УДК 004.046 Научная статья

https://doi.org/10.32603/2071-8985-2023-16-1-66-76

Разработка программно-алгоритмического обеспечения и системы коммуникации взаимодействия для управления роем беспилотных летательных аппаратов, выполняющих миссию по мониторингу в группе

Ж. Б. Нгуа Ндонг Авеле

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Санкт-Петербург, Россия

✉ avelejacques@yahoo.fr

Аннотация. Перспективным направлением использования дронов является их объединение в группы или стада. У насекомых соблюдается принцип роевой организации. В случае с дронами после объединения в рой каждый дрон управляется своей автоматикой, а поведением роя может управлять программа с элементами искусственного интеллекта или оператор. В статье рассмотрена задача повышения безопасности работы полета группы дронов в реальной жизни в агропромышленном комплексе. Программно-алгоритмическое обеспечение для управления роем беспилотных летательных аппаратов при мониторинге в агропромышленном комплексе было создано при помощи 3D-симулятора Coppeliasim и программного обеспечения PyCharm. Была реализована симуляция системы управления роем дронов при мониторинге растений в агропромышленном комплексе.

Ключевые слова: робототехника, роевая робототехника, программно-алгоритмическое обеспечение, рой беспилотных летательных аппаратов

Для цитирования: Нгуа Ндонг Авеле Ж. Б. Разработка программно-алгоритмического обеспечения для управления роем беспилотных летательных аппаратов // Изв. СПбГЭТУ «ЛЭТИ». 2023. Т. 16, № 1. С. 66–76. doi: 10.32603/2071-8985-2023-16-1-66-76.

Original article

Development of Software and Algorithmic Support and an Interaction Communication System for Controlling a Swarm of Unmanned Aerial Vehicles Performing a Monitoring Mission in Group

J. B. Ngoua Ndong Avele

Saint Petersburg Electrotechnical University, Saint Petersburg, Russia

✉ avelejacques@yahoo.fr

Abstract. A promising direction for the use of drones is their association into groups or herds. Insects observe the principle of swarm organization. In the case of drones, after combining into a swarm, each drone is controlled by its own automation, and the behavior of the swarm can be controlled by a program with elements of artificial intelligence or (several) an operator. The article considers the task of improving the safety of the flight of a group of drones in real life in the agro-industrial complex. The software-algorithmic software for controlling the swarm of unmanned aerial vehicles during monitoring in the agro-industrial complex was created using the 3D simulator Coppeliasim and the software PyCharm. A simulation of a drone swarm control system was implemented when monitoring plants in the agro-industrial complex.

Keywords: robotics, swarm robotic, software and algorithmic support, swarm of unmanned aerial vehicles

For citation: Ngoua Ndong Avele J. B. Development of Software and Algorithmic Support for Controlling a Swarm of Unmanned Aerial Vehicles // LETI Transactions on Electrical Engineering & Computer Science. 2023. Vol. 16, no. 1. P. 66–76. doi: 10.32603/2071-8985-2023-16-1-66-76.

Введение. Роевая робототехника имеет множество потенциальных применений. К ним относятся задачи, требующие миниатюризации, – задачи распределенного зондирования в микромашинах или человеческом теле. К наиболее перспективным применениям роевой робототехники относятся поисково-спасательные операции. Стаи роботов разного размера можно отправлять в места, куда спасатели не могут добраться безопасно, для исследования неизвестной среды с помощью бортовых датчиков. С другой стороны, групповая робототехника может подойти для приложений, требующих недорогого строительства, – таких, как горнодобывающая промышленность или сельскохозяйственные работы. Что еще более спорно, рои военных роботов могут формировать автономную армию. ВМС США испытали рой автономных лодок, которые могут самостоятельно проводить наступательные операции и управлять ими. Катера беспилотные и могут быть оснащены любым оборудованием для сдерживания и уничтожения кораблей противника. Во время гражданской войны в Сирии российские войска в этом регионе сообщали об атаках на их главную авиабазу в стране целыми роями самолетов с неподвижным крылом, начиненных взрывчаткой.

Рой дронов используется для наведения, отображения и доставки дронов. Ночью на дисплее дронов обычно используется несколько освещенных дронов для художественного отображения или рекламы. Рой доставленных дронов может одновременно доставлять несколько посылок в одно и то же место назначения и преодолевать ограничения полезной нагрузки и батареи одного дрона. Рой дронов может принимать различные формы полета, чтобы снизить общее энергопотребление из-за сил сопротивления.

Актуальность данной статьи заключается в разработке симуляции управления полета роем дронов для мониторинга растений в агропромышленном комплексе. В сельском хозяйстве активно внедряется практика применения дронов для разведки состояния сельскохозяйственных растений, мониторинга выполнения работ на полях, оценки качества урожая и т. п. В настоящее время развивающиеся страны выделяют из бюджета большие деньги на совершенствование и разработку новых образцов БПЛА – беспилотных летательных аппаратов. БПЛА получают все большее распространение, особенно беспилотники вертолетного типа с четырьмя двигателями (квадрокоптеры). Для выполнения большинства задач используется один дрон, при этом в силу того, что время его полета ограничено емкостью аккумулятора, для обслуживания, например, одного поля, требуется много времени, затрачиваемого на подзарядку, возвращение в начальную точку вылета и т. д. Если использовать рой, то увеличивается площадь охвата и сокращается время на выполнение задачи.

Постановка задачи. В статье рассмотрены анализ технологии роя беспилотных летательных аппаратов с целью повысить безопасность работы полета группы дронов в реальной жизни при мониторинге в агропромышленном комплексе; разработка математической модели для вычисления расстояния между группами дронов, а также особенности разработки программно-алгоритмического обеспечения для управления роем беспилотных летательных аппаратов при мониторинге сельскохозяйственных растений.

Анализ технологии роя беспилотных летательных аппаратов. Рой дронов, также называемый флотом дронов, представляет собой скоординированный набор дронов – воздушных, наземных, подземных или морских – с целью выполнения общей задачи в различных типах приложений, гражданских или военных. Это направление роевой робототехники. Специалисты холдинга «Российские космические системы», который входит в состав Роскосмоса, разработали систему динамического мониторинга климатически активных и опасных веществ в воздухе с помощью роя беспилотников. Отличительная особенность разработки состоит в том, что мониторинг ведется с помощью объединенных в группу беспилотных летательных аппаратов с установленными на них газоанализаторами – устройствами распознавания опасных веществ.

Беспилотники «Молния» рассчитаны на групповое применение. Аппараты будут запускать по цели массово, сразу с нескольких носителей; в полете они смогут взаимодействовать друг с другом, обмениваться информацией – на Западе это называется «роем дронов» (Drone Swarm). Со временем военное руководство США внедрит этот принцип во все свои воздушные операции. Даже современной системе противовоздушной обороны крайне сложно бороться с тучей малоразмерных, малозаметных и скоростных целей.

В нашем случае будет использоваться рой беспилотных летательных аппаратов при монито-

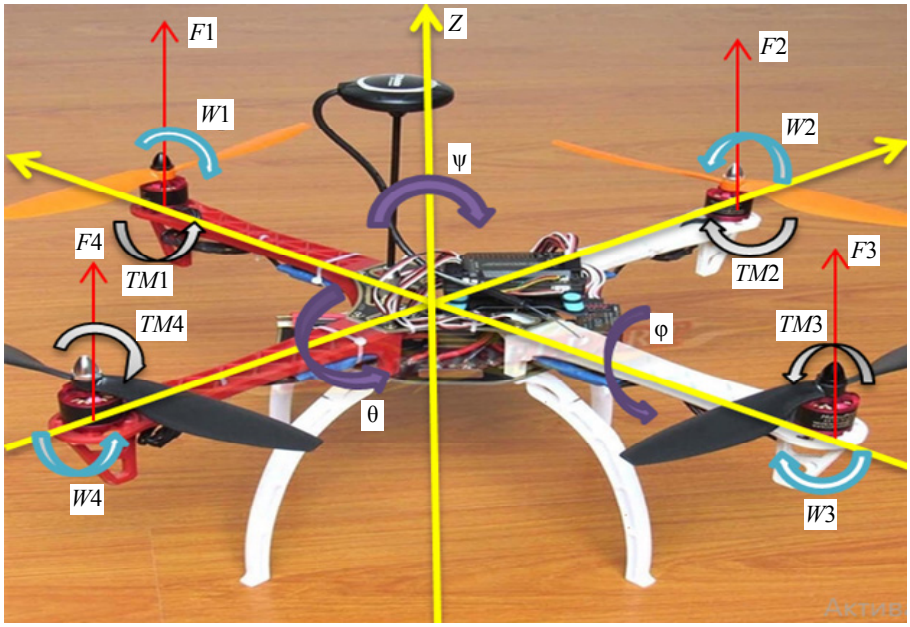


Рис. 1. Физическая схема квадрокоптера
Fig. 1. The physical scheme of the quadrocopter)

ринге в агропромышленном комплексе. Проблема состоит в том, что для обработки больших площадей одним дроном требуются значительные затраты времени. Решением может стать использование группы дронов, когда ведущим дроном управляет пилот (либо автопилот), а ведомые дроны согласованно работают в группе.

Физическая модель квадрокоптера. Квадрокоптер как описываемый объект управления имеет шесть степеней свободы, что в результате позволяет получить систему из шести уравнений [1]. Для составления физической модели управляемого движения квадрокоптера использована схема, представленная на рис. 1 [2], где можно посмотреть строение квадрокоптера: W – угловая скорость винтов; F – подъемная сила; TM – крутящий момент мотора.

Положение центра масс квадрокоптера в неподвижной связанной с землей системе координат обозначим $\boldsymbol{\varepsilon}$.

$\boldsymbol{\varepsilon} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ – вектор положения центра масс квадрокоптера; $\boldsymbol{\mu} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}$ – вектор углов Эйлера, где φ, θ, ψ – углы вокруг осей x, y, z соответственно; $q = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\mu} \end{bmatrix}$ – конкатенация векторов $\boldsymbol{\varepsilon}$ и $\boldsymbol{\mu}$.

Матрицы поворота вокруг оси x, y, z:

$$R_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix};$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix};$$

$$R_z(\psi) = \begin{bmatrix} \sin\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Чтобы найти углы φ, θ, ψ нужно узнать общую матрицу поворота 3 × 3:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix};$$

$$R = \begin{bmatrix} \cos_y \sin_z & \cos_z \sin_x \sin_y - \cos_x \sin_z & \sin_x \sin_z + \cos_x \cos_z \sin_y \\ \cos_y \sin_z & \cos_x \cos_z + \sin_x \sin_y \sin_z & \cos_x \sin_z \sin_y - \cos_z \sin_z \\ -\sin_y & \cos_y \sin_x & \cos_x \cos_y \end{bmatrix}.$$

Теперь можно найти углы φ, θ, ψ:

$$\varphi = \operatorname{atan2}(r_{32}, r_{33});$$

$$\theta = \operatorname{atan2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right);$$

$$\Psi = \operatorname{atan2}(r_{21}, r_{11}).$$

Математическое описание для определения расстояния между группами дронов при полете. Для составления математического описания для определения расстояния между группами дронов при полете использована следующая схема (рис. 2) [3].

Пусть m – минимальная допустимая дистанция между дронами; d – текущая дистанция между дронами R_i и R_j; L – радиус видимости каждого дрона R_i.

Тогда при выполнении условия $m < d < L$, вектор результирующей силы $\mathbf{F}_{ij}(k)$, определяющий требуемое действие a_i, будет направлен на дрон R и прямо пропорционален дистанции между ними d:

$\left\{\left|\mathbf{F}_{ij}(k)\right| \sim d,\ m < d < L\right.$ – при прямом направлении;

$\left\{\left|\mathbf{F}_{ij}(k)\right| \sim d^{-1},\ m < d < L\right.$ – при обратном направлении.

На рис. 2 показаны векторы силы $\mathbf{F}_{ij}(k)$ и дистанции между группами дронов. Группы дронов, владеющих информацией о требуемом направлении движения, будем называть «ведущими» дронами, а остальные – ведомыми [4].

Решение системы коммуникационного взаимодействия дронов, выполняющих миссию в составе группы в агропромышленном комплексе. При разработке проекта, использовали одиночно-групповую децентрализованную сеть роя. В одиночно-групповой децентрализованной сети роя (рис. 3) внутреннее взаимодействие не зависит от наземной инфраструктуры. В этой системе другие летательные аппараты служат узлами ретрансляции, которые передают данные внутри роя. Используя этот метод, БПЛА в рое могут обмениваться информацией о ситуации в режиме реального времени для оптимизации совместного управления и повышения эффективности. Одна из серьезных проблем, стоящих на пути использовании малоразмерных БПЛА, со-

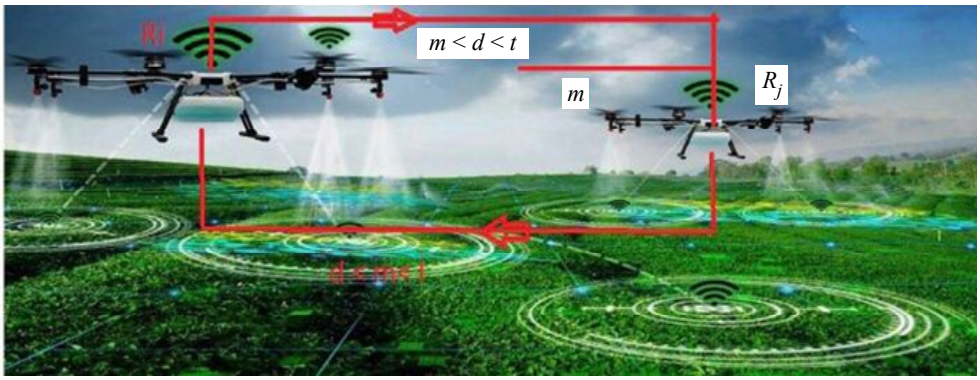


Рис. 2. Математическое условие для определения расстояния между дронами
Fig. 2. Mathematical condition for determining the distance between drones)

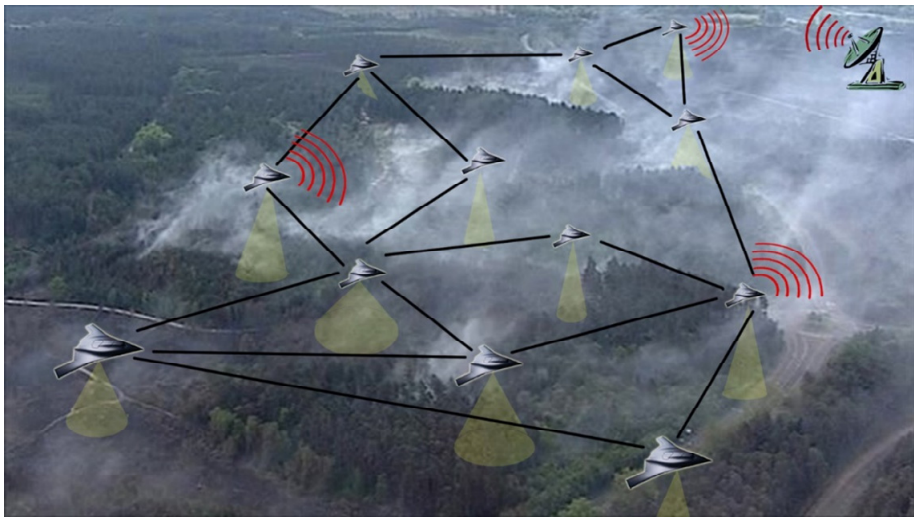


Рис. 3. Схема, изображающая одиночно-групповую децентрализованную сеть
Fig. 3. Scheme depicting a single-group decentralized network

стоит в сложностях обеспечения радиосвязи с операторским пультом управления. Вообще малые дроны накладывают существенные ограничения на запас бортового энергоресурса, большая часть которого предназначена для обеспечения движения, и лишь небольшая часть энергоресурса может использоваться приемопередающей аппаратурой. В итоге мощность радиопередатчиков сильно ограничена. Небольшие размеры БПЛА также ограничивают размеры антенн.

Разработка программно-алгоритмического обеспечения для управления роем дронов. При разработке программно-алгоритмического обеспечения для управления роем дронов мы использовали программную среду имитационного моделирования CoppeliaSim (рис. 4) и программного обеспечения PYCharm. На практике проекта использовали мини-дрон DJI Ryze Tello (рис. 5) что тестировать программу для синхронизации полета между двумя дронами.

Программа на языке Python для симуляции полета двух дронов в 3D-CoppeliasSim.

```
import sim
import sys
import time
import numpy as np
import math
from quadcopter import Quadcopter
from multiprocessing import Process, Manager, Pipe
hx=10
quad = None
#получаем хендлы и координаты меток
def init_coord():
ret, handle = sim.simxGetObjectHandle(clientID, '/target[0]', sim.simx_opmode_oneshot_wait) #главный коптер
```

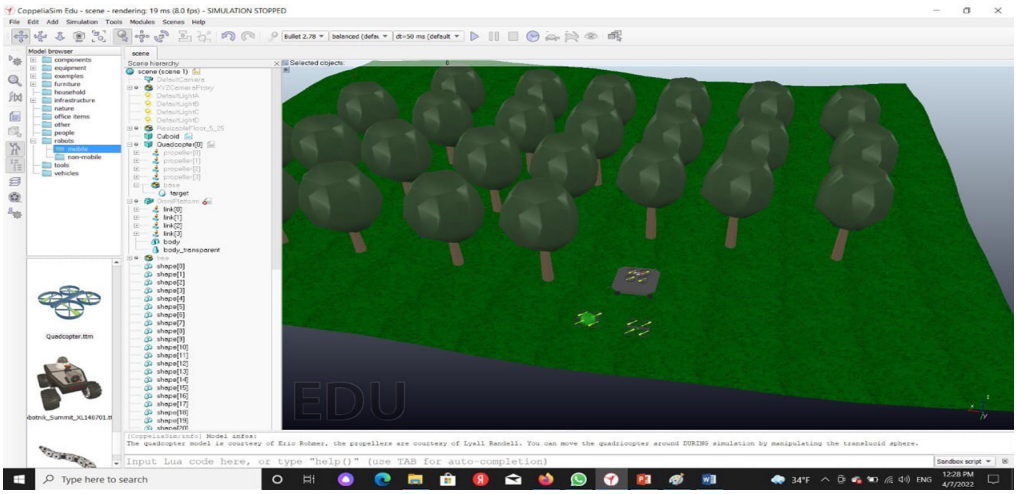


Рис. 4. Симуляция полета роя дронов в Coppeliasim
Fig. 4. Flight simulation by a swarm of drones in Coppeliasim

Рис. 5. Дрон-квадрокоптер DJI Ryze Tello
Fig. 5. Drone-quadcopter DJI Ryze Tello

```
    _, platf_pos = sim.simxGetObjectPosition (clientID, handle, -1, sim.simx_opmode_oneshot_wait) #платформа
    if (not ret == sim.simx_return_ok):
    print('Не получен указатель на цель головного коптера')
    sys.exit(1)
    # полетное задание отмечено метками
    # в программе мы получаем координаты этих меток
    ret, metka1 = sim.simxGetObjectHandle (clientID, '/metka1', sim.simx_opmode_oneshot_wait)
    ret, pos1 = sim.simxGetObjectPosition (clientID, metka1, -1, sim.simx_opmode_oneshot_wait)
    ret, metka2 = sim.simxGetObjectHandle (clientID, '/metka2', sim.simx_opmode_oneshot_wait)
    ret, pos2 = sim.simxGetObjectPosition (clientID, metka2, -1, sim.simx_opmode_oneshot_wait)
    ret, metka3 = sim.simxGetObjectHandle (clientID, '/metka3', sim.simx_opmode_oneshot_wait)
    ret, pos3 = sim.simxGetObjectPosition (clientID, metka3, -1, sim.simx_opmode_oneshot_wait)
    ret, metka4 = sim.simxGetObjectHandle (clientID, '/metka4', sim.simx_opmode_oneshot_wait)
    ret, pos4 = sim.simxGetObjectPosition (clientID, metka4, -1, sim.simx_opmode_oneshot_wait)
    ret, metka5 = sim.simxGetObjectHandle (clientID, '/metka5', sim.simx_opmode_oneshot_wait)
    ret, pos5 = sim.simxGetObjectPosition (clientID, metka5, -1, sim.simx_opmode_oneshot_wait)
    pos=[pos1, pos2, pos3, pos4, pos5] #это наше полетное задание
    return handle, platf_pos,pos1, pos
    def init_coord2(): #инициализация коптера2
    ret, handle2 = sim.simxGetObjectHandle (clientID, '/target[1]', sim.simx_opmode_oneshot_wait)
    if (not ret == sim.simx_return_ok):
    print('Не получен указатель на цель ведомого коптера')
    sys.exit(1)
    return handle2
    def vzlet(handle_copt,quad_copt):
    ret, pos = sim.simxGetObjectPosition(clientID, handle_copt, -1, sim.simx_opmode_oneshot_wait)
    if (not ret == sim.simx_return_ok):
    print('Failed to retrieve position for Quadricopter_target')
    sys.exit(1)
    new_pos = [pos[0], pos[1], 2.5] #высота взлета
    if quad_copt is None:
    quad_copt = Quadcopter(pos)
    time.sleep(5)  # время на запуск коптера
    # взлет
    ret = sim.simxSetObjectPosition(clientID, handle, -1, new_pos, sim.simx_opmode_oneshot_wait)
    time.sleep(0.5)
    return new_pos
    def control_copter2(handle_copt, new_pos,status):
    pos = [new_pos[0], new_pos[1] + 1.5, 2.5] # вычисляем координаты ведомого относительно ведущего
    sim.simxSetObjectPosition(clientID, handle_copt, -1, pos, sim.simx_opmode_oneshot_wait) #летим к нему
    status[1]=status[1]+1
    time.sleep(0.5)
    return status
    def control_copter_main(handle, path,handle2):
    r=0.6 #расстояние между коптерами
    l=len(path) #сколько точек
    i=0
    while i<l:
    # получили текущие координаты коптера
    _, pos = sim.simxGetObjectPosition(clientID, handle, -1,sim.simx_opmode_oneshot_wait) # вычисляем текущие координаты
    pos1=path[i]
    #вычисляем угол поворота к метке
    M=np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    posB=np.reshape(pos1, (3,1)) # координаты метки
    posA = np.reshape(pos, (3,1)) #координаты коптера
    M_inv = np.linalg.inv(M)
    ro=np.dot(M_inv, posB - posA) #координаты метки в системе коптера
    alfa=math.atan(ro[1]/ro[0]) #искомый угол поворота
    #поворачиваем коптеры в направлении метки
    sim.simxSetObjectOrientation(clientID, handle, -1, [0, 0, alfa], sim.simx_opmode_oneshot_wait)
    sim.simxSetObjectOrientation(clientID, handle2, -1, [0, 0, alfa], sim.simx_opmode_oneshot_wait)
    time.sleep(1)
    # летим к метке
    dx = (posB[0] - posA[0]) / 10
    x = posA[0]
    x1=x+0.5
    y=posA[1]
    y1=math.sqrt(r*2-(x-x1)*2)+y
    while (x-posB[0])*2+(y-posB[1])*2 > 0.2:  #если текущая позиция коптера не находится в окрестности цели
```

```
sim.simxSetObjectPosition (clientID, handle, -1, [x, y, 2.5], sim.simx_opmode_oneshot_wait) # летим к метке
sim.simxSetObjectPosition (clientID, handle2, -1, [x1, y1, 2.5], sim.simx_opmode_oneshot_wait)
time.sleep(0.4)
print(y, y1)
x = x + dx
x1=x+0.5
y=(x-posA[0])*(posB[1]-posA[1])/(posB[0]-posA[0])+posA[1]
y1 = math.sqrt(r * 2 - (x - x1) * 2) + y
i = i + 1
return 0
def posadka(handle_copt,platf_pos):
ret = sim.simxSetObjectPosition(clientID, handle_copt, -1, platf_pos, sim.simx_opmode_oneshot_wait)
time.sleep(0.5)
if _name=='main_':
status = [0, 0]
# just in case, close all opened connections
sim.simxFinish (-1)
PORT = 19999
ClientID = sim.simxStart ('127.0.0.1', PORT, True, True, 5000, 5)
# check if client connection successful
if clientID == -1:
print ('Could not connect to remote API server')
sys.exit (1)
print ('Connected to remote API server')
handle, platf_pos,pos1, path =init_coord() #получаем все хендлы головного коптера и координаты меток
handle2 = init_coord2() # получаем хендлы ведомого коптера
pos=vzlet(handle,quad) #взлетаем
time.sleep(1)
status[0]=status[0]+1 #главный взлетел
if status[0]==1:
status=control_copter2(handle2,pos,status)
if status[0]==1 and status[1]==1: #если взлетели все коптеры, то летим
control_copter_main(handle, path, handle2)
#posadka(handle, platf_pos) #посадк
```

Программа на языке Python для симуляции полета трех дронов в 3D-CoppeliasSim.

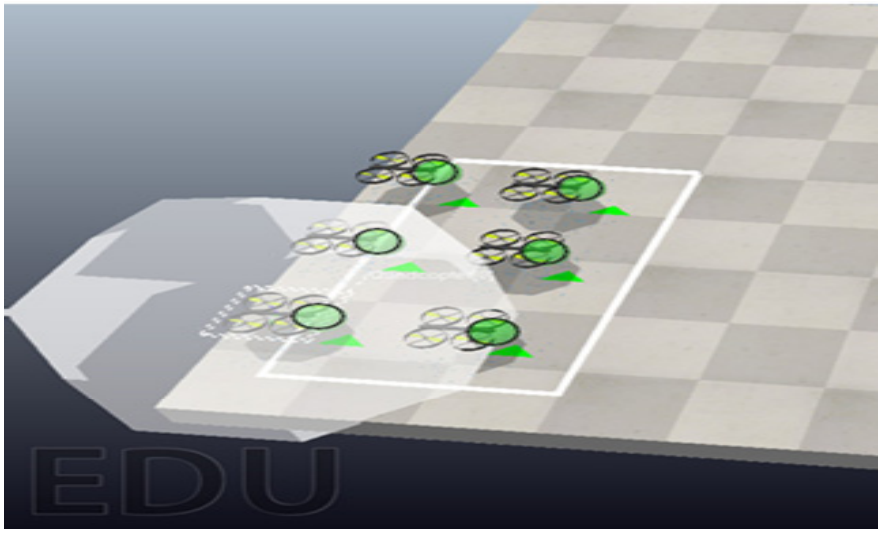
```
import sim
import sys
import time
import numpy as np
import math
from quadcopter import Quadcopter
from multiprocessing import Process, Manager,Pipe
hx=10
quad = None
#получаем хендлы и координаты меток
def init_coord():
ret, handle = sim.simxGetObjectHandle(clientID, '/target[0]', sim.simx_opmode_oneshot_wait) #главный коптер
_, platf_pos = sim.simxGetObjectPosition (clientID, handle, -1, sim.simx_opmode_oneshot_wait) #платформа
if (not ret == sim.simx_return_ok):
print('Не получен указатель на цель головного коптера')
sys.exit(1)
# полетное задание отмечено метками
# в программе мы получаем координаты этих меток
ret, metka1 = sim.simxGetObjectHandle (clientID, '/metka1', sim.simx_opmode_oneshot_wait)
ret, pos1 = sim.simxGetObjectPosition(clientID, metka1, -1, sim.simx_opmode_oneshot_wait)
ret, metka2 = sim.simxGetObjectHandle (clientID, '/metka2', sim.simx_opmode_oneshot_wait)
ret, pos2 = sim.simxGetObjectPosition(clientID, metka2, -1, sim.simx_opmode_oneshot_wait)
ret, metka3 = sim.simxGetObjectHandle (clientID, '/metka3', sim.simx_opmode_oneshot_wait)
ret, pos3 = sim.simxGetObjectPosition(clientID, metka3, -1, sim.simx_opmode_oneshot_wait)
ret, metka4 = sim.simxGetObjectHandle (clientID, '/metka4', sim.simx_opmode_oneshot_wait)
ret, pos4 = sim.simxGetObjectPosition(clientID, metka4, -1, sim.simx_opmode_oneshot_wait)
ret, metka5 = sim.simxGetObjectHandle (clientID, '/metka5', sim.simx_opmode_oneshot_wait)
ret, pos5 = sim.simxGetObjectPosition(clientID, metka5, -1, sim.simx_opmode_oneshot_wait)
pos=[pos1, pos2, pos3, pos4, pos5] #это наше полетное задание
return handle, platf_pos,pos1, pos
def init_coord2(): #инициализация коптера2
ret, handle2 = sim.simxGetObjectHandle (clientID, '/target[1]', sim.simx_opmode_oneshot_wait)
ret, handle3 = sim.simxGetObjectHandle (clientID, '/target[2]', sim.simx_opmode_oneshot_wait)
if (not ret == sim.simx_return_ok):
print('Не получен указатель на цель ведомого коптера')
```

```
    sys.exit(1)
    return handle2, handle3
    def vzlet(handle_copt,quad_copt):
    ret, pos = sim.simxGetObjectPosition(clientID, handle_copt, -1, sim.simx_opmode_oneshot_wait)
    if (not ret == sim.simx_return_ok):
    print('Failed to retrieve position for Quadricopter_target')
    sys.exit(1)
    new_pos = [pos[0], pos[1], 2.5] #высота взлета
    if quad_copt is None:
    quad_copt = Quadcopter(pos)
    time.sleep(5)  # время на запуск коптера
    # взлет
    ret = sim.simxSetObjectPosition(clientID, handle, -1, new_pos, sim.simx_opmode_oneshot_wait)
    time.sleep(0.5)
    return new_pos
    def control_copter2(handle_copt, new_pos, status, handle_copt2):
    pos1 = [new_pos[0], new_pos[1] + 1, 2.5]  # вычисляем координаты ведомого относительно ведущего
    pos2 = [new_pos[0], new_pos[1] + 2, 2.5]  # вычисляем координаты ведомого относительно ведущего
    sim.simxSetObjectPosition(clientID, handle_copt, -1, pos1, sim.simx_opmode_oneshot_wait)
    sim.simxSetObjectPosition(clientID, handle_copt2, -1, pos2, sim.simx_opmode_oneshot_wait) #летим к нему
    status[1]=status[1]+1
    time.sleep(1)
    return status
    def control_copter_main(handle, path,handle2, handle3):
    r=0.6 #расстояние между коптерами
    l=len(path) #сколько точек
    i=0
    while i<l:
    # получили текущие координаты коптера
    _, pos = sim.simxGetObjectPosition(clientID, handle, -1,sim.simx_opmode_oneshot_wait)  # вычисляем текущие координаты
    pos1=path[i]
    #вычисляем угол поворота к метке
    M=np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    posB=np.reshape(pos1, (3,1)) #координаты метки
    posA = np.reshape(pos, (3,1)) #координаты коптера
    M_inv = np.linalg.inv(M)
    ro=np.dot(M_inv, posB - posA) #координаты метки в системе коптера
    alfa=math.atan(ro[1]/ro[0]) #искомый угол поворота
    #поворачиваем коптеры в направлении метки
    sim.simxSetObjectOrientation(clientID, handle, -1, [0, 0, alfa], sim.simx_opmode_oneshot_wait)
    sim.simxSetObjectOrientation (clientID, handle2, -1, [0, 0, alfa], sim.simx_opmode_oneshot_wait)
    sim.simxSetObjectOrientation (clientID, handle3, -1, [0, 0, alfa], sim.simx_opmode_oneshot_wait)
    time.sleep(1)
    # летим к метке
    dx = (posB[0] - posA[0]) / 10
    x = posA[0]
    x1=x+0.5
    x3=x1-0.5
    y=posA[1]
    y1=math.sqrt(r*2-(x-x1)*2)+y
    y3 = math.sqrt(r * 2 - (x - x3) * 2) + y
    while (x-posB[0])*2+(y-posB[1])*2 > 0.2: #если текущая позиция коптера не находится в окрестности цели
    sim.simxSetObjectPosition (clientID, handle, -1, [x, y, 2.5], sim.simx_opmode_oneshot_wait)  # летим к метке
    sim.simxSetObjectPosition (clientID, handle2, -1, [x1, y1, 2.5], sim.simx_opmode_oneshot_wait)
    sim.simxSetObjectPosition (clientID, handle3, -1, [x3, y3, 2.5], sim.simx_opmode_oneshot_wait)
    time.sleep(0.4)
    print(y, y1)
    x = x + dx
    x1=x+0.5
    x3 = x1 - 0.5
    y=(x-posA[0])*(posB[1]-posA[1])/(posB[0]-posA[0])+posA[1]
    y1 = math.sqrt(r * 2 - (x - x1) * 2) + y
    y3 = math.sqrt(r * 2 - (x - x3) * 2) + y
    i = i + 1
    return 0
    def posadka(handle_copt,platf_pos):
    ret = sim.simxSetObjectPosition(clientID, handle_copt, -1, platf_pos, sim.simx_opmode_oneshot_wait)
    time.sleep(0.5)
    if _name=='main_':
    status = [0, 0]
    # just in case, close all opened connections
    sim.simxFinish(-1)
```

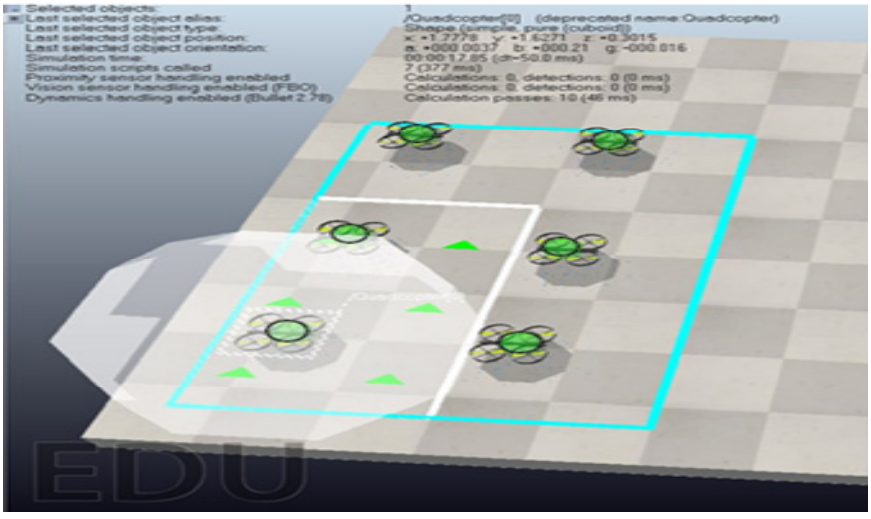
```
PORT = 19999
clientID = sim.simxStart('127.0.0.1', PORT, True, True, 5000, 5)
# check if client connection successful
if clientID == -1:
print('Could not connect to remote API server')
sys.exit(1)
print('Connected to remote API server')
handle, platf_pos,pos1, path =init_coord() #получаем все хендлы головного коптера и координаты меток
handle2, handle3 = init_coord2()  # получаем хендлы ведомого коптера
pos=vzlet(handle,quad) #взлетаем
time.sleep(1)
status[0]=status[0]+1 #главный взлетел
if status[0]==1:
status=control_copter2(handle2,pos,status, handle3)
if status[0]==1 and status[1]==1: #если взлетели все коптеры, то летим
control_copter_main(handle, path, handle2, handle3)
#posadka(handle, platf_pos) #посадка
```

Тело программы (функция main) будет выглядеть следующим образом:

1. Задание размеров поля и количества квадрокоптеров.
2. Создание поля и экземпляров квадрокоптеров.
3. Взлет роя квадрокоптеров.
4. Отработка траектории.
5. Посадка роя квадрокоптеров.



а



б

Рис. 6, *а* – равномерное распределение, территория 1;
б – равномерное распределение, территория 2
Fig. 6, *a* – uniform distribution, territory 1;
б – uniform distribution, territory 2

Пример результата для анализа алгоритма управления полетом шести дронов в 3D-симуляторе CoppeliaSim. Для управления полетом шести дронов было проведено тестирование с применением математического моделирования. С помощью прикладных средств языка программирования Python и с использованием программной среды имитационного моделирования CoppeliaSim были реализованы графики распределения БПЛА по территории и вычислены расстояния между условными центрами БПЛА. Было проведено экспериментальное исследование с использованием предложенного алгоритма участников роя БПЛА для мониторинга заданной территории. Изначальные параметры для проведения имитационного моделирования следующие:

– количество БПЛА, шт.: 6;

– размер территории 1 м^2: 1.2 × 2.2;

– размер территории 1 м^2: 2.2 × 3.3;

– время между сменой территории, с: 37.

При моделировании было две территории разного размера. Рой БПЛА начинает свой полет со стартовых позиций и движется до заданных позиций на первой территории. Пример равномерного распределения участников роя приведен на рис. 6, *а*. По истечении заданного времени размеры территории меняются. Соответственно, меняются и позиции БПЛА, и они продолжают свой полет с целью покрытия территории нового размера. Пример можно увидеть на рис. 6, *б*.

Результаты проведенного моделирования представлены в табл. У нас следующие параметры:

– n – условный номер БПЛА в рое;

– T_1 и T_2 – время полета БПЛА до позиций на первой и второй территориях соответственно;

– D_1 и D_2 – демонстрируют эти расстояния.

Также было рассчитано среднее время полета каждого БПЛА за 10 итераций до первой и второй позиций.

Ссылка на результат моделирования синхронизации полета двух дронов при мониторинге в агропромышленном комплексе (используемая программная среда имитационного моделирования: CoppeliaSim; используемое программное обеспечение: PyCharm): https://disk.yandex.ru/i/5V5cdEQcMMnaVg.

Выводы. В процессе работы проведен анализ технологии управления роем дронов и технологии автономного взлетно-посадочного бокса роем дронов; разработаны математические модели для вычисления расстояния между дронами при полете; разработано программно-алгоритмическое обеспечение для управления синхронизации полета роем дронов.

Преимущества использования роя дронов:

– снижение суммарной стоимости беспилотных летательных аппаратов;

– распределение полезной нагрузки на несколько сторон (возможность экономии на общей стоимости полезной нагрузки);

– снижение потерь от аварий;

– повышенная точность позиционирования каждого дрона за счет взаимного позиционирования;

– улучшение получаемых результатов за счет разных углов зрения разных дронов;

– ускорение получения результата в ряде применений.

Для эффективного управления роем и контроля взаимодействия между отдельными дронами необходимо решить две ключевые проблемы:

1. Проблему навигации и планирования путей для каждого дрона при недостатке информации об окружающем пространстве. В постоянно изменяющемся окружении помимо неподвижных препятствий присутствуют другие дроны из роя, столкновения с которыми необходимо избегать.

2. Создание мультикоптерами воздушных потоков, которые необходимо учитывать при близ-

Равномерное распределение: дистанция и среднее время полета каждого БПЛА
Uniform distribution: distance and average time of each agent agent

n	$\sum_{1}^{10} T_1/10$, с	$\sum_{1}^{10} D_1/10$, м	$\sum_{1}^{10} T_2/10$, с	$\sum_{1}^{10} D_2/10$, м
0	23.75	3.81	89.75	0.42
1	23.44	3.51	42.08	0.71
2	24.38	3.85	54.17	0.67
3	23.13	3.55	34.91	0.88
4	29.44	3.93	61.58	1.14
5	29.56	3.64	47.41	1.27

ком движении дронов, чтобы они не сдували друг друга с траектории.

Для применения группы дронов требуется решить следующие задачи:

– требуются высокие вычислительные возможности на борту беспилотных летательных аппаратов для взаимодействия БЛА в полете в составе группы и предварительной обработки собираемой информации в режиме реального времени;

– необходимы новые типы управляющего программного обеспечения;

– желательна интеграция системы управления группой дронов и программного обеспечения полезных нагрузок;

– беспилотники в рое не только не должны сталкиваться, но также не должны мешать друг другу создаваемыми ими воздушными потоками;

– беспилотники в рое должны иметь возможность коллективно распределять специальности и при необходимости перераспределять назначения, данные отдельным участникам роя, например при выходе из строя тех или иных специалистов.

Список литературы

1. Красовский А. Н., Суслова О. А. Упрощенная математическая модель управляемого движения квадрокоптера // Наукові записки Междунар. гуманитарного ун-та. 2016. Вып. 26. С. 146–150.

2. Красовский А. Н., Суслова О. А. Об оптимальном управлении движением дрона-квадрокоптера по критерию качества затрат энергии // Успехи современной науки и образования. 2017. Т. 4, № 3. С. 193–197.

3. Броников А. М., Круглов С. П. Упрощенные условия адаптируемой системы управления с идентификатор и эталонной моделью // Автоматика и телематика. 1998. № 7. С. 107–117.

4. Буков В. Н., Круглов С. П., Решетняк Е. П. Адаптируемость линейной динамической системы с идентификатором и эталонной моделью // Автоматика и телемеханика. 1994. № 3. С. 99–107.

Информация об авторе

Нгуа Ндонг Авеле Жак Бернис – магистрант кафедры квантовой и оптической электроники в технички СПбГЭТУ «ЛЭТИ». Обладатель степени бакалавра по мехатронике и робототехнике в Тамбовском государственном техническом университете, Россия.
E-mail: avelejacques@yahoo.fr

References

1. Krasovskij A. N., Suslova O. A. Uproshhennaja matematicheskaja model' upravljaemogo dvizhenija kvadrokoptera // Naukovi zapiski Mezhdunar. gumanitarnogo un-ta. 2016. Vyp. 26. S. 146–150. (In Russ.).

2. Krasovskij A. N., Suslova O. A. Ob optimal'nom upravlenii dvizheniem drona-kvadrokoptera po kriteriju kachestva zatrat jenergii // Uspehi sovremennoj nauki i obrazovanija. 2017. T. 4, № 3. S. 193–197. (In Russ.).

3. Bronikov A. M., Kruglov S. P. Uproshhennye uslovija adaptiruemoj sistemy upravlenija s identifikator i jetalonnoj model'ju // Avtomatika i telematika. 1998. № 7. S. 107–117. (In Russ.).

4. Bukov V. N., Kruglov S. P., Reshetnjak E. P. Adaptiruemost' linejnoj dinamicheskoj sistemy s identifikatorom i jetalonnoj model'ju // Avtomatika i telemehanika. 1994. № 3. S. 99–107. (In Russ.).

Information about the author

Jacques Bernice Ngoua Ndong Avele – master's student of the Department of Quantum and Optical Electronics in the Engineering Department of Saint Petersburg Electrotechnical University. He holds a bachelor's degree in mechatronics and robotics from Tambov State Technical University, Russia.
E-mail: avelejacques@yahoo.fr

Статья поступила в редакцию 22.09.2022; принята к публикации после рецензирования 25.11.2022; опубликована онлайн 30.01.2023.

Submitted 22.09.2022; accepted 25.11.2022; published online 30.01.2023.