Gesture commands for controlling high-level UAV behavior

John Akagi¹ · T. Devon Morris² · Brady Moon² · Xingguang Chen³ · Cameron K. Peterson²

Received: 22 September 2020 / Accepted: 15 April 2021

Published online: 01 May 2021

© The Author(s) 2021

Abstract

Directing groups of unmanned air vehicles (UAVs) is a task that typically requires the full attention of several operators. This can be prohibitive in situations where an operator must pay attention to their surroundings. In this paper we present a gesture device that assists operators in commanding UAVs in focus-constrained environments. The operator influences the UAVs' behavior by using intuitive hand gesture movements. Gestures are captured using an accelerometer and gyroscope and then classified using a logistic regression model. Ten gestures were chosen to provide behaviors for a group of fixed-wing UAVs. These behaviors specified various searching, following, and tracking patterns that could be used in a dynamic environment. A novel variant of the Monte Carlo Tree Search algorithm was developed to autonomously plan the paths of the cooperating UAVs. These autonomy algorithms were executed when their corresponding gesture was recognized by the gesture device. The gesture device was trained to classify the ten gestures and accurately identified them 95% of the time. Each of the behaviors associated with the gestures was tested in hardware-in-the-loop simulations and the ability to dynamically switch between them was demonstrated. The results show that the system can be used as a natural interface to assist an operator in directing a fleet of UAVs.

Article highlights

- A gesture device was created that enables operators to command a group of UAVs in focus-constrained environments.
- Each gesture triggers high-level commands that direct a UAV group to execute complex behaviors.
- Software simulations and hardware-in-the-loop testing shows the device is effective in directing UAV groups.

Keywords Autonomous vehicles · Cooperative control · Human–robot interaction · Gesture interface device

1 Introduction

While the capabilities and applications for Unmanned Aerial Vehicles (UAVs) have expanded drastically, their effectiveness is often limited by the number of available operators and the complexity of the assigned task. UAVs are generally flown by teams of two or more operators,

where one is the pilot and the others monitor data sent from the UAV [1].

Improvement in autopilot systems and sensor hardware have simplified UAV operation to the point that many tasks, such as obstacle avoidance, waypoint navigation, takeoff, and landing, can be done autonomously. This has led to the development of UAV systems with varying degrees of

✉ Cameron K. Peterson, cammy.peterson@byu.edu | ¹Department of Mechanical Engineering, Brigham Young University, 350 EB, Provo, UT 84062, USA. ²Department of Computer and Electrical Engineering, Brigham Young University, 450 EB, Provo, UT 84062, USA. ³School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, Guangdong, China.

autonomy. These range from autopilots that stabilize the UAV [2] to UAVs capable of independently exploring, mapping, and identifying targets in an area [3]. Despite continuous improvements, fully autonomous flight with many complex behaviors has not yet been realized and significant operator training may be required for effective control of a UAV [4]. One way to reduce the training needed is to provide the operator with the capability to select high-level commands such as flight direction, flight path, or mission. The UAV's position and attitude would then be driven by the low-level autonomy algorithms to complete the assigned goals.

For many tasks, the use of cooperating UAVs facilitates faster and more thorough completion. However, the need for at least one operator for each UAV outweighs many benefits obtained from a fleet. As the number of required operators increases, a fleet quickly becomes prohibitively expensive to field. Additionally, it also becomes increasingly difficult for their human operators to coordinate flight paths and prevent collisions. Algorithms and controllers that allow large numbers of UAVs to cooperate with a single human operator's oversight reduce both the cost of UAV fleet operations and minimize the workload placed on the operator.

This work presents a gesture device that triggers autonomy algorithms to command a fleet of UAVs to perform high-level behaviors. An operator may then effectively direct a UAV fleet with minimal interaction. The operator performs predefined hand gestures that are classified and mapped to desired UAV behaviors. Depending on the gestures made, different aspects of the UAVs' behavior are modified. This approach is beneficial in two ways. First, operators that would otherwise be needed to oversee operations of the UAV fleet are freed for other assignments. Second, the remaining operator may independently contribute to the mission beyond directing the UAVs.

The utility of this approach is driven by the UAV's ability to intelligently execute group behaviors that assist the operators in completing their tasks. To this end, we have developed a new decentralized cooperative path planning algorithm based upon the Monte Carlo Tree Search (MCTS). MCTS is a statistical anytime algorithm, meaning that the solution is guaranteed to continue to improve, but can be stopped at any time. This is particularly advantageous for real-time applications. We have enhanced the traditional MCTS algorithm to provide a decentralized algorithm that incorporates the dynamics of a fixed-wing UAV and is capable of cooperatively planning UAV paths with goals on both a macro and micro scale.

We implement a block coordinate descent MCTS (or CMCTS) method which conditionally optimizes the path of each UAV in turn, while holding constant the paths that the other UAVs are anticipated to follow. This results in conditionally optimizing individual UAV paths multiple times until

an overall solution is converged upon [5]. Using coordinate descent optimization causes the search space to grow linearly in the number of UAVs instead of exponentially.

Combined together, the CMCTS path planning algorithm and gesture device allows an operator to simply and intuitively control complex behaviors in a fleet of UAVs. Numerical and hardware-in-the-loop simulations are presented to demonstrate (a) the operator's ability to command a fleet of UAVs and (b) the utility of the gesture device in assisting an operator during search and track scenarios.

To test the capabilities of the gesture device, we developed a framework for cooperatively searching and tracking moving targets. UAV behaviors that would assist an operator while searching a region are defined and mapped to controller inputs. A depiction of the framework architecture is shown in Figure 1. From this figure, we see that an operator's gesture motion triggers data (accelerometer, gyroscope, and magnetometer) to be sent from the gesture device to a classifier module. The classifier identifies the gesture and maps it to a command which is sent, along with heading and position data, to each UAV. The UAVs then coordinate and fly their desired paths using the CMCTS algorithm.

The work presented in this paper builds upon and extends a previously published conference publication [6]. Specific additions to this work include (a) adding four additional gestures and corresponding high-level UAV behaviors that an operator may command, (b) validating the algorithms with hardware-in-the-loop experiments using an in-the-field operator and a virtual UAV, (c) incorporating a non-myopic control methodology that accounts for reward beyond the UAV's event horizon using artificial potential fields, and (d) showing through simulation experiments that the gesture commands are a viable method for directing groups of UAVs in search and track scenarios. These additions will be highlighted in greater detail in their relevant sections throughout the paper.

In the sections below we explain in detail the gesture device, UAV path planning algorithms, and experimental results. We begin by exploring previous relevant research in Sect. 2. In Sect. 3, we explain the CMCTS path planning algorithm. The gesture device hardware and mapped UAV behaviors are explained in Sect. 4. Simulation results are shown in Sect. 5 and conclusions are presented in Sect. 6.

2 Background

This paper builds on three fronts of previous research: (a) simplified UAV control interfaces, (b) autonomous algorithms for UAV path planning, and (c) improved cooperation between UAVs and their operators. Background

information for each of these areas will be discussed in the subsections below.

2.1 UAV-operator interface devices

The use of gestures was shown in [7, 8] to be an intuitive approach for giving commands to a UAV. In both of these studies, participants commanded the UAV to perform tasks to take-off, land, change altitude, move closer, etc., through hand gestures. The pilot observed the gestures and flew 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 implementations of gesture-based controllers have used various sensing instruments. In [9–12], cameras on board the UAV were used to detect human movements and respond to commands. On-board cameras simplify the control of a UAV and eliminate electronic communications between it and the operator. However, this limits the UAV's range since it needs to see the gestures.

Electromyography (EMG) sensors, such as the Myo armband, that detect electrical impulses in muscles have also been used to control robots via hand gestures in [13, 14]. In [13], a controller was implemented where the UAV can be commanded to take off, 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. In [14], a human-robot interface was developed with the Myo that allows the operator to pass objects back and forth to a robotic arm.

One final category of note uses sensors attached to the operator to measure and categorize motion. In [15], a three-axis accelerometer was 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 [16] to detect steps and to give commands such as "gain altitude," "lose altitude," and "take a picture." Beyond accelerometers, in [17] radio-frequency identification (RFID) tags were moved by the operator and interpreted as UAV commands.

A few devices implement some degree of control over a group of UAVs, typically by individually controlling UAVs or by directing a leader and having the other UAVs adjust around it. In [18], an operator used a pair of EMG sensors to give commands at both a group and individual level. This was done with hand gestures that scroll through a list of behaviors. While the system was shown to be usable, they also found that it was best suited for a pair of operators working in tandem.

This prior gesture-based research illustrates the potential for using simple interfaces in commanding UAVs to complete tasks. It indicates that gesture commands are an intuitive way for an operator to interface with a UAV. However, the majority of the gesture-based research has focused on directing single UAVs (individually or as a leader) that accomplish low-level tasks such as taking off or landing. In [6] we created a hand-worn gesture device capable of directing a single UAV. The device measures the operator's hand motions to translate them into UAV commands. In this work, we expand the capabilities of that device to provide additional gestures recognized by the device, a command acceptance trigger, and expanded the behaviors to work with UAV groups. Together these capabilities show that gesture commands can intuitively drive complex behavior in a fleet of UAVs tasked to search for and track targets.

2.2 Cooperative path planning

Cooperatively directing groups of UAVs can be challenging, especially when the environment contains obstacles, communication is unreliable, or vehicles are tasked with multiple objectives. Extensive research has been devoted to overcoming these and other cooperative path planning

Fig. 1 Architecture diagram showing the relationship between operator initiated commands originating in the gesture device to guidance commands for the UAVs, each of which has its own instance of the flight software

challenges. Overviews of this research are presented in [19–21] and include bio-inspired techniques [22–24], machine learning [25–27], and multi-objective optimization [28, 29].

Path planning techniques that pre-calculate UAV paths are undesirable since they can't adapt to dynamic environments. A proven technique that adjusts with learned information is a receding horizon controller (RHC). RHCs are a heuristic algorithm that replan frequently, mitigating the risk of executing suboptimal plans when operating in dynamic environments. However, RHC algorithms are notoriously computationally expensive since they evaluate the full decision space (out to an event horizon L) every time they replan.

Several approaches have been proposed to make RHCs computationally tractable. These include segmenting the timesteps [30], adapting the event horizons [31], separating the joint path planning of multiple vehicles into individual planners [32, 33] or clusters [34], incorporating Rollout Policies which greedily completes paths to their event horizon [35, 36], and using optimized tree search methods, such as MCTS [32, 33]. MCTS is a tree-search method that has found great success in the Artificial Intelligence community [37] and has subsequently been adapted to path planning techniques [32, 34, 38]. It provides an attractive alternative to Rollout Policies because it builds a tree to asymmetrically explore the available sample space and focuses on paths with high reward while ignoring less profitable subtrees. MCTS maintains the scalability that makes other mitigation techniques an attractive option, but also increases the breadth of search for large decision spaces.

MCTS has been used in prior research as an autonomous path planning algorithm for UAVs. Ref. [38] used MCTS as an RHC with a continuous control space and recursively reused the part of the tree associated with the previous policy. Ref. [34] created a belief grid to model coordinated UAVs response to disastrous events and found optimal policies using MCTS by factoring their tree over groups of spatially separated UAVs. Vehicle movement was modeled by jumping between adjacent grid cells. In [33] a decentralized MCTS algorithm was applied to the problem of optimizing routes of cooperating vehicles tasks while persistently revisiting a defined set of targets. In [32], a decentralized version of a block coordinate descent MCTS algorithm was presented to plan the action space of cooperating robots. Each robot accounted for the decisions of peer robots using a probability distribution over the joint-action space and the robot's joint-reward for a path was computed by sampling from the probability distribution to predict peer robot's plans.

Similar to [32, 33] the approach of this paper, first presented in [6], provides a decentralized MCTS algorithm

that cooperatively plans the paths of a group of UAVs. In this implementation, each UAV simulates the reward and decisions of the other vehicles using information (UAV positions and target estimates) that is communicated to the group. The algorithm requires a minimal number of messages to be shared among the UAVs, but assumes that all-to-all communication is available. This assumption is reasonable since the UAVs operate within close range of an operator. Furthermore, our MCTS variant, CMCTS, uniquely allows distant (macro) reward values (those beyond its planning horizon) to influence the paths chosen. This enables non-myopic control decisions that account for reward beyond the reachable space defined by the UAV's event horizon.

Expanding upon [6], and detailed in Sect. 3, the path planning algorithms developed in this paper combines artificial potential fields with CMCTS. This unique combination mitigates a shortcoming of RHC controllers in which they only consider reward values that may be reached within their event horizon. By integrating artificial potential fields with CMCTS, large reward values are still incorporated into the reward function, even when they lie outside the event horizon.

2.3 Human-UAV cooperation

In practically all UAV use cases, a balance exists between UAV autonomy and operator control. In systems with high UAV autonomy, the operator assigns broad mission goals while systems with high operator control allow for precise control of the UAVs. A high degree of autonomy may mean that the UAVs fail to perform exactly as the operator desires but if the operator has the sole power to make decisions their workload increases at an unsustainable rate when UAV groups grow in size.

The balance between operators and UAVs was categorized in [39] and [40] which examine various control architectures. In [39] the authors describe different architectures that have been seen in UAV-human systems such as how and when operators interact with UAVs, how UAVs determine trajectories, and how much autonomy UAVs are given. The author in [40] likewise compares implementations of human-robot systems and suggests that an ideal architecture would adjust its complexity depending on the level of control needed. One example is [41] where operators were tasked with controlling an unmanned vehicle (UV) group in a simulated Capture the Flag game. Using a graphical user interface (GUI), each operator could command the whole UV group, or any subgroup, using a combination of preconfigured "plays" and manual commands. The authors proposed a delegation style interface that allowed the operator to dynamically adjust the balance of control between them and the UV group. This

concept was further expanded in [42] with an expanded playlist and mission planning parameters. A similar idea is explored in [43] where the operator can decide what level of autonomy to grant the UAVs.

An additional consideration for human-UAV cooperation is the interface available to the operator. These can range from displays that simulate an airplane cockpit [44] to graphical interfaces operated with a mouse and keyboard [45] to smartphone-based controls [46]. Additionally, haptic devices are becoming increasingly attractive for some robotic interfaces since they allow high precision control [47]. While more detailed interfaces allow for greater information flow between the UAVs and operator, they also tend to restrict the mobility of the operator.

In general, research on UAV group control has not reduced the workload on the operator; rather it has tried to maximize the control an operator has without being overwhelmed [48–50]. This paper takes a different approach, where we attempt to allow the operator to work in parallel with a UAV rather than being a dedicated UAV operator. For example, a police officer searching for a fugitive may want to use a UAV group to search a large area. If the officer searches areas that would be occluded from the view of the UAVs (such as areas with heavy ground cover) while the UAVs fly over large exposed areas, the officer and UAV efforts complement each other and the combined effectiveness of the UAV/operator team is increased. In contrast, most current research assumes a dedicated operator would be controlling the UAV group from a remote location.

3 Path planning

This section describes the CMCTS path planning algorithm that is autonomously executed as an RHC within each UAV. CMCTS decides the low-level bank commands for each UAV based upon the reward structure described in this section. Changing the reward values offered to the UAVs creates different high-level behaviors for the group. A description of how the UAV reward values are modified for each gesture command is provided in Sect. 4.3

The UAVs plan their path using a modified MCTS algorithm. MCTS searches for optimal solutions to decision processes by randomly sampling the decision space and incorporating these samples into a tree structure. During MCTS, a tree is built asymmetrically by identifying the most profitable paths and expanding those paths' leaf nodes.

The explored nodes of a tree comprise the *tree policy*. A naïve tree policy would be to exhaustively search the tree, but this is generally computationally intractable, especially in real-time applications. Thus, a key question in MCTS is how to efficiently balance exploration and exploitation to choose optimal policies without searching in unprofitable areas of the tree.

The exploration–exploitation trade-off is solved using upper confidence bounds (UCBs) [51]. UCBs are a robust and efficient method to minimize regret, the difference in reward between the optimal strategy and the chosen strategy. When applied to MCTS, UCBs form the basis for the Upper Confidence Bound for Trees (UCT) algorithm [37]. Additionally, UCBs provide measures of finite-time performance, when many algorithms can only guarantee asymptotic optimality [51]. This allows UCT to be a statistical anytime algorithm, meaning that the policy improves with every iteration and can be stopped at any time. For real-time systems, a fixed number of model calls are allocated which can be completed in an allotted amount of time.

Although MCTS is typically used for single-agent systems, variants have been developed for multi-agent systems [32, 33, 52]. These methods expand MCTS by adding cooperative control with decentralized planning. In [6] we introduced a variant of MCTS called Coordinate Monte Carlo Tree Search (CMCTS). In this paper we expand our initial version of CMCTS to improve its robustness. During CMCTS, we follow the structure of MCTS by building a search tree for each UAV $T_j[n] = (N_j[n], E_j[n])$ at time step n , where W is the total number of UAVs and $j \in [1, W]$, N_j are the tree's nodes, and E_j the edges. Each node in the tree represents a specific UAV state that was reached by choosing one of the allowable control actions (for this application, a discrete set of roll commands).

For every node $k \in N_j[n]$ there is a unique sequence of node-edge pairs connecting the root to k . In CMCTS, we call this sequence of decisions a *control policy*. The length of each control policy is limited by the event horizon L . Since this control policy is unique for a given node, individual nodes have a one-to-one correspondence with control policies. The reward associated with a single UAV's control policy changes as the other UAV's control policies change, we track the average reward over all explored policies containing node k and denote it by $\bar{J}_{total}(k)$.

CMCTS adds nodes for each UAV in a cyclic fashion instead of constructing the entire tree for a UAV at each iteration. This allows each UAV to independently plan its path while considering its interaction with other UAVs. The CMCTS (and MCTS) algorithm consists of four steps:

1. **Selection:** a path is selected between the root node and a leaf node.
2. **Expansion:** new nodes are added to the leaf node.
3. **Simulation:** the path chosen during selection is simulated to the event horizon.
4. **Backpropagation:** the total reward associated with the chosen path is averaged into each node's reward value.

Each of these steps is iterated over for M cycles. Once completed, the control policy that provides the best average reward for the group of UAVs is selected. The first step in

this new control policy is executed and then, following RHC, the CMCTS tree-building process is repeated.

As an example of this process, consider a scenario with two UAVs. Each UAV will explore path options for itself while simulating the action of the second vehicle. To start, the first UAV will expand its tree by one node. It then simulates the second UAV, taking into account its own current best path, and expands the second UAV's tree by selecting a node. Then it will again expand its own tree through selecting a new node. It will calculate a combined reward by merging the currently selected path with the best found path of the second UAV. This cycle repeats until each tree is sufficiently sampled. CMCTS is run separately on each UAV. Since each UAV simulates its neighbors' paths, the only communication needed is to periodically update other UAVs of their current position, orientation, and target estimates.

Algorithm 1 Coordinate Monte Carlo Tree Search (CMCTS)

```

Initialize the set of actions  $B$  (bank angles)
Initialize  $W$  trees  $T_j[n] = (N_j[n], E_j[n])$ 
Initialize nodes  $k_j \leftarrow \text{root}(T_j)$ 
for  $i = 1$  to  $M$  samples do
    for  $j = 1$  to  $W$  UAVs do
        Selection:
         $k_j \leftarrow \text{root}(T_j)$ 
        while  $k_j$  is not a leaf do
             $k_j \leftarrow \arg \max_c UCT(k_j, c)$ 
             $q(k_j) \leftarrow q(k_j) + 1$ 
        end while
        Expansion:
        add children to  $k_j$  according to  $B$ 
        Simulation:
        simulate  $k_j$  to horizon  $L$  using default policy
        Calculate  $J_{total}$ 
        Backpropagation:
        for  $\ell = 1$  to  $W$  UAVs
             $m_\ell \leftarrow k_\ell$ 
             $\bar{J}_{total}(m_\ell) \leftarrow J_{total}$ 
            while  $m_\ell$  is not root( $T_\ell$ ) do
                 $m_\ell \leftarrow \text{parent}(m_\ell)$ 
                 $\bar{J}_{total}(m_\ell) \leftarrow \frac{(q(m_\ell)-1)\bar{J}_{total}(m_\ell) + J_{total}}{q(m_\ell)}$ 
            end while
        end for
    end for
end for

```

▷ iterate through a set number of samples
 ▷ iteratively build up each UAV's tree
 ▷ select nodes down the tree until a leaf is reached

▷ expand the children nodes of the chosen leaf

▷ simulate the UAV states to the event horizon

▷ calculate the total reward for all UAV control policies

▷ update the average local reward for nodes in the control policy

Each step of the CMCTS algorithm (selection, expansion, simulation, and backpropagation) is explained in greater detail in the following subsections. The overall process is also outlined in Algorithm 1.

3.1 Selection

In the selection step, children are *selected*, starting from the root node, to maximize the UCT criterion [53]

$$UCT(p, c) = \bar{J}_{total}(c) + \kappa_p \sqrt{\frac{\ln q(p)}{q(c)}}, \quad (1)$$

where p is the parent node, c are children nodes, q returns the number of times a node has been selected and κ_p is a tuning parameter to weight exploration. If $q(c) = 0$ we say $UCT = \infty$. The selected child is added to the control policy and we repeat this process recursively until a leaf node is reached. As mentioned, this function balances choosing nodes with known high values and nodes that have not been thoroughly explored.

3.2 Expansion

If the selected node is not a terminal node (i.e. the time horizon L has not been reached), then we *expand* it by adding children that represent the UAV's next state (i.e. positions they would reach under all of the UAV control decisions, which are the allowable discrete set of bank angles).

3.3 Simulation

During the *simulation* step, we propagate forward the UAV and target states until we meet the time horizon L . If we have not reached the time horizon in the expansion step, then we propagate the UAV state forward using a *default policy*. The default policy expands the path out to the time horizon by randomly choosing child nodes.

Once the path is simulated to the event horizon we use the current control policies of all peer UAVs to calculate a joint reward J_{total} . This is primarily based on the expected rewards gained by searching and tracking along the path described by the control policy. But it also includes an artificial potential field reward that draws the UAVs towards areas highlighted by the operator. The method for computing each of these expected reward values are described next.

The search reward incentivizes UAVs to visit areas that haven't been explored recently. The operating area is divided into a grid pattern and each cell is assigned a reward J_g . When the center of a grid cell falls within the

sensing radius of the UAV, the UAV receives the reward for that cell.

Grid cell reward values are updated as follows. If a grid cell $i \in [1, G]$, where G is the total number of grid cells, was seen during that timestep, the UAV is awarded a score equal to $J_{search,i}$ which is then set to 0. If the grid cell was not viewed, then the search reward value for cell i grows according to [54]

$$J_{search,i}[n+1] = J_g - (J_g - J_{search,i}[n])e^{-t\gamma}, \quad (2)$$

where $n \in [1, L]$ is the current timestep, $n+1$ is the next timestep, t is the length of time since the grid cell has been seen, and γ is a constant that describes the regrowth rate of the grid cell value. By increasing the reward over time, cells that have not been recently visited are prioritized.

The tracking reward incentivizes maneuvering the UAVs to positions where their measurements will reduce uncertainty of the target's state. The target's state is described by its position and velocity in a North-East coordinate frame. The state for target $v \in [1, V]$ at time step n is $x_v[n] = [X_v[n], Y_v[n], Z_v[n], \dot{X}_v[n], \dot{Y}_v[n], \dot{Z}_v[n]]^T$ with estimated state values denoted by $\hat{x}_v[n]$. The east and north positions are denoted by X and Y , respectively. The states $\dot{X}_v[n]$ and $\dot{Y}_v[n]$ are the derivatives with respect to time of the east and north positions. The target is assumed to reside on a flat earth at altitude $Z_v[n] = 0$, $\dot{Z}_v[n] = 0$. When target positions fall within the sensing radius of the UAV, the UAV obtains noisy range and bearing measurements that describe the target's planar location relative to the UAV.

For estimating the target states, we assume each target obeys a constant velocity model and update their estimated positions with an extended Kalman filter (EKF). With the EKF $\hat{x}_v[n]$ represents the mean estimated values at time step n and P_v is the target's error covariance. To calculate the reward gained by the UAV, the information matrix $I_v = P_v^{-1}$ for target $v \in [1, V]$ is calculated before and after the estimated target position is updated. The tracking reward is the total information gained for each target, where the gain is computed from the difference in the determinants of the information matrices as [54, 55]

$$J_{track,v}[n] = \ln |I_v[n]| - \ln |I_v[n-1]|. \quad (3)$$

An artificial potential field is used to attract UAVs to operator-defined areas. Let G_a be the a^{th} of A operator-defined search areas. The center of each area is assigned an attractive force U between each area G_a and UAV j , where $j \in [1, W]$, that scales with distance. Letting $z_j[n] = [N_j[n], E_j[n], Z_j[n]]^T$ be the position of UAV j at time n , the force U is defined by

$$U_{add,a,j}[n] = \begin{cases} \frac{\omega_a J_{search,G_a}}{(d_{a,j}[n])^{\lambda_a}} & z_j[n] \notin G_a \\ J_{search,G_a} & z_j[n] \in G_a \end{cases} \quad (4)$$

where ω_a determines the strength of the field and λ_a determines how the field changes with distance. The distance $d_{a,j}$ represents the distance between the center of the a^{th} area and the j^{th} UAV. The total reward associated with all the grid cells in the area G_a is J_{search,G_a} . Thus the attractive force for each area lessens with distance unless the UAV is inside of that area, in which case the attractive force is simply the value of the area. This attracts the UAVs to the operator-defined regions, but once they arrive it provides the freedom to explore the individual grids cells within the region.

A repulsive force is used to ensure that the UAVs avoid collisions and gathering redundant information. The repulsive force for the j^{th} UAV is calculated as

$$U_{rep,i,j}[n] = \begin{cases} \frac{\omega_r}{(d_{i,j}[n])^{\lambda_r}} & i \neq j \\ 0 & i = j, \end{cases} \quad (5)$$

where $d_{i,j}$ is the distance between the i^{th} and j^{th} UAVs and ω_r and λ_r are constant scalar values that determine the strength and scaling effect of the repulsive force.

In total, the reward gained from the artificial potential field for the j^{th} UAV is determined as

$$J_{potential,j}[n] = \sum_{a=1}^A U_{add,i,j}[n] - \sum_{i=1}^W U_{rep,i,j}[n]. \quad (6)$$

The total reward J_{total} for the joint control policy of the UAVs is calculated by combining Equations (2), (3), and (6) at each time step through the event horizon. This yields

$$J_{total} = \sum_{n=1}^L \sum_{j=1}^W \left(\sum_{i=1}^G J_{search,i}[n] + \sum_{v=1}^V J_{track,v}[n] + J_{potential,j}[n] \right). \quad (7)$$

The search reward is summed over each grid cell for every UAV and time step out to the event horizon $n \in [1, L]$. Likewise, the reward for sensing a target is summed for all targets, all vehicles, and all time steps in the control policy. Finally, the potential reward over the control policy is added.

The advantage of combining artificial potential field rewards with information-based rewards is that it allows each UAV to be directed at both a macro and micro scale. Typical receding horizon controllers are limited in their ability to plan by the choice of their event horizon. This can be problematic when large rewards lie outside of their planning space. This method still enables UAVs to effectively plan within their event horizon but adds an overall

bias towards moving to areas that have been specified by the operator.

3.4 Backpropagation

The final step in each iteration of CMCTS is to average or *backpropagate* the reward into all the nodes along all the UAV control policies. Each node k contains an average reward $\bar{J}_{total}(k)$, which represents the average reward from all previously explored policies that contain that node. During the backpropagation step, $\bar{J}_{total}(k)$ is updated by averaging in the total reward J_{total} to all nodes k in all the UAV's control policies.

This cycle of *selection*, *expansion*, *simulation*, and *backpropagation* is repeated until the stopping criteria is met.

4 Operator-UAV gesture interface

This section describes the gesture device and its capabilities. We will first describe the hardware (Sect. 4.1) and the ten motions it has been trained to recognize (Sect. 4.2). We then explain how the motions trigger high-level behaviors among the cooperating UAVs (Sect. 4.3).

4.1 Gesture device

A prototype gesture device was designed and built using a Raspberry Pi Zero board with an MPU-9250 9-axis inertial measurement unit (IMU) and a GP-20U7 GPS package and can be seen in Fig. 2. This device was enhanced from the one developed in [6] to include a push-button and an LED were added to provide a physical interface for the operator to use. The push-button provides a mechanism for the operator to indicate when gestures are being performed. The LED provides a feedback channel for the UAVs to alert the user of detected targets.

Information is gathered and sent to the Raspberry Pi Zero board from the IMU and GPS packages at different rates (Table 1). The operator's position and heading are updated using GPS at a rate of 1 Hz over a serial connection. When the button is pressed, accelerometer and gyroscope measurements are recorded at a rate of 100 Hz using an I2C connection to communicate the data to the Raspberry Pi. The direction the operator is pointing, with respect to north, is calculated using the integrated compass. The Raspberry Pi transmits the current position and heading of the operator as well as the classified gesture and compass heading to the UAVs over a TCP/IP socket connection.

Fig. 2 The components of the gesture device, in both a disassembled and assembled state

(a) The individual components of the disassembled gesture device. The bottom component has the IMU, LED, button, and interfaces for the Raspberry Pi (center) and GPS (top).

(b) The gesture device as assembled and worn. The GPS is located at the base of the thumb. The IMU is the small, uppermost board.

Table 1 The sensors and data gathered with the gesture device

Data	Sensor	Frequency (Hz)
Position	GPS receiver	1
Heading	GPS receiver	1
Acceleration (hand)	Accelerometer	100
Angular velocity (hand)	Gyroscope	100
Compass direction (hand)	Magnetometer	1

Position and heading are measured for the operator while the rest measure the movement and orientation of the gesture device

4.2 Gesture classifier

The gesture device was trained to identify 10 different hand and arm motions that are shown in Fig. 3 and described as follows: (a) a sweeping side to side motion in front of the operator with the palm facing down, (b) a repeated chopping or pointing motion made with the palm vertical, (c) a small counter-clockwise circle made with the palm down, (d) a large counter-clockwise circle made with the palm down, (e) a sustained point in a single direction with no motion, (f) a confirm gesture where the operator's fist

Fig. 3 Graphical depictions of the gesture commands used with the gesture device

makes an up-down “nodding” motion, (g) a decline gesture where the operator’s fist makes a left-right rolling motion, (h) an ‘X’ pattern traced in the air in front of the operator, (i) an up and down motion (like a fist pump) over the operator’s head, and (j) a repeated waving motion in front of the operator with the palm vertical. Gestures (d), (e), (f), and (g) represent gestures added beyond what was shown in [6]. All gestures were chosen because they are periodic and do not have determined start or stop positions. This simplifies recognizing commands since the operator does not need to coordinate the start of the gesture with the moment the gesture device starts recording data. Additionally, the gestures are unique and can express rough ideas such as direction, motion, and magnitude. The behavior triggered by each gesture will be discussed in the next section.

We used a logistic regression model to identify the gestures based on the frequency data calculated from the combined six axes of the accelerometer and gyroscope. The mean $\mu \in \mathbb{R}^6$ and standard deviation $\sigma \in \mathbb{R}^6$ were then found for each axis and saved for later use in both training and classifying the gestures.

For training, the raw data was split into segments with a rolling window so that each window was offset by five samples. This allows the periodic nature of the gestures to be leveraged and increases the training data set. Each individual segment $x \in \mathbb{R}^{6 \times 128}$ contained 128 measurements from each axis of the accelerometer and gyroscope. The previously calculated μ and σ for the full data set was used to normalize the raw axis measurements such that [56]

$$x'_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}, \quad (8)$$

where $i \in \{1, \dots, 6\}$ and $j \in \{1, \dots, 128\}$.

Classifying human motions by converting motion data to the frequency domain has been shown to be effective [57, 58] and we proceed in a similar manner. Each axis of the raw data segment was transformed to the frequency domain using a Fast Fourier Transform (FFT). This produced a (1×65) vector consisting of the magnitudes at each frequency. Additionally, the mean, μ_s and standard deviation σ_s of the raw data from each sensor axis was found for the segment. This provides orientation information of the gesture that would otherwise be lost when taking the FFT.

The FFT vectors for each sensor and axes were concatenated with the mean μ_s and standard deviation σ_s for each sensor and axis resulting in a single (1×402) vector.

For training the model, each gesture was repeated continuously by one of the authors for three 15-s segments for a total of 45 s of data per gesture. In order to limit introducing unintended artifacts into the data that could bias the logistic regression model, no two sequential 15-s training periods had identical gestures. A rolling window was used

to break up the measurements of each segment into individual gesture samples. This meant that each 15 second segment produced about 680 gesture samples that could be used to train the model. In total, over 20,000 equally divided samples of the 10 gestures were created and used to train the model. From these, 85% were used for training and the remaining 15% were used to validate the model.

When using the gesture device for actual operations, each gesture is identified using 148 measurements from each axis of the accelerometer and gyroscope. In order to limit the possibility of errors in the classification process, these 148 measurements are divided into 5 overlapping samples of 128 measurements that are staggered by 5 measurements. Each of these 5 samples is prepared identically to the training data by normalizing the measurements, taking the FFT, and concatenating those results with the sample means and standard deviations into a single vector. The logistic regression model classifies the state vector of each of the samples and the most commonly identified gesture is found.

4.3 Gesture commands

These simulations, gestures, and associated commands were developed around target searching and tracking missions where an operator is attempting to locate and identify various targets in a geographical area. We assume that the operator is in the area and has direct control of a group of UAVs. We further assume that the operator utilizes information that is not available to the UAVs. For example, the operator might notice a flash of light, hear vehicle noises, or obtain information from some outside source. This information will influence the operator in directing the UAVs.

The search commands described below are similar to those used in [6] but expanded for use with multiple UAVs so that any redundant effort is minimized. Also, the addition of the confirmation and rejection commands allows for more precise control of the UAV group without overwhelming the operator or reducing the group’s effectiveness.

The specific commands available to the operator, their associated gestures, and the high-level behavior this triggers are described below.

Wide-Heading Search (Sweeping Motion): The sweeping motion creates a shallow but wide area that incentivizes UAVs to search the region directly in front of the operator (see Figure 4 and 5). This is one of two related commands that increase the reward weight of grid cells located in front of the operator, as defined by the operator’s current heading θ_m . The heading is the operator’s direction of travel as calculated from GPS measurements. When the command is executed, a triangular area is defined that

Fig. 4 Example behavior of a UAV executing the heading based search command

Fig. 5 Parameters used to define the heading based search commands

begins at the operator's location, moving forward a distance d_s and extending θ_s degrees to either side of the centerline. The reward values of grid cells within this region are increased by an additional weight $J_{search,i}$ for the i^{th} grid cell. Giving a new weight of $(J_{search,i} + J_g)$ for each grid cell in the triangle. This incentivizes the UAVs to search the triangular area.

When a UAV observes a cell within the search region, the additional reward for that cell is set to zero. If the cell is not observed, the additional weight will gradually decrease so that the UAVs are not strongly influenced by stale instructions. The weight starts as $J_{search,i} = J_{search,max}$ and decreases according to the logistic curve [59]

$$\frac{dJ_{search,i}}{dt} = rJ_{search,i} \left(1 - \frac{J_{search,i}}{J_{search,max} + \delta} \right), \quad (9)$$

Fig. 6 Example behavior for the return to home and return and search commands. The return to home command results in the UAV loitering at a distance r_h about the operator while the return and search command has the UAV return to searching after the switching boundary r_s is reached

Fig. 7 Parameters used to define return to home (r_h) and return and search (r_s) commands

where $J_{search,max}$ is the additional initial reward value, r is the decay factor, and δ is a small value that moves the equation off the equilibrium position. The decay factor r is a negative constant and determines how fast the grid value decreases.

Deep-Heading Search (Chopping Motion): The chopping motion executes the second of the two heading-based commands by creating a deep forward search where the search area (similar to the wide area search, see Figs. 4, 5) is defined by a centerline distance away from the operator d_s and an angle θ_s that the area extends away from the centerline. The configurable search parameters are set so that

it creates a long but narrow corridor that expands out from the operator's position. As with the wide-heading search, the reward for grid cells in the area is initially increased by $J_{search,max}$ and decreases over time with the logistic curve Eq. (9).

Return-to-Home Command (Small Circling Motion): The small circling gesture commands the UAVs to return to home and circle the operator with a loiter radius specified by parameter r_ℓ (see Figs. 6, 7). This gesture can be used to quickly recall all UAVs to the operator's position in preparation to land or for extensive coverage of the area directly surrounding the operator.

In flying towards and orbiting the operator, the UAVs use an orbital path planner, rather than the CMCTS algorithm. The orbit path planner determines the necessary course heading to allow the UAV to maintain an orbit at radius r_ℓ around a point $c = [c_e, c_n]^T$ given UAV j 's current location $z_j = [X_j, Y_j, Z_j]^T$. The course heading is recalculated each simulation step to account for the changes in the aircraft position and changes in the loiter position (as the operator moves). If the UAV is located a large distance from the desired radius, the path planner first guides the UAV directly towards the point c then transitions it to an orbit.

The distance d_{cj} between the UAV and desired orbit position is found and then the commanded course heading is calculated as [60]

$$\chi^c = \phi + \lambda \left[\frac{\pi}{2} + \tan^{-1} \left(k_{orbit} \left(\frac{d_{cj} - r_\ell}{r_\ell} \right) \right) \right], \quad (10)$$

where

$$\phi = \arctan 2(X_j - c_e, Y_j - c_n) + 2\pi m,$$

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

The parameter λ defines the orbit direction and is allowed to be ± 1 where $\lambda = 1$ indicates a clockwise orbit while $\lambda = -1$ indicates 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. A smaller value of k_{orbit} indicates that the UAV will make a more gradual transition between flying towards the loiter location and orbiting the loiter location. After χ^c is calculated, a PID controller is used to command the necessary bank angle.

Return and Search (Large Circling Motion): The large circling gesture commands all the UAVs to move towards the operator's location as if executing the Return-to-Home command (i.e., in accordance with Eq. 10). However, once each UAV is within a switching radius r_s of the operator, it changes back to its default searching behavior (see Figs. 6, 7). This allows the operator to quickly refocus the UAV search closer to his current position.

Fig. 8 Example behavior of a UAV searching a circular area under the directed search command

Fig. 9 Parameters used to define the directed search command

Directed Search (Pointing Gesture): The pointing gesture identifies a specific location of interest for the UAVs to search. When the gesture is executed, the device determines the direction θ_m the operator is pointing, relative to magnetic north. That angle is used, with a preset distance d_m from the operator, to determine a specific location. All cells that lie within a distance r_m of that point are given an increased search reward $J_{search,max}$. This allows the operator to instruct the UAVs to an area by simply pointing to it (see Figs. 8, 9).

Confirm Target-Tracking (Fist Nodding Motion): The fist nodding motion is used to confirm that a UAV should continue tracking a target. The default UAV behavior is to search the area and then temporarily switch to tracking targets when one is discovered. Once enough information is gathered by the UAV (e.g., when it classifies the target) it resumes its searching behavior. However, in some cases,

such as when trying to maintain persistent surveillance, it is desirable for a UAV to continue tracking. This, and the following gesture commands, provide the ability to balance the searching and tracking behavior.

With the confirm gesture, the operator gives individual instructions to UAVs, when prompted. We assume there is some feedback from the UAVs that is accessible to the operator, such as a small screen showing images, data, or other telemetry. When a UAV encounters a target and obtains “enough” information, a request is sent to the operator. This may be an image or an estimated classification of the target. In our simulations, the target being tracked is circled in the color corresponding to the UAV who is tracking it. Once the operator gets the request, in this case via the LED flashing, they can instruct the UAV to continue tracking the target with the confirm gesture. Once confirmed, the UAV will solely focus on tracking that target and will not return to its search behavior.

Reject Target-Tracking (Fist Rolling Motion): The fist rolling motion is used to instruct the UAV to discontinue tracking a target. If rejected, the UAV will immediately return to the search behavior.

Confirm All (Up and Down Motion): The up and down motion confirms to the UAVs that the operator desires all known targets to be tracked. Since there will likely be a delay between a UAV requesting guidance and the operator confirming or rejecting the tracking mission, UAVs will continue to track their target while waiting. This can lead to cases where multiple UAVs are requesting tracking confirmation. Each vehicle’s request is queued and the operator may confirm or reject them sequentially. However, three gestures have also been implemented to allow for group approval and rejection. The first of these gestures is the confirm all command.

The confirm all command instructs the UAVs to track all known targets. This includes all targets that are currently queued and awaiting confirmation but also includes all targets that UAVs are tracking but haven’t gathered enough information on yet. If a UAV observes two or more targets when this command is issued, they will maximize the new information gained over all targets. Any UAVs not tracking a target when this command is issued will continue to search. In the event that these vehicles later find a target, they will track it, gather information, and then wait for confirmation from the operator; they will not immediately switch to a track only behavior.

Reject All (‘X’ Pattern): The reject all command, initiated with the ‘X’ pattern, instructs all UAVs to stop tracking their targets. This includes targets that are queued for operator approval as well as targets on which information is still being gathered.

Selective Decline (Waving Motion): The selective decline command, initiated with the waving motion, instructs the

UAVs to only track previously approved targets. All targets awaiting confirmation or not yet queued are not tracked.

5 Results

In this section, we discuss the results obtained by testing the gesture device’s classification accuracy and the effectiveness of the autonomy commands. Results are obtained through a combination of software simulations and hardware-in-the-loop experiments. The abilities of the gesture-autonomy architecture are verified incrementally by first testing in hardware the gesture classification accuracy. Second, by showing, with a multi-vehicle software simulation, the effectiveness of executing each autonomy command. Third, by verifying that operator-influenced autonomy will positively change the priorities of the UAV group with a software only simulation. And finally, providing a hardware-in-the-loop scenario that validates the ability of an operator to command the UAVs while maneuvering in an urban environment.

In the following subsections we will describe the multi-vehicle simulation environment (Sect. 5.1), testing of the CMCTS algorithm (Sect. 5.2), the gesture classification testing (Sect. 5.3), the behavior testing of the autonomy algorithms (Sect. 5.4), the simulations that validate the effectiveness of operator driven behaviors (Sect. 5.5), and the hardware-in-the-loop gesture-device field-test (Sect. 5.6).

5.1 Simulation environment

In each simulation, a (1×1) km² boundary was specified as the operating area and as shown in the thick dashed border. The area was further divided into (50×50) m² cells, which can be seen as the smaller squares with gray borders. Grid cells start with a reward value that correlates to the probability that a target exists inside the cell. When the UAV observes the center of a grid cell its reward value drops to zero. If a grid cell has not been seen by a UAV the reward gradually increases according to Eq. (2). Each UAV assumes that any object within its field of view is observed. Providing a sensor manager that schedules the field of regard is beyond the scope of this research.

Stationary targets are represented by black triangles in the simulation. The operator’s position and heading are shown as a black circle and arrow, both of which are measured by GPS and communicated to the UAVs. Finally, each UAV’s position, heading, and prior path are shown as the colored circle, arrow, and solid lines, respectively. The sensor footprint is indicated by the dashed circle surrounding them. It is assumed that UAVs fly at a constant, deconflicted altitude with collision avoidance further ensured through the CMCTS algorithm’s inherent bias to

Table 2 The configurable parameters used for each behavior

Behavior	Parameter	Value
Wide search	d_s (search depth)	200 m
	$J_{search,max}$ (additional grid Value)	40
	θ_s (search angle)	60°
	r (decay factor)	− 0.15
	δ (Offset)	0.01
deep search	d_s (search depth)	600 m
	$J_{search,max}$ (additional grid value)	40
	θ_s (search angle)	15°
	r (decay factor)	− 0.15
	δ (offset)	0.01
Return to home	r_ℓ (radius)	100 m
	λ (direction)	1 (clockwise)
Return and search	r_ℓ (radius)	100 m
	λ (Direction)	1 (clockwise)
	r_s (Switching radius)	115 m
Directed search	$J_{search,max}$ (additional grid value)	40
	d_m (Distance from operator)	250 m
	r_m (Search radius)	100 m
	r (Growth factor)	− 0.15
	δ (Offset)	0.01

separate the vehicles. Each UAV's communication range is large enough to encompass the operational space. Since we use a North-East-Down (NED) coordinate system and a flat Earth model, we assume that the down direction is constant for each UAV and from here on only concern ourselves with the North and East coordinates.

Table 2 provides the configurable UAV parameters (as described and illustrated in Sect. 4.3), separated by behavior, that were utilized for the simulations.

5.2 CMCTS testing

In this section the effectiveness of CMCTS is evaluated by comparing it to an exhaustive search and a Rollout Policy. The Rollout Policy provides a baseline for multi-vehicle cooperative applications. We used the method introduced by Ref. [36] where the paths of every UAV are searched exhaustively for two steps before greedily choosing the next best decision until reaching the event horizon. The exhaustive search provides the best possible outcome under this problem formulation. However, because of its computational cost we are only able to provide exhaustive search results for the most basic scenario.

The algorithms are compared using the unit-less, information reward, gained from each UAV at every time step. Each UAV counts information by summing the reward in observed grid cells (Eq. 2) and targets (Eq. 3). This *one-step*

Table 3 Average one-step information gain for three target scenario with two UAVs across randomized simulations

Algorithm	Event horizon, $L = 6$	Event horizon, $L = 8$
Exhaustive search	3.232	N/A
Rollout policy	2.557	2.581
CMCTS	2.564	2.594

The bold values highlight the algorithm that provides the highest reward level for each Monte Carlo simulations

reward is averaged across all the time steps in the simulation and again over all the randomized simulations. Results for a three target, two vehicles scenario show that, as expected, the exhaustive search performs the best with average one step reward of 3.232 for the $L = 6$ event horizon. The exhaustive search is prohibitively costly when searching beyond seven lookahead steps and reward values can not be computed. In this case, CMCTS performs better than the Rollout Policy as seen in Table 3.

Table 4 shows the results comparing Rollout Policy with CMCTS for different numbers of UAVs, $W \in \{3, 5, 7, 9\}$. In all cases, CMCTS outperformed the Rollout Policy in the average one step information measure. Note that the difference between the two lessens with an increase in the number of UAVs. In cases with saturating information (i.e. large numbers of targets or UAVs), we anticipate that the Rollout Policy will perform equally well as CMCTS since reward is more uniformly available and avoiding myopic control policies is less critical.

5.3 Gesture testing

The gesture device was tested to determine if it could accurately identify the 10 gestures. Each gesture was performed 50 times for a total of 500 trials. The order that the gestures were performed for the trials was randomly chosen. For each trial, a single set of 128 measurements on each axis were recorded as a sample. The sample was normalized, classified, and the result was recorded. The actual and predicted gestures were compiled into a confusion matrix (Fig. 10) to determine the classification accuracy.

Table 4 Average one-step information gain for five targets with $W \in \{3, 5, 7, 9\}$ UAVs across randomized simulations

Algorithm	$W = 3$	$W = 5$	$W = 7$	$W = 9$
Rollout policy	2.811	4.091	5.126	5.575
CMCTS	3.309	4.549	5.620	5.757

The bold values highlight the algorithm that provides the highest reward level for each Monte Carlo simulations

True Gesture	Sweep	50	0	0	0	0	0	0	0	0	
	Chop	0	50	0	0	0	0	0	0	0	
	Small Circle	1	0	40	4	0	0	1	1	3	
	Large Circle	2	0	1	46	0	0	1	0	0	
	Wave	0	0	0	0	50	0	0	0	0	
	X Pattern	0	0	0	0	0	46	1	0	3	
	Up Down	0	0	0	0	1	0	49	0	0	
	Point	0	0	0	0	0	0	1	49	0	
	Confirm	0	1	0	0	0	0	0	49	0	
	Decline	0	0	0	0	0	0	0	2	48	
			Sweep	Chop	Small Circle	Large Circle	Wave	X Pattern	Up Down	Point	Confirm
Classified Gesture											

Fig. 10 Confusion matrix showing the actual and identified gestures. Each gesture was performed 50 times. All gestures, other than the small circle, had over 90% accuracy

As seen in the figure, the gesture device correctly identified the operator's commands with high accuracy. The one exception is the Small Circle which was only correctly identified 40 of 50 times, for a total accuracy of 80%. This gesture was misinterpreted as the Large Circle 8% of the time, which is understandable given that both gestures have the same basic pattern. The remaining 12% of the time, the gesture was misidentified as the result of similarities between the circling motion and other gestures. For example, if the circle was too small then the circling motion identified as the fist nodding motion. Likewise, if the motion overemphasized side to side movement it was interpreted as a sweep movement.

To mitigate false identifications, five predictions are performed for each command. Provided there are no more than two incorrect predictions, the gesture device will correctly classify the command. With the five predictions, the Small Circle command is correctly identified 94% of the time. All other commands are correctly identified with even higher accuracy. These results show that the gesture device effectively identifies the gestures, although a more advanced model could be used to better distinguish between similar patterns. This is left as an item of future work.

5.4 Behavior testing

Six search behaviors were tested individually in software simulations to demonstrate that the commands matched the desired behaviors. All simulations were run on the same map for a total simulation time of 30 s. The starting positions of the target (black triangle), UAVs (colored circles), and operator (black circle) were the same across each simulation

(see Fig. 11). A single stationary target was active for the full simulation and was observable by the red UAV at the beginning of the simulation. In each simulation, the UAVs searched to find targets. Once a UAV had seen a target, it would track it while waiting for the operator to Confirm or Deny the target. In all these simulations, the target started within the field of view of the red UAV. This setup was intentional and provided a guaranteed way to test the UAV target tracking behavior. For most of the tests, these simulations were used to test the searching behavior (no targets required) and confirm that operator-commanded areas were explored without overlapping their sensor fields of view.

At the beginning of each simulation, the gesture device was used to command a single behavior. The gesture device was used again to send the Decline command when enough target information was gathered, but otherwise the UAVs received no further instructions from the operator. The only exception to this was the final simulation where no task was initially given but the Confirm gesture was used to command the UAV to continue target tracking. Even though some behaviors involve specific areas for the UAVs to search, they are simultaneously tasked with searching the entire bounded map area. Therefore, it is common (and desired) to have one or more of the UAVs continue searching the larger region (seemingly ignoring operator commands) when they know other UAVs will be more effective executing the search command.

5.4.1 Wide search

The first behavior was the wide area search. As seen in Fig. 11a, a wide, triangular area was specified, relative to the operator's position, and additional weight was given to areas within that boundary. Two UAVs can be seen cooperatively exploring that area. The magenta UAV flies a path similar to a lawn mower pattern. This pattern was not pre-planned but naturally arises from the CMCTS path planner. The blue UAV flies along the boundary and observes parts of the area although it never actually enters the region. Between these two UAVs, all the grid cells that lie within the indicated area were able to be searched. The third UAV observes the target and then, once given the Decline command, migrates toward the indicated area. Once it is clear that the area is fully covered by the other UAVs, the red UAV moves away to search other areas.

5.4.2 Deep search

The second behavior, a deep search, can be seen in Fig. 11b. This time, the UAV tracks show that all three UAVs move towards the triangular search area. The magenta and blue UAVs begin immediately while the red UAV tracks the target until

Fig. 11 Images showing the path of the UAVs as they executed the various behaviors in simulation

the Decline command is given and then proceeds towards the area. Despite all three UAVs converging on the area, they minimize duplicated coverage. The magenta and red UAVs cover most of the area while the blue UAV searches a corner.

The four dark gray grid cells show locations that are unsearched at 30 seconds. While the magenta UAV was moving to search three of the remaining areas, the final area was not searched. This is a consequence of the CMCTS algorithm deciding that the objective function would be maximized by searching the upper area of the triangle.

5.4.3 Return to home

The next behavior tested was the return-to-home command. As seen in Fig. 11c, all the vehicles leave their other tasks and orbit the operator. For this behavior, the UAVs have switched from path planning using CMCTS to using the circular path planner. As a result, no attempt is made to equally space themselves or otherwise cooperatively move. Also, while the majority of the behaviors only influence UAV behavior (by increasing the reward

of specific areas), the return-to-home command forces the UAVs to loiter.

5.4.4 Return and search

The return-and-search behavior (Fig. 11d) instructs the UAVs to return to orbit the operator's position and then switch to a searching behavior. This command overrides any current tracking behavior so even though the target is seen by the red UAV, it is ignored as the UAV returns to the operator's location. Since the condition for switching to a search behavior is the distance from each individual UAV to the operator, the UAVs do not switch at a uniform time or location.

5.4.5 Area search

In the area search behavior (Fig. 11e), a circular search area was identified. The blue UAV coming from the north cuts through and observes the entire area. The other two UAVs are initially drawn towards the area but move away once it is clear that the area has been fully searched.

5.4.6 Target tracking

The target tracking behavior can be seen in Fig. 11f. Here, the red UAV gathers information on the target and then asks the operator if it should continue observing the target. Upon receiving the Confirm command, the UAV continues to circle the target. In the absence of any other inputs, the other two UAVs search the map.

5.5 Simulated testing

The purpose of the simulation tests was to quantify how much the gesture commands can positively influencing the UAV's priorities. We compare the UAV's ability to find targets when directed to them by an operator with that of an undirected search, where the UAVs' search was not influenced by any operator commands. Using software-only Monte Carlo simulations, we measured the UAVs' ability to find the targets. The effectiveness was defined by how many targets were seen and the time that elapsed before the target was in the UAVs sensing radius. While software simulations were performed in [6], those did not quantify the improvement over a non-directed UAV group.

In these tests, 25 stationary targets were randomly dispersed throughout the bounded area. Upon the start of each simulation, and every 30-s afterwards, a subset of the 25 targets become active and remained so for 60-s. The only exception was that no targets were activated during the final 30 s of the simulation. The number of targets that

became active in each cycle was random but constrained by the requirement that at least one target needed to become active on each 30-s interval. Each target was only active for a single 60-s cycle. Outside of their active periods, each target was undetectable to the UAVs and the operator.

A total of 200 simulations were run where half were undirected and the other half were directed by a simulated operator. Each UAV planned paths using CMCTS with a time horizon of 10 s and explored 100 nodes of the decision-tree. In the undirected simulations, the only guidance available to the UAVs were the grid cell values. For the directed simulations, the simulated operator gave commands, in addition to the grid cell values, to guide the UAVs to areas with targets in them.

The simulated operator was purely a software construct and was assumed to always know the exact location of any targets in the area. It was not limited by visible range, reaction speed, the target's relative position, or other sources of error. Additionally, it was assumed that the gesture device correctly classified all gestures and the operator precisely indicated the desired direction of each gesture. While these assumptions are unrealistic, the purpose of this test was to verify that the UAV behaviors are a viable means of directing the UAVs, not to test an operator's ability to use the gesture controller.

The operator, located in the center of the area, determined the bearing and distance of each active target. The

Fig. 12 Visualization of simulated operator testing. The light gray circles represent the divisions between the distances, triangles represents targets, and the simulated operator's location is the dark circle

distance from the operator to the target was classified as close, medium, or far. A visualization of the setup can be seen in Figure 12. The black dot in the center is the simulated operator's position, the numbered triangles represent targets, and the gray circles show the divisions between the distance classifications. Thus, Target 1 would be close, Target 2 would be medium, and Target 3 would be far.

If the target was determined to be close, the operator commanded a return-and-search behavior. If the target was a medium distance away, the operator used the bearing to the target to command a directed search. Finally, if the target was far from the operator's position, the operator used the deep heading-based search command, centered on the bearing to the target. Example commands can be seen in Figure 13. A return-and-search command is used on Target 1, a directed search is used on Target 2, and a deep heading-based search is used on Target 3. If all three commands were executed in short order the UAVs would be expected to return to the operator's position and then begin searching with additional weight given to the areas highlighted by the operator's commands.

For comparison purposes, the same simulations were run with both the default behavior (the UAVs are guided only by the CMCTS algorithm) and the gesture control behavior (the UAVs are guided by both the CMCTS algorithm and the gesture commands). For each pair of simulations, the starting locations of the UAVs, the target locations, and the target appearance times were identical. We measured how many targets the UAVs found and how long it took to find them. The number of found targets was the

Fig. 13 Visualization of simulated operator commands

Fig. 14 Histogram showing the elapsed time between a target appearing until it was detected by a UAV. Any targets that were not seen over the course of the simulation were excluded from this analysis

number of targets that were seen by a vehicle during the simulation while the target was active. The time to find the targets was determined by the difference between the time the target became active and when it was first seen.

As expected, the directed UAVs found more targets faster. On average, the directed UAV group found 22.22 targets in each simulation. Each of these targets took an average of 20.85 s to find. By contrast, the undirected UAV group found 14.73 targets, requiring 25.73 s each, on average. The distribution of time need to find each target and

Fig. 15 Histogram showing the number of targets seen over all simulations. Each simulation had a possibility of seeing 25 targets

the number of targets seen in each simulation are shown in Figs. 14 and 15, respectively.

In Fig. 14, there is a higher than expected number of targets seen between 0 and 5 s. This is partly due to targets appearing in areas within the UAVs' sensing radius. There were a total of 123 (undirected) and 139 (directed) instances of targets that required 0 seconds to be seen. The similarity of these numbers indicate that these instances are independent of the UAVs being directed or not. When the directed and undirected populations were tested with a two-sided *T* test, they differ significantly in how long it takes to find targets with $p < 0.01$.

In Fig. 15, it is seen that there is a clear division between the number of targets each method observed. Only one simulation with the undirected UAVs saw 20 or more targets while only two of the directed simulations saw fewer than that. The undirected UAVs spread themselves throughout the simulation area and attempted to visit grid cells based on when they were last seen. Contrastingly, the UAVs with the simulated operator were primarily attracted to the areas designated by the operator. When the directed UAVs failed to see a target, it was typically because the number of targets outnumbered the UAVs and they couldn't search all the operator specified areas during the 60 seconds that the targets were active. Despite this limitation, the controlled UAV group was able to see significantly more of the targets with the populations being significantly different at $p < 0.01$ when compared using a two-sided *T* test.

The results of this test show that the UAV behaviors are a viable means for directing a group of UAVs to search for targets. Additionally, though the simulated operator is unrealistic in real-world scenarios, this data provides an indication of the improved performance possible when including an operator in directing the group of UAVs.

5.6 Outdoor hardware testing

A hardware-in-the-loop test was used to verify the effectiveness of the system while the operator was moving in an outdoor environment. Five stationary targets and 3 UAVs were randomly simulated within a $1 \times 1 \text{ km}^2$ area. The UAVs and targets were software simulated, but the operator moved in the real world and the gesture device tracked his position, heading, and any executed gestures. The classified gestures were transmitted to the simulation using WiFi. A simple visualization was available to the operator showing his location, the simulated UAV and target locations, and the weightings for each search area. For the test, the terrain was ignored by the UAVs (i.e. they did not maneuver to avoid buildings or trees). However,

the accuracy of the GPS receiver worn by the operator was impacted by environmental variables.

Similar to the simulated tests, the operator could identify the simulated targets by watching them appear/disappear on a laptop. He used the gesture commands to direct the UAVs to the target locations. A series of frames showing the progression of the simulation can be seen in Fig. 16. Each frame shows a time period of the simulation and the movement of the UAVs (colored circles) and the operator (black circle). The targets (triangles) were active and stationary for the full simulation. The dashed lines show areas that the operator directed to be searched and all grid cells that fall within that area are initially shown in gray to indicate their elevated reward status. Once the center of the grid cell falls within the sensing radius of a UAV (dashed circle), the cell turns white to indicate it has been observed. Any cell not observed remains gray although over time its value decays and ceases to attract UAVs. The UAVs were also rewarded for searching the unbounded areas, but the figures don't explicitly show which grid cells were seen. However, a rough approximation can be made based on the tracks of the UAV positions.

At the beginning of the simulation (Fig. 16a), the vehicles began searching their surrounding areas while waiting for directions from the operator. Ten seconds into the simulation, the operator uses the directed search to highlight the northern area that has two targets in close proximity to each other and the UAVs turn to begin moving towards that area (Fig. 16b). At 16 seconds, the operator used the heading-based search to indicate another area with a target in it (Fig. 16c). Both the blue and magenta UAVs find targets but the operator instructs them to discontinue tracking using the decline-all command at 28 s. While the UAVs continue to search in the northern half of the area, the operator moves south.

At 34 seconds, the operator uses a heading-based search to draw the UAVs towards the south area (Fig. 16d). The magenta and red UAVs choose paths with high reward while moving towards the new area. The blue UAV has found another target and circles around to keep it in its sensing radius. The operator then indicates two more areas with targets using a heading-based search and an area search at 52 and 77 s, respectively (Fig. 16e) that the magenta UAV searches out. Meanwhile, the operator confirms at 57 s that the blue UAV should continue to track the target and then at 63 s that the magenta UAV should not track the target it had found. At 81 s, the operator then ends the blue UAVs tracking with the drop-all command. The red UAV continues to search out the designated area.

Finally, 100 s into the simulation, the operator gives the return-to-home command and all the UAVs cease their other activities and return to orbit the operator (Fig. 16f) until the end of the simulation at 142 s. This includes the

Fig. 16 Images showing the UAVs (colored circles), operator (black circle), and targets (triangles) as well as their movement for different time intervals during the outdoor experiment. The dotted lines

indicate the operator specified search areas and gray cells represent grids within those areas that have not been searched

magenta UAV that had seen a target and was awaiting instructions about continued tracking.

The outdoor test showed that an operator who was moving around his environment could use simple gesture commands to influence a group of UAVs. The UAVs followed the commands and maintained good separation throughout the test. One issue encountered was caused by the unreliability of the GPS. This was evident during the

heading-based search commands, where a noisy operator position measurement could cause an inaccurate heading and alter the direction of the heading command. However, the inaccuracies in the operator's position were small enough that the indicated area was not noticeably different from what was expected. Since the directed area search relied on compass measurements, it was not affected by this problem.

6 Conclusion

We have shown effective control of a fleet of UAVs using a gesture device that requires minimal operator interaction. The gesture device classifies the operator's actions and maps them to high-level behavior commands for a fleet of UAVs. The UAVs autonomously execute the commands using a CMCTS algorithm that coordinates the paths of all the UAVs to maximize the reward gained by searching for and tracking targets. It was shown through simulation tests and hardware-in-the-loop experiments that the gesture device provided an effective interface that allowed an operator to command the fleet of UAVs.

Future work will include further testing of the device and its operational use, including a full outdoor flight test with multiple UAVs, evaluation of the operator's effectiveness in imperfect environments, and validation of the gesture effectiveness across a large group of operators.

Funding This work was funded by a Brigham Young University 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.

Availability of data and material This the work was partially funded by the C-UAS I/UCRC and as such the data resulting from these experiments are made available to all participants in that center.

Code availability This work was partially funded by the C-UAS I/UCRC and as such the code is made freely available to all participants of that center.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that the authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethics approval This study did not collect private, identifiable information and the data collected was not about human subjects. As such, the Institutional Review Board (IRB) determined that this scholarly activity does not meet the regulatory definition of human subjects research (IRB2021-119).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright

holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Murphy RR, Burke JK (2010) The safe human-robot ratio. In: Barnes M, Jentsch F (eds) Human-robot interactions in future military operations, Chap 3. Ashgate Publishing, Surrey (UK), pp 31–49
2. Hanford S, Long L, Horn J (2005) A small semi-autonomous rotary-wing unmanned air vehicle (UAV). In: Infotech@Aerospace, (Reston, Virginia). American Institute of Aeronautics and Astronautics, pp 1–10
3. Sampedro C, Rodriguez-Ramos A, Bavle H, Carrio A, de la Puente P, Campoy P (2019) A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *J Intell Robot Syst Theory Appl* 95(2):601–627
4. Arrabito GR, Ho G, Lambert A, Rutley M, Keillor J, Chiu A, Au H, Hou M (2010) Human factors issues for controlling uninhabited aerial vehicles: Preliminary findings in support of the canadian forces joint unmanned aerial vehicle surveillance target acquisition system project (incidence des facteurs humains sur le pilotage des vehicules aeriens telepilotes: Constatations preliminaires a l'appui du projet de systeme interarmees d'acquisition d'objectif au moyen de vehicules aeriens telepilotes de surveillance des forces canadiennes), tech. rep., Defence Research and Development Canada
5. Jäger S, Schöbel A (2020) The blockwise coordinate descent method for integer programs. *Math Methods Oper Res* 91(2):357–381
6. Akagi J, Moon B, Chen X, Peterson CK (2019) Gesture commands for controlling high-level UAV behavior. In: 2019 international conference on unmanned aircraft systems, ICUAS 2019, pp 1023–1030
7. Cauchard JR, E JL, Zhai KY, Landay JA (2015) Drone & me. In: Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing—UbiComp '15. ACM Press, New York, pp 361–365
8. Ng WS, Sharlin E (2011) Collocated interaction with flying robots. In: Proceedings—IEEE international workshop on robot and human interactive communication, pp 143–149
9. Naji J, Giusti A, Di Caro GA, Gambardella LM (2014) Human control of UAVs using face pose estimates and hand gestures. In: 2014 9th ACM/IEEE international conference on human-robot interaction (HRI). ACM, pp 252–253
10. Naji J, Giusti A, Gambardella LM, Di Caro GA (2014) Human-swarm interaction using spatial gestures. In: 2014 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 3834–3841
11. Monajjemi VM, Wawerla J, Vaughan R, Mori G (2013) HRI in the sky: creating and commanding teams of UAVs with a vision-mediated gestural interface. In: IEEE international conference on intelligent robots and systems, pp 617–623
12. Jiao R, Wang Z, Chu R, Dong M, Rong Y, Chou W (2020) An intuitive end-to-end human-UAV interaction system for field exploration. *Front Neurobot* 13:117
13. Li Z, Chen Y, Tan W (2017) Dronemyo: proactive control of unmanned aerial vehicle based on wearable devices. In: Chen Y, Christie M, Tan W (eds) Smart graphics. Springer, Cham, pp 211–214

14. Wang W, Li R, Diekel ZM, Chen Y, Zhang Z, Jia Y (2019) Controlling object hand-over in human–robot collaboration via natural wearable sensing. *IEEE Trans Hum Mach Syst* 49(1):59–71
15. Sandru LA, Crainic MF, Savu D, Moldovan C, Dolga V, Preitl S (2016) Automatic control of a quadcopter, ar. drone, using a smart glove. In: *Proceedings of the 4th international conference on control, mechatronics and automation*. ACM, pp 92–98
16. Lu Y, Han F, Xie L, Yin Y, Shi C, Lu S (2017) I am the UAV: a wearable approach for manipulation of unmanned aerial vehicle. In: *2017 IEEE international conference on smart computing, SMARTCOMP 2017*, pp 1–3
17. Zhang J, Yu Z, Wang X, Lyu Y, Mao S, Periaswamy SC, Patton J, Wang X (2018) RFHUI: an intuitive and easy-to-operate human-UAV interaction system for controlling a UAV in a 3d space. In: *Proceedings of the 15th EAI international conference on mobile and ubiquitous systems: computing, networking and services*, pp 69–76
18. Nagavalli S, Chandarana M, Sycara K, Lewis M (2017) Multi-operator gesture control of robotic swarms using wearable devices. In: *Proceedings of the tenth international conference on advances in computer–human interactions*, pp 25–33
19. Chung SJ, Paranjape AA, Dames P, Shen S, Kumar V (2018) A survey on aerial swarm robotics. *IEEE Trans Rob* 34:837–855
20. Aggarwal S, Kumar N (2020) Path planning techniques for unmanned aerial vehicles: a review, solutions, and challenges. *Comput Commun* 149:270–299
21. Zhang H, Xin B, Dou L hua, Chen J, Hirota K (2020) A review of cooperative path planning of an unmanned aerial vehicle group. In: *Frontiers of information technology and electronic engineering*, vol 21, no. 12, pp 1671–1694
22. Zheng C, Ding M, Zhou C, Li L (2004) Coevolving and cooperating path planner for multiple unmanned air vehicles. *Eng Appl Artif Intell* 17(8):887–896
23. Wu Y, Low KH, Lv C (2020) Cooperative path planning for heterogeneous unmanned vehicles in a search-and-track mission aiming at an underwater target. *IEEE Trans Veh Technol* 69(6):6782–6787
24. Biswas S, Anavatti SG, Garratt MA (2019) A time-efficient cooperative path planning model combined with task assignment for multi-agent systems. *Robotics* 8(2):1–16
25. Guo Y, Pan Q, Sun Q, Zhao C, Wang D, Feng M (2019) Cooperative game-based multi-agent path planning with obstacle avoidance. In *2019 IEEE 28th international symposium on industrial electronics (ISIE)*. IEEE, pp 1385–1390
26. de Witt CS, Peng B, Kamienny PA, Torr P, Böhmer W, Whiteson S (2020) Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint [arXiv:2003.06709](https://arxiv.org/abs/2003.06709)*
27. Lei L, Shen G, Zhang L, Li Z (2021) Toward intelligent cooperation of UAV swarm: when machine learning meets digital twin. *IEEE Net* 35(1):386–392
28. Qiu H, Duan H (2020) A multi-objective pigeon-inspired optimization approach to UAV distributed flocking among obstacles. *Inf Sci* 509:515–529
29. Ramirez-Atencia C, Camacho D (2019) Constrained multi-objective optimization for multi-UAV planning. *J Ambient Intell Humaniz Comput* 10(6):2467–2484
30. Bledt G, Kim S (2019) Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In: *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp 6316–6323
31. Peterson CK (2017) Dynamic grouping of cooperating vehicles using a receding horizon controller for ground target search and track missions. In: *2017 IEEE conference on control technology and applications (CCTA)*, pp 1855–1860
32. Best G, Cliff OM, Patten T, Mettu RR, Fitch R (2019) Dec-MCTS: decentralized planning for multi-robot active perception. *Int J Robot Res* 38(2–3):316–337
33. Peterson CK, Casbeer DW, Manyam SG, Rasmussen S (2020) Persistent intelligence, surveillance, and reconnaissance using multiple autonomous vehicles with asynchronous route updates. *IEEE Robot Autom Lett* 5(4):5550–5557
34. Baker CAB, Ramchurn S, Teacy WTL, Jennings NR (2016) Factored Monte-Carlo tree search for coordinating UAVs in disaster response. In: *In ICAPS proceedings of the 4th workshop on distributed and multi-agent planning (DMAP-2016)*, (London), pp 6–15
35. Gommans TM, Theunisse TA, Antunes DJ, Heemels WP (2017) Resource-aware MPC for constrained linear systems: two rollout approaches. *J Process Control* 51:68–83
36. Tian X, Bar-Shalom Y, Pattipati KR (2010) Multi-step look-ahead policy for autonomous cooperative surveillance by UAVs in hostile environments. *J Adv Inf Fusion* 5(1):3–17
37. Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. *IEEE Trans Comput Intell AI in games* 4(1):1–43
38. Schneider M (2016) Receding-horizon planning using recursive Monte Carlo tree search with sparse action sampling for continuous state and action spaces. In: *2016 American control conference (ACC)*, pp 5401–5406, July
39. Skorobogatov G, Barrado C, Salami E (2020) Multiple UAV systems: a survey. *Unmanned Syst* 8(2):149–169
40. Lewis M (2013) Human interaction with multiple remote robots. *Rev Hum Factors Ergon* 9(1):131–174
41. Parasuraman R, Galster S, Squire P, Furukawa H, Miller C (2005) A flexible delegation-type interface enhances system performance in human supervision of multiple robots: empirical studies with roboflag. *IEEE Trans Syst Man Cybern Part A Syst Hum* 35(4):481–493
42. Calhoun GL, Ruff HA, Behymer KJ, Frost EM (2018) Human-autonomy teaming interface design considerations for multi-unmanned vehicle control. *Theor Issues Ergon Sci* 19(3):321–352
43. Scherer J, Yahyanejad S, Hayat S, Yanmaz E, Andre T, Khan A, Vukadinovic V, Bettstetter C, Hellwagner H, Rinner B (2015) An autonomous multi-UAV system for search and rescue. In: *Proceedings of the first workshop on micro aerial vehicle networks, systems, and applications for civilian use*, pp 33–38
44. Eheim C, Dixon C, Argrow B, Palo S (2002) Tornadochaser: A remotely-piloted UAV for in situ meteorological measurements. In: *1st UAV conference*. American Institute of Aeronautics and Astronautics, Reston, p 3479
45. Perez D, Maza I, Caballero F, Scarlatti D, Casado E, Ollero A (2013) A ground control station for a multi-uav surveillance system: Design and validation in field experiments. *Journal of Intelligent and Robotic Systems: Theory and Applications* 69(1):119–130
46. Cummings M, Jackson K, Quimby P, Pitman D (2012) Development and testing of a quad rotor smartphone control system for novice users. *Int J Micro Air Veh* 4(3):165–178
47. Zhu M, Sun Z, Zhang Z, Shi Q, He T, Liu H, Chen T, Lee C (2020) Haptic-feedback smart glove as a creative human-machine interface (HMI) for virtual/augmented reality applications. *Sci Adv* 6(19):1–14
48. Chen JY, Barnes MJ (2014) Human—agent teaming for multirobot control: a review of human factors issues. *IEEE Trans Hum Mach Syst* 44(1):13–29
49. Crandall JW, Goodrich MA, Olsen DR, Nielsen CW (2005) Validating human–robot interaction schemes in multitasking environments. *IEEE Trans Syst Man Cybern Part A Syst Hum* 35(4):438–449
50. Goodrich MA, McLain TW, Anderson JD, Sun J, Crandall JW (2007) Managing autonomy in robot teams: Observations from four experiments. In: *HRI 2007—proceedings of the 2007 ACM/IEEE*

- conference on human-robot interaction—robot as team member, pp 25–32
51. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Mach Learn* 47(2):235–256
52. Kurzer K, Zhou C, Zollner JM (2018) Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search. In: *2018 IEEE intelligent vehicles symposium (IV)*, pp 529–536
53. Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. In: *European conference on machine learning*. Springer, pp 282–293
54. Newman AJ, Martin SR, DeSena JT, Clarke JC, McDerment JW, Preissler WO, Peterson CK (2009) Receding horizon controller using particle swarm optimization for closed-loop ground target surveillance and tracking. In: Kadar I (ed) *Signal processing, sensor fusion, and target recognition XVIII*, vol 7336. International Society for Optics and Photonics, SPIE, pp 285–296
55. Sinha A, Kirubarajan T, Bar-Shalom Y (2005) Autonomous surveillance by multiple cooperative UAVs. In: Drummond OE (ed) *Signal and data processing of small targets 2005*, vol 5913. International Society for Optics and Photonics, SPIE, pp 616–627
56. Jayalakshmi T, A S (2011) Statistical normalization and back propagation for classification. *Int J Comput Theory Eng (IJCTE)* 3(1):89–93
57. Wu W, Dasgupta S, Ramirez EE, Peterson C, Norman GJ (2012) Classification accuracies of physical activities using smartphone motion sensors. *J Med Internet Res* 14:e130
58. Guo Y, Xu G, Tsuji S (2002) Understanding human motion patterns. In: *Proceedings of the 12th IAPR international conference on pattern recognition (Cat. No.94CH3440-5)*, vol 2. IEEE Computer Society Press, pp 325–329
59. Pearl R, Reed LJ (1920) On the rate of growth of the population of the united states since 1790 and its mathematical representation. *Proc Natl Acad Sci* 6(6):275–288
60. Beard RW, McLain TW (2012) *Small unmanned aircraft: theory and practice*. Princeton University Press, Princeton
- Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.