A Multi Modal Interface Approach to Control an Unmanned Aerial Vehicle

Soroosh Mortezapoor
Institute of Computer Science and Applied Mathematics
University of Bern
Bern, Switzerland
Email:
soroosh.mortezapoor@student.unibe.ch

Mehdi Taale
Institute of Computer Science and Applied Mathematics
University of Bern
Bern, Switzerland
Email:
mehdi.taale@students.unibe.ch

Samaneh Soleimani
Institute of Computer Science and Applied Mathematics
University of Bern
Bern, Switzerland
Email:
samaneh.soleimani@students.unibe.ch

***Abstract*—This report demonstrates a new approach to control a simulated quad-rotor unmanned aerial vehicle by use of multi modality. Proposed method enables a human operator to control such a vehicle efficiently. Firstly, some modals selected for this project are introduced. After that using of each modal and process of controlling the quad-rotor vehicle based on information provided by these modals are discussed. Finally experimental results and a comparison made between proposed approach and traditional approach using mouse and keyboard are mentioned.**

I. Introduction

[1]Unmanned Aerial Vehicles referred also as UVAs are a kind crafts capable of flying without an onboard pilot. Most common approaches to control them are either controlling by a human operator remotely or being controlled by vehicle itself using a preprogrammed flight path what is called autonomous controlling. A hybrid is also possible meaning that it is human operator's task to make some destination points although selecting of path, velocity and other navigation parameters would be upon vehicles decision. Such flying objects had been developed by military for recognizance flights. However application of them went further and they were found useful in other domains such as manufacturing.

From the September $11th$ attacks on U.S Twin towers, a new application of robots started to gain ground. It was for the first time that some robots were utilized in a real rescue operation. However, the application of robots in rescue scenarios would be limited to probable catastrophes or small simulated environments which would be too expensive for researchers to access before computer aided simulation environments being utilized. In such cited cases, despite realistic simulation environments are appropriate choices for many researchers who are not able to use such real environments due to high expenses; they prefer computer aided simulation environments to realistic ones. USARSim was first designed as a high fidelity simulation of urban search and rescue (USAR) robots. The environment had been intended as a research tool for the study of human-robot interaction (HRI) and multi robot coordination as well. Soon after its initial release, USARSim became as a simulation companion to the National Institute of Standards' (NIST) Reference Test Facility for Autonomous Mobile Robots for Urban Search and Rescue (for further implementation details, reader is referred to references on USARSim). Thus, mentioned simulation environment is decided to be used to test and optimize proposed multi modal interface. Nevertheless it is possible to reuse the method on real robots as well.

Regarding what has been cited, various models of robots have been utilized in rescue scenarios. One of the most favorite types for researchers is aerial. Flyability is a great advantage for the locations where ground robots are unable to reach easily. Furthermore, using heterogeneous robots, it is now possible to identify human victims as well as locate them without the need of seeking for an appropriate ground path from the safe area.

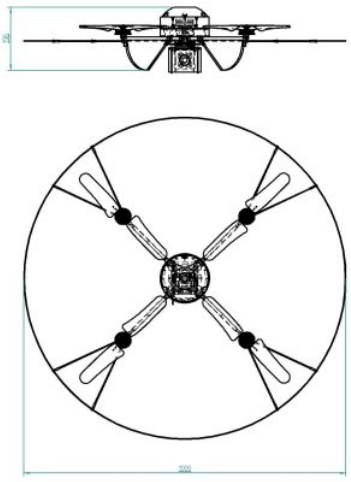

Fig. 1. A quad-rotor model

Unfortunately, current methods for manual controlling

[1]This report belongs to a project conducted as a course project of the course Multi Modal Interfaces which was held in University of Fribourg, Switzerland as a part of Swiss Joint Master of Science in Computer Science - Summer Semester 2012

such robots are usually based on using traditional mouse and keyboard or at most using joysticks. Nowadays humans are dealing with lots of intelligent devices capable of interacting using multi modal interfaces which means enjoying the benefit of humans having five senses. Using of sensors like accelerometers, gyroscopes, compasses, etc. in addition to new devices such as Kinect and Wiimote bring moth more exiting and more efficient methods for controlling a wide variety of electrical and mechanical stuff around. The idea behind this project is to develop a multi modal interface in order to control a UAV and use more of its capabilities. Thus, some sensors and devices are used to mobilize a quad-rotor. Thus, here one of these types of robots known as AirRobot GmbH, developed by AirRobot has been used.

Fig. 2. A real AirRobot

Beginning with going through the current problem in controlling a quad-rotor manually, a multi modal approach using an Android device enjoying accelerometer and gyroscope in addition to a Kinect device will be proposed in order to simplify the process of controlling this kind of robots manually. Subsequently we go in details about architecture of code and its sections respectively. Next, experimental results encountered during this project are discussed and finally, some future works are mentioned.

II. Problem

Regarding that quad-rotors are capable of movement in six degrees of freedom, meaning every possible movement in a 3D coordination, at least 12 buttons are required to control one of them ignoring intensity of movement pace in each direction. Even reducing degrees of freedom to four by considering only linear, lateral, altitude and yaw each with a signed movement speed will not help much since there are a lot of efforts should be put by a single human operator in order to control quad-rotor's navigation.

Besides of using traditional mouse and keyboard approach, nowadays some other tools such as gamepads and joysticks are being used in order to deliver human operator more abilities in controlling such vehicles. Although usage of game pads and joysticks is increasing, they are not perfect. Thus we decided to propose a new approach using multi modal interfaces.

III. Proposed approach

A. General Concept

As it is mentioned before in this report, proposed approacc is based on multi modal interfaces. It means that some new devices are used to interact with the system in order to achieve more convenience in addition to more efficient navigation control of quad-rotor. Among all possible interaction methods, a combination of position, touch and gesture recognition is used for the purpose. Consequently some devices are selected to deliver these services and help reducing development time.

As a tool for pose tracking purpose, an Android device with cited position sensors -accelerometer and gyroscope- is used. In addition to pose tracking, android device is capable of delivering touch information to the fusion system where all information from each section is gathered to make a final decision. Besides, Microsoft Kinect is used as a handy device to recognize operator predefined gestures during navigation. Later in this report, we will discuss how data provided by these devices are used as a reference to produce navigation control commands of quad-rotor.

B. More in Details

Combining data from accelerometer along with device's geomagnetic field sensors, Android devices provide a kind of data called Orientation which determines information about current position of a device relative to the earth's frame of reference (specifically, magnetic north).Using these two hardware sensors, an orientation sensor provides data for the following three dimensions:

- Azimuth (degrees of rotation around the z axis). This is the angle between magnetic north and the device's y axis. For example, if the device's y axis is aligned with magnetic north this value is 0, and if the device's y axis is pointing south this value is 180. Likewise, when the y axis is pointing east this value is 90 and when it is pointing west this value is 270.

- Pitch (degrees of rotation around the x axis). This value is positive when the positive z axis rotates toward the positive y axis, and it is negative when the positive z axis rotates toward the negative y axis. The range of values is 180 degrees to -180 degrees.

- Roll (degrees of rotation around the y axis). This value is positive when the positive z axis rotates toward the positive x axis, and it is negative when the positive z axis rotates toward the negative x axis. The range of values is 90 degrees to -90 degrees.

.

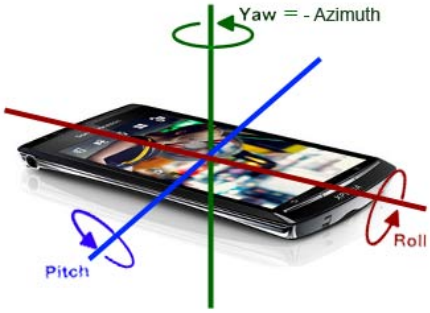

Fig. 3. Android Orientation

Bending to left side and right side are two gestures used as gesture references for this project. Standing in front of a Microsoft Kinect, operator is able to use bending side in order to interact with the system. The more operator bends to either left or right side, the more pressure will be exerted to rotors to make a higher movement speed. One of the benefits enjoyed during this project is that Kinect is capable of detecting skeleton of the person standing in front of that and returning position of joints. This feature is discussed later in this report where gesture recognition is explained.

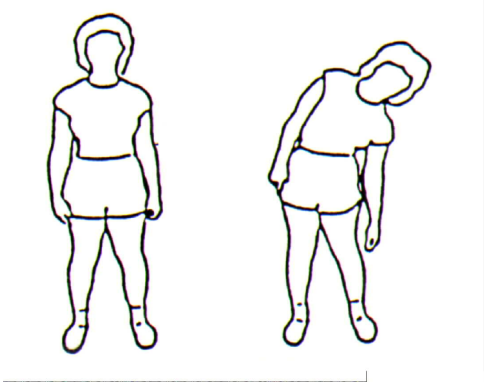

Fig. 4. Defined bending gesture

Touchability is an undeniable benefit of using an android phone. Two touchable buttons are designed as a GUI of android application to provide the system with information about touching state. In the other words, operator would be able to interact with the system through touching these buttons beside all other methods of interaction mentioned before simultaneously. Implementation of touch information usage is discussed within corresponding section.

In essence during this project, using orientation information and touchablity of android device, gesture recognition of Microsoft Kinect and definitely a central application running on a PC as data processor and combiner which is refered to MMARC[2] Central App here after, it is tried to make the process of controlling an unmanned aerial vehicle and specifically quad-rotor cushy as much as possible.

[2]MMARC stands for Multi Modal AirRobot Controller which refers to proposed approach all along with applications developed and devices used in this project

IV. Technical Details

A. Android

As major application developed for android devices use Java programming language, MMARC developed android application could enjoy concurrentablity foundable in Java. MMARC android application is built by three tiers. As usual, highest tier is devoted to GUI consisting of everything required to be shown to operator on a touch screen while middle tier is supposed to prepare data gathered from orientation[3] sensors and touch states to send to MMARC Central App. Below, general architecture of MMARC android application is illustrated.

Utilizing android device's wireless LAN, data from middle tier is sent to MMARC Central App through lowest tier using established TCP socket connection without any wire.

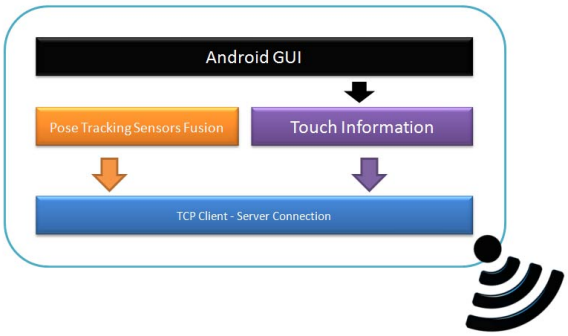

Fig. 5. MMARC android application architecture

B. Kinect

1) Skeleton Detection: Kinect is capable of detecting skeleton of operator standing in front of that. Consequently, it is decided to use information on detected skeleton joints as a reference to find desired gestures. This information includes the movement of shoulders to right and left with the quantum of speed at which they are rocked. As a result, information on related joints are provided in form of sets of $< X, Y, Z >$ in Cartesian Coordination in meters all along with current associated state which can be tracked, untracked or inferred. As a result if the operator bends to right or left, this motion is conveyed to the MMARC Central App.

2) Recognizing Gestures: What received and going to be used from the sensor are the raw values of X and Y for the right and left shoulders. Comparing these two values of each shoulder, bending gestures to either right ot left can be recognized. Altough Kinect delivers a magnificent service providing these information, there is still a filter required to be

applied due to differences in people's sizes and the distance between their two shoulders. This filter contrast sensitivity function approximates desired relative value through division of difference between Y parameters by a variable indicating total distance between two shoulders.

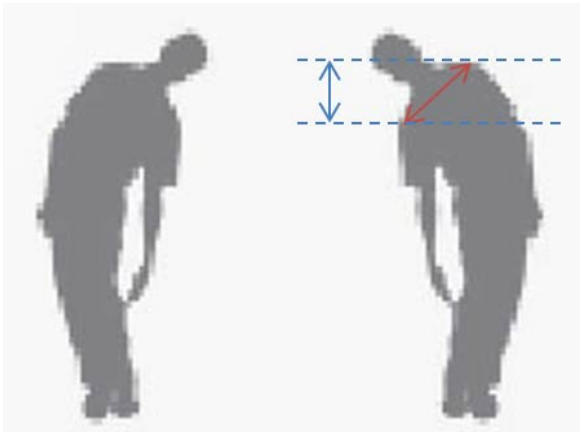

Fig. 6. Bending gestures detection by Kinect

C. Fusion System

1) Architecture: As it is shown in figure below, MMARC Central App is made up of three tiers. Lowest tier is responsible of connecting to sensors as well as USARSim environment which is mentioned before as our test rig. These sections in lowest tier connect to Kinect over a USB wire, to Android over a wireless TCP connection and finally to USARSim Server over a simple TCP connection either wired or wireless. Next, all important processes are done in second and middle tier. This is exactly where all information from sensors are gathered, interpreted, combined and command parameters are resulted. Following, this tier will be discussed and explained.

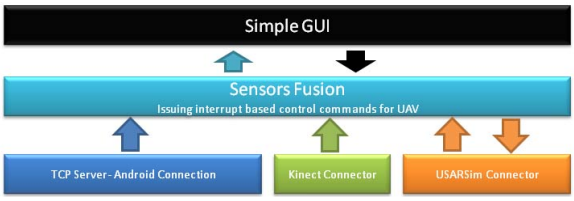

Fig. 7. Fusion application architecture

As usual, highest tier devoted to a windows form GUI which illustrates some information about state of connections to quad-rotor trajectory tracking.

2) USARSim Interface: Before it is mentioned that as a test environment during this project, AirRobot inside USARSim environment is used. In order to connect to AirRobot in USARSim, an interface is developed which is capable of connecting to USARSim server using a TCP socket connection. Over this TCP socket connection, data from AirRobot sensors and commands from developed PC application are transmitted from and to simulated AirRobot respectively. In order to control AirRobot navigation in USARSim environment, for parameters are required to be feed into USARSim through established connection. These parameters consist of four velocities, Altitude, Linear, Lateral and Rotational which controls vehicle in four degrees of freedom. Other two degrees also could be reached by special combinations of these parameters. These parameters are calculated in Unreal Development Kit (UDK) measurement units since USARSim uses UDK as a base for generating graphics and physics of environment along with all objects within including our quad-rotor vehicle. Thus main duty of the application is to produce these parameters upon sensed actions of the human operator. Changing this interface with every interface capable of connecting to a quad-rotor vehicle and control it using these four parameters, the system can be applied to every kind of quad-rotors whether simulated or real.

3) Fusion: Having orientation data, touching state information and recognized gestures, control commands should be issued. This is the main task of middle tier which is discussed earlier in this report. As data from each sensor is received based on interrupts, interval between two control commands demands on updating at least one sensor. In the other hand updating each sensor data, a new set of control parameters is produced, transmitted to USARSim environment and subsequently applied to AirRobot. Regarding structure of the system, both complementary and concurrent fusions can be considered for this project since on one hand each sensor is only responsible of providing raw data to produce one of the four navigation command parameters and on the other hand, there are some trajectories what cannot be gained without combining two or more received sets of data from diverse sensors. For instance, following a three dimensional spring-like path requires more than one parameter to change. Furthermore, raw data from each sensor must be transformed to another range of numbers with some equations to be fit for navigation commands. Following some functions are shown in order to transform raw amount sensed from sensors to velocities to be feed to AirRobot. Moreover, domains and ranges are important as can be seen.

$$f(x) = \begin{cases} 0 & \text{if } 26 < x < 34 \\ \frac{(\alpha(-x+30))^3}{3} & \text{if } -10 < x < 26 \\ & \text{or } 34 < x < 70 \\ 3 & \text{if } x > 70 \\ -3 & \text{if } x < -10 \end{cases} \quad (1)$$

Formula 1 transforms linear input in degrees between -10 and +70 to an exponential space between -3 to 3. This means that raw orientation data from android device is transformed to a float number between -3 and 3 in UDK unit. Furthermore, zero location is considered at $30(+/-)4$. Here $f(x)$ refers to linear velocity for AirRobot where α equals to 0.052002.

Next formula produces lateral velocity for AirRobot from data come from android device sensor. This formula similar to previous one transforms a linear space to an exponential space.

$$g(x) = \begin{cases} 0 & \text{if } -10 < x < 10 \\ \frac{(\alpha x)^3}{3} & \text{if } 10 < |x| < 60 \\ 3 & \text{if } x > 60 \\ -3 & \text{if } x < -60 \end{cases} \quad (2)$$

Again, $g(x)$ refers to lateral velocity of AirRobot where α equals to 0.030285. Nonetheless, rotational velocity is calculated based on data from Kinect which is discussed before. Following formula demonstrates how rotational velocity is made from raw data from Kinect where β equals to 40.

$$h(x) = \begin{cases} \frac{(x)^3}{\beta} & \text{if } -11.70 < x < 11.70 \\ 40 & \text{if } x > 11.70 \\ -40 & \text{if } x < -11.70 \end{cases} \quad (3)$$

Similar to previous formulas used to produce linear and lateral velocities, $h(x)$ which refers to rotational velocity maps linear input from Kinect to an exponential space between -40 and $+40$. This diference between range of $h(x)$ and two functions before traces back to diference between rotational velocity domain with other velocities' domains that USARSim accepts as navigation control commands for AirRobot.

Finally, altitude velocity is calculated diversly since altitude velocity is changed whenever operator touchs either up or down buttons inside developed android application's GUI. Touching one of these buttons, a timer inside the fusion application starts.

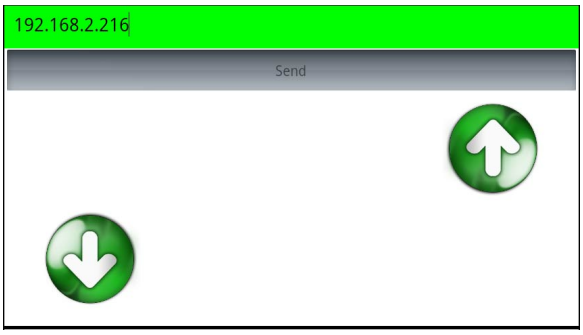

Fig. 8. Screenshot of developed android application's GUI along with altitude change buttons

As long as the button is pressed, the timer ticks. Each tick, the timer triggers altitude velocity calculation function what is a function of time passed. Unpressing the button, timer stops working, resets time and subsequently sets altitude velocity as 0. Below, function $p(x)$ is demonstrated which maps time passed by the timer to an exponential space in the range of $[-3, -1]$ or $[1, 3]$ when β equals to 5000.

$$p(x) = \begin{cases} 1 + \frac{(x)^3}{\beta} & \text{if } -1.25 < x < 1.25 \\ 3 & \text{if } x > 1.25 \\ -3 & \text{if } x < -1.25 \end{cases} \quad (4)$$

V. Experimental Results

Proposed system is tested and evaluated by eight users. Users and in the other words operators were requested to complete a predefined mission. The mission was taking off with an AirRobot from a specified location inside simulated environment, illuminating three target points in the area and finally landing on the roof a third target point which was a building. All operators were asked to complete this mission twice. First with traditional tools consisting of a simple set of keyboard and mouse and after that with MMARC. Results from these experiments are shown below.

Title	Trad.	MMARC
Avg. time of mission completion (minute)	2:30.50	1:18.24
More Efficient	37.5%	62.5%
More Exciting	12.5%	87.5%
Easier to use	50%	50%
Need training	25%	75%
Preference	25%	75%

Results of experiments

.

Title	Positive	Negative	Neutral
Using of gestures	62.5%	37.5%	0%
Using of touch	50%	0%	50%
Using of pose tracking	12.5%	87.5%	0%

Degree of satisfaction

VI. Conclusion

It can be concluded that using of MMARC, a human operator is able to control an unmanned aerial vehicle and specifically a quad-rotor more efficient and more exciting. Accourding to experimental results, users evaluated MMARC found this system preferable to current traditional method of interacting.

VII. Future works

This project aims to facilitate process of controlling a quad-rotor vehicle and definitely it can be a solid foundation for further developments. One considerable development which can be made upon this project is using of voice recognition to complement utilized modals. Another improvement could be using of some gestures as a reference for quad-rotor to do some preprogrammed behaviours.

ACKNOWLEDGMENT

The authors would like to thank all instructors of Multi Modal Interfaces course held at University of Fribourg, Switzerland. Another appreciation is expressed to USARSim commmunity for they guidance before and during this project.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Balaguer, B., Balakirsky, S., Carpin, S., Lewis, M., Scrapper, C., *"USARSim: a validated simulator for research in robotics and automation"*, IROS 2008.

[3] Scrapper, C., Balakirsky, S., Messina, E., *"MOAST and USARSim - A Combined Framework for the Development and Testing of Autonomous Systems,"* in Proceedings of the 2006 SPIE Defense and Security Symposium, 2006.

[4] Carpin, S., Lewis, M., Wang, J., S. Balarkirsky, C. Scrapper. *"USARSim: a robot simulator for research and education"*, Proceedings of the IEEE 2007 International Conference on Robotics and Automation, 1400-1405

[5] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, C. Scrapper. *"Bridging the gap between simulation and reality in urban search and rescue"*. In "Robocup 2006: Robot Soccer World Cup X", LNAI Vol. 4434, Springer, 2007, pp. 1-12

[6] AirRobot HomePage: http://www.airrobot-uk.com/air-robot-products.htm *(accessed: May 2012)*

[7] USARSim Homepage: http://usarsim.sourceforge.net *(accessed: May 2012)*

[8] Programming with the Kinect for Windows SDK *Published by Microsoft Reseach*

[9] Android Developer's Guide Homepage: http://developer.android.com/ *(accessed: May 2012)*