Controlling a drone with a wearable accelerometer

Fulvio Mastrogiovanni, Barbara Bruno, Carmine Tommaso Recchiuto, Erjon Hysa

***Abstract*—Gestural interaction with flying drones is now on the rise. Natural modalities, such as speech and gestures, have the potential to make human-robot interaction more natural. In this assignment we analyse interaction with UAV systems and we present a gesture-based controller to move a drone using a wearable accelerometer. We discuss and propose the gestural set and finally we implement the gesture based controller.**

I. INTRODUCTION

Accelerometer-based gesture control can be considered a natural communication channel which has not yet been fully explored in human-computer interaction(HCI). Recently, thanks to devices like smartwatches or other wearable devices like MYO, strong efforts have been made to develop gesture-based interfaces to control robots. Humans naturally use gesture to communicate so natural or intuitive body movements can be used as command interface to operate robots instead of keyboard, mouse, joystick or teach pendant that are generally considered unnatural methods of communication because they require prior training, which can be unpleasant.

In recent years drones have evolved as trending and popular topic of research. Innovative applications of drones like aerial filming, delivery, surveilance, aerial mapping, disaster response are massive across the globe and still continue to expand its scope. In this paper we will focus on controlling a quadcopter (Fig.1) with a wearable accelerometer (a smartwatch in our case), which will make it more user friendly as the user would not have to study or learn about instructions to control the drone with other controllers. At first we had to learn about human drone interaction from existing literature, and then we did a careful analysis to decide and modeling our gestures to control the drone.

Fig. 1. AscTec Firefly esacopter.

II. DRONE DYNAMICS

The motion of a drone is represented by three main axes, the yaw, the pitch, and roll along with uplift and downfall. These terms are definitely to know before we analyse gestures. To control these three axes, a minimum of four rotors with symmetrically pitched must be present. We take as example a quadcopter (scheme Fig.2), obviously the principles are the same for esacopters, octacopter etc.

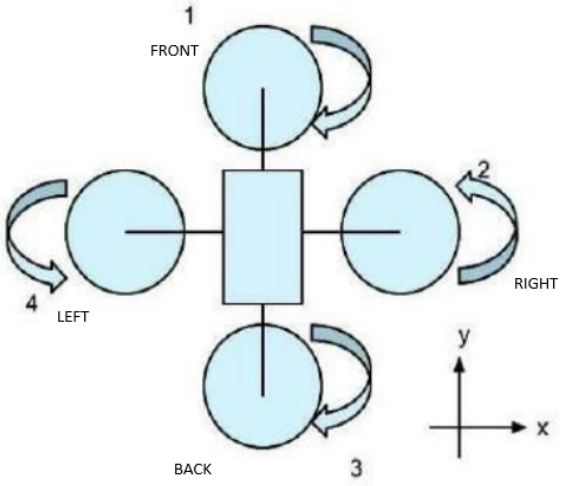


Fig. 2. Quadcopter scheme.

The circles represent the spinning rotors of the quadcopter and the arrows represent the rotation direction. Motors one and three rotate in clockwise direction using pusher rotors while motor two and four rotate in a counter-clockwise direction using puller rotors. Each motor produces a thurst and torque about the center of quadcopter, resulting in a net torque of zero. The symmetrical design allows for easier control of the overall stability of the aircraft.

PITCH: Pitch is the movement of quadcopter either forward and backward. The quadcopter turns along the x axis. Speed of respective rotor is increased while the speed of the diagonally opposite rotor is decreased.

ROLL: Roll is making the quadcopter fly sidewards, either to left or to right. The quadcopter turns along the y axis. It is provided by increasing the speed of left rotors (if we want the drone moves left) and decrease the speed of the diagonally opposite rotors or vice versa.

YAW: Yaw is provided by increasing (or decreasing) the speed of the front and rear motors or by increasing (or decreasing) the speed of the left and right motors. This causes

the quadcopter to turn along its vertical axis z in the direction of the stronger spinning rotors.

UPLIFT: Uplift is simply making the quadcopter fly high from the ground/ sea level. A vertical force is created by increasing speed of all the motors by the same amount. As the vertical forces overcome the gravitational forces of the earth, the quadcopter begins to rise in altitude.

DOWNFALL: Opposite of Uplift, Downfall makes the quadcopter come down to the ground/sea level. The speed of all the motors is decreased by the same amount.

In Figure below we have a *XYZ* representation of roll, pitch and yaw.

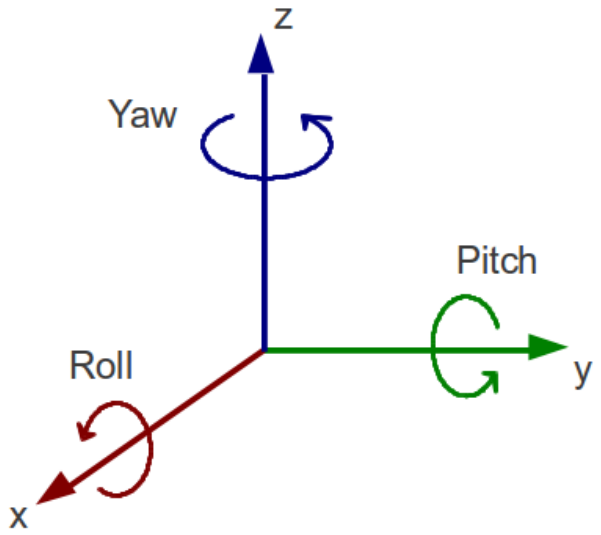


Fig. 3. Pitch, Roll and Yaw.

III. GESTURE ANALYSIS

In this project we use a smartwatch as a gestural interface that allows a real time interaction where the user can control the drone. Pitch, roll and yaw are associated with a set of gestures. Here we study how to establish a gestural set to navigate a drone. The full control of the drone motion is enabled by controlling the pitch, roll and yaw velocity, so we define a continuous mapping between gesture and the corrispective velocity. At the beginning velocities are set to zero and the drone waits until a move gesture is performed. Inertial data is continuously sent and processed in order to move the drone modifying pitch velocity, roll velocity or yaw velocity. So our aim is to identify a set of gestures which allows them to coexist with each other, without interferences during execution.

We have a one to one mapping between gestures and the control variables (pitch velocity, roll velocity and yaw velocity) so we have only three gestures, and this is good because is a small number to remember in order to not overload the user's short term memory. Besides, in according to Donald Norman's principles for a usable interface, we have chosen gestures easy to remember because every gesture reminds to the respective movement the drone has to accomplish, for example the user simply imitates the drone movements with the corresponding hand movements. A system is considered intuitive if the way it works corresponds to our expectations. Thus, it should be fast to learn and easy to use even for a novice user.

So, all of the gestures we propose are based on a single metaphor that implies the imitation of a drone movements. The interaction can be seen as a direct mapping of the user's movements to the drone motion.

The gesture set in Fig.4 shows our proposed gestures. It represents the gestures that satisfy the characteristics mentioned above and represent a coherent set indeed they are different from each other and do not create confusion. All gestures start with the arm positioned in the same way (arm stretched forward) that we called *neutral position*. The user performs a gesture to move the drone and as soon soon as the operator returns his or her arm to the neutral position, the drone stops its current movement.

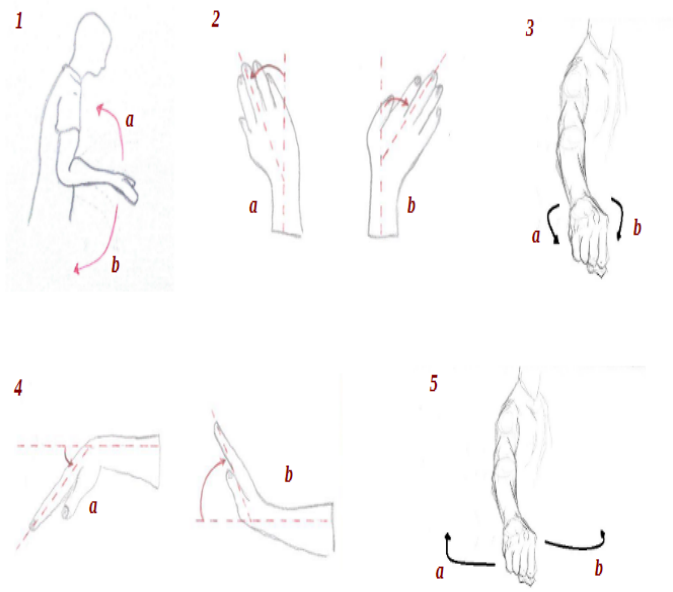
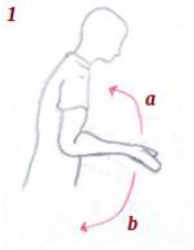


Fig. 4. Gestural set.

As we can see, the size of gesture set is bigger than we need so we had to choose the three gesture to map pitch, roll and yaw. We have already said that we propose only gestures that respect the properties they need so any three of them are fine. However, our final selection of gestures was based on aspects related to physical ergonomics. Thus, two users (a male and a female) have performed the various gestures many times and the results below were collected.



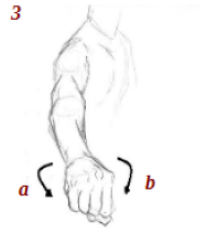
Angle around y axis.

Range [-1.948,1.9853]

a	Male user	Female user
max_x	0,4119	0,3657
max_y	2,2499	1,9853
max_z	1,4566	0,9480
min_x	-0,2338	-0,2784
min_y	-0,1615	-0,1785
min_z	-0,1513	-0,1158

b	Male user	Female user
max_x	0,4198	0,6694
max_y	0,2041	0,0544
max_z	0,1641	0,1063
min_x	-0,3989	-0,5128
min_y	-1,9489	-1,9540
Min_z	-1,3094	-0,8948

Fig. 5.



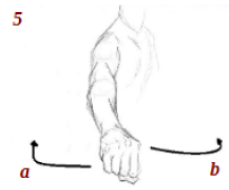
Angle around x axis.

Range [-1.5953,3.0751]

a	Male user	Female user
max_x	0,1766	0,2583
max_y	0,1094	0,1846
max_z	0,0910	0,2763
min_x	-1,5953	-3,2208
min_y	-0,1033	-0,1220
min_z	-0,0521	-0,0741

b	Male user	Female user
max_x	3,0751	4,2936
max_y	0,1241	0,3874
max_z	0,2232	0,4332
min_x	-0,2745	-0,3552
min_y	-0,4421	-0,2354
min_z	-0,1732	-0,1788

Fig. 6.



Angle around z axis.

Range [-1.6206,1.6204]

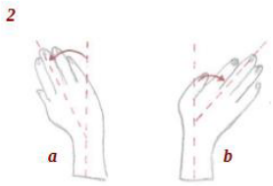
a	Male user	Female user
max_x	0,3701	0,2044
max_y	0,9089	0,8399
max_z	0,0632	0,1074
min_x	-0,1586	-0,4371
min_y	-0,0693	-0,1002
min_z	-1,6206	-1,7773

b	Male user	Female user
max_x	0,2946	0,6378
max_y	0,1355	0,1277
max_z	1,6204	1,8504
min_x	-0,3340	-0,3151
min_y	-0,6639	-0,3775
min_z	-0,1088	-0,0763

Fig. 7.

Each user has performed every gesture for ten times, then each value reported corresponds to the resulting avarage. At this point we chose 1,3 and 5 because they are the ones that are easier to perform in terms of fatigue and allow us to have a greater range of values for mapping pitch, roll and yaw.

More specifically, we propose a direct mapping between the angular velocity component ω_y of gesture 1 and the pitch of the drone. Analogously we propose a direct mapping between angular velocity component ω_x of gesture 3 and the roll of the drone while yaw is direct mapped with angular velocity



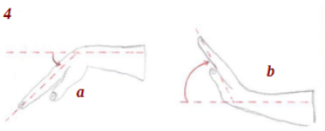
Angle around z axis.

Range [-0.4917,0.6982]

a	Male user	Female user
max_x	0,2887	0,6571
max_y	0,1300	0,1659
max_z	0,6982	0,7817
min_x	-0,1069	-0,1687
min_y	-0,1970	-0,4034
min_z	-0,0831	-0,0538

b	Male user	Female user
max_x	0,1745	0,2875
max_y	0,5152	0,3188
max_z	0,0862	0,0684
min_x	-0,1608	-0,4719
min_y	-0,0891	-0,1010
min_z	-0,5309	-0,4917

Fig. 8.



Angle around y axis.

Range [-0.5964,06929]

a	Male user	Female user
max_x	0,2068	0,2240
max_y	0,1028	0,0937
max_z	0,0948	0,0964
min_x	-1,3774	-0,2540
min_y	-1,4383	-0,5964
min_z	-0,8302	-0,2237

b	Male user	Female user
max_x	0,2208	0,3372
max_y	0,9026	0,6929
max_z	0,3083	0,1830
min_x	-0,2857	-0,2669
min_y	-0,0762	-0,0643
min_z	-0,0505	-0,0913

Fig. 9.

component ω_z of gesture 5. In this way, the user just needs to wear a transmitting device on his hand which includes a sensor which is an accelerometer in our case. Moving the hand in a specific gesture will move the drone in a specific direction.

IV. Implementation

We use an Android Wear compatible LG G watch R paired via Bluetooth with a smartphone LG G3. The smartphone acts as a gateway to a HP Pavilion 15 with an Intel Core i5 processor, 6 Gb RAM, connected through WIFI, in which Ubuntu 16.04 and ROS Kinetic are installed. The used drone is a ROS-enabled AscTec Firefly, shown in Fig.1.

Our project is based on the previous work by Enrique Coronado.Indeed, in order to establish communication between the smartwatch and the drone you will need to install two applications, one for the smartphone and one for the smartwatch.These applications have been developed by Enrique and you can find the source code in the following link:

https://github.com/enriquecoronadozu/smartwatches_apps

The architecture of our system in the simulation environment is depicted in Fig.10. As we can see from it, the system is based on publish/subscriber form of communication. In particular the controller node we implemented, also use this mechanism to send messages on topic /fcu/control which we use to interact with the drone.

We implemented our controller gesture-based as a ROS node in c++. The purpose of this node is to receive data continuously from the smartwatch, recognize the performed gesture, filter data and send them to the drone. For example, if the user wants to move the drone right, he must perform

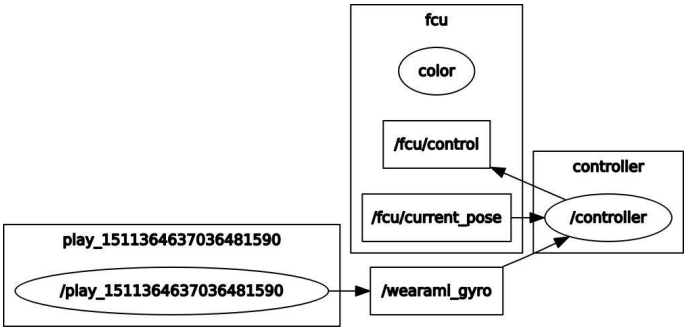
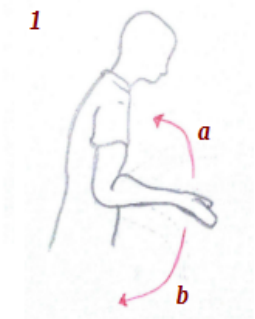


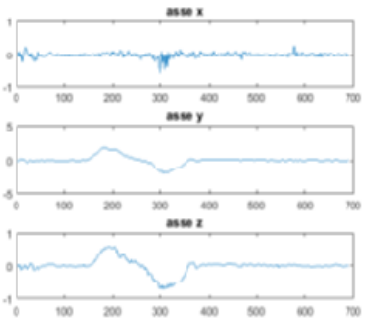
Fig. 10. System component graph.

the gesture 3-a because he wants to control the roll and we mapped the roll with gesture number three. According to our mapping, when a gesture is executed, only values of axis we choose to map that gesture must be sent to the drone while values of the other two axes must be zero, otherwise the drone could move in unexpected way.

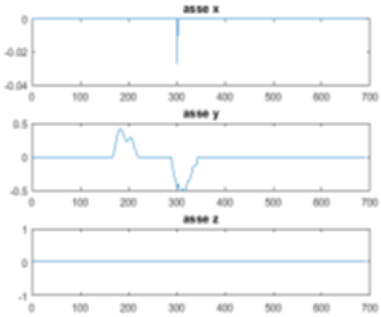
Here below we show the final results. For each gesture that is executed, the values registered by the smartwatch are represented, then they are manipulated by our controller-ROS-node and sent to the drone. As we can see from the charts, only the gyro data of one axis are sent to the drone and it's the axis with which we decided to map the pitch, roll or yaw, it depends on the gesture. Furthermore, as in or case these are indoor experiments so we set the maximum speed to 0.5, but is a parameter that can be changed.



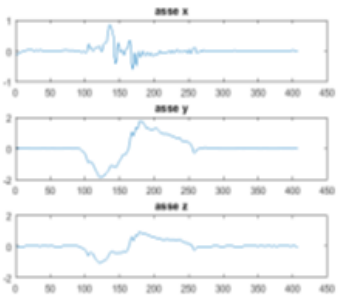
Pitch. We use the y-axis value obtained during the execution of this gesture.



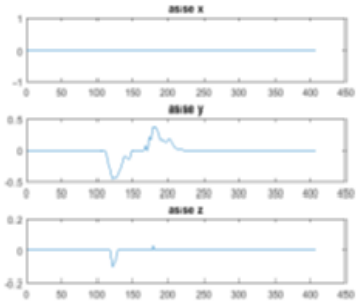
Smartwatch data gesture a.



Corrispective data sent to drone.



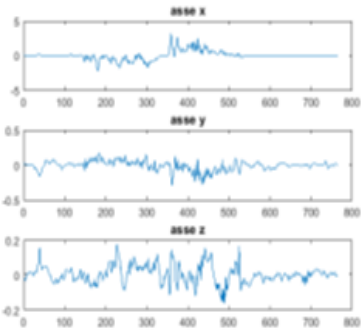
Smartwatch data gesture b.



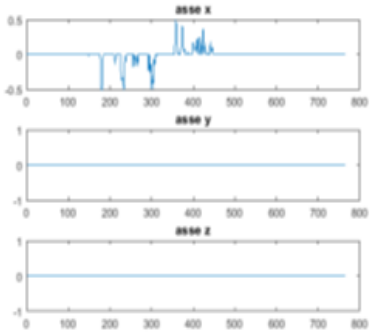
Corrispective data sent to drone.

3

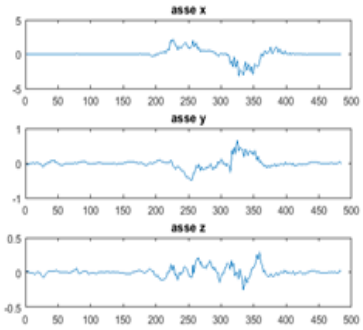
Pitch. We use the x-axis value obtained during the execution of this gesture.



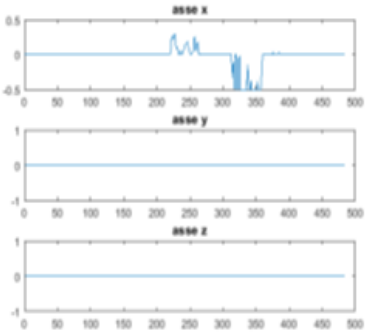
Smartwatch data gesture a.



Corrispective data sent to drone.

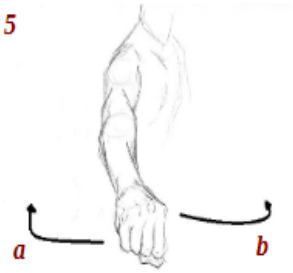


Smartwatch data gesture b.

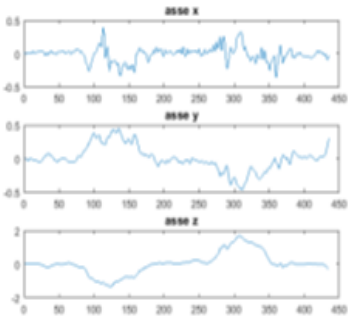


Corrispective data sent to drone.

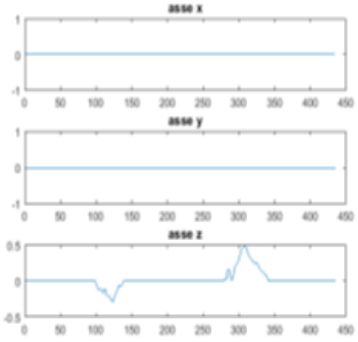
5



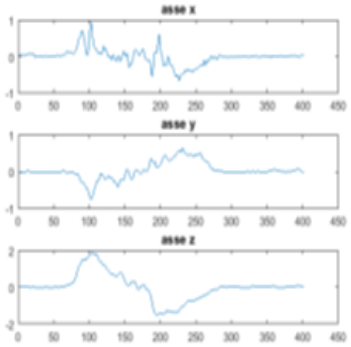
Pitch. We use the z-axis value obtained during the execution of this gesture.



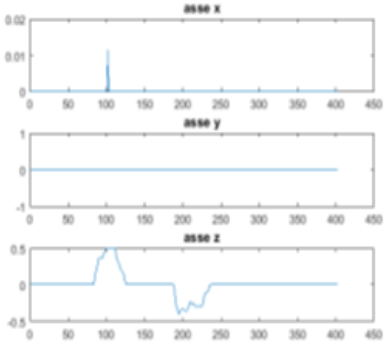
Smartwatch data gesture a.



Corrispective data sent to drone.



Smartwatch data gesture b.



Corrispective data sent to drone.

REFERENCES

[1] E. Coronado, J. Villalobos, B. Bruno and F. Mastrogiovanni, "Gesture-based robot control: Design challenges and evaluation with humans", 2017, 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore.

[2] David McNeill, So you think gestures are nonverbal? Psychological Review, 1985.

[3] David Efron, Gesture and Environment. King's Crown Press, Morningside Heights,New York, 1941.

[4] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009

[5] R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel. Natural human-robot interaction using speech, head pose and gestures. In the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), 2004.