2015

# Exploring Alternative Control Modalities for Unmanned Aerial Vehicles

David Qorashi
*Grand Valley State University*

# Exploring Alternative Control Modalities for Unmanned Aerial Vehicles

**David Qorashi**

A Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Computer Information Systems

at

Grand Valley State University

April 2015

Jonathan Engelsma, Assoc. Professor Date

# Table of Figures

# Acknowledment

Foremost, I would like to express my sincere gratitude to my advisor Prof. Jonathan Engelsma for the continuous support of my Master study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Jamal Alsabbagh and Prof. Hans Dulimarta, for their encouragement, insightful comments, and hard questions.

Last but not the least, I would like to thank my family: my parents Ahmad Qorashi and Fakhri Molavi, for supporting me spiritually throughout my life. I also thank my lovely sister Ac and my brother Yasser that without their support I was not able to achieve this.

# Abstract

Unmanned aerial vehicles (UAVs), commonly known as drones, are defined by the International Civil Aviation Organization (ICAO) as an aircraft without a human pilot on board. They are currently utilized primarily in the defense and security sectors but are moving towards the general market in surprisingly powerful and inexpensive forms. While drones are presently restricted to non-commercial recreational use in the USA, it is expected that they will soon be widely adopted for both commercial and consumer use. Potentially, UAVs can revolutionize various business sectors including private security, agricultural practices, product transport and maybe even aerial advertising. Business Insider foresees that 12% of the expected $98 billion cumulative global spending on aerial drones through the following decade will be for business purposes.[28] At the moment, most drones are controlled by some sort of classic joystick or multitouch remote controller. While drone manufactures have improved the overall controllability of their products, most drones shipped today are still quite challenging for inexperienced users to pilot. In order to help mitigate the controllability challenges and flatten the learning curve, gesture controls can be utilized to improve piloting UAVs.

The purpose of this study was to develop and evaluate an improved and more intuitive method of flying UAVs by supporting the use of hand gestures, and other non-traditional control modalities. The goal was to employ and test an end-to-end UAV system that provides an easy-to-use control interface for novice drone users. The expectation was that by implementing gesture-based navigation, the novice user will have an overall enjoyable and safe experience quickly learning how to navigate a drone with ease, and avoid losing or damaging the vehicle while they

are on the initial learning curve.  During the course of this study we have learned that while this approach does offer lots of promise, there are a number of technical challenges that make this problem much more challenging than anticipated.  This thesis details our approach to the problem, analyzes the user data we collected, and summarizes the lessons learned.

# Introduction

Unmanned aerial vehicles (UAVs), also known as drones, are aircrafts either controlled by ‘pilots’ from the ground or increasingly, autonomously following a pre-programmed mission. While they are currently popular in military, using them for commercial purposes is ramping up.

Up until recently, commercial use of drones was prohibited in the U.S.. In February 2015 the Federal Aviation Administration proposed a policy that would replace the current near-ban on flying drones for commercial purposes with a simple approval process for operators governing commercial flight of any drone up to 55 pounds. Operators would be required only to pass a written exam on FAA rules to obtain a certificate from the agency for drone flying, and to comply with certain safety requirements. The test must be passed again every two years. As the FAA works to complete its safety rules, its current effective ban on the commercial use of drones will remain in place. The only exceptions to that prohibition so far have been FAA approvals for 26 commercial-drone operators, but those approvals came with strict rules, including that operators have a private pilot’s license [1].

Much media attention has focused on Amazon and UPS seeking to use drones for shipping purposes. As another use case, drones are quickly moving to the farmer's field, on the verge of helping growers oversee millions of acres throughout rural America and saving them big money in the process [29].

## Quadcopters

This study will focus on a specific type of drone, quadcopter helicopters. A quadcopter is an aerial vehicle propelled by four rotors. They have a fixed pitch, which makes them mechanically simpler than a typical helicopter. [2]

Nowadays, quadcopters are able to perform quick and complex maneuvers [3], navigate autonomously in structured [4] and unstructured [5] environments and cooperate in a wide range of tasks [6].

## AR-Drone

In fall 2010, a French company released an affordable quadcopter named AR-Drone, equipped with the required sensors and with an easy-to-use software interface. Initially intended as a high-technology tool for augmented reality games, the drone swiftly caught attention of universities and research institutes. The fact that there is an open software stack in drones that makes it easily programmable accelerated this transition. Cornell University conducted experiments using AR-Drones in UAV visual autonomous navigation in structured environment [7]. In addition, Machine Learning algorithms were utilized to foresee the position errors of the UAV following an arbitrary flight path [8]. Presently, more research is conducted using them for surveillance tasks [9] and human-computer interaction [10]. For the purpose of this study, the newest generation of the AR-Drone, called Bebop is used. The Bebop was released on December 2014 (See Figure 1).

Figure 1 . Bebop AR-Drone Quadcopter

**Hardware and Software Specification**

The Bebop AR-Drone control computer is based on a Parrot P7 dual-core CPU Cortex 9 processor with a Quad core GPU running on a GNU/Linux. The manufacturer embedded a software interface, which allows communicating with the drone via an ad-hoc Wi-Fi network. The API provides access to preprocessed sensory information and images from onboard cameras [11] along with an interface to send commands to the drone.

The drone utilizes the following sensors:

- 3-axes magnetometer (AKM 8963)
- 3-axes gyroscope (MPU 6050)
- 3-axes accelerometer (MPU 6050)
- Optical-flow sensor: Vertical stabilization camera (Every 16 milliseconds, an image of the ground is taken and compared to the previous one to determine the speed of the Bebop Drone)
- Ultrasound sensor (Analyzes the flight altitude up to 8 meters)
- Pressure sensor (MS 5607)

## Controllability Issues

Presently there is no standard interface for piloting a drone. The de facto mechanism for controlling drones is using joysticks or d-pads. Innovative solutions use multitouch devices (e.g., the iPhone) and game controllers (e.g., Nintendo Wiimote). With current technology, getting efficient in piloting drones has a steep learning curve. Users will buy drones, unbox them, and attempt to fly them immediately. The problem is that they are not able to fly smoothly in their initial attempts. Most of the times, they crash the drone when they start flying them with joysticks and multitouch devices; which could lead to, damaging the drone or the operating environment. Another common issue is that novice users lose control of drones very easily. Drones will fly away and will never come back assuming that they do not use GPS devices. More importantly, problems in controlling drones can result in harming navigator or other people [15]. All these problems are caused by the fact that navigating drones is a complex task for inexperienced users due to undeveloped and non-intuitive remote controller technologies. This thesis will focus on investigating alternate control modalities that are aimed to mitigate these issues.

## Related Work

Human Robot Interaction (HRI) is a comparatively new subset of study in HCI (Human Computer Interaction). Lots of efforts in the field of Robotics have been spent on the evolution of hardware and software to expand the functionality of robotic platforms. In comparison to the considerable increase in the variety of robots and their capabilities, the techniques people utilize to command and control them remain fairly unchanged. As robots are being deployed in more challenging settings, necessity for more meaningful and intuitive approaches to control them has become apparent among HRI researchers.

In terms of interface design, Goodrich and Olsen [18] provided a general guide to HRI researchers on how to design an effective user interface.

Drury et al. [19] defined a set of HRI taxonomies and conducted a thorough study [20] on how to improve human operators' awareness of rescue robots and their surroundings. All the consecutive works are based on these terminologies.

Quigley et al. [12] investigated several different interfaces for controlling drones, ranging from an input interface where the user should enter numerical values for navigating the UAV, to utilizing a physical model of the drone as the controller. They utilized a PDA to control the non-physical interaction plans. Their focus in their studies was to control a remote drone rather than a collocated one. Further, they used a voice control interaction. In this part of their studies, they used talking mechanism to a PDA in order to control a remote drone. This study is not going to utilize the power of voice control but we see this kind of interaction modality as closely connected to our research effort.

The development of input device technologies caused recognition of the natural user interface (NUI) as the evident evolution of the human-machine interaction, following the shift from command-line interfaces (CLI) to graphical user interfaces (GUI).

Humans use gestures to communicate effectively between themselves. Gesture is the commonly used term for a symbol that gets used to command, instruct or converse what is expressed through hand postures or hand movements. Starting from early eighties, voice and gestures are used to control User Interfaces [16]. In a 2008 conference presentation "Predicting the Past," August de los Reyes described the NUI as the next evolutionary phase following the shift from the CLI to the GUI [17].

There have been numerous research studies conducted exploring gesture-based interaction for collocated ground robots [21].

Also Ng et al. tried to utilize hand gestures for controlling UAVs [10]. They used hand gestures to control both low-level and high-level movements of a robot; in these research studies the drone needed to see the user with the onboard camera, therefore drone and user needs to be collocated, which is a considerable limitation.

Most of gesture-based recognition research use robot’s built-in camera to run some image-processing task to recognize gestures. Several issues arise ranging from environments with complex backgrounds, different lighting conditions and also real-time execution limitations. For example, Iba et al. proposed architecture for gesture-based control of mobile robots [22]. A data glove captured the gestures and using Hidden Markov Model classifiers gesture recognition was done. The interpreted gestures were translated into commands to control the robot. As another study, markers got used instead of data gloves [23]. Using the information gathered with two cameras triangulation was done to position the hand markers. This allowed high precision controlling of a 6DOF robot using gestures. A trajectory-based hand gesture recognition, which uses kernel density estimation and the related mean shift algorithm, was presented in [24].

As mentioned before all these approaches are based on image processing techniques for gesture recognition.

Another novel study utilized a Kinect-based natural interface for quadrotor control [13]. They used body movements for navigation purposes [14]. The favorable aspect of this study was using Microsoft Kinect, a commercial motion sensing input devices that enables users to control and interact with their computer without the need for a game controller, through a natural user interface using gestures and spoken commands. Using this kind of motion sensing devices abate the problem of gesture recognition. The researcher can rely on the data provided by Kinect and they do not need to run image-processing algorithms for recognizing gestures because Kinect already solved this part of the problem. The downside of this study is that body part movements are not very comforting and receptive to control a drone compared to hand gestures. Intuitively, a user is less inclined to lean forward or backward or raise the right arm up/down to control the drone. This approach of controlling is very tedious when the navigator is trying to do it for a long time. Our research tries to address this problem.

A newer study tries to define a consensual and non-ambiguous set of gestures to Interact with UAVs [25]. Taralle et al. propose a methodology consisting of 4 steps for eliciting a gestural vocabulary. (1) Collecting gestures through user creativity sessions; (2) extracting candidate gestures to build a catalogue, (3) electing the gesture vocabulary and (4) evaluating the non-ambiguity of it.

## Thesis Hypothesis

Humans commonly use gestures in their everyday lives to communicate more efficiently. Researchers have designed and implemented several human-computer interaction (HCI) paradigms based on gestures, which are called Natural User Interfaces (NUIs). Among NUIs, gesture-based interfaces always played a pivotal part in human-robot communication, because they are intuitive to the human being. The naturalness and diversity of hand gestures in comparison to traditional interaction approaches, can propose distinctive opportunities for novel and attracting forms of Human Computer Interaction.

As discussed earlier, we propose that using an interactive hand-gesture based approach can help the novice users to have an overall enjoyable and safe experience quickly learning how to navigate a drone with ease, and decrease the chance of losing or damaging the vehicle while they are on the initial learning curve.

This study aims to explore a more intuitive mechanism for controlling a quadcopter, using hand gestures as a source of input for the drone.

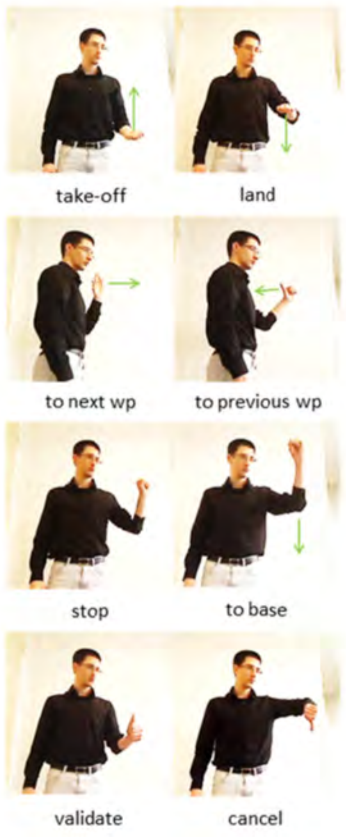
# Gesture Controller Design and Implementation

## Gesture Language

**Gestures** are defined as a hand movement with a recognizable start and end - for example, a wave, a tapping motion, a swipe right or left. A **pose** is a hand position that is held motionless for a few moments - for example, holding up some fingers to indicate a number, pointing, or making a stop sign.

With the growth of new technologies, gesture enables new human to Human Machine Interactions (HMI). Recently, Taralle et al. [25] performed a very thorough research on finding the right gesture set for interacting with a UAV. At the end they elected the gestures shown in figure 2.

Figure 2 - The 8 elected gestures for the symbolic commands of UAV defined by Taralle et al.



The process of finding and electing the right gesture was a daunting task for us as well. We tried lots of different gestures and measured their performance. The critical issue for us as researchers was that we needed to find gestures that are intuitive enough for an inexperienced user to be able to navigate smoothly. Also, we were limited by the Leap Motion technology as well.  Leap Motion sensor will sit on a table. As a result, we cannot use it for recognizing a "thumbs-up" gesture; basically because the Leap cameras are not able to see that. All this caused us to re-elect our gesture vocabulary several times during the course of this study.

To be able to pilot drones using gestures, the following gestures were elected. The gestures needed not only to be intuitive for the user but also easy enough for gesture recognizer to classify. They needed to have distinctive features.

Figure 3 - Elected Gesture for the "turn left" command

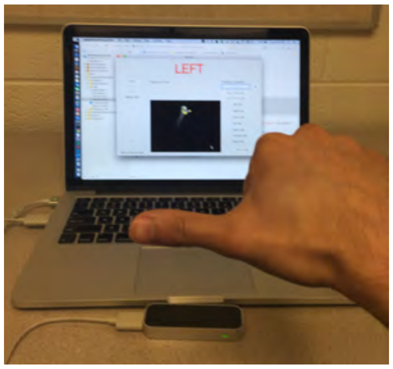


Figure 4 - Elected Gesture for the "turn right" command

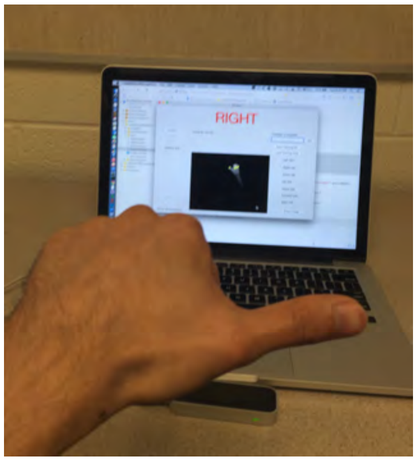


Figure 5 - Elected gesture for the "move forward" command

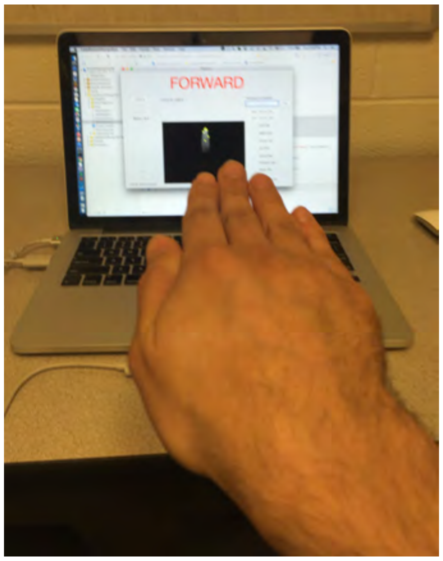


Figure 6 - Elected gesture for "ascend" command

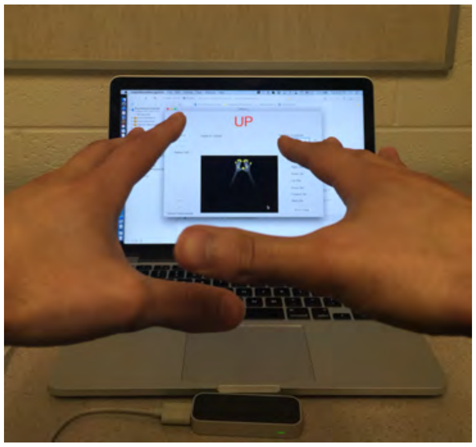
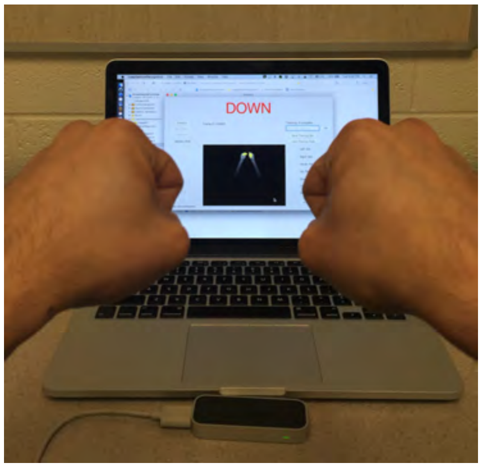
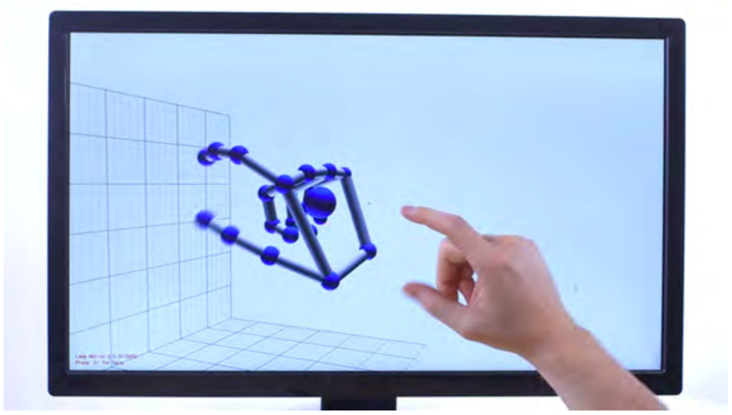


Figure 7 - Elected gesture for "descend" command



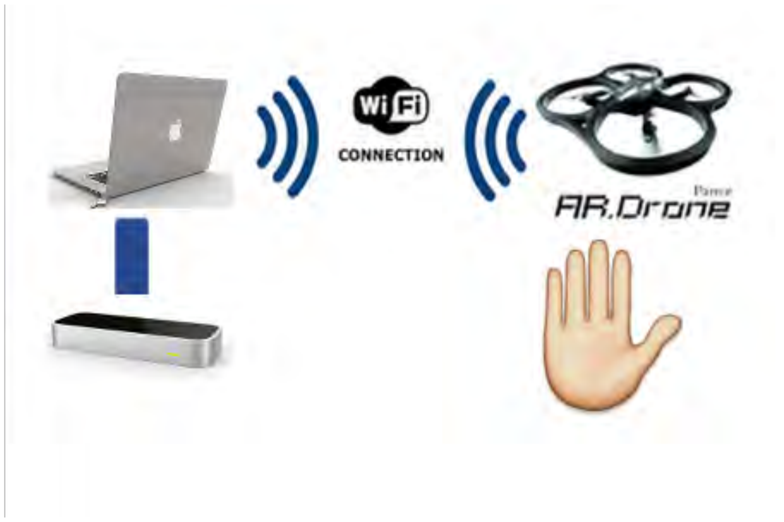
After defining the gesture language we used a Parrot Bebop quadcopter along with Leap Motion - a computer hardware sensor device that supports hand and finger motions as input, analogous to a mouse, but requiring no hand contact or touching (Figure 7) for the purpose of this research.

Figure 7. Hand gestures tracking with Leap Motion Sensor



The sensor connects to a computer using an USB port. The computer will gather raw data from the sensor and the software system embedded in the computer will decide based on the gesture type which command it should send to the quadcopter (Figure 8).

Figure 8. A high-level description of the system



## Architecture Design

The first thing that needed to be done was breaking the problem into smaller problems. The problems of navigating drones using hand gestures is comprised of the following sub-problems:

1. Implementing a gesture and pose recognition sub-system
2. Implementing a drone controller sub-system that connects to the drone.

The critical part here is to have a robust gesture/pose recognizer. Having a partially functional system wouldn’t be very useful due to the fact that piloting a drone is a highly critical task and there would be no place for a mistake.

Leap motion itself is comprised of three simple LEDs and two inexpensive cameras. The critical thing here was designing and investing on software algorithms, which can accurately specify what kind of gesture they are seeing.

## Machine Learning

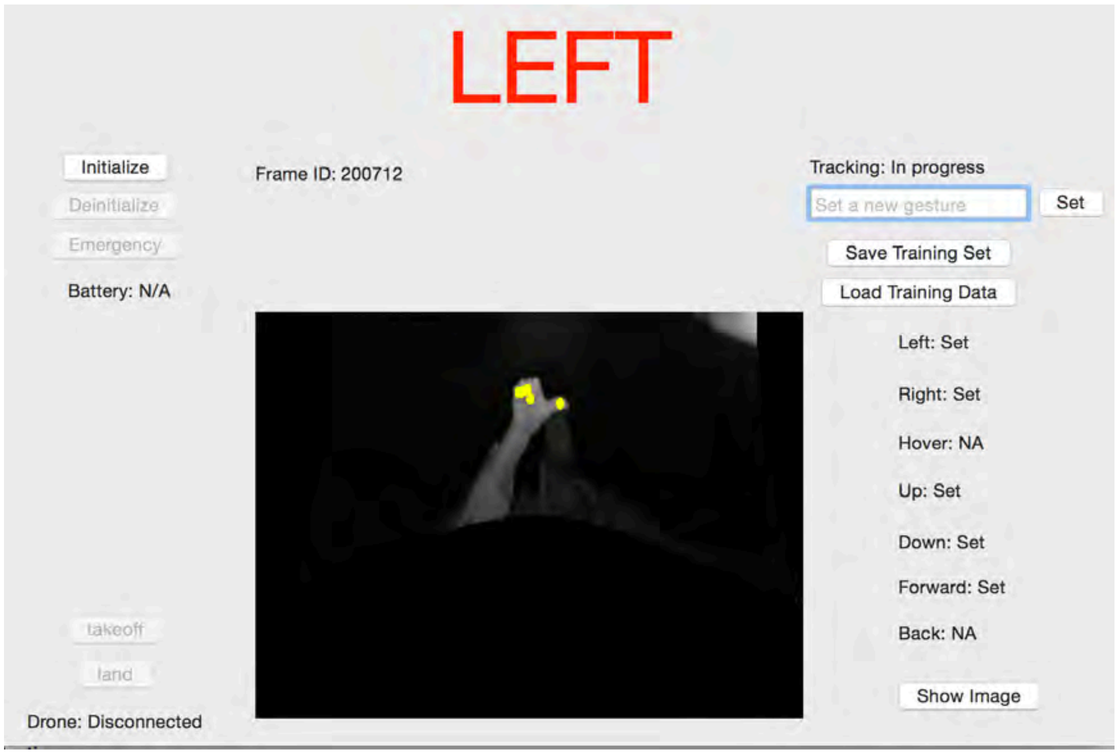
For decades, the computer systems have lots of limitations on how to conserve resources through clever, pre-built algorithms and instructions. Machine Learning helps to revolutionize this process. With the incredibly low cost of CPU cycles, it is now effective to throw large amounts of data at the computer and let it sort things out. In this study, to implement the gesture recognizer, a Supervised Machine Learning mechanism was utilized. The first step to do was to gather lots of quality data for training of the system. This data is many of frame videos captured from two different test subjects’ hand motions. Using the provided user interface (Figure 9), the trainer would define a gesture and then would ask two different people to perform the gesture when the program is in training mode. The program will capture hand movements and will associate them with the defined gesture. Later, when the application is in test mode, a new user can perform a gesture (from a list of predefined/pre-trained gestures) and the system will recognize it as a gesture. Below, the details involved in every phase are described.

Before being able to set the computer off to learn, the important thing is to collect quality data. Leap Motion API will provide the developer with data about every frame. In each frame, the developers will know how many hands are involved in the current gesture; they can have access to the position of each tip (as a vector containing x, y and z) (Figure 10), the stabilized palm position and velocity of fingers and palm position. Together, all these information can be used in the recognizer.

The method being used here is supervised learning which is the machine learning task of inferring a function from labeled training data.

Every supervised learning method includes two phases: (1) Training, (2) Testing. In training phase, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). The training data consists of a set of training examples. An example will make it clear. If for instance, the trainer desires to define a "left" gesture pose, he will first need to define the gesture type on the UI. After that the program will ask him to define this new "left" gesture by performing it in front of the motion-tracking sensor (training phase). After completing the gesture for a defined number of times, the system will learn about the new gesture. Therefore, the next time, when a new user completes a gesture/pose similar to the already defined "left" gesture, the recognizer system will calculate the probability of a match and if the number is greater than a predefined threshold, it will recognize it as a "left" gesture pose (testing phase).

Figure 9 - Application's User Interface. The yellow dots are the positions of the tips and palms, which are provided by the Leap Motion API.



Battery: N/A

Frame ID: 200712

Tracking: In progress

Set a new gesture

Left: Set

Right: Set

Hover: NA

Up: Set

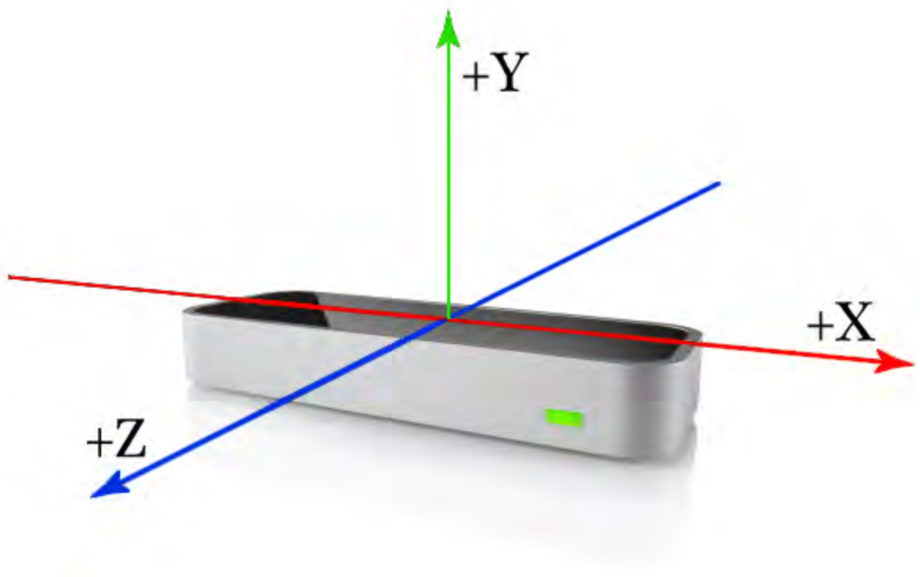
Down: Set

Forward: Set

Back: NA

Drone: Disconnected

Figure 10 - The Leap Motion controller uses a right-handed coordinate system



On this research, for training part, the data gathered using Leap Motion have to be hand tagged to indicate at which points the subjects performed each gesture. The human-tagged data is referred to as the "ground truth", essentially the gold standard the recognizer will be measured against.

To be able to have a gesture-based system we needed to be able to identify gestures and poses as defined earlier. The key point here is that gesture recognition starts when hand movement suddenly speeds up, and ends when it slows down or stops. For example, a quick swipe to the left will trigger gesture recognition algorithm. Pose recognition on the other hands, starts when movement suddenly stops and remains more or less unchanged for a short period, so just holding a thumps up will activate pose recognition mechanism.

Once capturing a large set of tagged data, the next step would be to decide which attributes or features are important to make the recognizer functional. This is a very important process and it's very probable that researcher makes a mistake on finding the right set of features. The features we wish to use have to be relatively easy to compute. For this research, during the training phase, for every frame associated to a gesture/pose, we collected six points of data: the center of the palm position along with position of tip for each finger.

The decision on what kind of feature should get extracted from the data was by trial and error. We tried different features and we ended up using the center of the palm position along with position of tip for each finger.

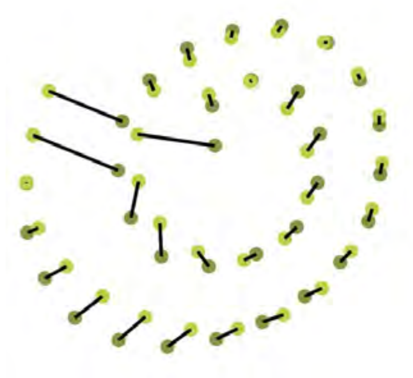
For instance, going back to our prior example, if the user is completing a left gesture with one hand, for every frame during gesture recording, we will have an array of six points. Also, due to the fact that every gesture/pose action contains of hundreds of frame we will have a huge array of points for each defined gesture.

Rather than the typical programming model of developing code that produces a desired output, machine learning systems like those used in this research rely on sets of desired output (the tagged data initially provided to the system. The tagged data – discussed earlier - allow the machine learning algorithm to produce a recognizer (essentially machine-generated code) that can be run on real data to identify the target gestures.

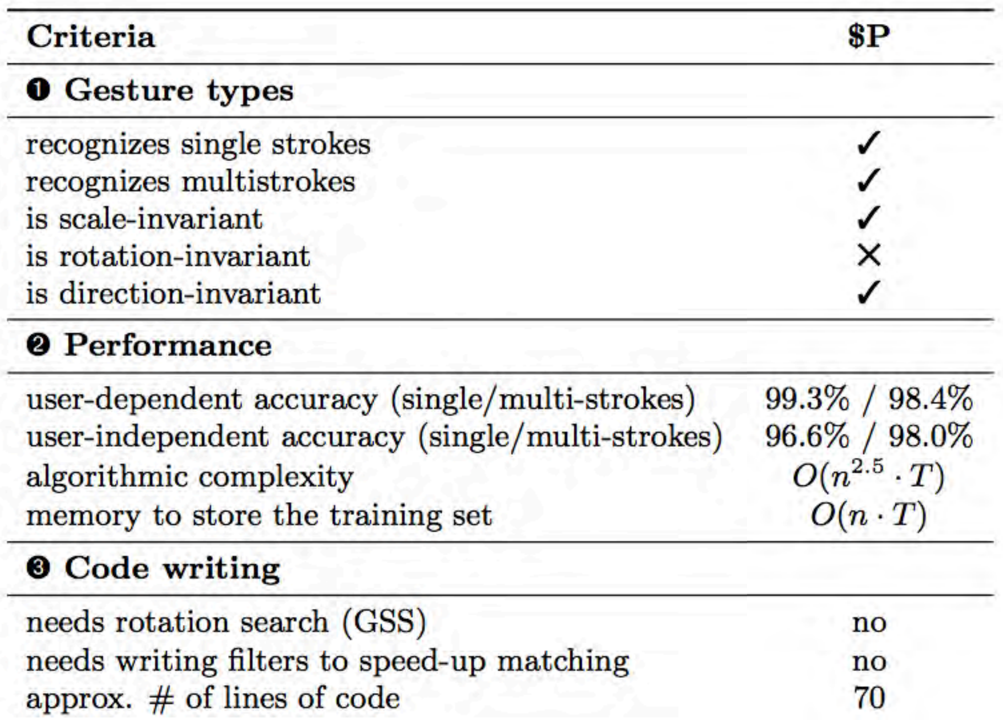
As the next step, the extracted features got fed into a machine learning system, typically a variant of one of several openly available systems implementing an algorithm like Decision Trees, SVM or $P recognizer.

The differences between various machine learning software algorithms is not very important if we have enough training tagged data [30]. With lots of data they will converge to very similar results. Although for this research, the choice of algorithm was a crucial step because we lacked having a huge set of quality data at the beginning. We needed to generate enough data as the first step and then feed it into the best algorithms suitable for those types of data. Initially the decision trees algorithm were used but the results were not very promising. As mentioned, if we had lots of training data, decision trees would be a nice choice but here we lacked having a huge set of training data. Later we switched to $P algorithm because it is designed initially for recognizing gestures.

**Figure 11 - Point alignments for two spiral gestures using $P algorithm**



The $P Point-Cloud Recognizer [26] is a 2-D gesture recognizer designed for rapid prototyping of gesture-based user interfaces. In Machine Learning terms, $P is an instance-based nearest-neighbor classifier with a Euclidean scoring function, i.e., a geometric template matcher. $P does not represent gestures as ordered series of points (i.e., strokes) but as unordered point clouds. By representing gestures as point-clouds, $P can handle both unistrokes and multistrokes equivalently. $P by default is a 2-D recognizer (supporting x and y). For use of this project, we extend it to support points in 3-D coordinate system (supporting z as well).



| Criteria | $P |
|---|---|
| **❶ Gesture types** | |
| recognizes single strokes | ✓ |
| recognizes multistrokes | ✓ |
| is scale-invariant | ✓ |
| is rotation-invariant | ✕ |
| is direction-invariant | ✓ |
| **❷ Performance** | |
| user-dependent accuracy (single/multi-strokes) | 99.3% / 98.4% |
| user-independent accuracy (single/multi-strokes) | 96.6% / 98.0% |
| algorithmic complexity | $O(n^{2.5} \cdot T)$ |
| memory to store the training set | $O(n \cdot T)$ |
| **❸ Code writing** | |
| needs rotation search (GSS) | no |
| needs writing filters to speed-up matching | no |
| approx. # of lines of code | 70 |

$P contributed a high level of accuracy in recognizing gestures to this research.

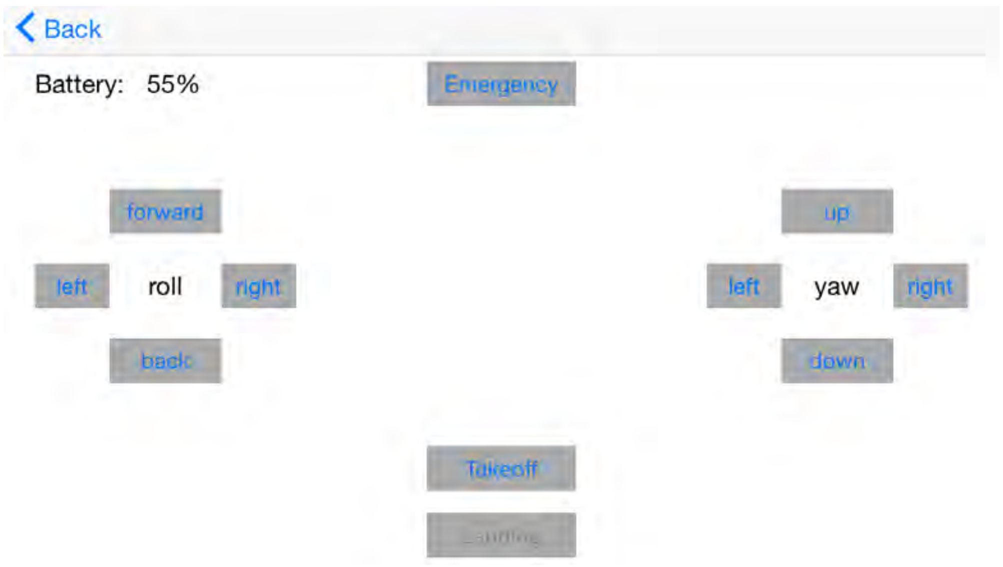
## Sending Commands to the Drone

After being done with the gesture recognition system, the second part of the research was interpreting the gestures and mapping them to an appropriate command for the drone, for example moving forward, backward, turning left and right and etc. For this part a complete C implementation for the Bebop drone was created. The API helps a user to connect to the drone and sends it appropriate commands. The gestures recognizer will recognize a gesture and after that it will send the appropriate command to the drone using the C API.

## Experiment Design

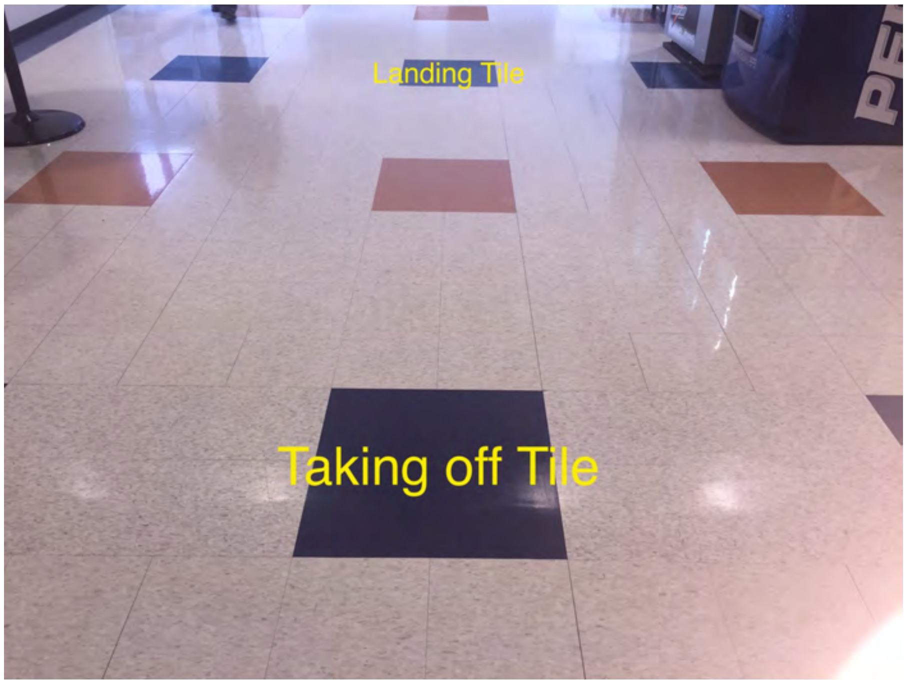
After implementing the proposed gesture-based system, the efficiency of the solution got examined. A comparative analysis involving the implemented gesture-based interface was ran against the traditional multitouch method. The Bebop drone comes with a multi-touch controller. For the purpose of the study a simpler multitouch controller implemented that only supported a subset of the original controller that paralleled the functionality in the experimental gesture-based controller.

Figure 12 - A simple multi-touch controller implemented for iOS systems



A sample of users was recruited to try out both systems. Then, we asked them to navigate a very basic pre-specified route using multi-touch conventional approach using a controller implemented for this purpose (Figure 12). Each user was asked to use the experimental gesture-based approach (Figure 9). The users were asked to perform a complete flight session from takeoff to landing using both provided methods. Before every run, the users were trained on the execution of each test. The trainer illustrated the features of the different interfaces to the trainee. The mission consisted of taking off the drone from a specific starting point, moving forward, landing in a specific point; taking off again from that point, turning and returning to the starting point (Figure 13).

**Figure 13 - Defined drone's mission**



Different commands were involved in this test making it a representative set of typical maneuvers to compare the different user interfaces. Each user was able to test each method twice. Only the best results from the two trials were included in the research data. The results were gathered in objective terms like the accuracy of visiting the checkpoints. If the drone completely fit into the tile after landing, a full grade (5) assigned to the pilot, points were reduced depending on how much of the drone was in the square. A grade of (1) was assigned if the drone was completely out of the target square upon landing. Moreover, subjective evaluations were considered. At the end, each user was asked to fill in a questionnaire about the usability of the proposed interface, including questions related to the perceived robustness of the gesture system, the level of intuitiveness of the gesture controller and easiness of using the system.

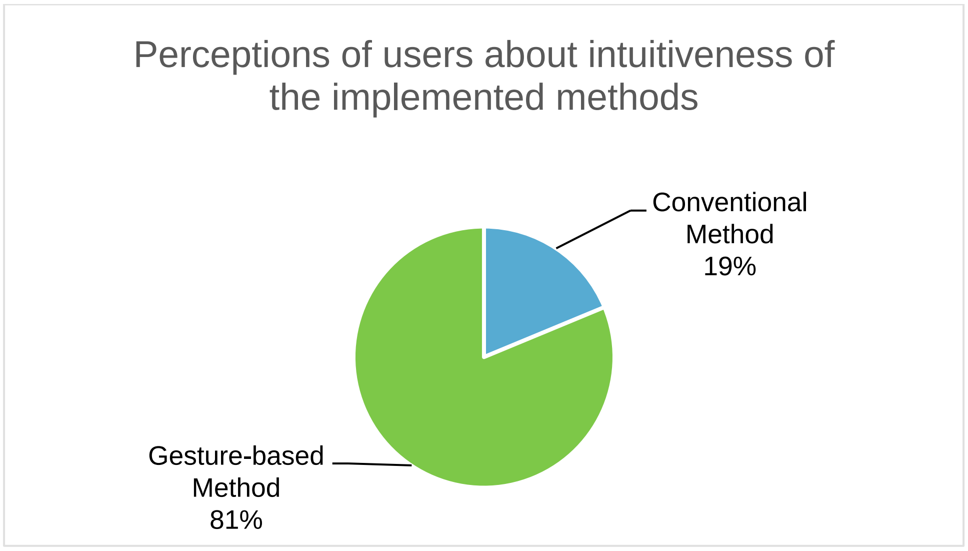
**Table 1 - Grading Criteria**

| ***Measure*** | ***Range of value*** | ***Comments*** |
|---|---|---|
| Perception | Conventional/Gesture-based | - |
| Robustness | 1 to 5 | 1: Less Robust<br><br>5: More Robust |
| Easiness to learn | 1 to 5 | 1: Hard to Learn<br><br>5: Easy to Learn |
| Accuracy | 1 to 5 | 1: Less Accurate in Landing<br><br>5: More Accurate in Landing |

## Analysis of Experimental Results

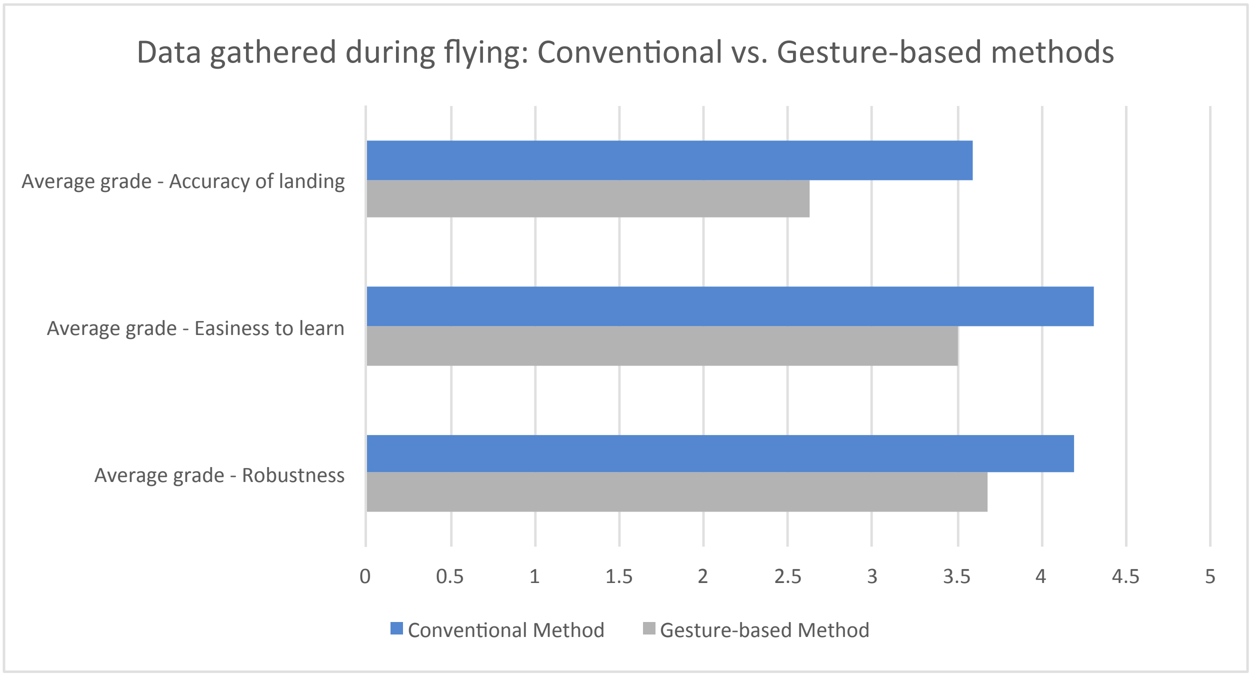
32 individuals were asked to fly the drone using both controlling approaches. Before piloting, almost 81% of them had the preconception that using the gestures to control a drone is more intuitive than multi-touch controller (figure 14). This was completely in accordance to thesis hypothesis at first that gesture based navigation should be perceived as a more intuitive approach for controlling drones.

**Figure 14 – User's perception on which method is more intuitive**



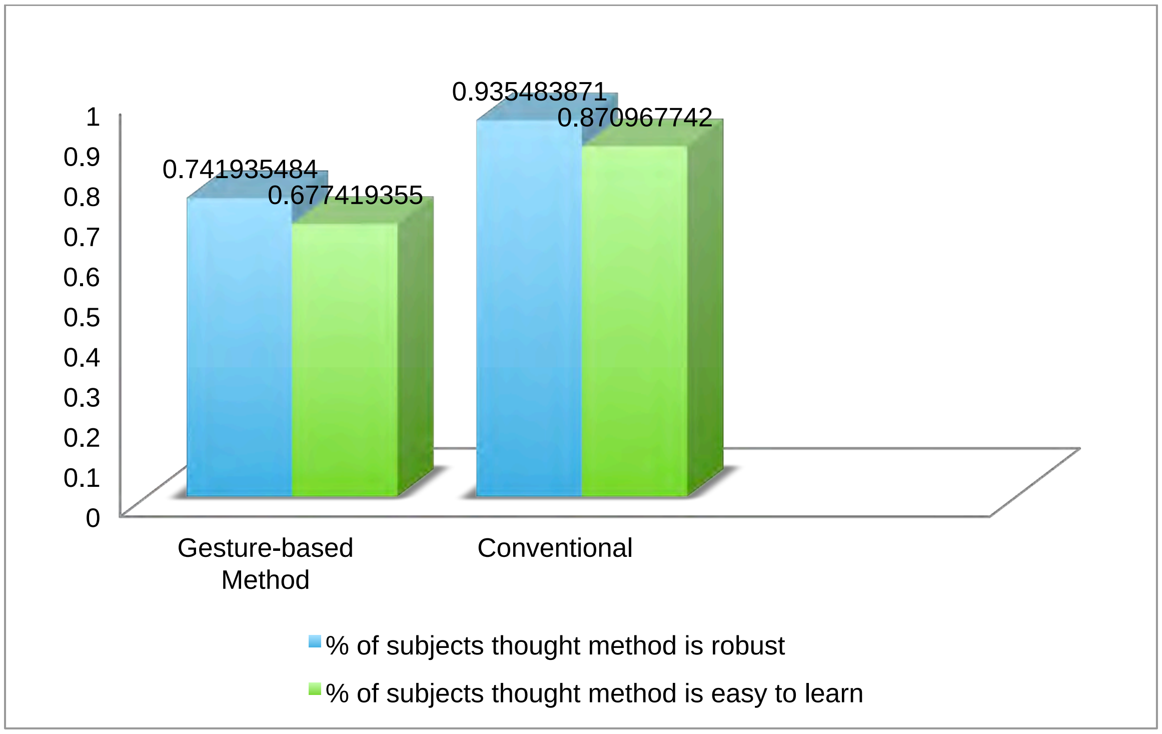
Further, the opinion of subjects was asked after flying to find out what method was more robust. Robustness was defined as "the ability of a system to resist change without adapting its initial stable configuration". Also, we asked users to provide us with data on how easy each method was to learn. After each flight, the trainer examined the accuracies of landings. The user had the chance to fly two times using each method. Only the highest grade was recorded for this study.

Figure 15 – Conventional vs. gesture-based methods



In addition, another comparison was conducted on different methods of controlling.

Figure 16 – Percentage of users favored one approach to the other



Although a majority of test subjects agreed about intuitiveness of gesture control (figure 14), there are still other considerations that need researcher's attention.

As you can see in figure 15 and 16, most subjects' impressions were that conventional method of controlling is more robust than the new proposed method (94% vs. 75%). Considering the fact that using gesture controller involves nondeterministic factors such as the quality of the sensor capturing hand gestures, quality of data gathered during training phase, lighting conditions of the environment during the test and the suitableness of the machine learning algorithm in hand, it makes sense that majority of trainees thought conventional method is more robust after testing both approaches. Based on our experience, it is safe to conclude that robustness and predictability  is a key characteristic of a gesture controller, and that it is far more difficult to achieve that robustness using conventional gesture recognition technology. Conventional multi-touch controllers are in the market for several years so far and they have a very deterministic nature. For instance, if pilots press and hold the forward button on the controller, drone will move forward and if they release the button drone will hover. Regardless of whether or not this approach is intuitive, the robustness does lend it the upper hand in our tests. On the other hand, using gesture controller, first there is a need to make sure that the gesture in use is sensitive enough and doesn't discriminate between different people with various hand structures. Then for the systems to be functional, we need to make sure that the training data set fed to the system covers different types of the same gesture. For example for a "left gesture" to be diverse enough, we need left gestures of different people with big, normal and small hands in varying lighting conditions. Ensuring that system has enough training data is a very crucial step for the machine learning algorithm. With enough training data the results would converge to very similar results [30]. For the purpose of this research, our training data set was consisted of 60 samples for each gesture type. The samples belonged to two different persons, one with big hands and another with small hands. The recognizer system almost on 95% of times captured and classified the hand pose/gesture as the anticipated gesture. Just in 5% of occasions, the system recognized a left gesture as a right gesture and vice versa. Having more training data the recognizer could overcome these limitations and have a more accurate gesture recognizer.

Surprisingly, more people revealed that the conventional controller method was easier to learn than the proposed gesture method (87% vs. 69%). Some test subjects ran into problems while trying to use the gesture input sensor. Although they were holding their hand correctly above the sensor and saw the infrared picture of their hand on the user interface, the data provided by the Leap Motion frame was invalid. It means that sometimes, with certain hand types, the sensor does not provide valid and reliable information to recognizer system. As a side effect, users were performing a specific kind of gesture but the drone didn’t do the appropriate command. The problem had a negative impact on the pilot’s perceived view of the whole system and users concluded that using the gesture system is harder and has a steeper learning curve.

This problem might be mitigated via using a different kind of sensor like Microsoft Kinect, which is another commercial motion sensing input device. When it comes to gesture recognition, there is no need to focus too much on the sensor’s hardware because the real magic is the software; although, having a reliable, competent sensor that provides us with valid information about the hand movements is essential for a highly-critical task like navigating a drone. The pilots need a dependable gesture recognition system in order to be able to pilot the drones smoothly; for example, when they issue a ‘move forward’ gesture they expect the drone move forward. The worst thing for a pilot is to have a fuzzy gesture recognizer that identifies a gesture occasionally and this requires a competent hardware. Another workaround for this problem would be to run some image processing techniques on the images provided by the sensor like the research Microsoft conducted for recognizing grip/release gestures using Microsoft Kinect [27].

For the purpose of this study, the researcher relied on the data provided by calling Leap Motion APIs. To get the position of the tips and palm along with the velocity of each point, in each recordable frame this piece of code ran:

```
- (void)recordFrame:(LeapFrame*)frame {
    for(LeapHand* hand in frame.hands) {
        [self recordVector:hand.stabilizedPalmPosition];
        for(LeapFinger* finger in hand.fingers) {
            [self recordVector:finger.stabilizedTipPosition];
        };
    };
}
```

It means that here we merely relied on the data provided by Leap via API. For a more detailed study to find position of the tips, center of the palm and their related velocity, a frame-to-frame analysis in the depth map at each pixel of the provided image could be used. These images are captured by the sensor (Both Kinect and Leap motion function using these images). This method of image processing can guarantee that valid data are being captured as long as the infrared cameras that sensor uses are able to capture images of the hands.

## Lessons Learned / Contributions

**Implementing a gesture-based controller to pilot a drone is substantially more difficult than originally anticipated.** Considering that the results gathered for gesture-based method are close to multi-touch method, it is highly plausible that utilizing a better sensor with better accuracy could very well yield a gesture-based interface that outperforms the conventional interface.  The fact that users' default perception that a gesture interface would be superior matched our own research hypothesis seems to indicate that having two methods with similar performance in terms of robustness/accuracy, the gesture-based method will be deemed superior. Relying on APIs provided by Leap Motion did not yield sufficient robustness/accuracy and hence were not preferred by the end user. It means that the accuracy of the hardware used in the gesture controller is as important as the algorithms employed.

**Though intuition indicates gesture-based controllers are more intuitive than the existing conventional controls for piloting a drone, a lack of robustness/dependability in the gesture controller implementation radically reduces the utility of this approach.** Piloting a drone is a highly critical task. To have a competitive controller compared to the conventional method, we need a robust accurate mechanism to eliminate the risk involved in piloting the drone. The gesture recognizer should function correctly 100% of times. Even 99% is not enough. Therefore, the training set should consist lots of data points gathered from different people in different conditions (small, normal, big hands in different lighting conditions and in different depths, with different skin colors).

**Coming up with an intuitive distinctive gesture language for essential drone commands was harder than what was expected initially.** The important thing when defining gestures was that we were looking for intuitive gestures; gestures that a new user can relate to quickly with no thought.  This limited the list of possible gestures. For instance, for "move backward" command, we were not able to find an intuitive gesture.

**An open source C / Objective-C library for interfacing with the Parrot Bebop Drone has been contributed by this effort.**  The Parrot Bebop Drone is brand new and because of that, the SDK is still young and immature. Open source library created for this research can be found here. (https://github.com/gvsucis/c-bebop-drone)

**Our gesture language implementation is now freely available via Leap Motion's website**. (https://developer.leapmotion.com/gallery/leap-gesture-recognizer)

## Further studies

The traditional navigation approaches for controlling drones (robots) have some significant flaws. This research's hypothesis was that using gesture-based mechanisms to control a drone is a potential solution for the problems discussed. Although using gestures offered promises, a perfect gesture-based controller is much more challenging than what seemed at first. As a plausible study, utilizing more accurate sensor along with more training data would be interesting. Further, using other Machine Learning Algorithms like Multilayer Neural Networks or Support Vector Machines could result in better outcomes.

Another interesting research would be to reduce the dimension of feature space from 6 points to just 3 points. The gestures defined in this research can be well-separated by using only three points (position of center of the palm, position of thumb, and position of the middle finger) therefore reducing the dimension of the feature space from 6D to 3D. The formal way of confirming this is to run some distance metrics and verify whether using the selected features we can minimize within cluster distances (a cluster must be compact) and maximize between cluster distances (two different clusters must be far apart). Also, use of some data visualization software may help to understand the structure of the high-dimensional feature space.

Another point of interest in the data collected is that the people who tried the gesture-recognition system several times had robust, accurate flights. This means that getting experienced in using gesture-based controller can result in more robust aerial navigations. To support this hypothesis, another research can be run.

The expectation is that by tweaking gesture-based controller and making it more sensitive, the novice user have an overall enjoyable and safe experience quickly learning how to navigate a drone with ease, and avoid losing or damaging the vehicle while they are on the initial learning curve.

As another kind of research designing and implementing simultaneous multimodal interfaces could be plausible. Gestures can be combined with other modals like speech interface and multi-touch interfaces to reach an ideal interface. No single modality is optimal in every commands/control situation; combining the best modalities for each individual interaction might produce a more intuitive user interface. As a possible application, we can have a multimodal interface that supports multitouch, voice and gesture interaction. As a scenario the pilot can specify several checkpoints on a map with touching the interface. After that, using voice command he can initiate the flying session. Using hand gestures, during the flight the pilot can reroute the drone or basically make it to stay put and hover over a specific point. This potentially will lead to a better, easier to learn interface for the users.

# References

[1] Wall Street Journal, (2015). FAA Proposes Rules to Allow Commercial Drone Flights in U.S. [online] Available at: http://www.wsj.com/articles/obama-issues-privacy-rules-for-government-drones-in-u-s-1424015402 [Accessed 15 April 2015].

[2] Krajník, Tomáš, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl. "AR-drone as a platform for robotic research and education." In Research and Education in Robotics-EUROBOT 2011, pp. 172-186. Springer Berlin Heidelberg, 2011.

[3] Mellinger, Daniel, Nathan Michael, and Vijay Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors." The International Journal of Robotics Research (2012): 0278364911434236.

[4] Achtelik, Markus, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments." In SPIE Defense, Security, and Sensing, pp. 733219-733219. International Society for Optics and Photonics, 2009.

[5] Blosch, Michael, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. "Vision based MAV navigation in unknown and unstructured environments." In Robotics and automation (ICRA), 2010 IEEE international conference on, pp. 21-28. IEEE, 2010.

[6] Michael, Nathan, Jonathan Fink, and Vijay Kumar. "Cooperative manipulation and transportation with aerial robots." Autonomous Robots 30, no. 1 (2011): 73-86.

[7] Bills, Cooper, Joyce Chen, and Ashutosh Saxena. "Autonomous MAV flight in indoor environments using single image perspective cues." In Robotics and automation (ICRA), 2011 IEEE international conference on, pp. 5776-5783. IEEE, 2011.

[8] Bills, Cooper, and Jason Yosinski. MAV stabilization using machine learning and onboard

sensors. Technical Report CS6780, Cornell University, 2010.

[9] Faigl, J., T. Krajnık, V. Vonásek, and L. Preucil. "Surveillance planning with localization uncertainty for mobile robots." In 3rd Israeli Conference on Robotics. 2010.

[10] Ng, Wai Shan, and Ehud Sharlin. "Collocated interaction with flying robots." In RO-MAN, 2011 IEEE, pp. 143-149. IEEE, 2011.

[11] Ar.Drone, http://ardrone2.parrot.com

[12] Quigley, Morgan, Michael A. Goodrich, and Randal W. Beard. "Semi-autonomous human-UAV interfaces for fixed-wing mini-UAVs." In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, pp. 2457-2462. IEEE, 2004.

[13] Sanna, Andrea, Fabrizio Lamberti, Gianluca Paravati, Eduardo Andres Henao Ramirez, and Federico Manuri. "A Kinect-based natural interface for quadrotor control." In Intelligent Technologies for Interactive Entertainment, pp. 48-56. Springer Berlin Heidelberg, 2012.

[14] Controlling the AR Drone with Microsoft Kinect, http://www.youtube.com/watch?v=jDJpb4xXAJM

[15] Remote Controlled Helicopter Kills Man in Brooklyn, http://blogs.wsj.com/metropolis/2013/09/05/remote-control-helicopter-kills-man-in-brooklyn/

[16] Bolt, Richard A. “Put-that-there”: Voice and gesture at the graphics interface. Vol. 14, no. 3. ACM, 1980.

[17] De los Reyes, August. "Predicting the Past". Web Directions South 2008. Sydney Convention Centre: Web Directions.

[18] Goodrich, Michael A., and Dan R. Olsen. "Seven principles of efficient human robot interaction." In Systems, Man and Cybernetics, 2003. IEEE International Conference on, vol. 4, pp. 3942-3948. IEEE, 2003.

[19] Yanco, Holly A., and Jill L. Drury. "Classifying human-robot interaction: an updated taxonomy." In SMC (3), pp. 2841-2846. 2004.

[20] Yanco, Holly A., Jill L. Drury, and Jean Scholtz. "Beyond usability evaluation: Analysis of human-robot interaction at a major robotics competition." Human–Computer Interaction 19, no. 1-2 (2004): 117-149.

[21] Rogalla, O., M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann. "Using gesture and speech control for commanding a robot assistant." In Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on, pp. 454-459. IEEE, 2002.

[22] Iba, Soshi, J. Michael Vande Weghe, Christiaan JJ Paredis, and Pradeep K. Khosla. "An architecture for gesture-based control of mobile robots." In Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on, vol. 2, pp. 851-857. IEEE, 1999.

[23] Kofman, Jonathan, Xianghai Wu, Timothy J. Luu, and Siddharth Verma. "Teleoperation of a robot manipulator using a vision-based human-robot interface." Industrial Electronics, IEEE Transactions on 52, no. 5 (2005): 1206-1219.

[24] Popa, Daniel, Georgiana Simion, Vasile Gui, and Marius Otesteanu. "Real time trajectory based hand gesture recognition." WSEAS Transactions on Information Science and Applications 5, no. 4 (2008): 532-546.

[25] Taralle, Florent, et al. "A Consensual and Non-ambiguous Set of Gestures to Interact with UAV in Infantrymen." Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems. ACM, 2015.

[26] Vatavu, Radu-Daniel, Lisa Anthony, and Jacob O. Wobbrock. "Gestures as point clouds: a $ P recognizer for user interface prototypes." Proceedings of the 14th ACM international conference on Multimodal interaction. ACM, 2012.

[27] Real-World Machine Learning: How Kinect Gesture Recognition Works: http://channel9.msdn.com/Events/Build/2013/3-704

[28] Assessing the potential for a new drone—powered economy http://www.businessinsider.com/the-market-for-commercial-drones-2014-2

[29] Amazon Prime Air, http://www.amazon.com/b?node=8037720011

[30] Domingos, Pedro. "A few useful things to know about machine learning." Communications of the ACM 55.10 (2012): 78-87.