УДК 004.031.2

Назаренко Павел Александрович, ассистент,
Санкт-Петербургский государственный университет
аэрокосмического приборостроения, Санкт-Петербург

Барякшева Валентина Павловна, ассистент,
Санкт-Петербургский государственный университет
аэрокосмического приборостроения, Санкт-Петербург

Попов Даниил Евгеньевич, студент,
Санкт-Петербургский государственный университет
аэрокосмического приборостроения, Санкт-Петербург

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АВТОНОМНОГО ПОЛЁТА МУЛЬТИРОТОРНЫХ СИСТЕМ НА ЯЗЫКЕ PYTHON 3.8

Аннотация: В данной статье, рассматривается программа, которая позволяет на основе заданных параметров ландшафта и БПЛА (беспилотных летательных аппаратов), построить оптимальный маршрут для выполнения задач БПЛА без осуществления управления оператора, т.е. в автономном режиме. На вход программы загружается ландшафт в формате stereolithography (stl), габариты летательного аппарата, точка старта и точка конца мисси.

Ключевые слова: БПЛА, программа, stereolithography, маршрут.

Stereolithography (stl) – формат для хранения трехмерных объектов, в котором информация об объекте храниться как список треугольных граней, которые описывают его поверхность. Для чтения данного формата в программе используется библиотека stl [1].

Данная программа считывает: файл ландшафта в формате stl, точку старта и точку конца мисси в 3х-мерных декартовых координатах. Считанный файл ландшафта представляет собой массив координат вершин треугольников, описывающих грани ландшафта. На основе координат точек старта и конца траектории строится прямой отрезок. Если отрезок пересекает хотя бы одну треугольную грань, то идет перестройка маршрута по алгоритму. После того как построенный маршрут не будет пересекать не одну из граней с учетом безопасной зоны маневра БПЛА, то программа должна будет сформировать код для полетного контроллера. Также для облегчения восприятия работы программы, реализована визуализация 3х-мерного пространства и маршрута при помощи библиотеки matplotlib [2].

Описание алгоритма постройки маршрута

Из точки старта в точку конца маршрута строится отрезок, если отрезок пересекает хотя бы одну из треугольных граней, то идет перестройка маршрута. Если отрезок не пересекает ни одну из треугольных граней, то траектория БПЛА будет представлять собой прямую [3].

Алгоритм перестройки маршрута представляет собой:

1. Нахождение координат точки пересечения отрезка и пересекаемых треугольных граней;
2. Нахождение координат точек пересечения нормалей, опущенных из точек пересечения на ребра треугольных граней. Одна из этих точек будет новой промежуточной точкой маршрута. Эти точки будут образовывать массив точек возможно излома траектории;
3. Следующим этапом идет постройка отрезков, соединяющих одну из промежуточных точек с наименьшим расстоянием с точкой старта и точкой конца траектории.

4. Затем пункты 1, 2 и 3 повторяются для каждого нового образованного отрезка, пока не будет ни одного пересечения с треугольными гранями. Тем самым данный алгоритм представляет собой рекурсивный характер.

5. В результате работы предыдущих пунктов получается траектория облета по поверхности пересекаемых препятствий. На этом этапе точки траектории смещаются на расстояние равное безопасной зоны маневра БПЛА и идет проверка отодвинутых точек по предыдущим пунктам.

В результате работы данного алгоритма будет получен маршрут облета препятствий с учетом безопасной зоны маневра БПЛА. Далее можно будет этот маршрут преобразовать в код для полетного контроллера [4].

Алгоритм нахождения координат точки пересечения отрезка и грани:

Данный алгоритм включает в себя алгоритм нахождения точки пересечения прямой и плоскости и алгоритм вхождения этой точки в грань.

Пример алгоритма нахождения точки пересечения прямой и плоскости, написанный на языке программирования Python 3.8 представлен на рисунке 1.

```
1 usage
def Point_cross_line(Plane, line_x, line_y, line_z):

    A, B, C, D = Plane
    koeff_parametr = A * line_x[0] + B * line_y[0] + C * line_z[0]
    koeff = A * line_x[1] + B * line_y[1] + C * line_z[1]
    if koeff_parametr != 0:
        parametr = (- D - koeff) / koeff_parametr
    else:
        parametr = 0

    Point_X = parametr * line_x[0] + line_x[1]
    Point_Y = parametr * line_y[0] + line_y[1]
    Point_Z = parametr * line_z[0] + line_z[1]

    return Point_X, Point_Y, Point_Z
```

Рис. 1

Алгоритм нахождения координат точек пересечения нормалей, опущенных из точки пересечения треугольной граи на его ребра представлен на рисунке 2.

```python
def Coords_Normal_Points(mesh, Cross_Point_Coord):

    x1, y1, z1 = Coord_Normal_Point(Cross_Point_Coord, mesh[0], mesh[1])
    x2, y2, z2 = Coord_Normal_Point(Cross_Point_Coord, mesh[1], mesh[2])
    x3, y3, z3 = Coord_Normal_Point(Cross_Point_Coord, mesh[0], mesh[2])

    X = [x1, x2, x3]
    Y = [y1, y2, y3]
    Z = [z1, z2, z3]

    return X, Y, Z

def Coord_Normal_Point(Point_Coord, Point_1_Line_Coord, Point_2_Line_Coord):
    x0, y0, z0 = Point_Coord
    xa, ya, za = Point_1_Line_Coord
    xb, yb, zb = Point_2_Line_Coord

    v = [xb - xa, yb - ya, zb - za] #Направляющий вектор линии
    u = [x0 - xa, y0 - ya, z0 - za]
    nx = v[1] * u[2] - v[2] * u[1]
    ny = -(v[0] * u[2] - v[2] * u[0])
    nz = v[0] * u[1] - v[1] * u[0]

    n = [nx, ny, nz] #Вектор нормали

    kanon_line_AB = [[-xa, v[0]], [-ya, v[1]], [-za, v[2]]]
    kanon_line_ON = [[-x0, nx], [-y0, ny], [-z0, nz]]

    param = ((x0*v[0] + y0*v[1] + z0*v[2]) - (xa*v[0] + ya*v[1] + za*v[2])) / (v[0]*v[0] + v[1]*v[1] + v[2]
    X_Cord = v[0] * param + xa
    Y_Cord = v[1] * param + ya
    Z_Cord = v[2] * param + za

    return X_Cord, Y_Cord, Z_Cord
```

Рис. 2

Преимущества данной программы:

- Позволяет упростить и автоматизировать процесс постройки маршрута для БПЛА
- Автоматизирует процесс программирования полетного контроллера БПЛА
- Может учесть движущиеся объекты и иные факторы, такие как инерция БПЛА, сопротивление воздуха и т.п.
- Может рассчитывать траекторию БПЛА для различной местности
- Сводит к минимуму работу программиста и оператора

Данная программа позволяет построить маршрут для любой местности, смоделированной в САПР (системе автоматизированного проектирования) программе и сохраненной в формате stl. Позволяет учесть множество внешних факторов, на основе которых, строится безопасная зона маневра БПЛА и с учетом этого строит маршрут. По этому маршруту будет создан код для полетного контроллера. Программа позволяет быстро запрограммировать БПЛА, для выполнения мисси в автономном режиме.

Список литературы:

1. Индекс пакетов Python (PyPI). URL: https://pypi.org/project/numpy-stl (дата обращения 16.11.2023)

2. Хантер Д., Дейл Д. 3D и объемные данные. URL: https://matplotlib.org/stable/plot_types/3D/scatter3d_simple.html#sphx-glr-plot-types-3d-scatter3d-simple-py (дата обращения 16.11.2023)

3. Файринг Э., Дреттбум М. Парные данные. URL: https://matplotlib.org/stable/plot_types/ basic/plot.html (дата обращения 16.11.2023)

4. Шахбаз Х. 3D-построение в Python с использованием matplotlib. URL: https://likegeeks.com/3d-plotting-in-python/ (дата обращения 16.11.2023)