2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Multi-UAV Testbed for Aerial Manipulation Applications

D. I. Montufar, F. Muñoz, E. S. Espinoza, O. Garcia and S. Salazar

Abstract— **This paper addresses the development and implementation of a testbed for object's manipulation by using unmanned aerial vehicles AR.Drone and the VICON Cameras System. Such testbed allows the user to choose among two development environments to perform aerial manipulation establishing a communication with LabVIEW, ROS and the C++ Qt library; thus, the user can employ the development environment more suitable for his application. This testbed allows us to establish communication between a computer and the A.R. Drone (v1 or v2) via a WiFi connection which sends and receives data using the communication protocol UDP and sockets for connection with the vehicle UDP ports. This platform sends control and navigation commands to the UAVs in order to position them in the space with user interaction by means of a graphical user interface. This testbed enables the implementation of more complex applications, such as complex controllers. In addition, to validate the effectiveness of this testbed, experimental results of aerial manipulation between two quadrotors are presented.**

I. Introduction

The recent technological advances, such as mini sensors and electronic components, have encouraged researchers to develop autonomous systems that execute tasks in dangerous places for the human being. These tasks must be controlled in the distance avoiding some risk. The Unmanned Aerial Vehicles (UAVs) are a class of these autonomous vehicles, which despite their great advantage, they are principally used in civilian and military applications related to observation and surveillance [1], [3]. Actually, the low cost of commercial platforms (UAVs) allows researchers to develop algorithms that involve multiple UAVs in complex tasks; such as the object's manipulation and transportation, the construction of structures under spatial conditions, and mapping applications. A kind of low cost UAVs is the Parrot AR.Drones which are suitable to be used in the development of new collaboration strategies.

In recent years, scientists have been developed research works in which the communication and control interfaces were developed for the Parrot AR.Drone [4], [5]. Such communication has been implemented by using AT commands and the UDP protocol to control the AR.Drone position and velocity through a Visual C++ interface in [6], while in [2] a control interface for a quadrotor with fire detection by using Visual C and OpenGL was developed. On the other hand, [7] and [8] developed a ground station for a quadrotor by using Java and Visual C++ software. In addition, [1] and [3] presented the development and use of an interface in order to implement complex systems which employ vision and trajectory tracking systems.

A first step in the generation of new UAVs applications is the development of indoor experiments by using testbeds. For this reason, we can find several articles about the construction of testbeds, which are used to test communications, control algorithms, vehicles cooperation, among others. For example, in [9] a Graphical User Interface (GUI), developed in Java, is used to communicate and to control the AR.Drone 2.0. In [10] a commercially (COTS) radio-controlled (RC) aircraft instrumented with flight avionics system is used as a testbed to demonstrate the integrated flight control development and testing framework. In [11], a testbed for a swarm of UAVs consisting of a ground station (a laptop running Windows XP and Borland Turbo C++) where a GUI is used to control and interact with the UAV was developed. As we can see, tesbeds have been used to develop more complex tasks, as in [12] and [13].

The main contribution of this paper is the development of a testbed based on the Parrot AR.Drone for object's manipulation that allows the user to choose between two different development environments. The first one is developed on a graphical programming software (LabVIEW) and the last one is based on the Robotics Operating System ROS. Such development environments allows the user to get and to send all the control and navigation information of at least three quadrotors and to get their orientation and position from the VICON Camera System. This testbed enables the implementation of complex applications, such as cooperative controllers for object's manipulation and transportation. The paper is organized as follows. Section II presents the description of the main elements of the testbed. Section III describes the Graphical User Interface (GUI) developed in the LabVIEW software. Section IV gives the information about the Robotics Operating System (ROS) used in the platform, and Section V shows the numerical and experimental results obtained by using the testbed for object's manipulation purposes. Finally, Section VI provides conclusions of this research work.

II. Testbed description

In this section, the components employed for the development of the testbed are presented.

D. I. Montufar, F. Muñoz and E. S. Espinoza are with the Polytechnic University of Pachuca, Zempoala, Hidalgo, Mexico. montufarmeca@micorreo.upp.edu.mx, {filiberto,steed}@upp.edu.mx

O. Garcia is with Aerospace Engineering Research and Innovation Center, CIIIA-FIME-UANL, Monterrey, 66616 Mexico. octavio.garcias@uanl.mx

F. Muñoz, E. S. Espinoza and S. Salazar are with The French-Mexican Laboratory on Computer Science and Control LAFMIA-UMI CNRS 3175, CINVESTAV-IPN, Mexico D.F., Mexico. sergio.salazar.cruz@gmail.com

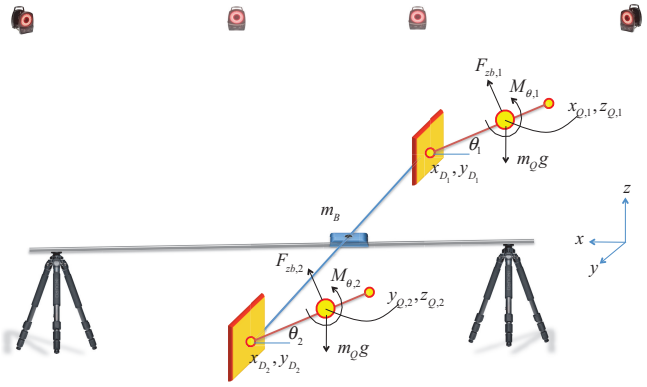

Fig. 1: Platform design for aerial manipulation.

1) Plattform description: The platform developed for the object's manipulation consists of a car mounted over a rail that allows an object (a bar with two contact surfaces) to move in a line or to pivot over its z axis, which allows us to test algorithms of object's manipulation of 1 or 2 DOF with holonomic constrains. In order to obtain the object and vehicles positions and orientation, we used a VICON Camera System. This system is composed of 12 VICON Bonita 10 Cameras and the Tracker Software. Fig. 1 shows a schematic of the platform design.

2) UAV description: The aerial vehicle used in this work is the AR.Drone V2 showed in Fig. 2, which is an unmanned aerial vehicle controlled via a WiFi connection and whose configuration is known as quadrotor. This UAV consists basically of a 1 Ghz 32 bits ARM Cortex A8 processor with 800 MHz video DSP TMS320DMC64x, a 1 Gb DDR2 RAM at 200 Mhz, an inertial sensor, an ultrasonic sensor, and two video cameras, besides, this vehicle includes an integrated WiFi network, allowing the vehicle to connect with mobile devices with different operative systems such as iOS, Android or Linux. Once the connection is done, the vehicle sends images and navigation data through the WiFi connection [14]. In order to communicate with the aerial vehicle, it creates a WiFi network in which the mobile device can access by using a network board. In this sense, the aerial vehicle generates its own network and opens communication protocols UDP to receive and send different command signals. This communication protocol uses three ports to send and receive information. Table I shows the UDP ports used by the AR.Drone.

Fig. 2: AR.Drone.

TABLE I: AR.Drone Ports.

Port Number	Description	Type
5556	Control and configuration	Read/Write
5554	Navigation Data	Read only
5555	Video package reception	Read only

3) Parrot AR.Drone SDK: Parrot company provides a SDK (Software Development Kit) for the AR.Drone. This SDK includes default and complex functions, and it is based on the standard C library AR.DroneTool. However, to perform a specific application, the SDK has commands that the AR.Drone recognizes like basic instructions designated as AT commands, which are the core of the library that the manufacturer has published [14]. AT commands are text strings of 8 bits, including ASCII characters and a newline at the end. A command consists of three characters "AT *" followed by the command name, the equal sign, a sequential number and optionally a list of arguments. Table II shows the basic AT commands to control the principal actions of the drone.

TABLE II: AR.Drone AT commands.

AT Command	Arguments	Description
AT*REF	Input	Take off/ Landing Emergency
AT*PCMD	Flag, roll, pitch thrust, yaw	Allow moving the drone
AT*FTRIM	-	Set the horizontal reference
AT*COMWDG	-	Reset the communication Watchdog

An example of an AT command construction is the following

```
AT*REF=1,290718208
where:
AT: The header
REF: The name of the command
1: The current sequence number of packet
290718208: The argument
```

where 290718208 is used to take-off and 290717696 is used for landing, for example.

III. LabVIEW Platform

LabVIEW provides a native implementation of the AR.Drone API with the AR.Drone Toolkit LVH. Using this Toolkit, programmers can create applications that control the AR.Drone via a small set of VIs. To control the AR.Drone, it is necessary to establish a connection with its broadcasted WiFi network. Then the AR.Drone Toolkit can communicate with the drone using network protocols such as TCP and UDP in order to send commands as defined in the AR.Drone programmer's guide found on the AR.Drone website. After installation by using the VI Package Manager, the provided VIs in the Toolkit are available in the block diagram through the AR Drone toolkit palette.

In this palette, there are VIs to perform the principal operations over the AR.Drone. The *Open* VI allows us to start browsing data and video, this VI contains an input cluster which is used to specify the IP address of both AR.Drone and the PC. The *Close* VI finalizes communication with the drone. In the toolkit there is a VI called *Control Drone*, which sends basic control commands to the AR.Drone, including the take-off and landing, emergency landing, hover mode and movement commands. Since this VI uses AT commands to control the drone, the sent values must be from -1 to 1, which internally are converted to the IEEE 754 format.

The AR.Drone Toolkit also includes the functions *Initialize NavData* and *Read NavData* to get the drone navigation data (navdata). This information includes the roll, pitch and yaw angles, the angular velocities, the percentage of available battery, among other information. The toolkit also provides functions to perform other operations such as: demos (animations), manual control of the the drone using an Xbox 360 joystick.

On the other hand, in order to implement the control laws developed by the users, a Graphical User Interface was developed in LabVIEW to manipulate at least three Parrot AR.Drones. Fig. 3 shows the front panel of this GUI, which depicts the information that the Read NavData VI returns to the computer. The first graph shows the roll, pitch and yaw angles, the second one shows the velocities v_x, v_y and v_z, and the third one shows the set points for the roll, pitch angles and the vertical and yaw velocities, which can be introduced from an external radio control (Futaba 7 Ch, for manual operation) or directly obtained from the implemented control law (autonomous operation). Finally, the information about altitude and battery percentage is also displayed.

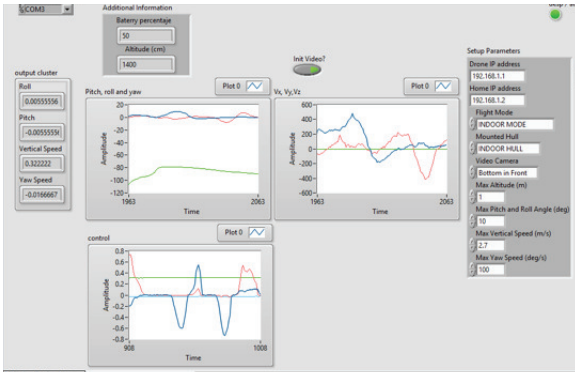

Fig. 3: LabVIEW based Graphical User Interface.

Fig. 4 shows the GUI block diagram which controls the AR.Drone behavior. In order to perform manually control of the AR.Drone an external radio control can be used. First, the configuration of PPM decoder (from the external radio) and the AR.Drone is initialized outside of the while loop. Second, the code inside the while loop allows obtaining from the PPM decoder the set points of the roll, pitch and yaw angles which are send to the drone by using the *Control Drone* VI. Next, the navigation data is obtained by using the *Read NavData* VI and it is plotted in the waveform chart. The *while* loop is executed until the user press the Stop Button. Finally, the ports for communication with the PPM decoder and the drone are closed.

Algorithm 1 summarizes all the necessary steps to carry out the Control and Visualization of the parameters of the AR.Drone in a Graphical User Interface.

Algorithm 1 LabVIEW Based GUI

Require: Install the AR.Drone Toolkit LVH

1: Initialize PPM and AR.Drone communications, *Open* and VISA VI
2: **while** Stop Button == false **do**
3: Obtain the desired roll and pitch angles and the yaw and vertical speeds from the PPM decoder, from the GUI buttons or from the user control law VI
4: Send the control commands to the drone by using the *Control Drone* VI
5: Read and display the navigation data with the *Read NavData* VI
6: **end while**
7: Close communication ports of the PPM decoder and the AR.Drone with the VISA Close and *Close* VI

In order to get the connection between LabVIEW and the VICON System, it is necessary to install the Vicon DataStreamSDK in the computer. To use the SDK, we need to employ the library ViconDataStreamSDK DotNET.dll (DS-SDK), which is included in the installation folder. Then, we need to use .NET connectivity for adding a reference to .NET assembly.

In this way, the function calls, within the SDK, allow connecting and requesting data from the VICON DataStream.

Finally, Algorithm 2 summarizes the necessary steps to get the object's position and orientation from the Vicon Camera System based on LabVIEW.

Algorithm 2 Vicon in LabVIEW

Require: Install the .NET 4.0 Framework on your development machine

Require: Install Vicon DataStreamSDK.msi

1: Use the a Constructor Node like client using DS-SDK
2: Insert an "Invoke Node" to connect LabVIEW with the Vicon Camera and to enable the segment data
3: **if** No error **then**
4: Get the segment global rotation and translation
5: Use an "Invoke Node" to obtain the object's position and orientation
6: **end if**
7: Disconnect the segment data

IV. ROS PLATFORM

ROS (Robot Operating System) is an open-source, meta-operating system used in several robotics applications, see for example [15], [16], [17], [18] and [19]. This operating

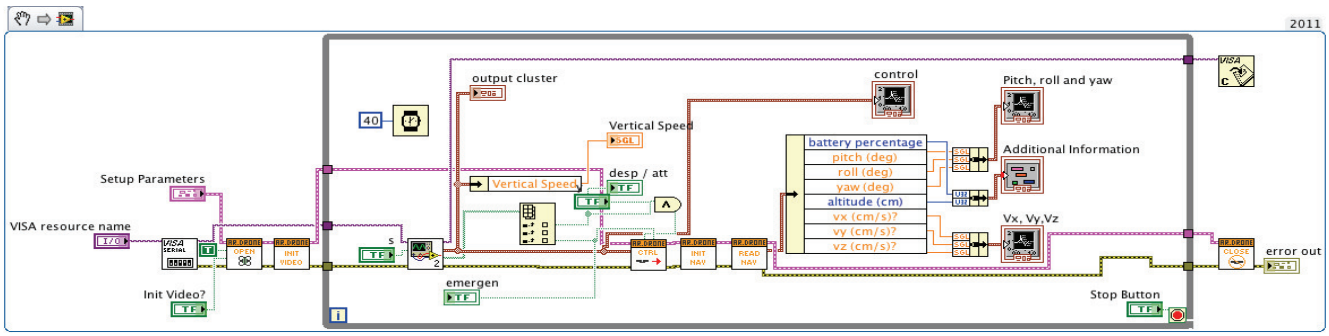

Fig. 4: GUI Block Diagram.

system provides services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between process, and package management [20]. It also provides tools and libraries for obtaining, building, writing and running code across multiple computers. Like LabVIEW, ROS is not a realtime framework, though it is possible to integrate ROS with realtime code. The use of the AR.Drone, with ROS performing camera-based navigation, was presented in [21].

In this work, we use the package called ardrone_autonomy, which employs the SDK 2.0.1 by Parrot AR.Drone and sopports the versions 1.0 and 2.0. The node created for the package of ROS is the ardrone_driver.

This package uses the dependencies such as roscpp, image_transport, sensor_msgs, camera_info_manager and std_srvs which are employed for programming on C++ language. These dependencies are used for transporting images in low-bandwidth compressed formats, to create messages for sensors, and to calibrate the cameras, respectively. Before running any package, it is necessary to execute the roscore program since it is a collection of nodes and programs.

The package of ardrone_autonomy consists of four topics[1] of input and twenty eight topics of output. The topics of input include the /ardrone/reset (emergency), /ardrone/land (landing), /ardrone/takeoff (takeoff) and /cmd_vel (roll, pitch, yaw, thrust), while the topics of output provide the navigation and camera data.

If the information of the navigation data is required, it is necessary to create a node subscriber which is the bridge between the information of the ardrone_autonomy and our application. However a simple form, to watch the data, is to read the topic in a terminal by using the command line rostopic list. The /ardrone/navdata topic displays the information of the battery level, the position in x, y, z, the angular position (roll, pitch, yaw), the altitude, the pressure, and other state variables.

The drone's data transmission update frequency can be configured in the frequency range between 15 Hz or 200 Hz, which depends on the navdate_demo parameter. On the other hand, the driver can operate in two modes: real-time or fixed rate, where we can select the configuration by using the realtime_navdata parameter.

Fig. 5 shows a Graphical User Interface (GUI) developed in the software *Qt Creator* under the ROS environment. This figure depicts the interfaces to control three different Parrot AR. Drones, where the principal commands are available to the user, such as Take-off, Landing, Emergency, and the four control signals (Roll, Pitch, Yaw and Thrust), which take values from -1 to 1.

Fig. 5: Qt and ROS based GUI.

This GUI records the information in a file .msg which is used to publish elements and parameters from the message and service, which connect the signals of the application with the topic /ardrone/takeoff, /ardrone/land, /ardrone/reset, /cmd_vel. In order to use several vehicles at the same time, it is recommended to create a file .launch to execute different packages in a terminal which avoids to have many terminals open simultaneously.

On the other hand, in order to get the vehicle position x, y and z, we used the VICON motion capturing system, which is communicated with ROS through the communication protocol VRPN (Virtual-Reality Peripheral Network). This protocol is a set of classes within a library and a set of servers that are designed to implement a network-transparent interface between application programs and the set of physical devices used in a virtual-reality system. In order to use this communication protocol, we employed the package called ros_vrpn_client which is a client for VRPN and publishes a TF frame and TransformStamped for tracked bodies. Fig. 6 depicts all the testbed hardware and software

[1]Topics are named buses over which nodes exchange messages

components, whose operation is explained on Algorithms 3, 4, 5.

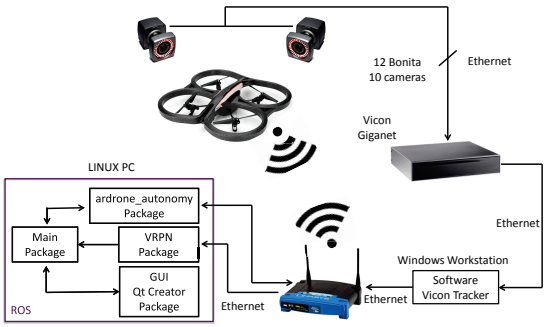

Fig. 6: Nodes used in ROS.

Algorithm 3 Graphical User Interface node

Require: Run roscore in a terminal
Require: Run Qt Creator Package ROS
1: Include the address of the file .msg
2: **while** ros::ok **do**
3: Get the set points and the operation commands (from the GUI) to send to the AR.Drone
4: Publish the variable's values in the file .msg
5: **end while**

Algorithm 4 VICON node

Require: Run ros_vrpn_client
Require: Run VRPN Package ROS
1: Include the address of the file .msg
2: **while** node.ok **do**
3: Get variable's values of translational and angular position $(x, y, z, \phi, \theta, \psi)$ from the Vicon System
4: Publish the variable's values in the file .msg
5: **end while**

It is important to mention that the program have been tested at different frequencies such as 30 Hz, 100 Hz and 50 Hz, in which some disconnection problems were presented for 100 Hz.

A. Use of multiple ARDrone

To employ multiple drones, it is necessary to change the default network configuration of the vehicle. Since the drone creates its own network at which we can connect a device with Android or iOS.

To change the default configuration network of the ARDrone, the TelNet (TELecommunication NETwork) network protocol is required allowing the connection remotely to the embedded system of the drone which works with the operating system based on Linux 2.6.32. In this sense, the vehicles are connected to a router in order to control multiple drones in a PC. Once the vehicles are connected to the router,

Algorithm 5 Main Package ROS

Require: Run ardron_autonomy package
Require: Run Main Package ROS
1: Include the address of the file .msg
2: Get the information published in the file .msg from the Qt Creator and VRPN node
3: **while** node.ok **do**
4: **if** Some operation command==true **then**
5: Publish an "Empty ROS message" to the corresponding topic (ardrone/takeoff, ardrone/land or ardrone/reset)
6: **end if**
7: **if** flying==true **then**
8: Publish the set points values in order to control the force and moments in the /cmd_vel topic
9: **end if**
10: **end while**

we need to change the SDK 2.0.1, which is located inside the package of ardrone_autonomi, due to a problem which causes that we can not connect more than one ARdrone using the same package. To solve this problem, it's necessary to modify the file called vp_com_socket.c, which is located in the path of ARDroneLib/VP_SDK/VP_Com, where needs to be changed the line of code

```
res = VP\_COM\_ERROR;
```

for this one

```
res = VP\_COM\_OK;
```

and then to compile again the package.

V. Simulation and Experiments

This section presents the simulation and experimental results of the aerial manipulation test for two quadrotors.

A simulation platform was developed by using the Nastran software, which communicates with Matlab to validate the proposed control laws. This simulation platform consists of a Nastran virtual model which interacts with a Matlab/Simulink program. Fig. 7 shows the virtual model created by using the Nastran software for an experiment which involves two quadrotors moving an object in a flat structure.

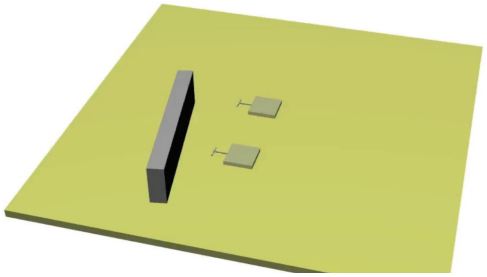

Fig. 7: Simulation platform.

Fig. 8: Testbed platform for aerial manipulation between two quadrotors.

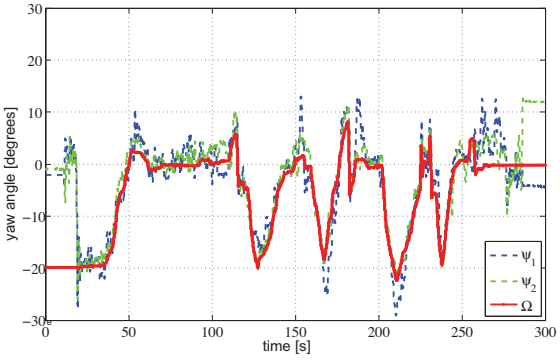

Fig. 9: Experimental Results.

In addition, this section presents experimental results of an aerial manipulation test between two quadrotors, see Fig. 8. A quadrotor is an aerodynamic configuration with unstable dynamics which possess some coupled state variables. The measurements of these coupled state variables are complex in a GPS denied environments such as indoor. Tasks in indoor, such as aerial manipulation, require the information of all state variables in order to achieve these complex applications.

In the reported results, two quadrotors AR.Drone V2 are used to manipulate an object of two degrees of freedom by using the multi-UAV testbed presented in this paper. In this experiment both quadrotors are pushing an object that pivots over a car which slides in a rail. Fig. 9 shows the results of the above experiment, where we can see how the two quadrotors first align the object yaw angle Ω to zero and then pushing the object over the rail under different kind of manual perturbations. The following link shows the test described before: `http://youtu.be/fLHYcdzweFQ`

VI. Conclusion

In this paper, a multi-UAV testbed has been presented for UAVs complex applications such as manipulation and transportation tasks. This testbed is based on graphical programming software LabVIEW, VICON Camera System and the Robotics Operating System ROS for aerial manipulation of quadrotors Parrot AR.Drone. The testbed allows the user to obtain the control and navigation of at least three quadrotors. Finally, simulation results, based on Nastran software and experimental results of the manipulation and pushing of an object of two degrees of freedom by using two quadrotors AR.Drone V2 were presented in order to validate the effectiveness of the developed testbed.

References

[1] I. Mellado, L. Mejias, P. Campoy, and M. A. Olivares, Rapid Prototyping Framework for Visual Control of Autonomous Micro Aerial Vehicles, 12th International Conference on Intelligent Autonomous System (IAS-12), 2012, Island Korea.

[2] X. Zhekui, F. Yongchun, Z. Ge, and S. Hui, Experiment platform for pan-tilt control of a small scale autonomous helicopter, 29th Chinese Control Conference (CCC), 2010.

[3] D. Cavett, M. Coker, R. Jimenez, and B. Yaacoubi, Human-Computer Interface for Control of Unmanned Aerial Vehicles, Systems and Information Engineering Design Symposium, 2007.

[4] A. Visser, N. Dijkshoorn, M. van der Venn, R. Jurriaans, Closing the gap between simulation and reality in the sensor and motion models of an autonomous AR.Drone, International Micro Air Vehicles Conference and Competitions 2011 (IMAV 2011), 2011.

[5] A. Gururaj, S. Tulpule, A. Chaturvedi, and J. N. Charvet, Controlling the Position and Velocity in Space of the Quad-Rotor UAV AR.Drone Using Predictive Functional Control and Image Processing in Open CV, 2012 International Conference on Signal Processing Systems (ICSPS-2012).

[6] L. Yang, X. Bin, W. Fu, and L. Shibo, Development of the Ground Control Station for a Quadrotor unmanned aerial vehicle based on Java programming, 31st Chinese Control Conference (CCC), 2012.

[7] S. Xiao-yan, D. Wen-rui, W. Zhi-mian, and L. Xin-jun Design of software for UAV Ground Control System based on VC++, The Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics, 2009.

[8] R. Purta, M. Dobski, A. Jaworski, and G. Madey, A Testbed for Investigating the {UAV} Swarm Command and Control Problem Using {DDDAS}, Scientia Iranica, Vol. 18, pp. 2018-2027, 2013.

[9] Y. C. Paw, and G. J. Balas, Development and application of an integrated framework for small {UAV} flight control development, Mechatronics, Vol. 21, No. 5, pp. 789-802, 2011.

[10] M. Jamshidi, A.S. Jaimes Betancourt, and J. Gomez, Cyber-physical control of unmanned aerial vehicles, Scientia Iranica, Vol. 18, No. 3, pp. 663-668, 2011.

[11] C. Martinez, T. Richardson, P. Thomas, J. Luke du Bois, and P. Campoy, A vision-based strategy for autonomous aerial refueling tasks, Robotics and Autonomous Systems, Vol. 61, No. 8, pp. 876-895, 2013.

[12] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, Precision flight control for a multi-vehicle quadrotor helicopter testbed. Control Engineering Practice, Vol. 19, No. 9, pp. 1023-1036, 2011.

[13] S. Piskorsky, N. Brulez, and P. Eline, ARDrone SDK 1.7 Developer Guide, 2011, pp. 33-38.

[14] J. Scholz, S. Chitta, B. Marthi, and M. Likhachev, Cart Pushing with a Mobile Manipulation System: Towards Navigation with Moveable Objects, IEEE International Conference on Robotics and Automation, pp. 6115-6120, 2011.

[15] J. Mason, and B. Marthi, An Object-Based Semantic World Model for Long-Term Change Detection and Semantic Querying, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3851-3858, 2012.

[16] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, S. Chitta, Navigation in Three-Dimensional Cluttered Environments for Mobile Manipulation, IEEE International Conference on Robotics and Automation, pp. 423-429, 2012.

[17] D. Mellinger, and V. Kumar, Minimum Snap Trajectory Generation and Control for Quadrotors, IEEE International Conference on Robotics and Automation, pp. 2520-2525, 2011.

[18] V. Grabe, M. Riedel, H. H. Bulthoff, P. R. Giordano, and A. Franchi, The TeleKyb Framework for a Modular and Extendible ROS-based Quadrotor Control, European Conference on Mobile Robots (ECMR), pp. 19-25, 2013.

[19] A. Martinez, and E. Fernandez, Learning ROS for Robotics Programming, *Packt Publishing*.

[20] J. Engel, J. Sturm, and D. Cremers, Camera-Based Navigation of a Low-Cost Quadrocopter, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2815-2821.