Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Московский политехнический университет»

**факультет информационных технологий**

**Кафедра СМАРТ-технологий**

Дисциплина: Разработка систем сбора и обработки данных

Отчёт по лабораторной работе №2

«Обработка структурированных пространственных данных и отслеживание объектов»

Работа выполнена (ФИО):

Ф И О

Научный руководитель

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

Ф И О

**Цель работы:**

Разработать алгоритм считывания и визуализации потоковых данных в виде динамически обновляемого облака точек многолучевого 3D-лидара

**Задачи:**

* Подготовить приложение считывания данных с 3D-лидара, сохраненных в файле потоковом формате и вычисления облака точек;

• Разработать функцию динамического обновления данных в заданной структуре (объекте) для хранения данных облака точек;

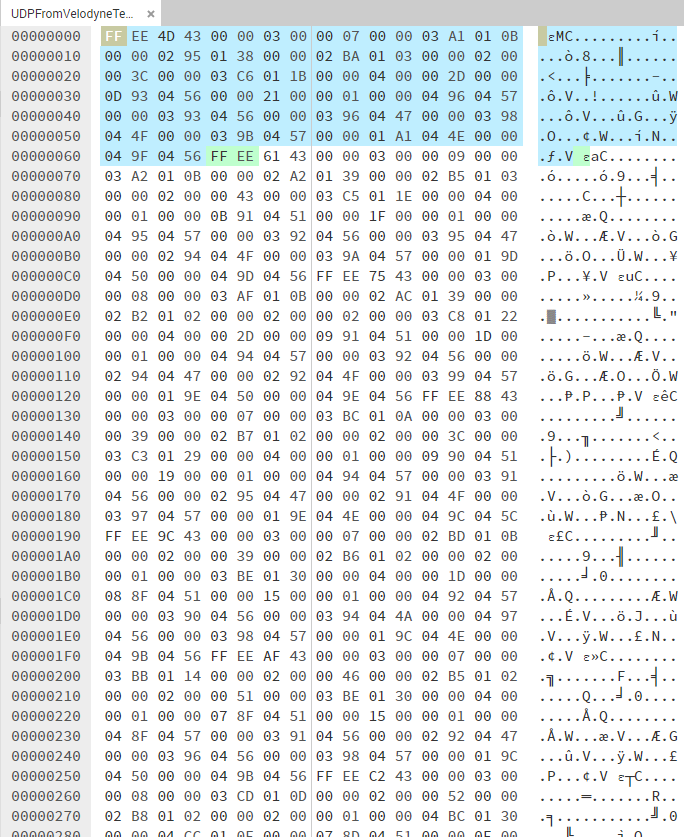
• Разработать функцию покадровой визуализации облака точек в изометрической проекции и в режиме «вид сверху» с заданным (настраиваемым) фреймрейтом (частотой);

• Разработать функцию фильтрации данных с использованием плоскостей отсечения (куб интереса);

• Разработать функцию детектирования объектов методом кластерного анализа;

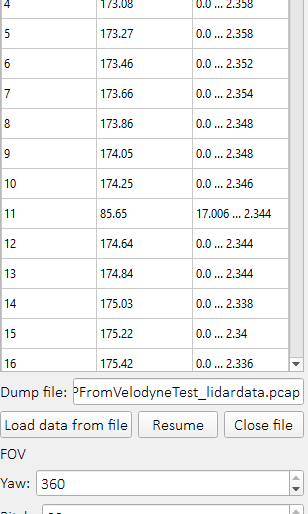
• Разработать метод отслеживания «коридора проходимости» для прямолинейного движения.**Ход работы**

В качестве источника данных выступает дамп с 3D-лидара имеющий следующую структуру:



Дамп последовательно передаёт пакеты по 100 байт, содержащие измерения дистанции с 32-ух лазеров. Каждый пакет имеет свой угол рыскания. Т.к. лидар вращается, последовательно передаются измерения длин во всём угле обзора 360 градусов.

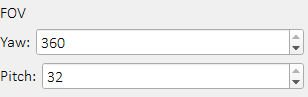
Для начала чтения файла, на форме необходимо указать расположение дампа и нажать кнопку Load data from file. В процессе, чтение можно поставить на паузу и продолжить с помощью кнопки Pause / Resume. Также, чтение можно оставить и закрыть файл, нажав на кнопку Close file.



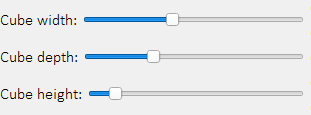
В процессе чтения данные в реальном времени отображаются в таблице. Первая колонка – номер пакета, вторая – угол по оси рыскания, третья – первое и последнее число расстояния из пакета.

Для отображения существуют 2 ограничения:

* Поле зрения (настраивается спин боксами в разделе FOV. Можно ограничить угол по оси рыскания или тангажа)



* “Куб” видимости. По заданию – 6 плоскостей, образующих параллелепипед, внутри которого отображаются данные. Настраивается 3-мя ползунками на форме:



Файл парсится в буфер длиной 100 байтов. Как только обнаруживается пара символов EE-FF происходит обработка буфера. В начале вычисляется угол лидара по оси рыскания:

laser\_yaw = (int(bytes\_buffer[1] & 0xFF) << 8) | int(bytes\_buffer[0] & 0xFF)

Далее, для каждой тройки байтов (32 луча) считается дистанция

distance = (int(bytes\_buffer[byte\_position + 1] & 0xFF) << 8) | int(  
 bytes\_buffer[byte\_position] & 0xFF)  
distance \*= 0.002

И, соответственно, угол тангажа, по указанным в задании значениям

self.laser\_angles = [-30.67, -9.33, -29.33, -8.00, -28.00, -6.66, -26.66, -5.33, -25.33, -4.00, -24.00, -2.67,  
 -22.67, -1.33, -21.33, 0.00, -20.00, 1.33, -18.67, 2.67, -17.33, 4.00, -16.00, 5.33,  
 -14.67, 6.67, -13.33, 8.00, -12.00, 9.33, -10.67, 10.67]

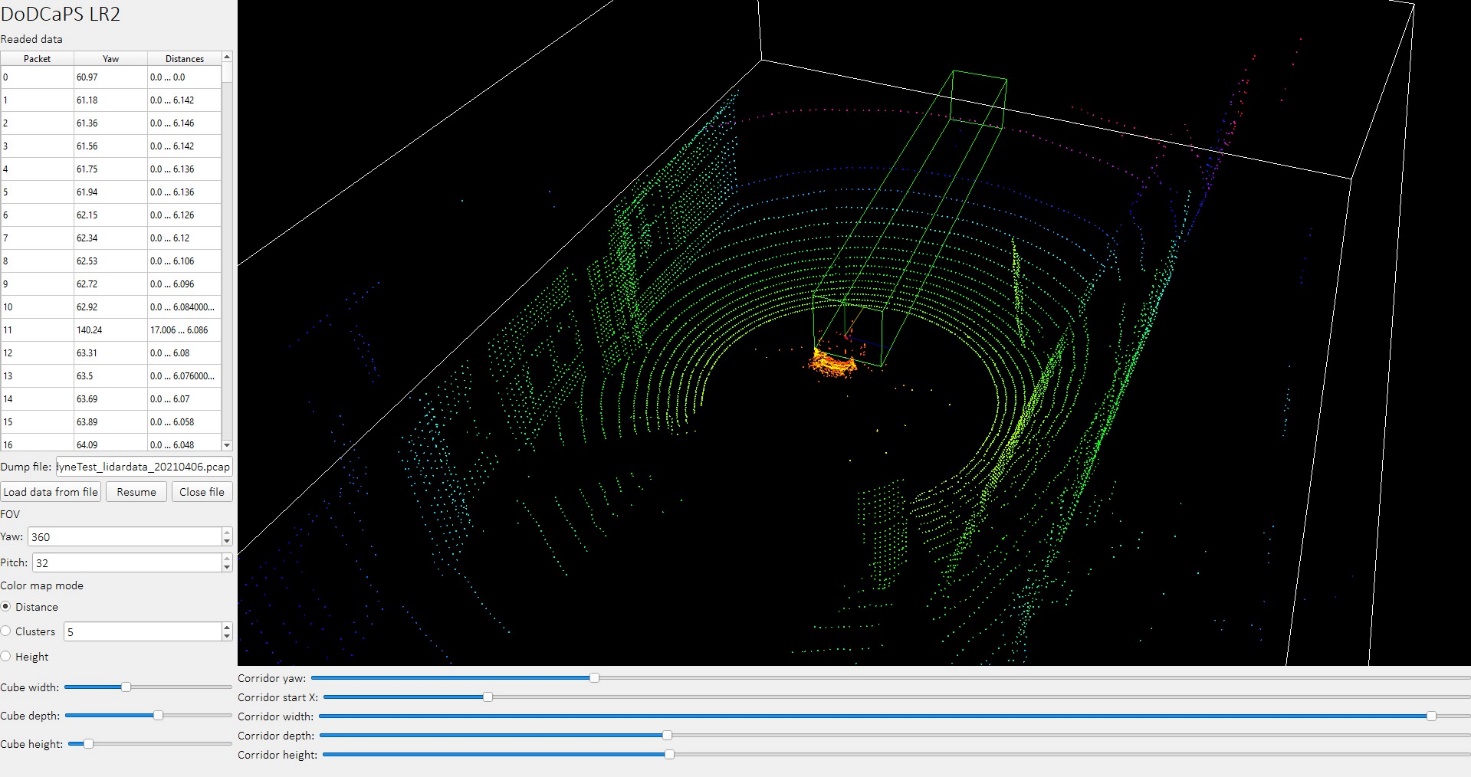
После, дистанция и 2 угла передаются на вход функции поворота, которая вычисляет 3 матрицы поворота (крен, тангаж, рыскание) и применяют к дистанции. Что даёт на выходе точке в декартовой системе координат:

def rotate3d(pos, roll, pitch, yaw):  
 *"""  
 Rotates point on angle  
 :param pos: (X, Y, Z) tuple position  
 :param roll: Roll angle in radians  
 :param pitch: Pitch angle in radians  
 :param yaw: Yaw angle in radians  
 :return:  
 """* roll\_sin = math.sin(roll)  
 roll\_cos = math.cos(roll)  
 pitch\_sin = math.sin(pitch)  
 pitch\_cos = math.cos(pitch)  
 yaw\_sin = math.sin(yaw)  
 yaw\_cos = math.cos(yaw)  
  
 mat\_roll = np.array([[1, 0, 0], [0, roll\_cos, -roll\_sin], [0, roll\_sin, roll\_cos]])  
 mat\_pitch = np.array([[pitch\_cos, 0, pitch\_sin], [0, 1, 0], [-pitch\_sin, 0, pitch\_cos]])  
 mat\_yaw = np.array([[yaw\_cos, -yaw\_sin, 0], [yaw\_sin, yaw\_cos, 0], [0, 0, 1]])  
  
 pos = np.array(pos)  
  
 pos = np.dot(pos, mat\_roll)  
 pos = np.dot(pos, mat\_pitch)  
 pos = np.dot(pos, mat\_yaw)  
  
 return pos

Перед заполнением массива координат, происходит проверка на вхождение в заданный «куб» и угол обзора:

*# Check cube and FOV*cube\_w = self.slider\_cube\_w.value() / 2  
cube\_h = self.slider\_cube\_h.value() / 2  
cube\_d = self.slider\_cube\_d.value() / 2  
if not (-cube\_w < rotated\_point[0] < cube\_w and  
 -cube\_h < rotated\_point[1] < cube\_h and  
 -cube\_d < rotated\_point[2] < cube\_d and  
 abs(laser\_pitch) <= self.spin\_fov\_pitch.value() and  
 laser\_yaw <= self.spin\_fov\_yaw.value()):  
 rotated\_point = [0, 0, 0]  
 distance = 0

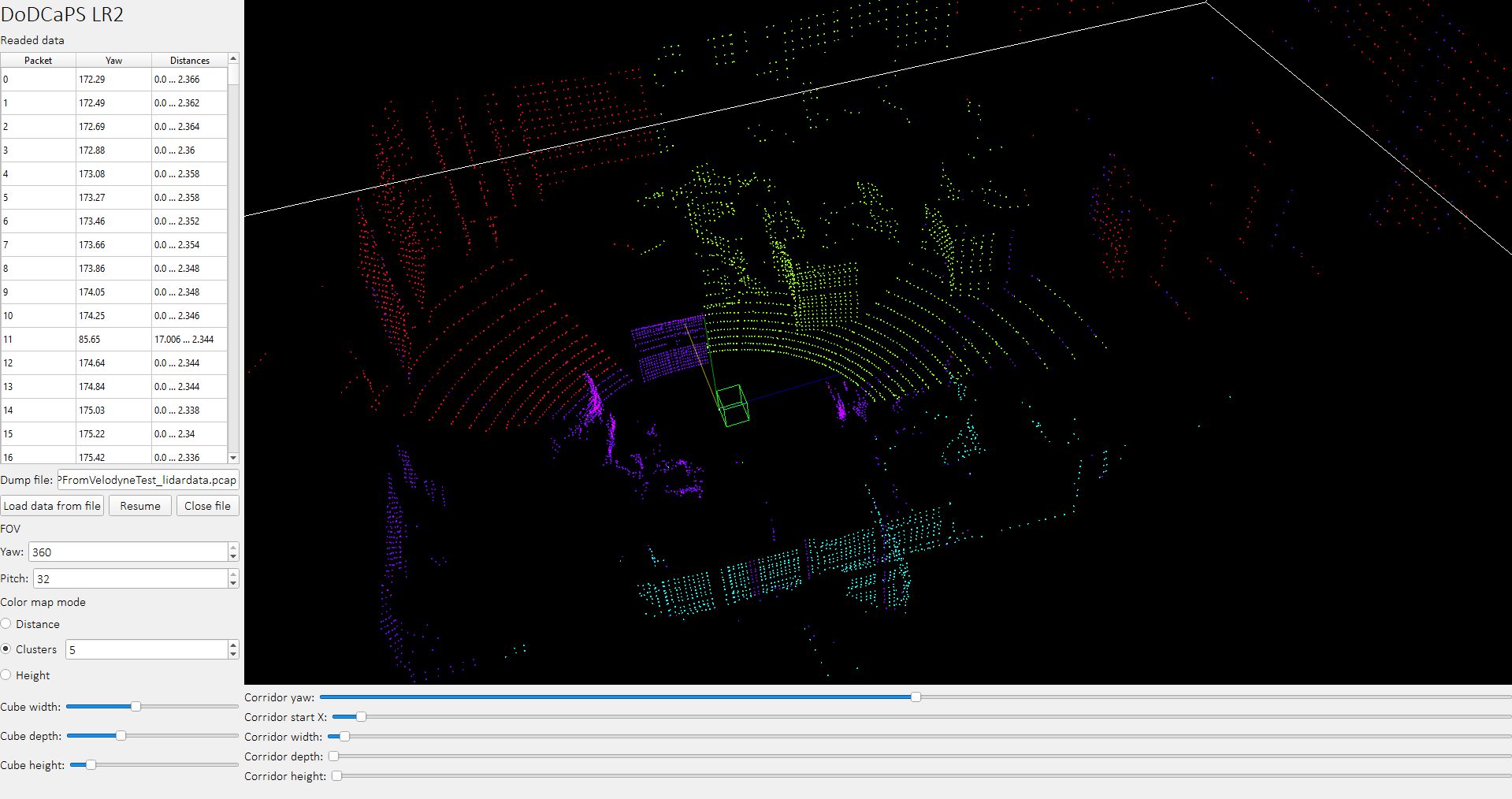
В результате чего, после прохождения нескольких таких пакетов, образуется облако точек:



Также, по заданию был реализован «коридор проходимости». Для настройки его параметров есть ползунки на форме.

По мере отрисовки точек проверяется количество точек внутри этого коридора, чем их более, чем более красным цветом он отрисовывается, сигнализируя о том, что необходимо остановиться чтобы не врезаться в препятствие.

Согласно заданию, цвета точек отображают расстояние до объектов. Помимо этого, имеется возможности отображать цвета по кластерам K-means. Для чего на форме имеются соответствующие переключатели и возможность указать количество кластеров.



**Вывод**

В ходе данной работы было создано приложение, реализующее алгоритм считывания и визуализации потоковых данных с 3D-лидара

**Исходный код**

*"""  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
"""*import math  
import os  
import sys  
import threading  
import time  
  
import matplotlib.pyplot as plt  
import numpy as np  
import pyqtgraph.opengl as gl  
from PyQt5 import uic, QtWidgets, QtCore  
from PyQt5.QtWidgets import QApplication, QMainWindow, QTableWidgetItem  
from sklearn.cluster import KMeans  
  
  
def rotate3d(pos, roll, pitch, yaw):  
 *"""  
 Rotates point on angle  
 :param pos: (X, Y, Z) tuple position  
 :param roll: Roll angle in radians  
 :param pitch: Pitch angle in radians  
 :param yaw: Yaw angle in radians  
 :return:  
 """* roll\_sin = math.sin(roll)  
 roll\_cos = math.cos(roll)  
 pitch\_sin = math.sin(pitch)  
 pitch\_cos = math.cos(pitch)  
 yaw\_sin = math.sin(yaw)  
 yaw\_cos = math.cos(yaw)  
  
 mat\_roll = np.array([[1, 0, 0], [0, roll\_cos, -roll\_sin], [0, roll\_sin, roll\_cos]])  
 mat\_pitch = np.array([[pitch\_cos, 0, pitch\_sin], [0, 1, 0], [-pitch\_sin, 0, pitch\_cos]])  
 mat\_yaw = np.array([[yaw\_cos, -yaw\_sin, 0], [yaw\_sin, yaw\_cos, 0], [0, 0, 1]])  
  
 pos = np.array(pos)  
  
 pos = np.dot(pos, mat\_roll)  
 pos = np.dot(pos, mat\_pitch)  
 pos = np.dot(pos, mat\_yaw)  
  
 return pos  
  
  
def inside\_test(points, cube3d):  
 *"""  
 Checks the location of points relative to the cube  
 Code from https://stackoverflow.com/a/53559963  
 :param points: array of points with shape (N, 3)  
 :param cube3d: numpy array of the shape (8,3) with coordinates in the clockwise order.  
 first the bottom plane is considered then the top one  
 :return: indices of the points array which are outside the cube3d  
 """* b1, b2, b3, b4, t1, t2, t3, t4 = cube3d  
  
 dir1 = (t1 - b1)  
 size1 = np.linalg.norm(dir1)  
 dir1 = dir1 / size1  
  
 dir2 = (b2 - b1)  
 size2 = np.linalg.norm(dir2)  
 dir2 = dir2 / size2  
  
 dir3 = (b4 - b1)  
 size3 = np.linalg.norm(dir3)  
 dir3 = dir3 / size3  
  
 cube3d\_center = (b1 + t3) / 2.0  
  
 dir\_vec = points - cube3d\_center  
  
 res1 = np.where((np.absolute(np.dot(dir\_vec, dir1)) \* 2) > size1)[0]  
 res2 = np.where((np.absolute(np.dot(dir\_vec, dir2)) \* 2) > size2)[0]  
 res3 = np.where((np.absolute(np.dot(dir\_vec, dir3)) \* 2) > size3)[0]  
  
 return list(set().union(res1, res2, res3))  
  
  
def list\_direction(data):  
 *"""  
 Determines whether an array is increasing or decreasing  
 :param data:  
 :return:  
 """* inc\_points = 0  
 dec\_points = 0  
 for i in range(len(data)):  
 average = sum(data[i:]) / len(data[i:])  
 if data[i] < average:  
 inc\_points += 1  
 elif data[i] > average:  
 dec\_points += 1  
  
 if inc\_points > dec\_points:  
 return 1  
 elif dec\_points > inc\_points:  
 return 0  
 else:  
 return None  
  
  
class Window(QMainWindow):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 *# Load GUI file* uic.loadUi(**'LR2.ui'**, self)  
  
 *# Laser angles* self.laser\_angles = [-30.67, -9.33, -29.33, -8.00, -28.00, -6.66, -26.66, -5.33, -25.33, -4.00, -24.00, -2.67,  
 -22.67, -1.33, -21.33, 0.00, -20.00, 1.33, -18.67, 2.67, -17.33, 4.00, -16.00, 5.33,  
 -14.67, 6.67, -13.33, 8.00, -12.00, 9.33, -10.67, 10.67]  
  
 *# System variables* self.dump\_file = None  
 self.reader\_running = False  
 self.dump\_paused = False  
 self.points = []  
 self.distances = []  
 self.points\_surface = gl.GLScatterPlotItem(pos=np.array([[0, 0, 0]]))  
 self.cube\_lines = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
 self.corridor\_lines = gl.GLLinePlotItem(pos=np.array([[0, 0, 0], [0, 0, 0]]))  
 self.estimator = KMeans(n\_clusters=3)  
  
 *# Connect GUI controls* self.btn\_load\_data.clicked.connect(self.load\_data)  
 self.btn\_stop\_reading.clicked.connect(self.stop\_reading)  
 self.btn\_pause.clicked.connect(self.pause)  
 self.plot\_timer = QtCore.QTimer()  
 self.plot\_timer.timeout.connect(self.update\_opengl)  
 self.plot\_timer.start(100)  
  
 *# Initialize table* self.init\_tables()  
  
 *# Initialize openGL view* self.init\_opengl()  
  
 *# Show GUI* self.show()  
  
 def init\_tables(self):  
 *"""  
 Initializes table of packets and setup table (whitelist table)  
 :return:  
 """* self.points\_table.setColumnCount(3)  
 self.points\_table.verticalHeader().setVisible(False)  
 self.points\_table.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)  
 self.points\_table.setHorizontalHeaderItem(0, QtWidgets.QTableWidgetItem(**'Packet'**))  
 self.points\_table.setHorizontalHeaderItem(1, QtWidgets.QTableWidgetItem(**'Yaw'**))  
 self.points\_table.setHorizontalHeaderItem(2, QtWidgets.QTableWidgetItem(**'Distances'**))  
 header = self.points\_table.horizontalHeader()  
 header.setSectionResizeMode(0, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(1, QtWidgets.QHeaderView.Stretch)  
 header.setSectionResizeMode(2, QtWidgets.QHeaderView.Stretch)  
  
 def init\_opengl(self):  
 *"""  
 Initializes charts  
 :return:  
 """* self.openGLWidget.addItem(gl.GLAxisItem())  
 self.openGLWidget.addItem(self.points\_surface)  
 self.openGLWidget.addItem(self.cube\_lines)  
 self.openGLWidget.addItem(self.corridor\_lines)  
  
 def update\_opengl(self):  
 *"""  
 Draws points over openGL view  
 :return:  
 """* if len(self.points) > 0:  
  
 *# Height color map* color\_data = np.array([item[2] for item in self.points])  
  
 if self.radio\_cmap\_dist.isChecked():  
 *# Default color map* color\_data = np.array(self.distances)  
 elif self.radio\_cmap\_clusters.isChecked():  
 if not self.dump\_paused:  
 *# Fit into k-means estimator if enabled* self.estimator = KMeans(n\_clusters=int(self.spin\_clusters.value()))  
 self.estimator.fit(np.array(self.points))  
 if hasattr(self.estimator, **'labels\_'**):  
 *# Clusters color map* color\_data = np.array(self.estimator.labels\_)  
  
 color\_map = plt.get\_cmap(**'hsv'**)  
 min\_data = np.min(color\_data)  
 max\_data = np.max(color\_data)  
 if max\_data == min\_data:  
 max\_data = 1  
 rgba\_img = color\_map((color\_data - min\_data) / (max\_data - min\_data))  
  
 self.points\_surface.setData(pos=np.array(self.points), color=rgba\_img,  
 size=2, pxMode=True)  
  
 cube\_w = self.slider\_cube\_w.value() / 2  
 cube\_h = self.slider\_cube\_h.value() / 2  
 cube\_d = self.slider\_cube\_d.value() / 2  
  
 tl\_f = [-cube\_w, cube\_h, -cube\_d]  
 tr\_f = [cube\_w, cube\_h, -cube\_d]  
 br\_f = [cube\_w, -cube\_h, -cube\_d]  
 bl\_f = [-cube\_w, -cube\_h, -cube\_d]  
 tl\_n = [-cube\_w, cube\_h, cube\_d]  
 tr\_n = [cube\_w, cube\_h, cube\_d]  
 br\_n = [cube\_w, -cube\_h, cube\_d]  
 bl\_n = [-cube\_w, -cube\_h, cube\_d]  
  
 cube\_points = [tl\_f, tr\_f, br\_f, bl\_f, bl\_n, tl\_n, tl\_f, bl\_f, bl\_n, br\_n, tr\_n, tr\_f, br\_f, br\_n, tr\_n,  
 tl\_n]  
  
 self.cube\_lines.setData(pos=np.array(cube\_points), color=[1, 1, 1, 1])  
  
 corr\_yaw = self.slider\_corr\_yaw.value()  
 corr\_x = - self.slider\_corr\_x.value() / 100  
 corr\_w = self.slider\_corr\_w.value() / 10 + 0.1  
 corr\_h = self.slider\_corr\_h.value() / 50 + 0.1  
 corr\_d = self.slider\_corr\_d.value() / 50 + 0.1  
  
 tl\_f = rotate3d([-corr\_w, corr\_h, -corr\_d], 0, 0, math.radians(corr\_yaw))  
 tr\_f = rotate3d([corr\_x, corr\_h, -corr\_d], 0, 0, math.radians(corr\_yaw))  
 br\_f = rotate3d([corr\_x, -corr\_h, -corr\_d], 0, 0, math.radians(corr\_yaw))  
 bl\_f = rotate3d([-corr\_w, -corr\_h, -corr\_d], 0, 0, math.radians(corr\_yaw))  
 tl\_n = rotate3d([-corr\_w, corr\_h, corr\_d], 0, 0, math.radians(corr\_yaw))  
 tr\_n = rotate3d([corr\_x, corr\_h, corr\_d], 0, 0, math.radians(corr\_yaw))  
 br\_n = rotate3d([corr\_x, -corr\_h, corr\_d], 0, 0, math.radians(corr\_yaw))  
 bl\_n = rotate3d([-corr\_w, -corr\_h, corr\_d], 0, 0, math.radians(corr\_yaw))  
  
 *# Check obstacles* cube\_test = (bl\_f, br\_f, br\_n, bl\_n, tl\_f, tr\_f, tr\_n, tl\_n)  
 outside\_ids = inside\_test(np.array(self.points), cube\_test)  
 density = (len(self.points) - len(outside\_ids)) / len(self.points)  
 density \*= 100  
 if density > 1:  
 density = 1  
  
 corr\_points = [tl\_f, tr\_f, br\_f, bl\_f, bl\_n, tl\_n, tl\_f, bl\_f, bl\_n, br\_n, tr\_n, tr\_f, br\_f, br\_n, tr\_n,  
 tl\_n]  
  
 self.corridor\_lines.setData(pos=np.array(corr\_points), color=[density, 1 - density, 0, 1])  
  
 def load\_data(self):  
 *"""  
 Loads dump file  
 :return:  
 """* if not self.reader\_running:  
 if os.path.exists(self.data\_file.text()):  
 print(**'Loading data...'**)  
 self.dump\_file = open(self.data\_file.text(), **'rb'**)  
 self.reader\_running = True  
 thread = threading.Thread(target=self.dump\_reader)  
 thread.start()  
 else:  
 print(**'File'**, self.data\_file.text(), **'doesn**\'**t exist!'**)  
  
 def pause(self):  
 *"""  
 Pauses data stream  
 :return:  
 """* self.dump\_paused = not self.dump\_paused  
 if self.dump\_paused:  
 self.btn\_pause.setText(**'Resume'**)  
 else:  
 self.btn\_pause.setText(**'Pause'**)  
  
 def stop\_reading(self):  
 *"""  
 Stops reading data from dump file  
 :return:  
 """* self.reader\_running = False  
 self.dump\_file.close()  
  
 def dump\_reader(self):  
 *"""  
 Reads dump from file  
 :return:  
 """  
 # Clear table and data arrays* self.points\_table.setRowCount(0)  
  
 *# Create temp buffers* bytes\_buffer = [0] \* 100  
 bytes\_buffer\_position = 0  
 previous\_byte = 0  
 packets\_read = 0  
 self.distances = [0.] \* 360 \* 32  
 self.points = [[0., 0., 0.]] \* 360 \* 32  
 *# Array of indexes* indexes = np.array([[0] \* 32] \* 360)  
 index = 0  
 for yaw in range(360):  
 for pitch in range(32):  
 indexes[yaw][pitch] = index  
 index += 1  
  
 *# Continue reading* while self.reader\_running:  
 incoming\_bytes = self.dump\_file.read(1024)  
 if incoming\_bytes is None or len(incoming\_bytes) == 0:  
 self.reader\_running = False  
 break  
  
 for incoming\_byte in incoming\_bytes:  
  
 while self.dump\_paused:  
 time.sleep(0.1)  
  
 bytes\_buffer[bytes\_buffer\_position] = incoming\_byte  
 if bytes\_buffer[bytes\_buffer\_position] == 238 and previous\_byte == 255:  
 bytes\_buffer\_position = 0  
 if bytes\_buffer.count(0) < len(bytes\_buffer) - 2:  
 *# Check for not 00 packet  
  
 # Calculate yaw angle* laser\_yaw = (int(bytes\_buffer[1] & 0xFF) << 8) | int(bytes\_buffer[0] & 0xFF)  
 laser\_yaw /= 100.0  
 while laser\_yaw >= 360:  
 laser\_yaw -= 360.0  
  
 first\_distance = 0  
 last\_distance = 0  
 byte\_position = 2  
 laser\_num = 0  
 while byte\_position <= 96:  
 *# Calculate distance* distance = (int(bytes\_buffer[byte\_position + 1] & 0xFF) << 8) | int(  
 bytes\_buffer[byte\_position] & 0xFF)  
 distance \*= 0.002  
  
 *# First and last distances for table* if laser\_num == 0:  
 first\_distance = distance  
 elif laser\_num == 31:  
 last\_distance = distance  
  
 if distance < 50:  
 *# Calculate pitch angle* laser\_pitch = self.laser\_angles[laser\_num]  
  
 *# Rotate distance over pith and yaw* rotated\_point = rotate3d((distance, 0, 0), 0, math.radians(laser\_pitch),  
 math.radians(laser\_yaw))  
  
 *# Find flat array index* array\_position = indexes[int(laser\_yaw)][laser\_num]  
  
 *# Check cube and FOV* cube\_w = self.slider\_cube\_w.value() / 2  
 cube\_h = self.slider\_cube\_h.value() / 2  
 cube\_d = self.slider\_cube\_d.value() / 2  
 if not (-cube\_w < rotated\_point[0] < cube\_w and  
 -cube\_h < rotated\_point[1] < cube\_h and  
 -cube\_d < rotated\_point[2] < cube\_d and  
 abs(laser\_pitch) <= self.spin\_fov\_pitch.value() and  
 laser\_yaw <= self.spin\_fov\_yaw.value()):  
 rotated\_point = [0, 0, 0]  
 distance = 0  
  
 *# Fill arrays* self.points[array\_position] = \  
 [rotated\_point[0], rotated\_point[1], rotated\_point[2]]  
 self.distances[array\_position] = distance  
  
 byte\_position += 3  
 laser\_num += 1  
  
 *# Write packet to the table* row\_number = self.points\_table.rowCount()  
 self.points\_table.insertRow(row\_number)  
 self.points\_table.setItem(row\_number, 0, QTableWidgetItem(str(row\_number)))  
 self.points\_table.setItem(row\_number, 1, QTableWidgetItem(str(laser\_yaw)))  
 self.points\_table.setItem(row\_number, 2, QTableWidgetItem(str(first\_distance) +  
 **' ... '** + str(last\_distance)))  
 packets\_read += 1  
 *# time.sleep(0.001)* else:  
 previous\_byte = bytes\_buffer[bytes\_buffer\_position]  
 bytes\_buffer\_position += 1  
 if bytes\_buffer\_position >= 100:  
 bytes\_buffer\_position = 0  
  
 self.dump\_file.close()  
 print(**'File reading stopped. Read'**, packets\_read, **'packets'**)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app = QApplication(sys.argv)  
 app.setStyle(**'fusion'**)  
 win = Window()  
 sys.exit(app.exec\_())

*<?*xml version="1.0" encoding="UTF-8"*?>  
<!--  
This is free and unencumbered software released into the public domain.  
  
Anyone is free to copy, modify, publish, use, compile, sell, or  
distribute this software, either in source code form or as a compiled  
binary, for any purpose, commercial or non-commercial, and by any  
means.  
  
In jurisdictions that recognize copyright laws, the author or authors  
of this software dedicate any and all copyright interest in the  
software to the public domain. We make this dedication for the benefit  
of the public at large and to the detriment of our heirs and  
successors. We intend this dedication to be an overt act of  
relinquishment in perpetuity of all present and future rights to this  
software under copyright law.  
  
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,  
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF  
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  
IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR  
OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,  
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR  
OTHER DEALINGS IN THE SOFTWARE.  
  
For more information, please refer to <https://unlicense.org>  
 -->*<ui version="4.0">  
 <class>MainWindow</class>  
 <widget class="QMainWindow" name="MainWindow">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>962</width>  
 <height>691</height>  
 </rect>  
 </property>  
 <property name="font">  
 <font>  
 <family>Calibri Light</family>  
 <pointsize>12</pointsize>  
 </font>  
 </property>  
 <property name="windowTitle">  
 <string>DoDCaPS LR2</string>  
 </property>  
 <widget class="QWidget" name="centralwidget">  
 <layout class="QGridLayout" name="gridLayout">  
 <item row="0" column="0">  
 <widget class="QLabel" name="label">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="font">  
 <font>  
 <pointsize>22</pointsize>  
 </font>  
 </property>  
 <property name="text">  
 <string>DoDCaPS LR2</string>  
 </property>  
 </widget>  
 </item>  
 <item row="0" column="1" rowspan="5">  
 <widget class="GLViewWidget" name="openGLWidget">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Expanding" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="minimumSize">  
 <size>  
 <width>640</width>  
 <height>480</height>  
 </size>  
 </property>  
 </widget>  
 </item>  
 <item row="1" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_2">  
 <item>  
 <widget class="QLabel" name="label\_4">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Readed data</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QTableWidget" name="points\_table">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Minimum" vsizetype="Expanding">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="sortingEnabled">  
 <bool>true</bool>  
 </property>  
 <attribute name="horizontalHeaderShowSortIndicator" stdset="0">  
 <bool>true</bool>  
 </attribute>  
 <attribute name="verticalHeaderShowSortIndicator" stdset="0">  
 <bool>false</bool>  
 </attribute>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="2" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout">  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout">  
 <item>  
 <widget class="QLabel" name="label\_19">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Dump file:</string>  
 </property>  
 <property name="alignment">  
 <set>Qt::AlignCenter</set>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QLineEdit" name="data\_file">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>LR2\_data/UDPFromVelodyneTest\_lidardata.pcap</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_2">  
 <item>  
 <widget class="QPushButton" name="btn\_load\_data">  
 <property name="text">  
 <string>Load data from file</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_pause">  
 <property name="text">  
 <string>Pause</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QPushButton" name="btn\_stop\_reading">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Close file</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="3" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_4">  
 <item>  
 <widget class="QLabel" name="label\_3">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>FOV</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_12">  
 <item>  
 <widget class="QLabel" name="label\_12">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Yaw:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSpinBox" name="spin\_fov\_yaw">  
 <property name="minimum">  
 <number>1</number>  
 </property>  
 <property name="maximum">  
 <number>360</number>  
 </property>  
 <property name="value">  
 <number>360</number>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_11">  
 <item>  
 <widget class="QLabel" name="label\_13">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Pitch:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSpinBox" name="spin\_fov\_pitch">  
 <property name="minimum">  
 <number>1</number>  
 </property>  
 <property name="maximum">  
 <number>32</number>  
 </property>  
 <property name="value">  
 <number>32</number>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="4" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_6">  
 <item>  
 <widget class="QLabel" name="label\_14">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Color map mode</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QRadioButton" name="radio\_cmap\_dist">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Distance</string>  
 </property>  
 <property name="checked">  
 <bool>true</bool>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_13">  
 <item>  
 <widget class="QRadioButton" name="radio\_cmap\_clusters">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Clusters</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSpinBox" name="spin\_clusters">  
 <property name="minimum">  
 <number>2</number>  
 </property>  
 <property name="value">  
 <number>5</number>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <widget class="QRadioButton" name="radio\_cmap\_height">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Height</string>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item row="5" column="0">  
 <layout class="QVBoxLayout" name="verticalLayout\_3">  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_3">  
 <item>  
 <widget class="QLabel" name="label\_5">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Cube width:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_cube\_w">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="maximum">  
 <number>50</number>  
 </property>  
 <property name="value">  
 <number>20</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_4">  
 <item>  
 <widget class="QLabel" name="label\_6">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Cube depth:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_cube\_h">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="maximum">  
 <number>50</number>  
 </property>  
 <property name="value">  
 <number>15</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_5">  
 <item>  
 <widget class="QLabel" name="label\_2">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Cube height:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_cube\_d">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Preferred" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="maximum">  
 <number>50</number>  
 </property>  
 <property name="value">  
 <number>5</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 <item row="5" column="1">  
 <layout class="QVBoxLayout" name="verticalLayout\_5">  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_9">  
 <item>  
 <widget class="QLabel" name="label\_10">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Corridor yaw:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_corr\_yaw">  
 <property name="maximum">  
 <number>360</number>  
 </property>  
 <property name="value">  
 <number>180</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_10">  
 <item>  
 <widget class="QLabel" name="label\_11">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Corridor start X:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_corr\_x">  
 <property name="maximum">  
 <number>100</number>  
 </property>  
 <property name="value">  
 <number>2</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_6">  
 <item>  
 <widget class="QLabel" name="label\_7">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Corridor width:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_corr\_w">  
 <property name="maximum">  
 <number>100</number>  
 </property>  
 <property name="value">  
 <number>50</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_7">  
 <item>  
 <widget class="QLabel" name="label\_8">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Corridor depth:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_corr\_h">  
 <property name="maximum">  
 <number>100</number>  
 </property>  
 <property name="value">  
 <number>30</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 <item>  
 <layout class="QHBoxLayout" name="horizontalLayout\_8">  
 <item>  
 <widget class="QLabel" name="label\_9">  
 <property name="sizePolicy">  
 <sizepolicy hsizetype="Fixed" vsizetype="Fixed">  
 <horstretch>0</horstretch>  
 <verstretch>0</verstretch>  
 </sizepolicy>  
 </property>  
 <property name="text">  
 <string>Corridor height:</string>  
 </property>  
 </widget>  
 </item>  
 <item>  
 <widget class="QSlider" name="slider\_corr\_d">  
 <property name="maximum">  
 <number>100</number>  
 </property>  
 <property name="value">  
 <number>30</number>  
 </property>  
 <property name="orientation">  
 <enum>Qt::Horizontal</enum>  
 </property>  
 </widget>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </item>  
 </layout>  
 </widget>  
 <widget class="QMenuBar" name="menubar">  
 <property name="geometry">  
 <rect>  
 <x>0</x>  
 <y>0</y>  
 <width>962</width>  
 <height>21</height>  
 </rect>  
 </property>  
 </widget>  
 <widget class="QStatusBar" name="statusbar"/>  
 </widget>  
 <customwidgets>  
 <customwidget>  
 <class>GLViewWidget</class>  
 <extends>QOpenGLWidget</extends>  
 <header>pyqtgraph.opengl</header>  
 </customwidget>  
 </customwidgets>  
 <resources/>  
 <connections/>  
</ui>