Отчёт

Постановка задачи: написать программу, генерирующую неструктурированную сетку(обтекающую архитектурные препятствия), с фотографии

Подзадачи

1. Написать алгоритм кластеризирующий отдельные архитектурные препятствия с фотографии
2. Написать алгоритм, возвращающий граничные точки полигона (препятствия)
3. Написать алгоритм, принимающий граничные точки полигонов и генерирующий сетку
4. Валидация сетки на использование её при решении методов конечных объемов

\*\* Задачи Стаса: 3, 4; задачи Елданы 1, 2, 4

## 1.1 Для кластеризации использовался метод DBSCAN: Density-Based Spatial Clustering of Applications with Noise. DBSCAN определяет плотность в терминах точек данных в пределах указанного радиуса (ϵ) вокруг точки данных.

## Точка данных считается "основной точкой" (core point), если по крайней мере заданное количество точек данных (MinPts) находятся в ее ϵ-окрестности.

## Алгоритм начинает с выбора произвольной точки данных. Если точка является основной точкой, формируется кластер путем рекурсивного добавления всех достижимых основных точек в пределах ϵ-окрестности.

## Точки данных, которые не являются основными точками или не могут быть достигнуты напрямую из основной точки, рассматриваются как шум или выбросы.

## 1.2 Код

## from sklearn.cluster import DBSCAN

## dbscan = DBSCAN(eps=1e-3, min\_samples=3)

## labels = dbscan.fit\_predict(coor\_obst)

## 1.3 Результат

## 2.2 Далее обрабатывался контур препятствий.

## 2.2.1 Перевод пиксельного контура в контур пикселей и граней

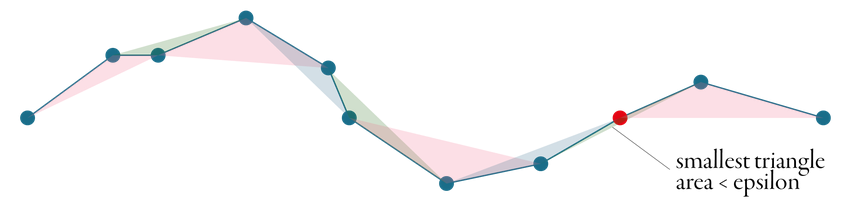
## Изображение выглядит как круг, диаграмма Автоматически созданное описание

2.2.2 Применить алгоритм упрощения линий (The Visvalingam-Whyatt algorithm)

Изображение выглядит как диаграмма, зарисовка

Автоматически созданное описание

Visvalingam-Whyatt, на первом этапе, генерирует треугольники между точками, как показано на рисунке ниже.

Затем он выявляет наименьший из этих треугольников и проверяет, является ли его площадь меньшей или большей, чем значение эпсилон. Если она меньше, точка, связанная с этим треугольником, отбрасывается, и процесс повторяется - генерируются новые треугольники, выявляется наименьший, проводится проверка, и так далее. Алгоритм завершается, когда все сгенерированные треугольники больше, чем эпсилон.

2.3 Код

temp = np.tile(0, res.shape)

for i in range(len(res)):

for j in range(1, len(res[i])-1):

if (res[i][j-1] == 1 and res[i][j+1]==1):

temp[i][j] = 0

else:

temp[i][j] = res[i][j]

for i in range(1, len(res)-1):

for j in range(len(res[i])):

if (res[i-1][j] == 1 and res[i+1][j]==1):

temp[i][j] = 0

else:

temp[i][j] = temp[i][j]

Изображение выглядит как снимок экрана, черный, диаграмма, астрономия

Автоматически созданное описание

X\_bndr = list()

Y\_bndr = list()

for i in range(len(temp)):

for j in range(len(temp[i])):

if temp[i][j] == 1:

X\_bndr.append(j)

Y\_bndr.append(i)

X\_bndr = np.array(X\_bndr)

Y\_bndr = np.array(Y\_bndr)

X\_bndr=np.array(X\_bndr)/len\_

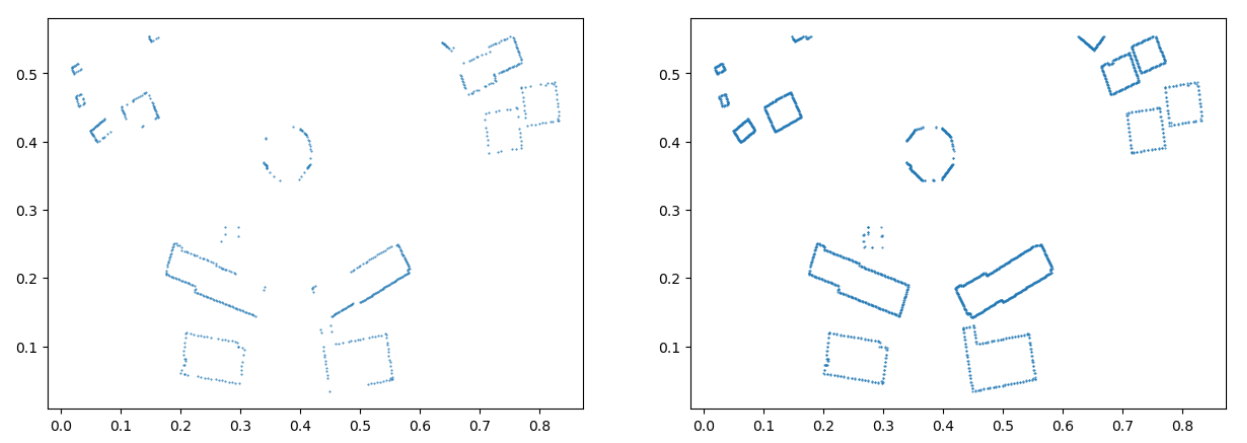
Y\_bndr=np.array(Y\_bndr)/len\_

coor\_bndr = np.stack([X\_bndr, Y\_bndr]).T

import visvalingamwyatt as vw

simplifier = vw.Simplifier(coor\_bndr)

simplified =simplifier.simplify(ratio=0.3)



3.1 Для создания сеток используется библиотека meshpy в Python, готовый генератор сеток. Был разработан класс MeshBuilder, для функционального удобства пользователя.

Код:

from \_\_future\_\_ import division

from \_\_future\_\_ import absolute\_import

%matplotlib inline

import meshpy.triangle as triangle

import numpy as np

import numpy.linalg as la

from six.moves import range

import matplotlib.pyplot as plt

from scipy.spatial import Delaunay

areas = []

def needs\_refinement(vertices, area):

# print(np.array(vertices))

bary = np.sum(np.array(vertices), axis=0)/3

max\_area = 0.001 + (la.norm(bary, np.inf))\*0.01

# print(area, max\_area)

areas.append(area)

return bool(area > max\_area)

def needs\_refinement2(vertices, area):

# areas.append(area)

return bool(area <= 0.0000015)

def round\_trip\_connect(start, end):

return [(i, i+1) for i in range(start, end)] + [(end, start)]

class MeshBuilder:

def \_\_init\_\_(self, max\_volume):

self.points, self.facets, self.last, self.holes = [], [], 0, []

self.max\_volume = max\_volume

self.inner = []

def addBuilding(self, nodes):

self.points.extend(nodes)

self.facets.extend(round\_trip\_connect(self.last, len(self.points) - 1))

self.last = len(self.points)

self.holes.append(((nodes[0][0] + nodes[1][0] + nodes[2][0] + nodes[3][0])/4, (nodes[0][1] + nodes[1][1]+nodes[2][1]+nodes[3][1])/4))

height = round(abs(nodes[1][1] - nodes[2][1]), 2)

width = round(abs(nodes[0][0] - nodes[1][0]), 2)

for i in range(50):

for j in range(50):

self.inner.append([nodes[0][0] + i \*(width/50) + 0.001, nodes[0][1] + j \* (height/50)+ 0.001])

def setFrame(self, borders):

self.points.extend(borders)

self.facets.extend(round\_trip\_connect(self.last, len(self.points) - 1))

self.last = len(self.points)

def clear(self):

self.points, self.facets, self.last, self.holes, self.inner = [], [], 0, [], []

def build(self):

info = triangle.MeshInfo()

info.set\_points(self.points)

info.set\_holes(self.holes)

info.set\_facets(self.facets)

self.mesh = triangle.build(info,max\_volume=self.max\_volume,

min\_angle = 33, volume\_constraints = True)#,min\_angle=30, refinement\_func=needs\_refinement)#, volume\_constraints=True)

self.mesh\_points = np.array(self.mesh.points)

# self.mesh\_tris = np.array(self.mesh.elements)

# self.mesh\_facets = np.array(self.mesh.facets)

tri = Delaunay(self.mesh\_points, incremental = True)

print(tri.points == self.mesh\_points)

self.mesh\_tris = np.array(tri.vertices)

p = tri.find\_simplex(self.inner)

self.mesh = np.delete(tri.simplices, np.unique(p), 0)

def show(self):

plt.figure(figsize=(20, 20))

plt.triplot(self.mesh\_points[:, :1].flatten(), self.mesh\_points[:, 1:2].flatten(), self.mesh)

def del\_write\_dat(self,f):

if type(f) is str:

f = open("coord.txt", 'w')

mesh\_points = np.array(self.mesh\_points)

mesh\_tris = np.array(self.mesh\_tris)

p\_count = mesh\_points.shape[0]

t\_count = mesh\_tris.shape[0]

f.write(str(p\_count))

f.write('\n')

for i in range(p\_count):

f.write(f'{i + 1} ')

f.write('{:0.15e}'.format(mesh\_points[i][0]))

f.write(' ')

f.write('{:0.15e}'.format(mesh\_points[i][1]))

f.write('\n')

f.close()

f = open("links.txt", 'w')

f.write(str(t\_count))

f.write('\n')

for i in range(t\_count):

f.write(f'{i + 1} 0 tri ')

f.write(str(mesh\_tris[i][0] + 1))

f.write(' ')

f.write(str(mesh\_tris[i][1] + 1))

f.write(' ')

f.write(str(mesh\_tris[i][2] + 1))

f.write('\n')

Класс поддерживает методы:

1) addBuilding(self, nodes) - разметка препятствия, необходимо в массив nodes передать tuple координат граничных точек препятствий.

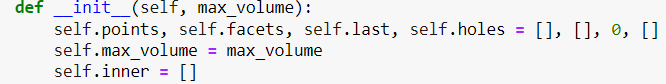
2) setFrame(self, borders) - разметка общих границ, т.е. основной домейн.

3) clear(self) - удаление нынешних настроек(расстановок граничных точек)

4) build(self) - построение сетки

5) show(self) - визуализация сетки

6) del\_write\_dat (self, f) - запись координат и связей между узлами в два файла (“coord.txt”, “links.txt”)

Инициализация обьекта класса запрашивает параметр max\_volume, обозначающий максимальную допустимую площадь треугольников сетки. 

3.2 Тестирование кода:

guy = MeshBuilder(2e-4)

borders = []

sep = 0.02

for i in range(51):

borders.append((0, round(i \* sep, 2)))

for i in range(1, 51):

borders.append((round(i\*sep, 2), 1))

for i in range(1, 51):

borders.append((1, round(1 - i\*sep, 2)))

for i in range(1, 50):

borders.append((round(1 - i\*sep, 2), 0))

# guy.setFrame([(0, 0), (0, 1), (1, 1), (1, 0)])

guy.setFrame(borders)

# guy.addBuilding([(0.7, 0.7), (0.7, 0.72), (0.7, 0.74), (0.7, 0.76), (0.72, 0.76), (0.74, 0.76),(0.76, 0.76),(0.76, 0.74),(0.76, 0.72),(0.76, 0.7),(0.74, 0.7),(0.72, 0.7)])

# guy.addBuilding([(0.2, 0.2), (0.3, 0.2), (0.3, 0.3), (0.2, 0.3)])

guy.addBuilding([(0.4, 0.4), (0.6, 0.4), (0.6, 0.6), (0.4, 0.6)])

guy.addBuilding([(0.1, 0.1), (0.3, 0.1), (0.3, 0.3), (0.1, 0.3)])

# guy.addBuilding([(0.75, 0.75), (0.95, 0.75), (0.95, 0.95), (0.75, 0.95)])

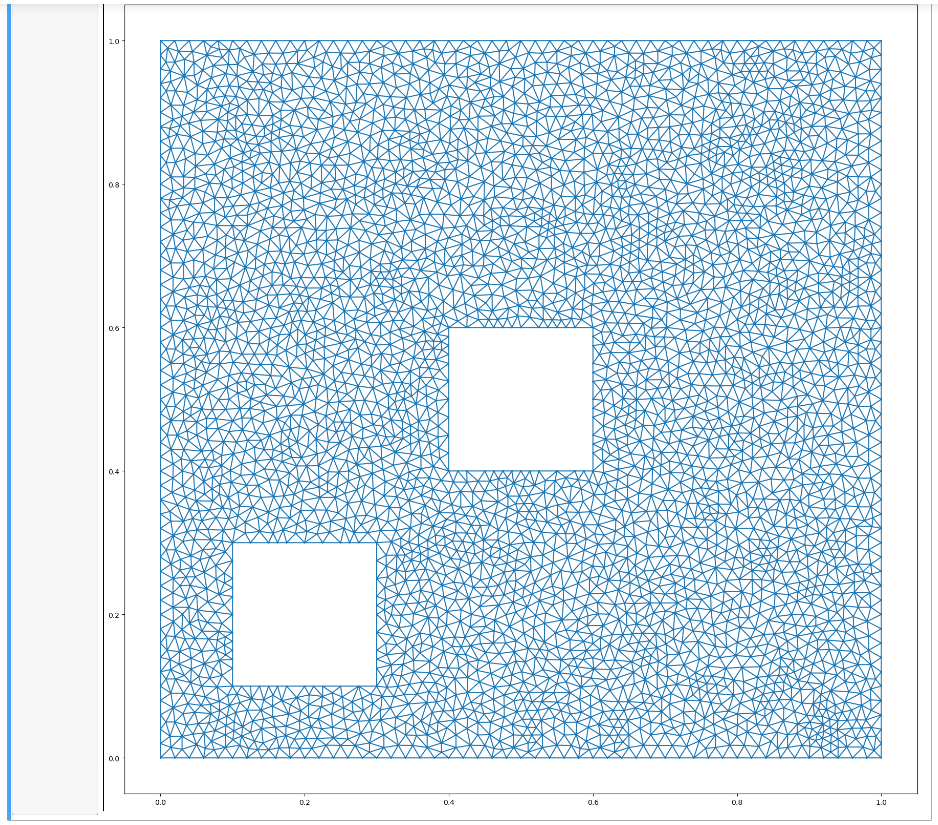
# guy.addBuilding([(0.1, 0.8), (0.3, 0.8), (0.3, 0.9), (0.1, 0.9)])

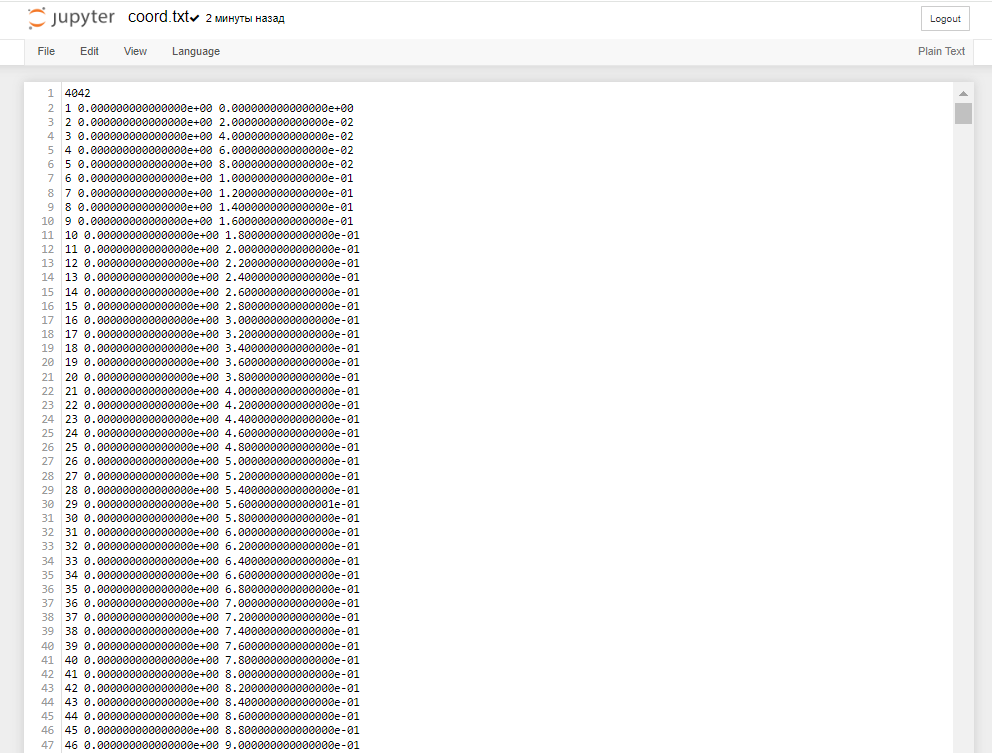
guy.build()

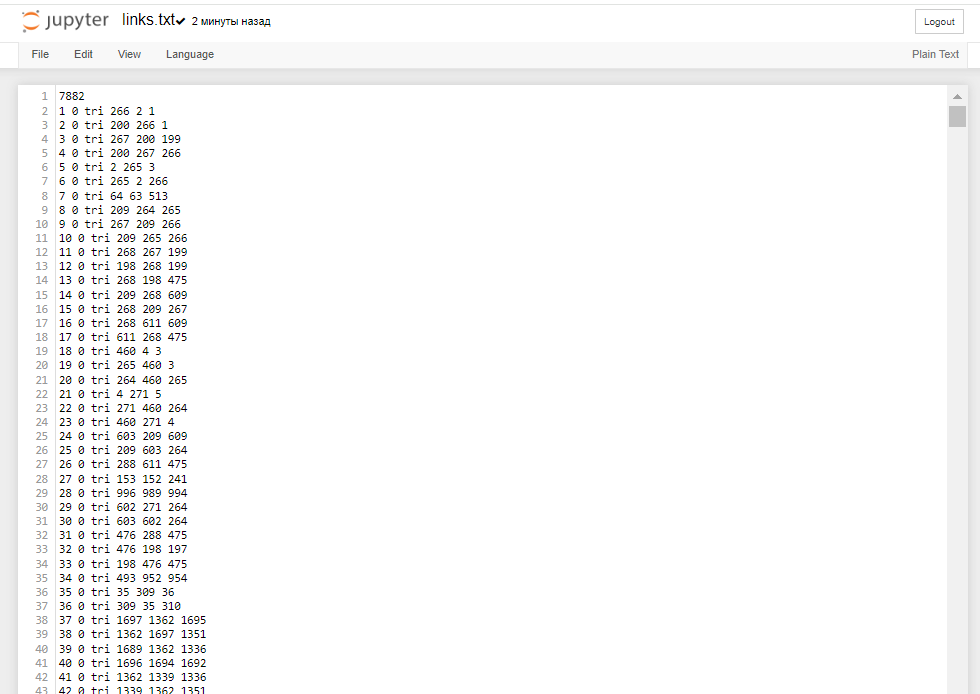
guy.show()

guy.del\_write\_dat("aba.txt")

3.3 Вывод:







Однако, у данного генератора сеток есть несовершенства, большинство генерируемых сеток не пригодны для численного моделирования, причем, чем более сложная геометрия сетки, тем меньше вероятность ее пригодности.

4.1 При запуске сетке для решения Навье Стокса не все сетки являются валидными.

