Simulation of Efficient Cooperative UAVs using Modified PSO Algorithm

G.Pradeep Kumar[1], Dr.B.Sridevi[2]
Department of Electronics and Communication Engineering
Velammal College of Engineering and Technology, Madurai[1]
Velmmal Institute of Technology, Chennai[2]
Tamilnadu, India
gpk@vcet.ac.in[1], bsd@vcet.ac.in[2]

Abstract: - Unmanned Aerial Vehicles (UAVs), better known as drones, are one of the major technological developments of today. Groups of UAV are of special interest for their abilities to coordinate simultaneous coverage of large areas, or cooperate to achieve goals such as mapping. Cooperation and coordination in UAV groups also allows increasingly large numbers of aircraft to be operated by a single user. Our project aims to develop an algorithm for commanding multiple UAV's to co-operatively perform multiple tasks and to assign specify tasks to each vehicle without collision, congestion and overlapping by exact localizing each drone. We are planning to develop a real time algorithm that works possibly under communications constraints and other uncertainties and failures. For simulating of our algorithm, we use Dronekit, Ardupilot, pixhawk v and MAVlink protocol. Then we can embed the algorithm in real time in a Quadcopter using Ardupilot controller.

Key-Words: - Cooperative UAV; path planning; collision avoidance

1 Introduction

An Unmanned Aerial Vehicle (UAV), commonly known as a drone, is an aircraft or an airborne system that is remotely operated by a human operator or autonomously by an onboard computer. A UAV is defined as a "powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expandable or recoverable, and can carry a lethal or nonlethal payload. They are used for a variety of applications especially in military applications. UAVs are increasingly used because they have the advantage of not placing human life at risk and lowering operation costs. To realize these advantages, UAV must have a higher degree of autonomy and preferably work cooperatively in groups. Groups of UAVs, which work together as a single unit to complete a particular mission such as mapping, are referred to as Cooperative UAV.

Our Project focuses on the simulation of cooperative UAV with embedded path planning and collision avoidance algorithms such that they work autonomously. In this paper, we have simulated the UAVs using Dronekit SITL (Software In The Loop) and Mission Planner. The advantage of our method is that we can easily debug and modify the design as it is in simulation. Also, we can easily embed this in hardware as the software emulates that in the hardware board.

2 Related Works

Here we review the papers and algorithms that has been proposed in recent literature publications.

A. Cooperative UAV methods

With the advancement in advanced sensing and information technology cooperative UAV control can be achieved by using a variety of sensors. The existing methods deal about controlling the multiple UAVs using one remote controller. The safe flight of multiple UAVs can be done by maintaining a particular distance between each UAVs. The distance maintenance behavior can be achieved by using the principle that if two copters are so close to each other then the copters must move away autonomously similarly if the copters are away from each other then the copters must move closer autonomously. UAVs send their GPS co-ordinates to other UAVs thereby maintaining the distance. Cooperative UAV control can be achieved by using sensors and remote controllers.

B. Path planning

The key factor for cooperative UAV is path planning. The existing methods for path planning can be classified into two types: 1.Predefined flight path based search 2.Dynamic path planning. In predefined flight path method first the flight paths are generated in advance and the paths are followed during the execution. They are done by using sweep line based search. This method is effective so that no search areas are missed but not efficient due to the predefined paths and cannot be used for searching the dynamic targets. Dynamic programming, artificial intelligence, model predictive control can be used for an effective path planning. The optimal path has to be found out. There are several algorithms for path planning some of them are sampling-based algorithms, node-based algorithms, mathematical model based algorithms, Bio-inspired algorithms, and multi-fusion based algorithms.

C. Collision Avoidance

The UAVs are required to fly in the defined path and also to avoid collisions between each other. First the UAVs has to detect obstacles and then move accordingly. If an obstacle is detected then the UAV has to change it's location. Several sensors can be used to detect obstacles like ultrasonic and infrared sensors. These sensors detect the obstacles and inform the UAVs about the obstacles. Likewise after the detection of obstacles it has to know the position in which the objects are. The distance at which the objects are present can also be detected by using sensors. Sensors like object detection sensor and Fourier tracking sensor can be used. Collision avoidance generally refers to the ability of the vehicle to acknowledge dangers that are not originally known and act simultaneously.

3 Proposed Architecture

We use the Dronekit software for the simulation of the UAV. The Dronekit SITL is used for simulating the vehicle without the real one. We need virtual machine for simulating multiple drones as the SITL can simulate only one UAV. The Dronekit allows us to control the UAV using python programming language and to test bug fixes and other changes to the autopilot. This method uses MAVproxy to make the initial connections. As shown in the Fig.1, we run multiple UAVs in Dronekit SITL and connect to Mission planner using different TCP ports.

They each connect to the Mission planner using either TCP or UDP connection. The Mission planner provides a virtual environment to simulate the vehicle and implement the path planning and collision avoidance algorithms. We can set the waypoints and targets in the Mission planner in guided mode or .program them using python programming and thus run the vehicle autonomously.

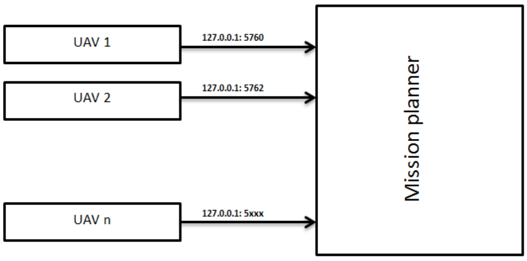


Fig. 1: Block Diagram

A. Dronekit-python

Dronekit-python is an open source and community-driven project. It is installed using python pip tool on all platforms. It is a project of Ardupilot created for connecting, controlling and monitoring a vehicle. Dronekit helps you to create powerful apps for UAVs. These apps can run in their companion computers. They can also perform tasks that are computationally intensive and use a low latency link. It provides compatibility with vehicles that communicate using the MAVlink protocol. One of its major advantages is that it uses python programming language. Python is an high-level programming language and easy to program and interpret. Dronekit-python runs on Linux, Mac OS X, or Windows.

Installation

Dronekit-python can be installed by the command:

```
pip install dronekit
```

B. Dronekit-python SITL

Dronekit-SITL is the fastest, simplest and easiest way to simulation on Windows, Linux, or MacOS X. The SITL simulator can be used to test the algorithms and processes for UAV without the real/physical vehicle.

1. Installation

Dronekit-SITL can be installed in all platforms using the following command:

```
pip install dronekit-sitl
```

The commands work only if the computer has python 2.7 or above.

2. Running SITL

We can run the any SITL vehicles using default settings

```
dronekit-sitl copter
```

or can input parameters such as home location, the vehicle model type (e.g. "quad"), etc.

```
dronekit-sitl plane-3.3.0 --
home=-
35.363261,149.165230,584,353
```

C. MAVlink

MAVLink or Micro Air Vehicle Link is a protocol for communicating with unmanned vehicles. It is used for communication between a ground control station and unmanned vehicles , and in the inter-communication of the subsystem of the vehicle. It can also be used to transmit the of the state vehicle, its GPS coordinates and other parameters.

D. ArduPilot

Ardupilot is open source autopilot software. It is the most advanced software capable of controlling any vehicle system imaginable, from conventional airplanes, multirotors, and helicopters, to boats and even submarines. It has both hardware and software thus allowing for testing before implementation.

E. Mission planner

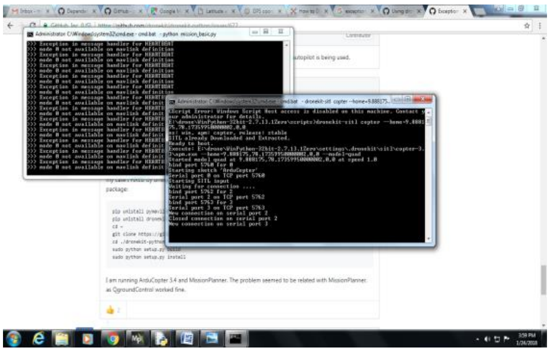
Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project. It has various options and features that help in monitoring and controlling the vehicle. In mission planner, we can view the location of the vehicle, its state parameters, RPY axis, and plenty of other options. We can also record the running of the vehicle as a video. Waypoints and missions can be programmed and can even be saved for later use. We can view the running of the Dronekit SITL in mission planner using a TCP or UDP connection.

4 Simulation of Dronekit SITL

After installing dronekit and dronekit SITL we can run the SITL for various vehicles such as copter, plane, quad, etc. To run the latest version of copter we can simply call:

```
dronekit-sitl copter
```

SITL will start and wait for TCP connections on 127.0.0.1:5760. We can also specify the version and other parameters of the vehicle.



```
>>> Exception in message handler for HEARTBEAT
>>> mode 0 not available on mavlink definition
>>> Exception in message handler for HEARTBEAT
>>> mode 0 not available on mavlink definition
>>> Exception in message handler for HEARTBEAT
>>> mode 0 not available on mavlink definition
>>> Exception in message handler for HEARTBEAT
>>> mode 0 not available on mavlink definition
>>> Exception in message handler for HEARTBEAT
>>> mode 0 not available on mavlink definition
```

```
Script Error: Windows Script Host access is disabled on this machine. Contact y
our administrator for details.
E:\drone\WinPython-32bit-2.7.13.1Zero\scripts>dronekit-sitl copter --home=9.8881
75,78.1735995000002,0,0
os: win, apm: copter, release: stable
SITL already Downloaded and Extracted.
Ready to boot.
Execute: E:\drone\WinPython-32bit-2.7.13.1Zero\settings\.dronekit\sitl\copter-3.
3\apm.exe --home=9.888175,78.1735995000002,0,0 --model=quad
Started model quad at 9.888175,78.1735995000002,0,0 at speed 1.0
bind port 5760 for 0
Starting sketch 'ArduCopter'
Serial port 0 on TCP port 5760
Starting SITL input
Waiting for connection ....
bind port 5762 for 2
Serial port 2 on TCP port 5762
bind port 5763 for 3
Serial port 3 on TCP port 5763
New connection on serial port 2
Closed connection on serial port 2
New connection on serial port 3
```

I am running ArduCopter 3.4 and MissionPlanner. The problem seemed to be related with MissionPlanner as QgroundControl worked fine.

Fig. 2: The dronekit SITL running in terminal window

We can connect the Mission planner to the Dronekit SITL after running it (Fig. 3). Also we can run the other python programs in the command prompt and view the results in Mission planner. The graph shown in the Fig. 4, of Mission planner is the graph of RPY axis while the drone is flying.

Fig. 3: Quad flying over a given location

The graph shown in the Fig. 4, of Mission planner is the graph of RPY axis while the drone is flying. We can give the location of the place we want to simulate the drone. The GPS location of the Fig. 4 is Velammal College of Engineering and Technology, Madurai.

Fig. 4: Cooperative UAVs simulation in Mission planner

In the simulation of cooperative UAVs in Fig. 4, we use different TCP ports for each vehicle. Also we can control the drones and position them in mission planner using swarm function. The drone can be either in auto mode or guided mode. In auto mode we can fly the UAV autonomously using the algorithm embedded using python code. Thus the cooperative UAV is achieved autonomously. The simulation of the drones are done using the Dronekit SITL in command prompt and it can be viewed using Mission planner.

In the mission planner, we can create the waypoints for the UAV to fly.

5 Proposed Algorithm

One of the existing algorithms for cooperative UAV is Particle Swarm Optimization (PSO). The main aim of PSO algorithm is to minimize the error between the UAV and target. The major drawback of PSO algorithm is when it is applied in real-time it doesn't produce greater results. The drawback of existing algorithms is the constant change of the speed of the vehicle. The vehicle speed has a direct relationship with the efficiency of vehicle. We have developed an algorithm with collision avoidance and with optimal path planning. For collision avoidance each UAV must have to maintain a particular distance. Our algorithm aims to maintain a constant velocity in order to avoid collisions with objects. To avoid collision between vehicles a threshold boundary value has to be set. If the vehicle enters into the threshold boundary then it has to automatically change it's location. To maintain constant velocity a linear approach is followed. The constant velocity increases the efficiency of vehicle as well as it reduces the unnecessary strain on the vehicle. If an obstacle is detected then the vehicle has to change its direction instead of the speed of vehicle. By changing the direction the turn angle will be minimum which reduces the sudden path change of a vehicle thereby maintaining constant velocity. We have simulated our algorithm in Mission Planner and we have written our program in Python. Other parameters like turn angle, visual range and altitude are still adjustable in the program.

5 Collision Avoidance & Path Planning

In cooperative UAV, the major challenge is obstacle/collision avoidance and path planning. While the groups of UAVs are flying, they need to discover a path which is obstacle free and also does not cause them to collide with each other. To achieve this we need an algorithm to find the best possible path while the UAV is flying. The various algorithms that can be used to find the best path are Dijkstra's algorithm, Bellman Ford's algorithm, Floyd-Warshall's algorithm and the AStar algorithm. Among these algorithms AStar algorithm is found to be best as it doesn't chooses next state only with lowest heuristics value but one that gives lowest value when considering its heuristics and cost of getting to that state.

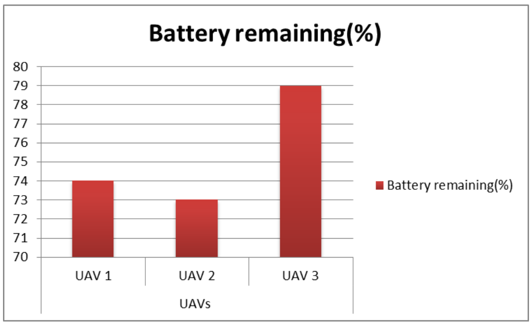
Also to avoid collisions between the UAVs that are flying we program them to maintain a particular distance between them. This is made possible by getting the GPS location of each drone. If the drones move too close to the critical distance of each other then they are made to move farther. If they are too far away then they are made to come closer to maintain the formation. Thus by this method we achieve both collision avoidance and path planning.

6 Results and Discussions

In Table I, the results obtained from implementing modified PSO algorithm in Mission Planner are tabulated. We have used our algorithm to determine the correct heading and maneuver multiple UAVs toward the destination at a constant velocity. For keeping a constant velocity, our modified PSO algorithm can be used instead of the previous algorithms. To reach the destination, the direction the UAV is heading needs to change rather than the velocity. Our approach is simpler and more stable since the UAVs will not quickly change directions and locations. Maintaining a constant speed while searching and moving will ensure better fuel efficiency. Our algorithm has faster computational time, and a straightforward linear simulation that can be easily scaled for a larger search space.

Parameters	**UAVs**		
	UAV 1	***UAV 2***	***UAV 3***
Altitude(m)	1.20	0.97	2.38
Battery remaining(%)	74	73	79
Vertical speed(m/s)	5.86	5.98	5.83
Dist. to WP(m)	0.05	0.08	0.06
Dist. traveled(m)	3.5	3.2	4.1
Yaw(deg.)	154.17	344.36	99.19
Wind Speed(m/s)	3	1.8	2.2
Threshold distance(m)	1.5	1.5	1.5

TABLE I. Readings from the Mission planner



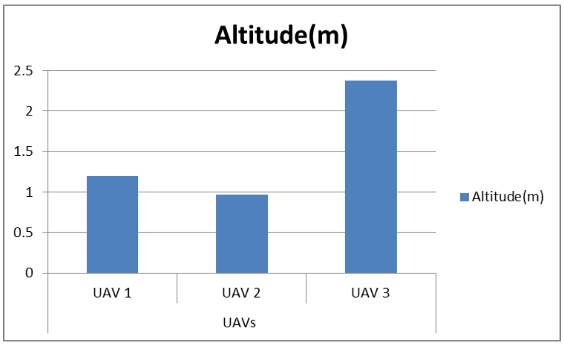


FIG. 6: GRAPH FOR BATTERY PERCENTAGE AND ALTITUDE

The graph in Fig. 6 shows the relationship between UAV parameters altitude and battery percentage.Table II gives the comparison of readings from existing PSO algorithm and our modified PSO algorithm.

PARAMETERS	PSO			MODIFIED PSO		
	Vehicle 1	Vehicle 2	Vehicle 3	Vehicle 1	Vehicle 2	Vehicle 3
Velocity(m/s)	5.1	5.033	4.465	5.86	5.98	5.83
Fuel efficiency(%)	64	72	71	74	73	79
Average cost time(ms)	76	80	93	65	67	73

TABLE II. Comparison of readings from PSO and Modified PSO

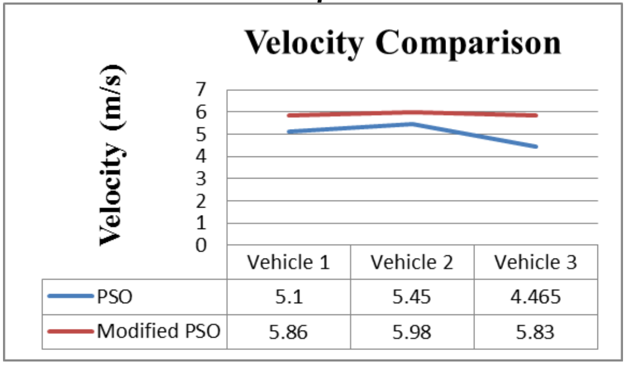


Fig. 7: Comparison graph for velocity

The graph in Fig. 7 indicates that the modified PSO has a constant velocity. Our approach is simpler and more stable since the UAVs will not quickly change directions and locations. Maintaining a constant speed while searching and moving will ensure better fuel efficiency. Our algorithm has faster computational time, and a straightforward linear simulation that can be easily scaled for a larger search space.From the graph it is inferred that modified PSO algorithm is more fuel efficient than conventional PSO algorithm.. Our results show that the velocity is constant and thereby it increases the fuel efficiency and reduces the strain on the platform.

7 Conclusion

In this paper, we have simulated the cooperative UAV with collision avoidance and path planning algorithm. Also, we have created an effective method for cooperative UAVs to perform a mission autonomously. The future scope for this project is that the aerial images captured using cooperative UAVs can be used the images capture by the drones in 2D Orthomosaics to form a complete map of the given area. The 2D orthomosaics is only one of the applications of cooperative UAV. Some of the other applications are precision agriculture, 3D mapping, emergency rescue missions and various other applications.

References:

[1] Christopher M. Eaton , Edwin K. P. Chong and Anthony A. Maciejewski "Multiple-Scenario Unmanned Aerial System Control" (January,2015)
[2] Xiaohua Wang , Vivek Yadav , and S.N. Balakrishnan "Cooperative UAV Formation Flying with Obstacle/Collision Avoidance "(June 2010)
[3] Stephen Lazzaro "Flying multiple drones from 1 remote controller" (April,2008)
[4] Csaba Virágh , Gábor Vásárhelyi, Norbert Tarcai , Tamás Szörényi , Gergő Somorjai, Tamás Nepusz, Tamás Vicsek "Flocking algorithm for autonomous flying robot"s (September,2015)
[5] C. (Cees) Trouwborst "Control of Quadcopters for Collaborative Interaction "(June,2014)
[6] Xiaoting Ji, Xiangke Wang, Yifeng Niu, and Lincheng Shen "Cooperative Search by Multiple Unmanned Aerial Vehicles in a Nonconvex Environment "(June,2015)
[7] Yosi Ben-Asher, Moran Feldman, Sharoni Feldman and Pini Gurfil "Distributed Decision and Control for Cooperative UAVs using Ad-Hoc Communication" (April,2008)
[8] Andrew Zulu, Samuel John "A Review of Control Algorithms for Autonomous Quadrotors(Open Journal of Applied Sciences, 2014)
[9] Prateek Burman, B.S.E.E "Quadcopter stabilization with Neural Network"(December 2016)

[10] H. de Plinval, A. Eudes, P. Morin "Control and estimation algorithms for the stabilization of VTOL UAVs from mono-camera measurements"(June 17, 2014)

[11] Cameron Roberts, Electrical Engineering "GPS Guided Autonomous Drone" (April 25, 2016)

[12] Tyler Ryan and H. Jin Kim "Smartphone Quadrotor Flight Controllers and Algorithms for Autonomous Flight"(December 2017)

[13] Luis E. Romero, David F. Pozo, Jorge A. Rosales "Quadcopter stabilization by using PID controllers" (2014)

[14] Quintero, S.A.," Optimal UAV coordination for target tracking using dynamic programming. Decision and Control "(CDC) (2010)

[15] Yosi Ben-Asher, Moran Feldman, Sharoni Feldman and Pini Gurfil " Distributed Decision and Control for Cooperative UAVs using Ad-Hoc Communication" (2012