2019 International Conference on Unmanned Aircraft Systems (ICUAS)
Atlanta, GA, USA, June 11-14, 2019

Gesture Commands for Controlling High-Level UAV Behavior

John Akagi, Brady Moon, Xingguang Chen, Cameron K. Peterson

Abstract— **In this paper, an accelerometer and gyroscope are used to sense gesture commands, which are then classified using a logistic regression model. Seven gestures were chosen and mapped to specific behaviors that a fixed wing unmanned air vehicle could accomplish. These behaviors specified various searching, following, and tracking patterns that could be used in a dynamic environment. The system was trained to recognize the seven gestures and then tested in a hardware-in-the-loop simulation. The system was able to identify all gestures with an overall accuracy of 90% and with five of the seven gestures being accurately identified at least 94% of the time. Each of the behaviors associated with the gestures was tested in simulation and the ability to dynamically switch between behaviors was proven. The results show that the system can be used as a natural interface to assist an operator in directing an unmanned air vehicle's behaviors.**

I. Introduction

The usage and applications of Unmanned Aerial Vehicles (UAVs) have grown immensely in recent years as a result of decreasing prices and increasing controllability and versatility. However, UAV operation continues to require continual monitoring, close attention, and training. As a result, UAV operators are typically focused exclusively on manually piloting their UAV and relaying necessary information from the UAV to those around them. This research seeks to reduce the complexity of UAV operations by developing high-level algorithms that enable UAVs to autonomously perform behaviors with minimal guidance from an operator. Additionally, we seek to reduce the attention burden on a UAV operator by developing an interface that is capable of intuitively directing UAV actions with minimal interruption to other tasks the operator may be performing.

Consider a UAV operator who is attempting to walk through an area while avoiding hostile targets. Using the traditional method of UAV operation, a two joystick controller and image data being streamed from the UAV, the operator can only focus on either flying the UAV or walking through the area. Attempting to simultaneously walk while flying the UAV can lead to suboptimal UAV behavior and a distracted operator. This paper develops a method of reducing the complexity required for operators to work together with a UAV.

In addition to the concentration requirement mentioned above, the traditional method of controlling UAVs with two joysticks requires training and an understanding of UAV dynamics. Both take time and resources to obtain. This has led to investigations of more intuitive control interfaces. Some of these interfaces have used verbal commands [1]–[3], eye-tracking [4]–[6], and physical model manipulation [7] to control UAVs.

This project uses gesture commands to send desired behavior objectives to a UAV. The use of gestures has been shown in [8] and [9] to be an intuitive method that is easy for human operators to understand. In both of those studies, a human pilot and UAV worked together to imitate a vision-based control scheme where a participant would gesture to direct the UAV. The pilot would fly the UAV based on what he felt the gesture represented. Although the participants were aware the UAV was actually being manually controlled, they reported that commanding the UAV in this manner was natural and not physically or mentally demanding.

Actual implementation of gesture based controllers have used various sensing instruments, namely on-board cameras, off-board stereo vision, touchpads, electromyograph sensors, and accelerometers. In [10] and [11], a camera on board the UAV is used to detect human movements and respond to commands. This approach is attractive since there is no need for electronic communications between the operator and UAV but is limited in range since the UAV needs to be able to see the gestures.

In [12] and [13], a Leap Motion Controller was used to detect an operator's commands. The Leap Motion Controller is a sensor that is able to detect 3 dimensional hand gestures using stereo vision and IR LEDs [14]. This capability has been used to build paths by selecting flight segments [12] and to instruct a UAV to takeoff, land, or perform a flip [13]. The commercial availability of the Leap Motion Controller offers a relatively quick and simple way to implement gesture control, but its design makes it difficult to use when the operator is walking.

A touchpad was also used to detect touch gestures consisting of taps and swipes in [15]. This study had participants control a UAV and a gimbaled camera using both the traditional two joystick method and touch gestures. The major contribution of this study was the abstraction of the combined UAV and camera dynamics. Instead of iteratively moving the UAV and camera to reach the desired orientation, all commands were interpreted as being relative to the camera field of view and then automatically remapped to provide the individual commands for the UAV and camera. The results indicated that while both the joystick and touch gestures methods were able to complete the assigned tasks, the touch gesture method reduced the workload on the operator since

J. Akagi is with the Department of Mechanical Engineering, Brigham Young University, akagi94@gmail.com

B. Moon is with the Department of Electrical and Computer Engineering, Brigham Young University, bradygmoon@gmail.com

X. Chen is with the School of Electronics and Information Technology, Sun Yat-sen University, chenxg8@mail2.sysu.edu.cn

C. Peterson is faculty in the Department of Electrical and Computer Engineering, Brigham Young University, cammy.peterson@byu.edu

they only needed to worry about the higher level task they were supposed to accomplish.

Electromyography (EMG) sensors that detect the electrical impulses in muscles have also been used to control UAVs via hand gestures in [16] and [17]. In [17], a controller is implemented where the UAV can be commanded to takeoff, land, move forward, backward, left, or right depending on the hand gesture made. Additionally, magnetometer and gyroscopic measurements are used to distinguish directions and eliminate noise from non-gesture movements. A similar command scheme is used in [16] except the user is controlling a group of robots by either directing a leader or issuing instructions to the full group.

The final common method for detecting gestures is the use of devices worn on a user's body that measure motion. In [18], a three-axis accelerometer is used to detect the orientation of the user's hand and command the UAV to move forward, backward, left, or right depending on that orientation. This device was primarily tested to assist users with limited mobility who may be unable to properly operate the joysticks of a typical controller. The accelerometers in wearable smart devices were used in [19] to detect steps and to give commands such as "gain altitude", "lose altitude", and "take a picture".

While all of the alternative control schemes simplify the training needed to operate a UAV, they still require an operator's undivided attention. Generally, the operator provides low-level commands to the UAV such as move forward, backward, turn, land or takeoff. These commands all require that the UAV operator be aware of the location and attitude of the UAV. Additionally, UAVs are unable to execute their desired behavior without constant operator attention.

This paper examines a user protection mission where a UAV is used to scout ahead, track discovered targets, circle the operator at various distances, and search a bounded area as directed by an operator. A gesture controller was designed and prototyped which allows the operator to send these high-level behaviors to the UAV without needing to divert attention from their other tasks. As mentioned above, the use of gestures to control UAVs has been shown to reduce the strain on an operator. The contribution of this paper is to show that UAV commands can be further abstracted to enable the UAV to execute behaviors with minimal guidance.

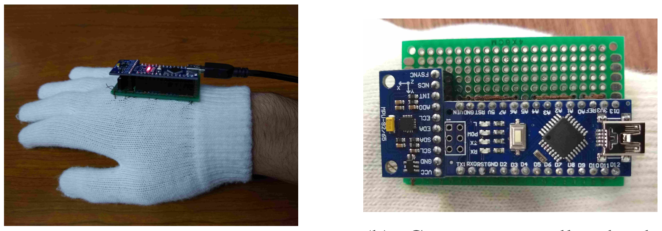
We discuss the framework developed to test the gesture controller and explain the gesture controller hardware, simulation environment, UAV behaviors, and path planning algorithm in Section II. Simulation results are shown in Section III and conclusions are presented in Section IV.

II. Methodology

The detailed explanations of the gesture controller, UAV behaviors, and simulation are described in the following subsections.

A. *Gesture Controller*

A prototype gesture controller was designed and built using an Arduino Nano and an MPU-9250, a 9-axis inertial



(a) Gesture controller as worn while in use.

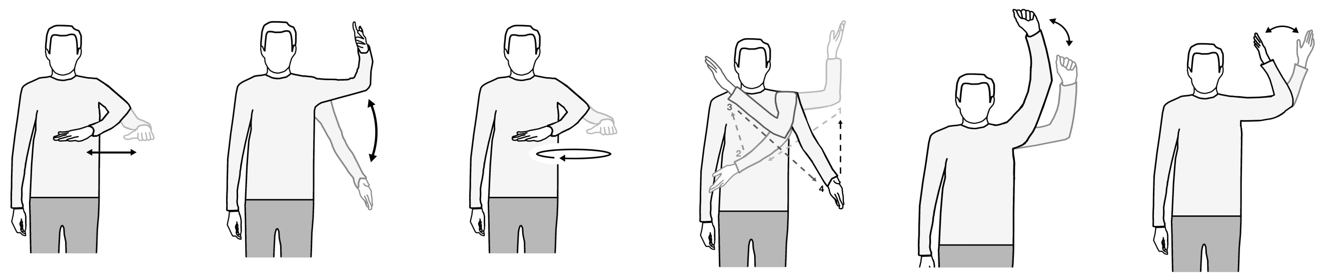
(b) Gesture controller hardware.

Fig. 1: The gesture controller as worn when attached to the glove. The Arduino board is the longer board closer to the wrist, while the IMU is the shorter board closer to the knuckles.

measurement unit (IMU) consisting of an accelerometer, gyroscope, and compass. The electrical components were placed on protoboard and attached to the back of a glove (see Figure 1). As a result of the limited computing capabilities of the Arduino Nano, it was used only to communicate data from the IMU to the computer running the simulation. The accelerometer and gyrosocope measurements for each of their 3 axes were reported at a rate of approximately 100 Hz using an I2C connection between the IMU and the Arduino and serial communication over USB between the Arduino and the desktop computer.

The gesture controller was used to collect data for 7 different hand and arm motions which are shown in Figure 2 and described as follows: 1) a sweeping side to side motion in front of the operator with the palm facing down, 2) a repeated chopping or pointing motion made with the palm vertical, 3) a small counter-clockwise circle made with the palm down, 4) a large counter-clockwise circle made with the palm down, 5) a repeated waving motion in front of the operator with the palm vertical, 6) an 'X' pattern traced in the air in front of the operator, and 7) an up-down motion or fist pump over the operator's head. These gestures were chosen because they are periodic and do not have determined start or stop locations. This simplifies recognizing commands since the operator does not need to coordinate the start of the gesture with the moment the gesture controller starts recording data. Additionally, the gestures are unique and can express rough ideas such as direction, circling, and size. The specific behaviors triggered by each gesture can be found in Section II-B.

We used a logistic regression to identify the gestures based on the frequency data calculated from the combined six axes of the accelerometer and gyroscope. Data is collected from all six axes and transmitted to the off board computer, via the Arduino Nano, at a rate of 100 Hz. For training data, 3 minutes of data for each gesture was collected in 1 minute periods with an author performing each gesture repeatedly. The data was continually passed from the Arduino to an offboard computer and recorded using MATLAB. The mean μ and standard deviation σ were then found for each each



(a) Sweeping motion (b) Chopping or pointing motion (c) Small or large circling motion (d) 'X' pattern (e) Up and down motion or fist pump (f) Waving motion

Fig. 2: The different gestures that were used with the gesture controller.

axis, and the data was normalized such that [20]

$$x' = \frac{x-\mu}{\sigma}. \tag{1}$$

This mean and standard deviation was saved and used to normalize the data for gesture prediction during the simulations.

The training data was then split into 1 second segments with a rolling window so that each window was offset by one sample. Using the rolling window allows the periodic nature of the gestures to be leveraged and increases the training data set. In the end, these 1 second raw data segments consisted of 100 raw acceleration and angular velocity data points along the combined six axes.

Each axis of the raw data segment was transformed to the frequency domain using a Fast Fourier Transform (FFT). The amplitudes of all six axes were concatenated together and comprised one set of data that was fed into the logistic regression. From the generated data, 85% was used to train the logistic regression and the remaining 15% was used to validate the model.

B. Behavior Commands

As mentioned, each gesture is mapped to a unique behavior or task that the UAV executes. The behaviors are 1) search a wide area in front of the operator, 2) search a deep area in front of the operator, 3) search in a small circle around the operator, 4) search in a large circle around the operator, 5) search the entire area, 6) track targets, and 7) search the entire area while tracking targets. Each of these behaviors correspond to the respective gestures in Section II-A.

1) Search Behavior: The area search reward function incentivizes UAVs to visit unexplored areas. The reward function utilizes a grid pattern of i equally spaced grids, each with a reward J_g. The reward corresponds to a probability that the grid cell contains a target. When the center of a grid cell falls within the sensing radius of the UAV, it is assumed that the UAV was able to see the whole grid cell and detect any targets that were within it. The UAV is rewarded by the summation of all grid cell rewards seen during that timestep.

The grid cell values are updated as follows. If the grid cell was seen during that timestep, then $J_i = 0$. If the grid cell was not seen during that timestep, then the value is updated such that [21]

$$J_i[n+1] = J_g - (J_g - J_i[n])\, e^{-t\gamma} \tag{2}$$

where n is the current timestep, $n+1$ is the next timestep, t is the length of time since the the grid cell has been seen, and γ is a constant that describes how fast the value of the grid cell returns after being seen.

For the deep area search and wide area search behaviors, the UAV is instructed to only search the grid cells whose centers lie within a certain triangular area in front of the operator. The area is defined by an angle θ and a distance r such that the points of the triangle are the operator's location, a point r meters in front and θ degrees to the left, and a point r meters in front and θ degrees to the right of the operator. These search areas are both relative to the heading of the operator and so will translate and rotate as the operator changes location and orientation.

If the UAV is outside of this area when commanded to do a deep or wide area search, or if the operator turns such that the UAV is no longer in the area, the UAV uses a PID controller to fly towards the midpoint of the search area. Once the UAV's position is within the search area, it stops using the PID controller and returns to maximizing the search reward.

2) Loiter Behavior: The two loiter behaviors have the UAV continually circle around the location of the operator. The operator's location is assumed to be measured by GPS and transmitted to the UAV such that it always knows the operator's current position. Given the location of the operator $\mathbf{c}$, the location of the UAV $\mathbf{p}$, the current heading of the UAV χ, the desired loiter radius ρ, and the distance between the operator and UAV d, the desired course heading for the UAV can be calculated [22] as

$$\chi^C = \phi + \lambda\left[\frac{\pi}{2} + \tan^{-1}\left(k_{orbit}\left(\frac{d-\rho}{\rho}\right)\right)\right], \tag{3}$$

where

$$\phi = \arctan 2\left(\mathbf{p}_e - \mathbf{c}_e, \mathbf{p}_n - \mathbf{c}_n\right) + 2\pi m,$$

with $m \in \mathbb{N}$ such that $-\pi \leq \phi - \chi \leq \pi$.

The parameters λ and k_{orbit} are used to define the loiter circle and transition to into a circular orbit. A value of $\lambda = 1$ indicates the UAV will circle in a clockwise orbit while a value of $\lambda = -1$ indicates it will circle in a counter-clockwise orbit. The value $k_{orbit} > 0$ defines the rate at which the UAV will transition from a straight line path to a circular orbit as the UAV approaches the desired loiter location. Finally, the radius ρ is set before the simulations begins and is constant throughout the duration.

3) Target Tracking: When target positions fall within the sensing radius of the UAV, the UAV is able to obtain noisy range and bearing measurements that describe the target's location relative to the UAV.

The state for target m at time step n is given by $\mathbf{x}_m[n] = [N_m[n], E_m[n], \dot{N}_m[n], \dot{E}_m[n]]^T$ with estimated state values denoted by $\hat{\mathbf{x}}_m[n]$ where N and E are the north and east positions, respectively. In this simulation, we assume that each target obeys a nearly-constant velocity model and update our estimated position of targets with an extended Kalman filter. To calculate the total information gained by the UAV, the information matrix $I = P^{-1}$, where P is the target's error covariance, is calculated before and after the estimated target position is updated. The total information gained for each target is then [21], [23]

$$J_T = \ln |I[n]| - \ln |I[n-1]|. \tag{4}$$

4) Combined Searching and Tracking: The combined searching and tracking behavior follows the same principles referred to in Sections II-B.1 and II-B.3 where reward is gained for grid cells and targets seen. The only difference is that in the combined version, the rewards for each of these individual behaviors are directly added. This incentivizes the UAV to search out as much area as possible while still keeping the target within its sensing radius. Additionally, if the uncertainty in the target's position is low enough, the UAV may stop observing the target to instead check areas where additional, unseen targets may be located. The behavior is primarily influenced by setting the the max grid cell value, J_g, prior to the mission beginning where a higher value prioritizes the search behavior.

C. Simulation

The simulated environment consists of targets, grid cells, an operator, and a fixed wing UAV. The targets are modeled on vehicles travelling on real world road networks. The road networks were based on maps from OpenStreetMap [24] and used with Simulation of Urban Mobility (SUMO) [25] to create realistic driving patterns. The actual positions and headings of each target are indicated by a red 'x' and arrow, respectively.

The grid cells indicate discrete areas where targets have a possibility of existing. As mentioned, grid cells start will a certain value that correlates to a probability that a vehicle exists inside the cell. When the UAV is able to see the center of a grid cell its reward value drops to zero. If a grid cell has not been seen by a UAV the cell value gradually increases according to Equation 2.

The operator's position and heading are indicated by a black 'x' and arrow, respectively. Both these values are assumed to be measured by GPS and communicated to the UAV so that it always knows the operator's current position. The operator is assumed to be a person walking at approximately 1 m/s and is not constrained to a road network.

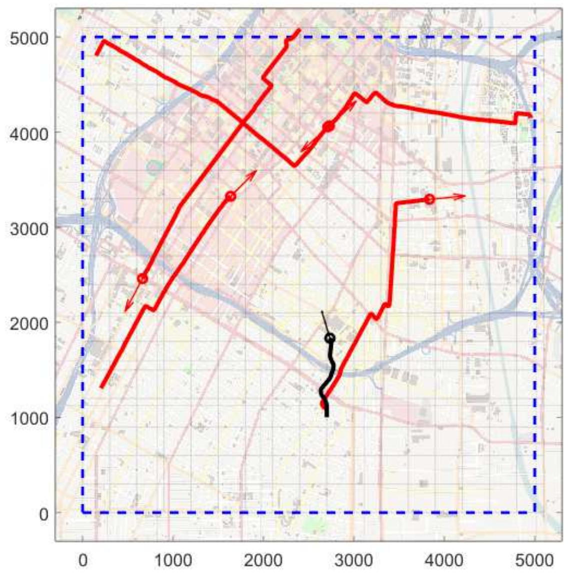


Fig. 3: A representation of the constant data in the simulation. The underlying road network can be see as well as the target paths (red), the operator path (black), and the boundaries (dashed blue).

The UAV position and heading are indicated by a small blue circle and arrow, respectively. Additionally, a large, dashed circle indicates the sensor footprint. It is assumed that the UAV is able to measure all targets and grid cells that fall within its footprint It is assumed thet the UAV flies at a constant altitude.

Depending on the behavior assigned to the UAV, there are two possible control schemes for determining its path. The first method is an orbit following path as described in Section II-B.2. This method is used for the two loiter behaviors and defines a course heading that points towards the center of the loiter when the UAV is far from the desired location, and is tangental to the desired loiter when the UAV is close to the desired location. A PID controller is used to adjust the roll angle so that the UAV heading matches the calculated heading.

The second method is a rollout method based on a receding horizon control. In this method, the possible UAV bank angles are discretized into three potential options: a full left bank, a full right bank, or no bank. The rollout policy works by initializing a set number of paths using an exhaustive search consisting of all possible combinations of left, right, and no turns. Once the desired number of paths are initialized, each path expands out to the time horizon using an exhaustive search algorithm where the three possible next steps are simulated and the step that is predicted to give the highest reward from the search and/or tracking behaviors is chosen. Once all paths have been expanded out to the time horizon, the path with the highest expected overall gain is chosen and the UAV begins travelling down that path.

TABLE I: A confusion matrix showing the number of performed gestures and the number of gestures identified by the logistic regression model. Although the Small Circle gesture proved difficult for the model to identify, the remainder of the results show good performance.

		Actual Gestures						
		Sweep	Point	Small Circle	Large Circle	Wave	X Pattern	Fist Pump
Predicted Gestures	Sweep	43			1			
	Point	1	50	11	1			
	Small Circle			28				
	Large Circle	2		11	47			
	Wave					50		
	X Pattern	4			1		47	
	Fist Pump						3	50

When the simulation is running, the gesture controller only transmits data when instructed to do so in order to reduce computation requirements and avoid unintended gesture commands. When the operator wants to change the behavior assigned to the UAV, a keypress triggers a flag that alerts the MATLAB code to listen for data from the Arduino and interpret the gesture. The data is recorded, normalized, transformed to the frequency domain, interpreted using the logistic regression model, and the UAV task is updated according to the classified gesture.

III. Simulation Results

In this section, we discuss the results obtained by testing the gesture controller accuracy and by running hardware-in-the-loop simulations to verify the individual UAV behaviors. Finally, the results of a simulation where varying behaviors were commanded is discussed. The parameters used in each simulation are listed in Table II.

TABLE II: A list of the parameters used for each behavior. The target tracking behavior has no configurable parameters.

Behavior	Parameter	Value
Wide Search	r (Search Depth)	800 m
	θ (Search Angle)	60°
Deep Search	r (Search Depth)	100 0m
	θ (Search Angle)	45°
Small Loiter	ρ (Radius)	100 m
	λ (Direction)	1 (Clockwise)
Large Loiter	ρ (Radius)	800 m
	λ (Direction)	1 (Clockwise)
Full Area Search	J_g (Grid Value)	40
	γ (Growth Rate)	1/5000
Combined Search and Track	J_g (Grid Value)	40
	γ (Growth Rate)	1/5000

A. Gesture Testing

When in use during the hardware-in-the-loop simulations, 120 samples or 1.2 seconds of data are collected from the each axis of the accelerometer and gyroscope while the gesture is being performed. The data is then divided into segments 100 samples long using a rolling window with each segment offset by five samples. The data was normalized using the mean μ and standard deviation σ of each sensor axis, which were calculated and recorded during the training phase. Each individual axis of each segment was transformed into the frequency domain using an FFT and the amplitudes of these transformations were all concatenated together. With the offset and rolling window, a total of five vectors were created. Each of these were evaluated using the logistic regression model and the gesture that was identified the most from the five data sets was chosen. The use of five data sets creates redundancy to reduce the impact of noise while only slightly increasing the time needed to input a gesture.

The training and verification data showed a high degree of accuracy when training the model, 99.87% and 99.85% respectively. Additionally, testing was done to validate that gestures could be performed, measured, and identified live with high accuracy. In order to test this, a single gesture was performed repeatedly while the 1.2 seconds of data were gathered. The predicted gesture for each of the 5 data sets was recorded as well as the actual gesture that was being performed. This process was repeated for each of the 7 gestures to conclude one round of data collection. Overall, this process was repeated 10 times so that each of the 7 gestures were classified 50 times. The actual and predicted gestures were then compiled into a confusion matrix (Table I) to determine the accuracy of the gesture controller and the similarity of the various gestures.

Overall, the gesture controller was able to correctly identify the gestures with good accuracy. The one gesture of note is the Small Circle gesture which was only correctly identified 56% of the time. 22% of the time, the gesture was misinterpreted as the Large Circle gestures which is understandable given that the gestures have the same basic pattern. The other 22% of the time, the gesture was mistaken to be the Point gesture, although the similarities with the Point gesture are less clear. The inclusion of both the Small Circle and Large Circle gestures were to test the ability of the gesture controller to additionally measure a parameter of the gesture being performed, in this case the radius of the circle. These results show that measuring parameters is possible but a more sophisticated model, such as a Recursive Neural Network, is likely needed in order to achieve high accuracy.

For the remainder of the gestures, the accuracy is high enough for practical use. Additionally, because enough data is gathered to perform 5 predictions, a level of protection is added against false identifications. As long as there are no more than 2 incorrect predictions, the gesture controller will still be able to, overall, correctly identify the gesture. These results show that the gesture controller is able to identify the main gestures although more advanced models would be useful to identify small differences in gestures.

B. Behavior Testing

There were 7 different simulations run to test the gesture controller and each individual behavior. All simulations were run on the same map, for a simulation time of 300 seconds, with the starting positions of the targets, UAV, and operator being the same. Additionally, the paths of the targets and operator did not change between the simulations (as shown in Figure 3). The starting location of the operator and UAV were such that they were both in close proximity to each other and to one of the targets so that the track target behavior could be performed without needing the UAV to first happen across a target.

For these simulations, the UAV was initialized with the small loiter behavior prior to the simulation being run. Approximately 30 seconds into each simulation, one of the 7 behaviors was then commanded using the gesture controller and the UAV performed that behavior for the duration of the run. The paths of the targets, operator, and UAV for each simulations are shown in Figure 4. Note that the figures have been cropped to better show the movement of the UAV.

1) Wide Search: The first behavior, a wide search of the area in front of the operator, was commanded using a side to side sweeping gesture with the operator's palm facing down. The wide search involved an area defined as extending 800 m in front of the operator and sweeping out 60° to either side (see Section II-B.1). When this behavior was being executed, the UAV stayed in an area almost directly in the center of the search area and did not search the far left or far right (see Figure 4a). As a result of the discretization of the search area, the grid cells to be searched (highlighted in green) do not perfectly match the defined search area. This incentivized the UAV to stay centered.

2) Deep Search: The second behavior, a deep search of the area in front of the operator, was commanded by using the pointing or chopping gesture and resulted in the UAV searching an area farther away from the operator (see Figure 4b). This area was defined as extending to a point 1000 m in front of the operator and sweeping 45° to either side. As with the wide area search, the areas that are considered to be in front of the operator are highlighted in green. Both the deep search and the wide search behaviors were successful in keeping the UAV within the designated area. However, in future work, methods to improve the coverage within that area will be explored.

A potential problem comes from the fact that relatively small changes in the heading of the operator can cause large changes in the defined search area boundaries. As the UAV moves away from the operator's location, a change in the operator's heading is more likely to move the search boundaries such that the UAV is outside the prescribed region. Although the UAV automatically flies toward the center of the prescribed area, this results in time spent outside the designated area. For example, note that in Figure 4b, the UAV ends the simulation on the border of the defined search area. This is because the UAV was searching the right side of the prescribed area when the operator began a left turn. The UAV is then outside the boundary and immediately begins to turn to return to the boundary. Overall, the impact of this is minor, but the behavior could be modified to better account for and predict operator motion.

3) Protect Behavior: The next two behaviors are a small loiter and large loiter around the operator. These behaviors represent the operator keeping the UAV close to their position or searching out an area centered around their position. The small search (see Figure 4c) was defined as a circle with radius of 100 m with a clockwise motion. The large search (see Figure 4d) was 800 m with a clockwise motion. Overall, the behaviors performed as expected with the UAV recalculating the desired orbit based on the changing operator positions. The orbits appear as elongated circles due to the continuous operator movement. It is assumed that the speed of the UAV is greater than the speed of the operator to ensure the UAV can sucessfully orbit its operator.

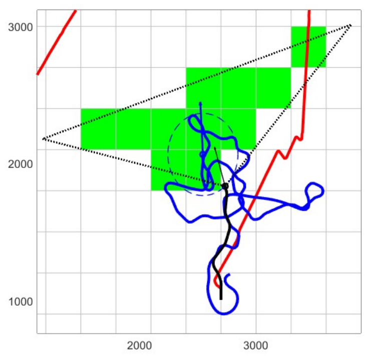
4) Full Area Search: The full area search behavior (Figure 4e) was commanded by the 'X' pattern and resulted in the UAV flying a pattern similar to an outward spiral. It should be noted that the pattern was not programmed and simply is the result of the UAV attempting to maximize the number of new grid cells that it can observe (see Section II-B.1). As a result, the overall path is similar to a spiral but the UAV tends to zig-zag in order to observe grid cells that are just out of sight on the left and right. As with the wide and deep area searches, the behavior of the full area search can be influenced by changing the sizes of the grids as well as the rate that the individual grid cell rewards increase (see Table II).

5) Target Tracking: The target tracking behavior was commanded with the up down gesture. As can be seen in Figure 4f, the UAV wtays as close to the target as possible. Since the UAV speed is higher than the target speed, the UAV is forced to make a series of loops and circles to stay in close proximity to the target. The behavior for tracking a single target could be changed to a simple loiter about the estimated target position. However, this would force the UAV to focus on a single target, while the tracking behavior as implemented allows the UAV to optimize its position when tracking multiple targets.

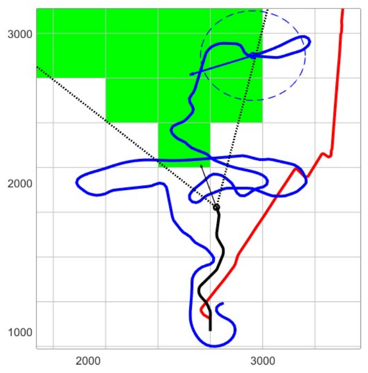
6) Combined Search and Track: The combined search and track behavior was commanded using the wave gesture and can be seen in Figure 4g. The UAV begins by primarily tracking the target and then, when the attitude of the target is sufficiently known, switches to primarily searching out new areas. The balance between tracking and searching is determined by the maximum value the UAV can get for seeing each grid cell (see Table II).

C. Multiple Behavior

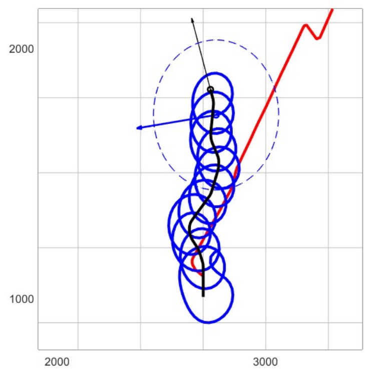
A final simulation (Figure 4h) involved multiple behaviors being commanded depending on feedback from the UAV. As with the other simulations, the UAV started with a small loiter behavior commanded. Approximately 30 seconds into the simulation, the large loiter behavior was commanded



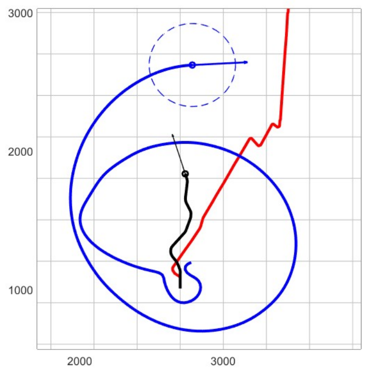
(a) The UAV was assigned to search a wide area directly in front of the operator. The areas to search during the final simulation step are highlighted in green.



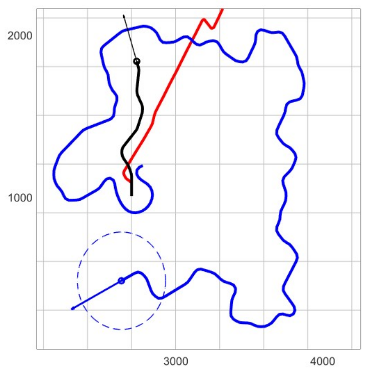
(b) The UAV was assigned to search a deep area in front of the operator. The areas to search during the final simulation step are highlighted in green.



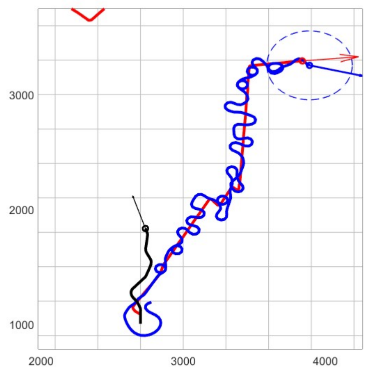
(c) The UAV was assigned to loiter around the operator at a distance of 100 m.



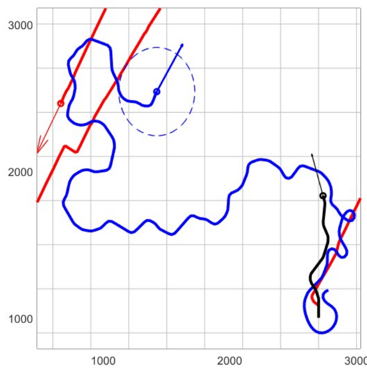
(d) The UAV was assigned to loiter around the operator at a distance of 800 m.



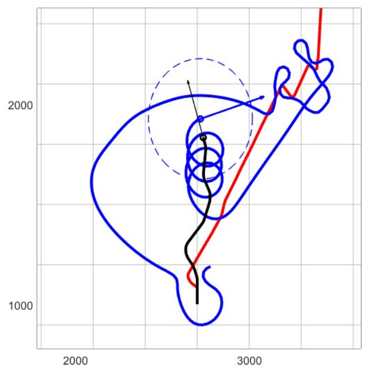
(e) The UAV was assigned to search the entire area.



(f) The UAV was assigned to follow known targets.



(g) The UAV was assigned to balance searching new areas and following known targets. Note how the UAV initially tracks the vehicle but eventually prioritizes searching.



(h) The UAV behaviors were changed in response to the environment. The UAV performs a large loiter, is switched to target tracking, and then is switched to a small loiter.

Fig. 4: Cropped images showing the path of the UAV as it executed the various behaviors in simulations

with the gesture controller. While executing this behavior, the UAV saw a target and was commanded by the operator to track it. Eventually, the small loiter behavior was commanded again and the UAV returned to the operator.

These behaviors were chosen because they represent a series of behaviors an actual operator may desire. Starting with little to no information about their environment, an operator could command a large loiter behavior to get a rough idea about their surroundings and see if there are any targets close by. Upon being alerted by the UAV that there was a target nearby the operator could command the UAV to track the target in order to better observe the target. Once sufficient information has been gathered, the operator may have the UAV return to them so that it is available to investigate another area or target.

IV. Conclusion

We have shown that accelerometer and gyroscope measurements can be used with a logistic regression model in order to identify a variety of gestures. These gestures were then mapped to individual behaviors that an operator might desire a UAV to perform, such as searching a specific area or following targets. The gesture controller was used in combination with a simulated environment to show that an operator could command multiple behaviors using only periodic gestures. The operator was able to react to the simulation and use the gestures to switch behaviors dynamically.

Since the controller has been proven in a hardware-in-the-loop simulation, the next step will be to implement the system in an outdoor flight experiment.

Acknowledgements

This work was funded by a BYU undergraduate research grant through the Office of Research & Creative Activities (ORCA) and by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award No. IIP-1650547 along with significant contributions from C-UAS industry members.

References

[1] A. C. Trujillo, J. Puig-Navarro, S. B. Mehdi, and A. K. McQuarry, "Using natural language to enable mission managers to control multiple heterogeneous uavs," in *Advances in Human Factors in Robots and Unmanned Systems*, pp. 267–280, Springer, 2017.

[2] M. Landau and S. van Delden, "A system architecture for hands-free uav drone control using intuitive voice commands," in *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 181–182, ACM, 2017.

[3] M. Chandarana, E. L. Meszaros, A. Trujillo, and B. D. Allen, "'fly like this': Natural language interface for uav mission planning," 2017.

[4] J. M. Ettikkalayil, "Design, implementation, and performance study of an open source eye-control system to pilot a parrot ar. drone quadrocopter," 2013.

[5] J. P. Hansen, A. Alapetite, I. S. MacKenzie, and E. Møllenbach, "The use of gaze to control drones," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 27–34, ACM, 2014.

[6] M. Yu, Y. Lin, D. Schmidt, X. Wang, and Y. Wang, "Human-robot interaction based on gaze gestures for the drone teleoperation," *Journal of Eye Movement Research*, vol. 7, no. 4, pp. 1–14, 2014.

[7] M. Quigley, M. A. Goodrich, and R. W. Beard, "Semi-autonomous human-uav interfaces for fixed-wing mini-uavs," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2457–2462, IEEE, 2004.

[8] J. R. Cauchard, K. Y. Zhai, J. A. Landay, *et al.*, "Drone & me: an exploration into natural human-drone interaction," in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pp. 361–365, ACM, 2015.

[9] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *RO-MAN, 2011 IEEE*, pp. 143–149, IEEE, 2011.

[10] J. Nagi, A. Giusti, G. A. Di Caro, and L. M. Gambardella, "Human control of uavs using face pose estimates and hand gestures," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pp. 252–253, ACM, 2014.

[11] J. Nagi, A. Giusti, L. M. Gambardella, and G. A. Di Caro, "Human-swarm interaction using spatial gestures," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 3834–3841, IEEE, 2014.

[12] M. Chandarana, A. Trujillo, K. Shimada, and B. D. Allen, "A natural interaction interface for uavs using intuitive gesture recognition," in *Advances in Human Factors in Robots and Unmanned Systems*, pp. 387–398, Springer, 2017.

[13] A. Sarkar, K. A. Patel, R. G. Ram, and G. K. Capoor, "Gesture control of drone using a motion controller," in *Industrial Informatics and Computer Systems (CIICS), 2016 International Conference on*, pp. 1–5, IEEE, 2016.

[14] A. Colgan, "How does the leap motion controller work." http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/, 2014.

[15] H. Kang, H. Li, J. Zhang, X. Lu, and B. Benes, "Flycam: Multitouch gesture controlled drone gimbal photography," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3717–3724, 2018.

[16] S. Nagavalli, M. Chandarana, K. Sycara, and M. Lewis, "Multi-operator gesture control of robotic swarms using wearable devices," in *Tenth International Conference on Advances in Computer-Human Interactions (ACHI)*, 2017.

[17] Z. Li, Y. Chen, and W. Tan, "Dronemyo: Proactive control of unmanned aerial vehicle based on wearable devices," in *International Symposium on Smart Graphics*, pp. 211–214, Springer, 2015.

[18] L. A. Sandru, M. F. Crainic, D. Savu, C. Moldovan, V. Dolga, and S. Preitl, "Automatic control of a quadcopter, ar. drone, using a smart glove," in *Proceedings of the 4th International Conference on Control, Mechatronics and Automation*, pp. 92–98, ACM, 2016.

[19] Y. Lu, F. Han, L. Xie, Y. Yin, C. Shi, and S. Lu, "I am the uav: A wearable approach for manipulation of unmanned aerial vehicle," in *Smart Computing (SMARTCOMP), 2017 IEEE International Conference on*, pp. 1–3, IEEE, 2017.

[20] T. Jayalakshmi and A. Santhakumaran, "Statistical normalization and back propagation for classification," *International Journal of Computer Theory and Engineering*, vol. 3, no. 1, pp. 1793–8201, 2011.

[21] A. J. Newman, S. R. Martin, J. T. DeSena, J. C. Clarke, J. W. McDerment, W. O. Preissler, and C. K. Peterson, "Receding Horizon Controller using Particle Swarm Optimization for Closed Loop Ground Target Surveillance and Tracking," *Signal Processing, Sensor Fusion, and Target Recogniton*, vol. 7336, no. 1, pp. 73360M–1–73360M–12, 2009.

[22] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[23] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom, "Autonomous surveillance by multiple cooperative uavs," in *Signal and Data Processing of Small Targets 2005*, vol. 5913, p. 59131V, International Society for Optics and Photonics, 2005.

[24] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ." `https://www.openstreetmap.org`, 2017.

[25] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.