объектов: 1



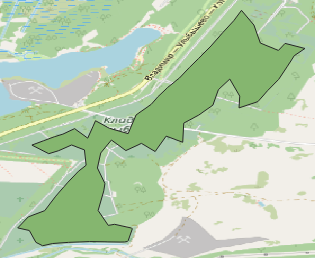
1. Исходные данные

Этапы работы:

1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 1 структуры. В ходе данного этапа было выделено 1 объектов структуры.



1. Абстрактные структуры
2. Построение абстрактной структуры. В ходе данного этапа было построено 1 полигональный абстрактный объект структуры. Алгоритм успешно справляется с постройкой полигональных абстрактных структур.

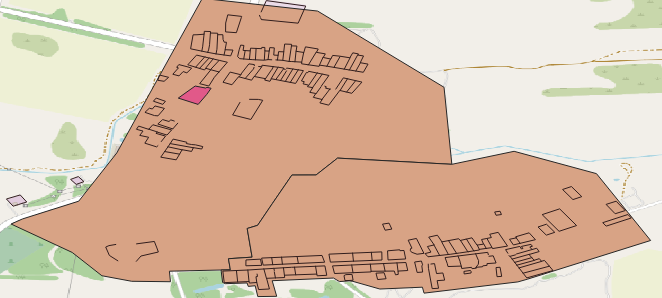


1. Результат алгоритма

**Эксперимент №6**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом.

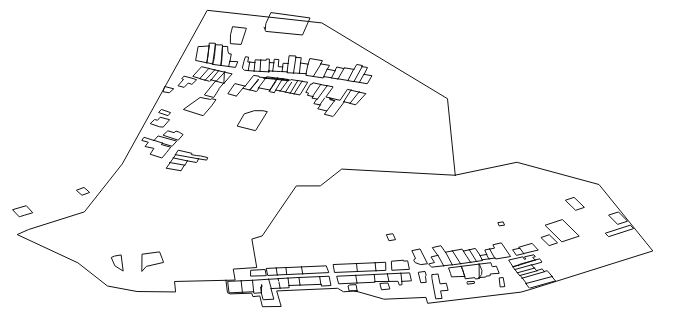
Исходные данные: 3 векторных слоев: слой линий и 2 слоя полигонов. Предполагаемое количество объектов: 30



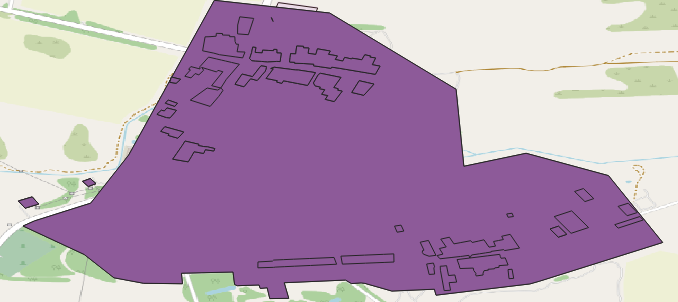
1. Исходные данные

Этапы работы:

1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 30 структуры. В ходе данного этапа было выделено 30 объектов структуры. В отличае от других опытов, в данном опыте количество выделенных объектов структуры совпадает с количеством предпологаемых структур. Это связано с упорядочностью структур (слой линий является «оградой» и в целом мало отличается от полигонов) и отсутствием точек.



1. Абстрактные структуры
2. Построение абстрактной структуры. В ходе данного этапа было построено 27 полигональный абстрактный объект структуры. В результате анализа было выделено 23 отношения типа «Входящий» и 1 отношение «Пересекающий».

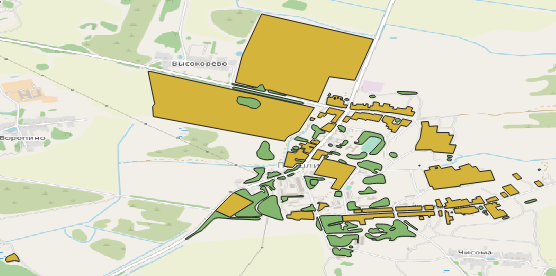


1. Результат алгоритма

**Эксперимент №7**

Цель: поиск абстрактных структур и взаимоотношений их друг с другом.

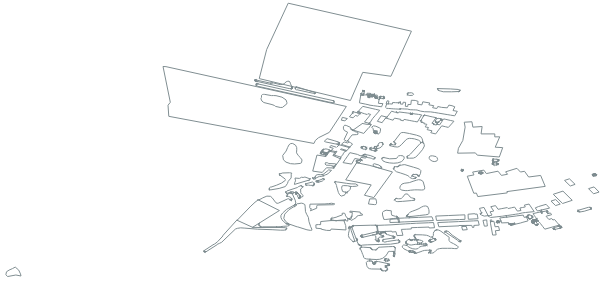
Исходные данные: 2 векторных полигональных слоев: Предполагаемое количество объектов: 60



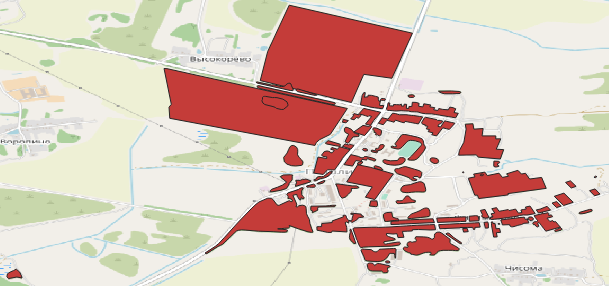
1. Исходные данные

Этапы работы:

1. Поиск абстрактных структур. В качестве вводных данных предполагается наличие 60 структуры. В ходе данного этапа было выделено 60 объектов структуры. Аналогично с экспериментом номер 6 количество абстрактных структур равно предполагемым так как анализ проводится с полигональными векторами.



1. Абстрактные структуры
2. Построение абстрактной структуры. В ходе данного этапа было построено 60 полигональный абстрактный объект структуры. В результате анализа было выделено 2 отношения типа «Входящий» и 2 отношение «Пересекающий».



1. Результат алгоритма

В результате проведенных опытов выяснилось, что алгоритм способен строить абстрактные фигуры. Персистентная гомология хорошо подходит для подобных задач. Однако алгоритмы, построенные на персистентной гомологии не подходят для последующего анализа расположения абстрактных структур из-за особенности «поглащения» ближайших структур, из-за чего отношение типа «Соприкосновение» попросту не смогут быть выявлены алгоритмом. Отношения типа «Пересечения» рискуют быть «съедены» если точки 2 абстрактных объектов распологаются слишком близко. Аналогичной ситуации находится и тип «Вложение», если находится слишком близко к краю.

Исходя из проведенных расчетов можно сделать вывод о том, что точность классификации системы равна приблизительно 54%. Данная цифра говорит о сложной структуре строения молочных желез, которое содержит в себе множество различных сегментов, требуемые тчательного анализа специалиста.

# ЗАКЛЮЧЕНИЕ

В данной пояснительной записке приведен процесс реализации алгоритма по анализу расположения абстрактных пространственных структур. Данный алгоритм позволяет распознать абстрактные структуры из набора пространственных объектов, состоящих из точек, линий и полигонов.

В рамках работы над проектом был проведен анализ предметной области, определены данные к проекту, а также предъявлены требования к разрабатываемому алгоритму. Далее была произведена разработка алгоритма, который конструирует сбстрактный объект из разных пространственных объектов и находит их взаимосвязи.

Алгоритм основан на технологии Persistent Homology, в частности на фильтрации Вьеториса-Рипса, с помощью которого алгоритм получал числа Бетти и на их основе конструировал абстрактную пространственную структуру.

В результате проведенных опытов выяснилось, что алгоритм способен строить абстрактные фигуры. Алгоритм успешно строит абстрактные пространственные структуры, однако алгоритм не подходят для анализа расположения абстрактных структур из-за его особенности «поглащения» ближайших структур. Так отношения типа «Соприкосновение» попросту не могут быть выявлены и будут «съедены» другим объектом. Отношения типа «Пересечения» и «Вложения» также рискуют быть «съедены» если точки 2 абстрактных объектов распологаются слишком близко.

Итогом работы является сбор статистики и анализ полученных данных в процессе исследования программы для выявления точности работы алгоритма.

# СПИСОК ЛИТЕРАТУРЫ

1. Персистентные гомологии. //URL:http://intsysjournal.org/pdfs/23-2/ kushnar.pdf (дата обращения: 10.03.21).
2. giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration, Tauzin et al, J. Mach. Learn. Резолюция 22.39 (2021): 1-6.
3. Лялин М.С. Моделирование и анализ топологических отношений абстрактных структур на карте. 2020г.
4. Число Бетти [электронный ресурс] //URL:http://poivs.tsput.ru/ ru/Math/ Geometry/Topology/TopologicalInvariants/BettyNumber
5. Сенюков Л.В. Реализация вероятностного алгоритма вычислительной топологии [электронный ресурс] //URL: <https://se.math.spbu.ru/thesis/> texts/ Senjukov\_Leonid\_Vladimirovich\_Bachelor\_Report\_2020\_text.pdf