2017 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 13-16, 2017, Miami, FL, USA

Multi-UAV Control Testbed for Persistent UAV Presence: ROS GPS Waypoint Tracking Package and Centralized Task Allocation Capability

Brad Hyeong-Yun Lee[1], James R. Morrison[1†] and Rajnikant Sharma[2]

[1]Department of Industrial and Systems Engineering, KAIST, Daejeon, Korea
[2]Department of Aerospace Engineering and Engineering Mechanics, University of Cincinnati, OH, 4221, USA
[†]Corresponding author (Tel : +82-42-350-3127; E-mail: james.morrison@kaist.edu)

Abstract: A system of unmanned aerial vehicles (UAVs) can be used to provide uninterrupted service to customers by handing the mission from a weary UAV to a fresh one in an automated manner. Toward such a system, we discuss the development of two components: ROS GPS waypoint tracking package and centralized task allocation network system (CTANS). The ROS GPS waypoint tracking package is the first open source software developed for the AR Drone 2.0 UAV that enables integration with a larger system. CTANS connects the UAV host computers to a central computer, receives the UAV status information and distributes GPS flight directives. CTANS is responsible for distributing the commands to the UAVs to ensure mission handoffs are conducted as required. This multi-UAV control software testbed was used to implement a small scale, outdoor, persistent, GPS based patrol mission with two UAVs.

Keywords: Unmanned aerial vehicle (UAV), Robot Operating System (ROS), UAV GPS waypoint tracking, Network programming, Multi-UAV systems.

1. INTRODUCTION

Though UAVs can be used to conduct various missions, in the case of small UAVs, they suffer from short mission duration due to limited battery or fuel capacity. As such, a single UAV has limited potential. To overcome the limitation of a single UAV, systems of UAVs may be used to extend mission duration and provide uninterrupted service; e.g., [1, 2]. Such persistent service can be realized by handing off the mission from one UAV to another. Thus, when a UAV is conducting a mission and its battery or fuel level has drained to a desired threshold, a fresh replacement UAV should be directed to assume responsibility for the mission. The weary UAV may then return to a fuel service station for replenishment.

The handoff of a mission from one UAV to another is depicted in Fig. 1. A patrol route has been provided and a UAV should follow the path from start to end. UAV1 prosecutes the initial stages of the patrol, but grows weary midway through. An autonomous system should send a replacement UAV to conduct the second half of the patrol. UAV1 thus hands off the mission to UAV2. UAV1 may then retire to a fuel service station.

While there are many components required to implement such a system, here we describe the development of two: ROS GPS waypoint tracking package and CTANS for use with the Robot Operating System (ROS). We focus on the ROS as it is open source middleware for robots systems and is a key resource for the robotics community.

The work was supported by National Research Foundation of Korea NRF-2016R1A2B4010132

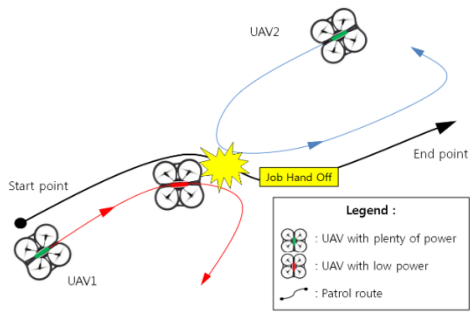


Fig. 1 UAV1 hands off the patrol mission to UAV2.

The first component's purpose is to deliver directives to the UAV autopilot. We refer to this component as the UAV director. In our system, we want the UAVs to patrol an area or follow paths defined by GPS waypoints. As such, the directive function is provided by GPS waypoint tracking software. Though GPS waypoint tracking software exists for use with the AR Drone, none are directly applicable for use in an integrated multi-UAV system. We developed an ROS package for this purpose by extending the existing open source software ardrone_autonomy [3] and tum_ardrone [4].

The second component's purpose is to conduct network level communication between the local task manager (LTM) for each UAV and a central task manager (CTM). The CTM should deliberate on the system state and objectives and determine which tasks will be assigned to each UAV. The LTM for each UAV will receive and execute these commands as well as provide UAV state information to the CTM.

The physical hardware and connections for a multi-UAV system are depicted in Fig. 2. The system consists of a central computer, router, UAV host computers and UAVs.

To demonstrate the joint operation of these two components developed for the ROS, we constructed a multi-UAV system. It consists of two UAVs, two UAV host computers, a router and one central computer. We conducted an experiment with this system that provides a persistent patrol service about a region defined by GPS waypoints. The persistent patrol is accomplished via mission handoff from one UAV to another as dictated by the CTM.

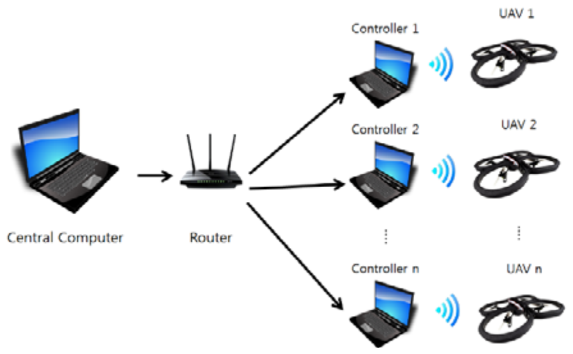


Fig. 2 Multi-UAV system.

To our knowledge, the existing UAV systems that have been developed for persistent operations and handoffs have been conducted indoors under the watchful eye of Vicon infrared position detection systems. Our persistent handoff based system operates outdoors in a GPS enabled environment in which the UAV state information (such as UAV position and velocity) is obtained only from onboard sensors and is developed for the ROS.

The contributions of this paper are as follows. We

- Develop an open source ROS package for GPS waypoint tracking which is compatible with multi-UAV systems conducting autonomous missions.
- Develop a CTANS with periodic automatic data transfer.
- Provide a tutorial on the installation and use of the ROS package and CTANS.
- Develop a small scale prototype multi-UAV control testbed for the ROS capable of conducting autonomous outdoor GPS based missions and implementing mission handoffs.

The paper is organized as follows. In Section 2, we review existing open source software which provides UAV control capability and some related test beds. Section 3 describes how we developed our ROS package and provides a tutorial of its use. The design of CTANS and the other components are introduced in Section 4. In Section 5, we describe our testbed and patrol demonstration. Concluding remarks are provided in Section 6.

2. RELATED SOFTWARE AND TESTBEDS

Here we review relevant existing ROS tracking packages and UAV testbeds intended for long term mission fulfillment via mission handoff.

2.1 ROS tracking packages

GPS waypoint tracking capability is important for UAVs to perform missions such as border patrol in an outdoor environment. There are many UAV products with built in GPS tracking capability. There are also many software programs that have been developed to allow UAVs to track GPS waypoints. Examples of software for GPS waypoint tracking include ardrone_autonomy, which is Autonomy Lab's ROS package for the AR Drone 2.0 [5], and QGroundControl [6], which is also compatible with ROS and the AR Drone 2.0.

Although ardrone_autonomy is capable of GPS waypoint tracking and is compatible with other ROS packages, ardrone_autonomy is not sufficient for our objectives. Three capabilities are not provided: autonomous missions cannot be performed, waypoint information must be input manually, and a GUI is not provided [3].

QGroundControl provides a fairly complete GPS waypoint tracking capability for use with the ROS. It can take inputs from a file and provides a GUI. However, it cannot be conveniently combined with other software and is difficult to modify [6].

FalkorSystems' tum_ardrone [4] is vision based tracking software for the ROS. tum_ardrone can read input files, provides a GUI and is easy to combine with other ROS packages. It is not for GPS tracking.

We seek to develop an ROS package for GPS waypoint tracking with the AR Drone 2.0 that can be used in a larger system. To do so, we extend the tum_ardrone package, which relies on ardrone_autonomy, to include GPS tracking functionality.

2.2 Relevant multi-UAV testbeds

Much research has been conducted toward the development of outdoor UAV systems. In [7, 8], an overview of aircraft systems is provided with a focus on sense and avoid technologies. While numerous testbeds have been developed for UAV systems, few exist for use in persistent missions. The RAVEN testbed [9-12], developed by the Aerospace Controls lab at MIT, is one with such capability. Their indoor facility uses the Vicon infrared motion capture system to accurately estimate UAV position. They accomplish long term persistence with the use of autonomous recharge stations [13-16].

Our goal is to develop components and a testbed for use in outdoor GPS enabled environments with the ROS system for AR Drone 2.0 UAVs.

3. ROS PACKAGE DESIGN

ROS is middleware used for developing robot software. ROS provides various useful libraries and tools to help developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more [17]. ROS is open source and widely used in the robot community. While ROS supports various languages, we use C++ for this research.

Parrot's AR Drone 2.0 can be equipped with a GPS module capable of providing the current UAV location in terms latitude and longitude [18].

Our ROS package was developed by exploiting features of the ardrone_autonomy ROS package and the tum_ardrone ROS package. While the ardrone_autonomy package allows GPS waypoint targets to be specified and tracked, it requires manual input. We extend the input capability of the tum_ardrone for autonomous operation and integrated it with ardrone_autonomy. The resulting package provides GPS tracking functionality with the desired autonomous capability.

The remainder of Section 3 is organized as follows. In Section 3.1, we provide a tutorial for setting up the development environment including Ubuntu, ROS, ardrone_autonomy and tum_ardrone installation. Note that the tum_ardrone installation is not required for our package, but may be desired by a developer. In Section 3.2, we provide a discussion on the development of our package. In Section 3.3, we provide a tutorial for the installation of our package and brief instructions for use with a single UAV. (Use with multiple UAVs will be discussed in Section 4.)

3.1 Development environment set up

The first step is to install Ubuntu 14.04 LTS and ROS Indigo on a PC. For Ubuntu 14.04, it is recommended to download Ubuntu 14.04 from the official website ([19]) on a DVD or USB and follow steps of the Ubuntu installation wizard. ROS Indigo ([20]) can be easily installed by referring to wiki.ros.org. Note that ROS Indigo only supports Ubuntu 13.10 and 14.04. The installation procedure for Ubuntu 14.04 and ROS follows.

1. Configure Ubuntu repositories.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

2. Setup sources.list.

```
sudo apt-key adv --keyserver
hkp://ha.pool.sks-keyservers.net --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

3. Set up keys.

```
sudo apt-key adv --keyserver
hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

4. Install ROS Indigo.

```
sudo apt-get update
sudo apt-get install ros-indigo-desktop-full
```

5. Initialize rosdep.

```
sudo rosdep init
rosdep update
```

6. Environment setup.

```
echo "source /opt/ros/indigo/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

7. Download rosinstall.

```
sudo apt-get install python-rosinstall
```

8. Create a ROS workspace.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
cd ~/catkin_ws/
catkin_make
```

One may now proceed to install ardrone_autonomy and tum_ardrone.

The ardrone_autonomy is a ROS driver for the Parrot AR.Drone 1.0 and 2.0 quadcopters. This driver is based on the official AR-Drone software development kit (SDK) version 2.0.1. The package ardrone_autonomy is a fork of the AR-Drone Brown driver [21]. In our work, we used the gps-waypoint branch of the ardrone_autonomy package. The ardrone_autonomy package can be installed on the UAV host computer as follows [5]:

```
cd ~/catkin_ws/src
git clone
https://github.com/AutonomyLab/ardrone_autonomy.git
-b indigo-devel
cd ~/catkin_ws
rosdep install --from-paths src -i
catkin_make
```

The tum_ardrone is a ROS driver for the Parrot AR.Drone 1.0 and 2.0 quadcopters. tum_ardrone uses the ardrone_autonomy package. The tum_ardrone package can be installed on the UAV host computer by executing the following commands [4]:

```
cd ~/catkin_ws/src
git clone https://github.com/tum-vision/
tum_ardrone.git -b indigo-devel
cd ~/catkin_ws
rosdep install tum_ardrone
catkin_make
```

3.2 ROS package development

The ROS ardrone_autonomy package provides manual GPS waypoint tracking capability. A UAV can track a single GPS waypoint target using the following commands.

```
roscore
rosrun ardrone_autonomy ardrone_driver
_enable_navdata_gps:=True
rosservice call ardrone/setgpstarget "target:
   id:
      uuid: "uuid_msgs/UniqueID"
   position:
      latitude: 36.367164
      longitude: 127.363472
      altitude: 1.5
   props:
   - key: 'velocity'
      value: '0.1'"
rostopic pub ardrone/takeoff std_msgs/Empty
rosservice call ardrone/setautoflight "enable: True"
rostopic pub ardrone/land std_msgs/Empty
```

As mentioned previously, tum_ardrone uses ardrone_autonomy. In Fig. 3, the connections between three of the nodes in tum_ardrone and the ardrone_drive node of ardone_autonomy are depicted.

Our new package extends integrates the functionality of these packages and extends them to allow for automatic operations. The flow of operations in our package is provided next.

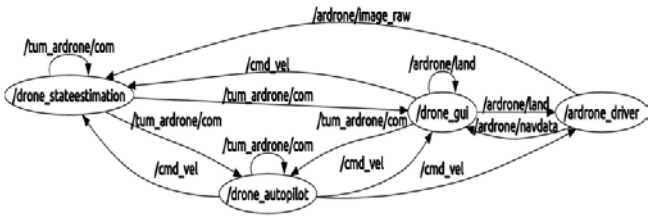
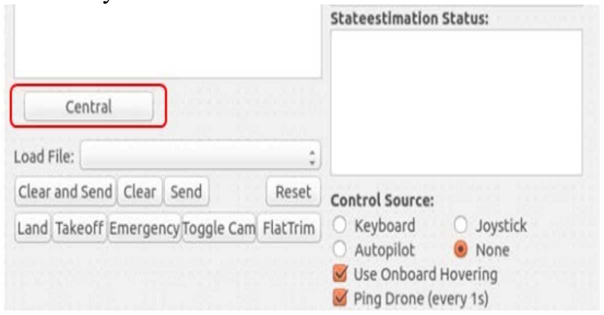


Fig.3 Node graph of existing ROS packages.

The QT GUI interface for our package is provided in Fig. 4. This QT GUI extends the QT GUI of the original tum_ardrone by adding the "Central" button. (We used QT Creator for this purpose.) This button will execute software so that the UAV will follow the commands provided by the CTM.



Control Source:
Keyboard Joystick
Autopilot [✓ None]
[✓ Use Onboard Hovering]
[✓ Ping Drone (every 1s)]

Fig.4 Modified QT GUI of tum_ardrone.

In our implementation, tum_ardrone periodically sends the GPS waypoint tracking command to the UAV. Clicking the "Central" button causes the package to execute the following operations repeatedly on a periodic basis.

1. Read text file of commands from the CTM.

ifstream is used to read the text file and push_back is used to write values on a vector. The text file of CTM commands, located in the "flightPlans" folder, contains three lines of numeric values for latitude, longitude, and altitude. The manner in which the CTM generates this file is described in Section 4.

2. Send target latitude, longitude, and altitude from tum_ardrone to ardrone_autonomy.

The GPS target parameters are sent from tum_ardrone to ardrone_autonomy via the message ardrone_autonomy::SetGPSTarget with the vector of values for latitude, longitude, and altitude.

3. Send target latitude, longitude and altitude from ardrone_autonomy to the UAV.

Execute the ROS call service function with the message ardrone_autonomy::SetGPSTarget to the service ardrone/setgpstarget.

4. Set auto flight mode.

Execute the ROS call service function with the message ardrone_autonomy::RecordEnable to the service ardrone/setautoflight. This will cause the UAV to enter the auto flight mode and move to the designated waypoint.

For the UAV to track a fixed target GPS position, our package uses feedback control based on the UAV's current GPS position and battery level. The required UAV state data stream is obtained when our package subscribes to the ardrone/nav_data and ardrone/navdata_gps topics inside of ardrone_autonomy with the messages ardrone_autonomy::Navdata and ardrone_autonomy::navdata_gps, respectively.

For the CTM to track the UAV progress, determine if a waypoint has been reached, provide a new waypoint target and monitor the UAV battery level, our package must send the UAV state information to the CTM. This is accomplished via a LTM resident on the UAV host computer. The UAV state data for the LTM is provided from our ROS package via ofstream. ofstream is used to write the appropriate numeric value in the latitude, longitude and battery text files.

3.3 Installation and use

Our ROS package has been uploaded on GitHub at https://github.com/Brad-Lee/xS3D_GPS [22]. Our extensions to tum_ardrone can be found in the section "//added part". Our package can be installed as follows.

```
cd ~/catkin_ws/src
git clone https://github.com/Brad-Lee/xS3D
-GPS.git
cd ~/catkin_ws
rosdep install tum_ardrone
catkin_make
```

Note that it not necessary to also install the original tum_ardrone as our package contains all of its functionality.

After installation, connect the UAV to a PC using the Wi-Fi signal from the AR Drone 2.0. After a connection has been established, it is recommended to use a different Wi-Fi channel for each UAV to avoid Wi-Fi signal interference. For example, if three UAVs are to be employed simultaneously, the use of channels 1, 6, 11 for UAVs 1, 2 and 3, respectively, is recommended. The Wi-Fi channel can be changed as follows (where # indicates the channel number to be used.)

```
telnet 192.168.1.1
iwconfig ath0 channel # commit
```

Next, run the launch file (this starts the master server and ROS nodes) with the following commands.

```
roslaunch tum_ardrone ardrone_driver.launch
roslaunch tum_ardrone tum_ardrone.launch
```

The PC will now show 3 windows including the QT GUI window. The buttons has the following functions:

- Takeoff: UAV will takeoff.
- Land: UAV will land.
- Central: UAV will track the target designated by the CTM.

Note that the "Central" button will only function in conjunction with the CTM.

4. MULTI-UAV SYSTEM DESIGN

In this paper, a multi-UAV system which is capable of controlling more than a single UAV using the developed UAV director, GPS waypoint tracking ROS package, was designed and built. The developed multi-UAV system aims at autonomous operation of UAV control for mission execution in addition to continuous execution of specific tasks. This section provides ideas and how to design and build these multi-UAV systems.

The process of executing the mission is as follows. First, the flight plan to be executed by each UAV is generated according to the mission that the CTM wishes to perform. Then, when the flight plan is transmitted to the UAV host computer corresponding to each UAV through the router, the LTM receives the flight plan and transmits the appropriate command to the connected UAV. In order to implement the concept of the multi-UAV system described, connection between the central computer and the UAV host computer must be constructed, and the CTM and the LTM must do their respective tasks in order to perform the mission. From the overall system point of view, each component is represented by architecture as in Fig. 5.

The network level communication components, including the CTM and each LTM, is referred to as the CTANS. CTANS serves to deliver tasks from the CTM to the LTM for each UAV as well as send information on the UAVs status from the LTM to the CTM. Our focus here is on the network communication component of CTANS and not the CTM itself. The CTM that we use is rudimentary and provides basic directives. Our network software that performs these functions was developed with the Java network programming library and compatible for use with the ROS GPS tracking package.

The CTM resides on the central computer. The network communication components of CTANS consist of a network server on the central computer and a network client in the LTM for each UAV. A router connects the central computer to the UAV host computers which have different IP and port addresses. Further, UAV host computers also serve to send those directives to the UAV autopilot via the AR Drone 2.0 onboard Wi-Fi signal.

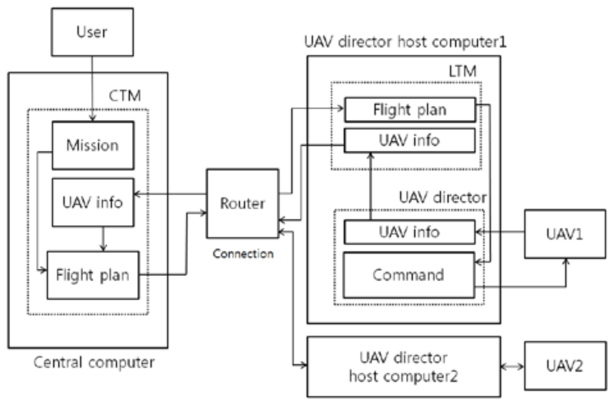


Fig. 5 Multi-UAV system architecture.

When the user inputs a mission for the required service on the central computer, the CTM receives information about the mission and current UAV status and generates the flight plan. At this time, the flight plan is determined differently according to each UAV, and the corresponding flight plan is transmitted to each LTM through the router. The LTM creates commands to control the UAV according to the received flight plan and controls the UAV with it. The UAV receives the command and sends information about its current GPS position and battery status to the UAV host computer and sends information about the current mission status and its mission persistence back to the central computer. Collectively, the multi-UAV system conducts network level feedback control.

In order to construct the above system, it is necessary to construct a network through a router first. In this study, we used Java network programming for this purpose. Network programming is a communication program that sends and receives data to and from another computer.

In Java network programming, servers and clients exist. A server is a computer or program that provides information and a client is a computer or program that requests information. In our system, the central computer is basically providing the flight plan to the UAV host computer according to the given mission. Therefore, the communication program is written with the central computer as the server and the UAV host computer as the client as Fig. 6.

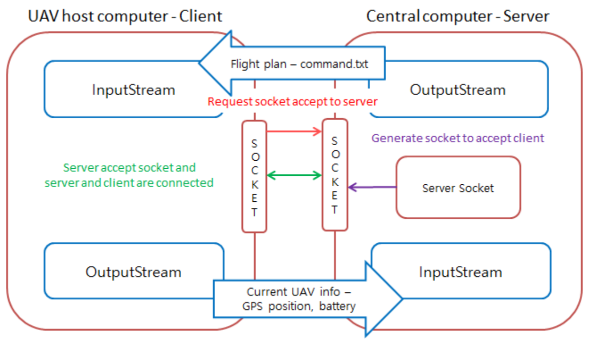

UAV host computer - Client
Central computer - Server
Flight plan – command.txt
InputStream
OutputStream
Request socket accept to server
Generate socket to accept client
Server accept socket and server and client are connected
SOCKET
SOCKET
Server Socket
OutputStream
InputStream
Current UAV info – GPS position, battery

Fig.6 Network programming architecture.

4.1 Central computer - server

The central computer acts as a server in the network program. It transmits the flight plan to the UAV host computer (client) according to the mission to be performed. The flight plan is generated based on the current GPS position and battery information which are transmitted to the UAV host computer by the UAV.

The source code server.java, which contains the CTM program, can be downloaded from GitHub https://github.com/Brad-Lee/xS3D_Network [21]. Executing the file will conduct the following tasks which will then allocate the desired actions to the LTM residing on the UAV host computers.

The tasks conducted by the CTM are as follows:

1. Open a new server socket to accept the socket from the client.

2. Query InputStream from socket to obtain information about the current status of UAV from client.

3. Compile the mission to be performed with the UAV information received, and create it as a command text file which will be located in “flightPlans” folder through FileWriter.

4.2 UAV host computer - client

The UAV host computer acts as a client in the network programming. It transmits the current GPS position and battery information of UAV to the central computer (server). Client receives the flight plan to generate the command that each UAV should execute from server, and our ROS package generates the command and sends it to the UAV to control it. These tasks are accomplished as follows.

1. Read text file and send the information to central computer based on the text file. (This text file is generated by ROS package, as explained in Section 3.)

2. Create a socket and request a connection to the server. (Each UAV host computer should have different IP each other.) The generated socket is used to send the UAV information to the server through OutputStream.

The source code client.java, which contains the LTM program, can be downloaded from GitHub https://github.com/Brad-Lee/xS3D_Network [21].

LTM program will conduct above tasks to receive task allocated by CTM.

In order to run CTANS, the server in CTM should be run first and the client in LTM should be run. With running CTANS, the information about current mission operation includes position of UAVs and battery state of UAVs will be detected repeatedly. Based on that information, CTM sets the mission and transmit the GPS waypoint target to LTM, and LTM receives GPS waypoint target and send it to ROS package on the UAV host computer.

In order to operate the multi-UAV system, network program should be operated first, and then, each UAV host computer should run the ROS package so that appropriate task assignment and execution are performed. In other words, in the central computer, only the CTM (server) is executed and in the UAV host computer, LTM (client) and ROS package are executed. Additional UAVs and their host computers can be added by installing LTM and our ROS package on the host (with different IP, port addresses and Wi-Fi channels).

5. EXPERIMENT

In this study, we developed a prototype system that performs specific tasks using the developed multi-UAV system. The prototype system was constructed to a minimum size. Two UAVs, two UAV host computers, one router and a central computer are used. UAV host computers and central computer are connected to each other via router. See Fig. 7.

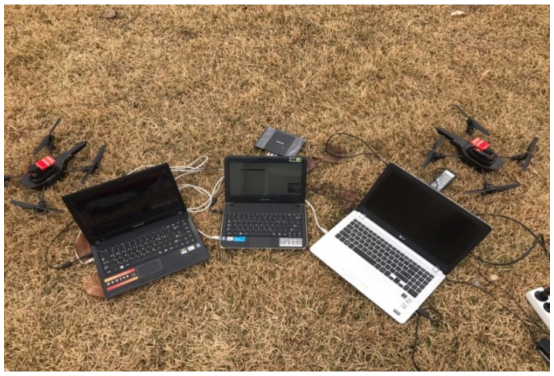


Fig. 7 Multi-UAV system components.

With this prototype system, we performed a simple mission that explores relatively small area continuously. The waypoints are designated in advance as shown in Fig. 8. When the mission is placed, the first UAV will track the 4 waypoints sequentially and perform the search mission.

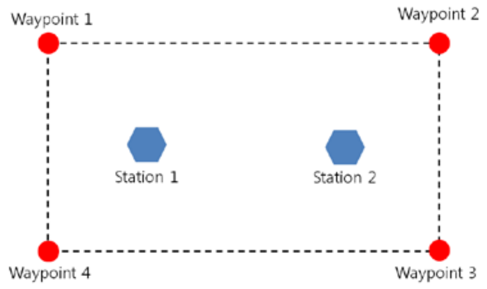


Fig.8 Designated waypoints.

A single UAV has a limited mission time, so when a UAV has reached insufficient battery capacity (30%) after a while, the second UAV will take over and carry on first UAV's mission alternatively.

All control during the patrol mission is autonomous. That is, the user is no longer involved in the system, and there is no need to do any operation except takeoff and land. These autonomous systems are made possible by the use of loop statement in each program, and detailed instructions can be found in the code at GitHub [22,23].

In conclusion, the experiment was successful. When the first UAV tracks the waypoints and performs a local patrol mission that draws a square, it returns to the station at the moment of the battery level drops below 30%. The second UAV went to the point at where the first UAV finished its patrol mission to continue the mission that was in progress. In other words, two UAVs maintained the persistency of the mission by implement mission handoffs. The central computer was able to monitor the whole progress of these tasks. The experiment video is available on YouTube ([24]). The steps of the experiment are depicted in Fig. 9-13.

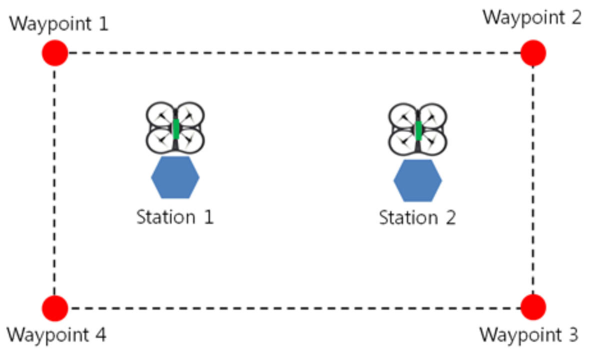


Fig. 9 Step 1 of experiment

Two UAVs with plenty of battery are placed at stations respectively at the first moment of experiment.

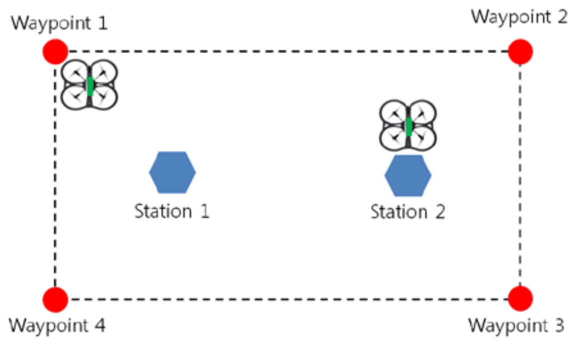


Fig. 10 Step 2 of experiment

When patrol mission was started, the first UAV went to the waypoint1 and draw a square.

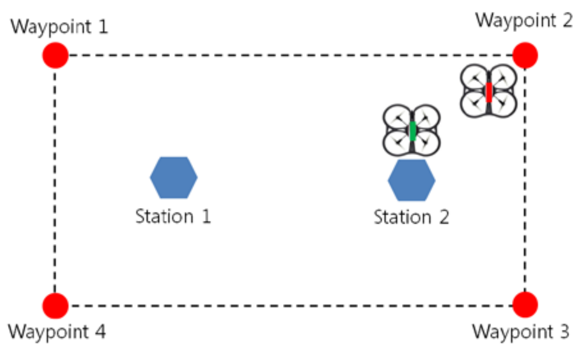


Fig. 11 Step 3 of experiment

After drawing a square several times, the first UAV had insufficient battery level (30%) to continue the mission at waypoint2.

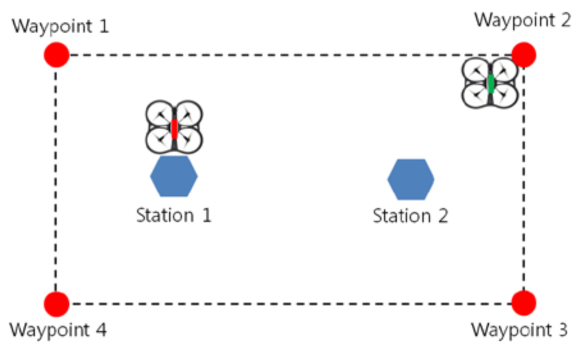


Fig. 12 Step 4 of experiment

The second UAV with plenty of battery went to waypoint2 to carry out mission instead of old UAV and old UAV went back to station1.

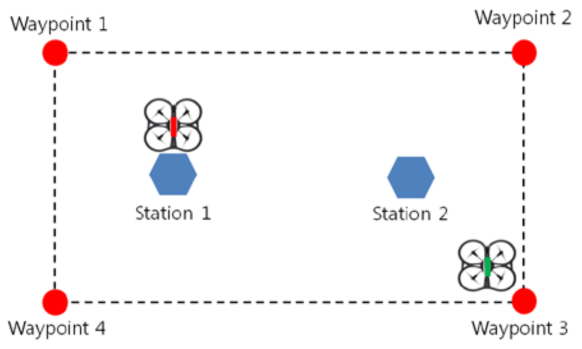


Fig. 13 Step 5 of experiment

The second UAV went to waypoint3 and draw a square to maintain persistency of patrol mission.

6. CONCLUDING REMARKS

Looking toward a persistent multi-UAV system in an outdoor environment, we developed several components: an open source ROS GPS waypoint tracking software, tutorial for the development and use of the software and a network system to connect the system components. An open source GPS waypoint tracking ROS package was developed which allows UAV to track GPS waypoint target. It is more convenient than the existing software and configured for use in automated systems. Moreover, the GPS waypoint tracking function could be customized to perform certain tasks. We constructed a CTANS to control multiple UAVs simultaneously. This allows the central computer (CTM) to monitor and operate UAV host computers (LTM). Such a system is centralized, so our multi-UAV system has centralized task allocation capability. Based on the developed ROS package and CTANS, this research built a prototype of multi-UAV system which is able to implement mission handoff by building and testing the system.

Future work includes the integration of the existing GPS tracking capability with vision based tracking. Further work on landing on a recharge platform is required.

REFERENCES

[1] Song, BD., Scheduling Problem for Persistent Automated Robotic Service, Ph.D. Thesis, Korea Adv. Inst. Science. Techn., Daejeon, Republic of Korea, 2015.
[2] Kim, J., Morrison, J.R.: On the concerted design and scheduling of multiple resources for persistent UAV operations. In: Proceedings of ICUAS, 2013.
[3] AutonomyLab, Github, AutonomyLab/ardrone_autonomy, ROS driver for Parrot AR-Drone 1.0 and 2.0 quadrocopters, December 06, 2016, Last accessed May 7, 2017,
" https://github.com/AutonomyLab/ardrone_autonomy/"
[4] FalkorSystems, Github, FalkorSystems/tum_ardrone, Last accessed May 7, 2017,
"https://github.com/FalkorSystems/tum_ardrone"
[5] Installation — ardrone_autonomy indigo-devel documentation, Installation, Last accessed May 7, 2017, "http://ardrone-autonomy.readthedocs.io/en/gps-waypoint/installation.html"
[6] QGroundControl - Drone Control, QGC, Last accessed May 7, 2017, "http://qgroundcontrol.com/"
[7] Xiang Yu and Youmin Zhang, Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects, Progress in Aerospace Sciences, vol. 74, pp. 152-168, 2015.
[8] Yu Fu, Xiang Yu, and Youmin Zhang, An Advanced Sense and Collision Avoidance Strategy for Unmanned Aerial Vehicles in Landing Phase, IEEE Aerospace and Electronic Systems Magazine, vol. 31, pp. 40-52, 2016.
[9] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. IEEE Control Systems Magazine, 28(2):51-64, April

2008.
[10] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. Autonomous Robots, 30(1):73-86, January 2011.
[11] S. Lupashin, A. Schollig, M. Hehn, and R. D'Andrea. The flying machine arena as of 2010. In IEEE International Conference on Robotics and Automation (ICRA), pages 2970-2971. IEEE, 2011.
[12] G Chowdhary, D. M. Sobers, C. Pravitra, C. Christmann, A. Wu, H. Hashimoto, R. Kalghatgi, C. Ong, and E. Johnson. Self-contained autonomous indoor flight with ranging sensor navigation. AIAA Journal of Guidance Control and Dynamics, 35(6):1843-1854, November-December 2012.
[13] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H. Choi, and J. P. How. Distributed 137 planning strategies to ensure network connectivity for dynamic heterogeneous teams. IEEE Journal on Selected Areas in Communications, 30(5):861 -869, June 2012.
[14] T. Toksoz and N. K. Ure and G. Chowdhary and A. Geramifard and J. P. How. 3 hours persistent search and track mission with autonomous recharge station. Website, “http://acl.mit.edu/projects/recharge.htm”, 2012.
[15]] N. K. Ure, G. Chowdhary, T. Toksoz, J. P. How, M. Vavrina, and J. Vian. Automated battery management system for enabling multi-agent persistent missions. IEEE Transactions of Mechatronics, 2012.
[16] Tuna Toksoz. Design and Implementation of an Automated Battery Management Platform. Master's thesis, Massachusetts Institute of Technology, August 2012.
[17] ROS.org, Powering the world's robots, Last accessed May 7, 2017, “http://www.ros.org/”
[18] Parrot for Developers, Welcome, Parrot Developers, Last accessed May 7, 2017,
“ https://developer.parrot.com”
[19] How to Install Ubuntu 14.04 Trusty Tahr, Last accessed May 7, 2017,
“http://howtoubuntu.org/how-to-install-ubuntu-14-04-trust y-tahr”
[20] Builtvisible, How to Install Ubuntu: The Ubuntu Installation Guide, Last accessed May 7, 2017,
“https://builtvisible.com/the-ubuntu-installation-guide/”
[21] ardrone_autonomy — ardrone_autonomy indigo-devel documentation, ardrone_autonomy, Last accessed May 7, 2017,
“ https://ardrone-autonomy.readthedocs.io/en/latest/”
[22] Brad-Lee, GitHub, GPS Waypoint Tracking Package, Last accessed May 7, 2017,
“https://github.com/Brad-Lee/xS3D_GPS”
[23] Brad-Lee, GitHub, Network Programming, Last accessed May 7, 2017,
“ https://github.com/Brad-Lee/xS3D_Network”
[24] Brad-Lee, YouTube, Patrol mission and job hand off, Last accessed May 7, 2017,
https://www.youtube.com/channel/UClljtAs2dCgIscvwq5 wnMSA”