HYBRID CONTROL FOR UAV-ASSISTED SEARCH AND RESCUE

Allison D. Ryan
Dept. of Mechanical Engineering
University of California, Berkeley
Berkeley, California 94720
allison@vehicle.me.berkeley.edu

David L. Nguyen
Dept. of Mechanical Engineering
University of California, Berkeley
Berkeley, California 94720
davidln@berkeley.edu

J. Karl Hedrick
Dept. of Mechanical Engineering
University of California, Berkeley
Berkeley, California, 94720
khedrick@newton.berkeley.edu

ABSTRACT
We develop a decentralized hybrid controller for fixed-wing UAVs assisting a manned helicopter in a United States Coast Guard search and rescue mission. The UAVs assist the manned helicopter by providing an expanded sensor footprint using onboard cameras. We consider two UAVs, one flying on either side of the helicopter, with constant velocity and maximum turn rate constraints. Tracking the helicopter around sharp corners will be difficult due to these constraints and the difference in path lengths for the two UAVs. To solve this problem, we propose a hybrid controller that allows the UAVs to swap positions in an attempt to improve the tracking and ground coverage performance of the formation. We discuss tracking control, the position swapping algorithm and collision avoidance. Simulation results demonstrate improved search efficiency and aircraft safety.

keywords- unmanned aerial vehicles, waypoint tracking, hybrid control, conflict resolution

INTRODUCTION
The search and rescue scenario includes two fixed-wing UAVs tracking a manned helicopter in a standard United States Coast Guard (USCG) search pattern [1]. It is desired to maintain an aircraft formation that provides continuous sensor coverage of the ground although the helicopter is more maneuverable than the UAVs. The following assumptions are made:

1. UAVs fly at a constant altitude and airspeed and have a limited turn rate.
2. The helicopter has a much smaller minimum turn radius than the UAVs. Its maximum speed is less than or equal to the airspeed of the UAVs and it flies at a different altitude.
3. The search and rescue flight pattern executed by the helicopter consists of sharp turns that the UAVs will not be able to execute.
4. There is two-way communication between the UAVs, and the positions of all aircraft are known using GPS.

Each UAV is equipped with an autopilot such as the Piccolo from Cloud Cap Technologies. This system controls the UAV’s non-linear flight dynamics, receiving commands such as desired altitude and yaw rate from a higher-level application. The search and rescue controller produces a desired heading angle ($\psi_{des}$) for each UAV, and then commands the desired yaw rate to the autopilot. The yaw rate command is generated by a simple proportional controller shown in Eqn. (1), where $\psi$ is the heading angle and $\psi_{des}$ is the yaw rate command.

$$\dot{\psi} = K(\psi_{des} - \psi)$$ (1)

In general, the hybrid search and rescue controller is designed for two UAVs tracking a more maneuverable aircraft. It assigns each UAV to one of three modes: portside, starboard, and collision avoidance. UAVs are able to exchange assigned positions when in order to aid tracking. The collision avoidance mode will be activated when a conflict is detected between the UAVs, and will provide safe control until the conflict is resolved. Figure 1 shows the architecture of the hybrid controller running on each UAV’s onboard computer.
At all times, the control requires two categories of computation. The UAV turn rate command is calculated by either a tracking controller or the collision avoidance controller, depending on its assigned mode. The second category includes the swap algorithm and conflict detection, which run in addition to the turn rate control. The purpose of these calculations is to determine if a mode transition is necessary. The swap algorithm, along with the guard conditions on transitions $\epsilon_1$ and $\epsilon_2$, determines if reassigning the port and starboard tracking modes would prove beneficial. The conflict detection calculation determines when the UAVs should enter or exit the collision avoidance mode. For decentralized control, it is necessary for the UAVs to independently make the decision to swap at the same time, and to transition in and out of the collision avoidance mode at the same time. Hence, if $\{q_1^i, q_2^i\}$ and $i \in \{1, 2, 3\}$ represents the modes that UAV1 and UAV2 are in at any given time, then the only possible configurations are the following: $\{q_1^1, q_2^2\}$, $\{q_1^2, q_2^3\}$, and $\{q_1^3, q_2^3\}$.

**TRACKING CONTROL**

In this section, we consider a single UAV tracking a helicopter. We begin with the definition of the UAV’s ideal position and desired flight path. The UAV’s ideal position is a constant distance $d_{\text{span}}$ from the helicopter, measured perpendicular to the helicopter’s velocity. The UAV’s desired flight path is the trace of the ideal position, and will provide the desired sensor coverage, but does not satisfy the UAV’s kinematic constraints. The tracking control strategy balances tracking the ideal position with covering the desired flight path. To accomplish this, we employ a slightly modified version of the Aerosonde lateral track control law detailed in [2].
Figure 3. INFLUENCE OF THE PARAMETER $k_2$ ON THE MODIFIED AEROSONDE LATeRAl TRACK CONtROL LAW. IN THIS SIMULATION, $D_{SPAN} = 100$ METERS AND THE HELICOPTER IS MAKING A 90 DEGREE TURN AT TWICE THE MAXIMUM TURN RATE OF THE UAV.

$$X_{track} = \max \{0, X_{track} \}$$

This method causes the UAV to track the wayline whether it is ahead of or behind the helicopter. Figure 3 shows how the parameter $k_2$ effects the trajectory of a UAV that is initially 100 meters in front of the helicopter. When $k_2 = 0$, the control law is the original ALTCL and the UAV attempts to reach the ideal position, which is behind it. This shows that the original Aerosonde lateral track control law is not suitable for this application in situations when the UAV is in front of the helicopter. Through simulation, $k_1 = 0.5$ and $k_2 = 100$ were found to yield satisfactory results.

POSITION SWAPPING

In the previous section, we discussed the control strategy for a single UAV tracking on one side of the helicopter. In this section, we consider one UAV on each side of the helicopter, and discuss the concept of position swapping. Figure 4 shows two different control strategies for a team of two UAVs assisting in a search and rescue scenario. In the first scenario, UAV1 is permanently assigned the portside position and UAV2, the starboard position. As the helicopter turns the corner, both UAVs depart from their ideal paths, leaving gaps in sensor coverage. The second scenario allows UAV1 to swap positions with UAV2 in an attempt to achieve better tracking of the helicopter and to achieve better ground coverage.

The design of the swapping algorithm must satisfy three criteria. The decision to swap must be decentralized, occur simultaneously between UAVs, and benefit the combined tracking and ground coverage of the team of UAVs. To accomplish this, the swapping cost function is defined as follows, where the sums are taken over the two UAVs. The definition of tracking error is discussed in the following sections.

$$J = \frac{\sum_{1,2} \text{current tracking error}}{\sum_{1,2} \text{tracking error after swap}}$$ (3)

According to Fig. 1, $q_1$ represents the portside mode and $q_2$, the starboard mode. Guard conditions on the edges, $e_1$ and $e_2$ will utilize $J$ to determine when a position swap will benefit the system as a whole. The following subsections discuss methods of quantifying the tracking error.

If UAV1 is in the portside mode, we will denote its tracking error by $\text{error}_1^p$ and if it is in the starboard mode, $\text{error}_1^s$. Hence, for UAV1 currently in the portside mode and UAV2 currently in the starboard mode the swapping cost function is as follows:

$$J = \frac{\text{error}_1^p + \text{error}_2^s}{\text{error}_1^p + \text{error}_2^p}$$ (4)

The guard conditions on $e_1$ and $e_2$ are of the form $J > J_{\text{swap}}$. As soon as $J$ is greater than $J_{\text{swap}}$ UAV1 will transition into $q_2$ and UAV2 into $q_1$. After the transition has been made, $J$ becomes the inverse of Eqn. (4).

Tracking Performance using Relative Distances

The simplest method of quantifying the tracking performance is to define tracking error as the Euclidian distance between a UAV and its ideal position. In simulation, this metric rarely resulted in position swapping, even in cases where it would clearly be beneficial. This is due to the fact that the Euclidian
distance between a UAV and its destination does not reflect the distance a UAV has to travel to reach it, due to turn rate constraints.

**Tracking Performance using Dubins Curves**

Dubins’ well-known result [3] provides the optimal path between two points with specified heading angles for a vehicle with a minimum turn radius. This optimal path is proven to be a combination of at most three maneuvers of the following types: turning with minimum turn radius, or traveling straight. The path length of the Dubins path from a UAV to the point C (from Fig. 2) is a useful measure of tracking error because it represents the minimum distance the UAV must travel to reach the point. However, the Dubins path length calculation was too complex to run in our development environment, as further described in the simulations section. Also, a cost function based on Dubins curves would have large discontinuities where the Dubins path types change, which is undesirable. However, the intuition gained from Dubins paths led to development of a tracking error metric that accounts for the UAV’s heading angle and is simple to compute.

**A Heuristic metric for Tracking performance**

The decision to swap waypoints is based on the tracking error metric in Eqn. (5). $\Delta p$ is the Euclidian distance between a UAV and the point it is tracking, and $\Delta \theta$ is the difference in orientation between the UAV and its wayline. The parameter $f$ affects the swap algorithm’s sensitivity to the helicopter’s turn rate, and is tuned by the designer. The parameter value 4.0 was selected based on simulations of UAVs tracking a helicopter through 90 degree turns, as are present in USCG search patterns. For smaller values of $f$, swapping rarely occurs, and larger values cause unwanted sensitivity to small perturbations in the helicopter’s heading angle.

$$tracking\ error = \Delta p(1 + f \Delta \theta)$$ (5)

Having defined $J$, guard conditions for $e_1$ and $e_2$ must be selected via the design parameter, $J_{swap}$. As we increase $J_{swap}$, the set of states for which a transition is possible becomes smaller; however, for values too close to one, zeno behavior may occur. This is because $J$ is inverted when the UAVs swap sides, so $J_{swap}$ must not be allowed to equal its inverse. The value $J_{swap} = 1.05$ was selected through simulation.

**COLLISION AVOIDANCE**

Aircraft safety is guaranteed by two components: conflict detection and a collision avoidance controller. The conflict detection algorithm is evaluated at each time step as the guard for the transitions $e_3, e_4, e_5$, and $e_6$ to or from the collision avoidance mode. It calculates an approximation of the finite time reach set for each UAV, which is the set of all points that the UAV could reach within a particular time. If the finite time reach sets of the aircraft intersect, then there exists a control for each aircraft that could cause them to collide within that time. The approximation of the finite time reach set is conservative, so false alarms are possible, but potential collisions will always be detected. Therefore, when an intersection is detected, there is a potential conflict, and the UAV controllers switch from portside or starboard tracking mode to collision avoidance mode, as shown in Fig. 1. The collision avoidance mode ensures that only safe commands will be generated until the UAVs are no longer in conflict. When a conflict is no longer present, the controller switches back to its previously assigned tracking mode.

**Conflict detection**

Conflict detection is based on checking for an intersection of the approximations of the UAVs’ finite time reach sets. The reach set for a UAV traveling in two-dimensions is defined in the horizontal plane $\mathbb{R}^2$. Given infinite time, a UAV’s reach set includes all of $\mathbb{R}^2$, but limited to the near future, a UAV cannot arrive at the points directly behind itself or on either side. The conflict detection algorithm detects conflicts that could occur within $T$ seconds, where $T$ is the horizon time of the finite reach set calculation. The finite time reach set represents the set of locations that the UAV could pass through within $T$ seconds, and can be easily calculated due to the simplicity of the model. A UAV’s finite time reach set is shown in Fig. 5, along with its rectangular approximation. The curves adjacent to the UAV are determined by its maximum turn rate, and the forward curves are determined by $T$ and the UAV’s constant velocity. If the UAV is located at the origin and has heading angle $\psi$, the right side boundaries of the finite time reach set are given by the following equations, where $R$ is the UAV’s minimum turn radius and $V$ is its forward velocity. The left boundaries are calculated similarly.

Right turn limit:

$$x = R \sin \psi + R \cos(\pi/2 + \psi - \phi)$$
$$y = -R \cos \psi + R \sin(\pi/2 + \psi - \phi)$$
$$\phi \in \left[0, \frac{V T}{R}\right]$$

Forward right limit:

$$x = R_I \sin \psi + R_I \cos \left(\pi/2 + \psi - \frac{V T}{R_I}\right)$$
$$y = -R_I \cos \psi + R_I \sin \left(\pi/2 + \psi - \frac{V T}{R_I}\right)$$
$$R_I \in [R, \infty]$$
The conflict detection algorithm must check for intersections of the UAVs’ reach sets in real time. For a reach set defined by the curves above, this could be computationally intensive. Therefore, the reach set is approximated by a rectangle as shown in Fig. 5. The rectangle’s two vertices nearest the UAV \((x_1, y_1)\) and \((x_2, y_2)\) are the endpoints of the turn-rate limit curves (as in Eqn. (6)). The height \(H\) of the rectangle is determined by the UAV’s turn radius and speed as in Eqn. (8).

\[
H = VT - R \sin \left( \frac{VT}{R} \right) \tag{8}
\]

After exactly \(T\) seconds, the UAV will be located not only within its reach set, but within the rectangular approximation. This is because its maximum forward progress is limited by its forward velocity to \(VT\), the distance to the far side of the rectangle. Its minimum forward progress is limited by its maximum turn rate and constant velocity, represented by the near side of the rectangle. Therefore, a potential collision will be detected at least \(T\) seconds before it could occur.

The algorithm for detecting the intersection of two rectangles depends on the fact that in order for the rectangles to transition from a non-intersecting state to an intersecting state, a vertex of one rectangle must enter the other rectangle. Thus, if the rectangles do not previously intersect, their intersection can be detected based on the locations of their vertices. This is accomplished by transforming the coordinates of a vertex into a coordinate system aligned at the center of the other rectangle. The following boolean expression tells whether a point \((x, y)\) falls inside a rectangle centered at \((x_0, y_0)\), oriented at angle \(\theta\), and with width \(W\) and height \(H\). This expression can easily be calculated in real time for each vertex of each UAV’s reach set approximation. If it is true for any vertex, the rectangles intersect and there is a conflict. This condition causes the UAVs to transition into collision avoidance mode.

\[
(x, y) \in \text{Rectangle}(x_0, y_0, \theta, W, H) \iff \\
|\cos \theta (x - x_0) + \sin \theta (y - y_0)| \leq W/2 \\
\wedge \\
|\cos \theta (y - y_0) - \sin \theta (x - x_0)| \leq H/2 
\tag{9}
\]

Safe maneuver selection

When the UAVs transition from either the portside or starboard mode to the collision avoidance mode, the rectangular approximations of their reach sets have just begun to intersect. At this point, there may exist sequences of commands \(\psi_1\) and \(\psi_2\) for the two UAVs that could cause them to collide. The goal of the collision avoidance controller is to safely command the UAVs until their reach sets no longer intersect, while also taking into account their assigned tracking positions. Conflict resolution for aircraft is a widely studied problem, and many centralized and decentralized solutions have been proposed [4]. The solution presented here is decentralized and simple to compute, but is not optimal.

The simplicity of the collision avoidance algorithm comes from limiting each UAV to three possible actions: maximum turn rate left (L), maximum turn rate right (R), and continue straight (S). A joint collision avoidance maneuver is then defined as an action assignment for UAV1 and an action assignment for UAV2, such as (L,R). Thus, we must guarantee that at least one of the nine such maneuvers will be safe, and choose from among the safe ones based on cost. The cost of a maneuver is defined to penalize an action that turns a UAV away from its desired trajectory. The desired trajectory depends on the tracking mode that the UAV was in before the conflict was detected. For example, if a UAV was in the starboard mode before the conflict was detected, its desired heading \(\psi_{des}\) is that which points toward the starboard ideal tracking position. The cost function \(J\) of a collision avoidance maneuver is defined in Eqn. (10), where \(\psi_{new}\) is the new heading that would result from applying the maneuver to each UAV. Note that each UAV evaluates the cost of a maneuver by summing the individual costs for both UAVs. With shared state information, each UAV’s controller arrives at the same cost for each combined maneuver.

\[
J = \sum_{UAVs} |\psi_{des} - \psi_{new}| 
\tag{10}
\]

Assume a UAV has heading angle \(\psi\), desired heading angle \(\psi_{des}\), velocity \(V\), and turn radius \(R\), and that the time horizon of the reach set is \(T\). Then the cost of each action L, R, or S is as
follows. Note that the cost of a collision avoidance maneuver is very simple to compute.

\[
\begin{align*}
J_L &= |\psi_{des} - \psi - VT/R| \\
J_R &= |\psi_{des} - \psi + VT/R| \\
J_S &= |\psi_{des} - \psi|
\end{align*}
\]  

(11)

**Geometric safety calculation**

Having established a metric for whether a maneuver is favorable, it remains to be calculated whether a maneuver is safe. An added advantage to restricting a UAV to the three actions L, R, and S is that the resulting trajectory will be either a circular arc or a line segment. Conditions for the intersection of these trajectories can be derived as simple boolean expressions, and will be used to evaluate the safety of collision avoidance maneuvers. If the UAVs’ trajectories intersect within the time horizon \( T \), the maneuver is considered unsafe. As implemented, this method does not include a safety buffer around each aircraft. In other words, the two UAVs may pass very close to one another without colliding, and the maneuver will be considered safe. However, the method described could easily be extended to include a buffer zone using similar geometric conditions.

The safety of a collision avoidance maneuver is evaluated in 3 steps. First, the corresponding action (L, R, or S) for each UAV is assigned. Then, the trajectory parameterization corresponding to that action is calculated. For example, for action S (go straight), the trajectory is a line segment that starts at the current UAV location \((x, y)\) with orientation \( \psi \) and length \( VT \). The L or R action produces a circular arc, which is parameterized by a center point \((x, y)\) and starting and ending angles \( \theta_1 \) and \( \theta_2 \). Finally, the appropriate geometric condition tests the parameters of the UAVs’ trajectories for intersection. For example, if the collision avoidance maneuver is (S,S), then the maneuver is safe if the following is false. The conditions for intersection of two arcs, or a line segment and an arc, can be derived similarly, but will not be shown.

\[
\begin{align*}
x_1 &\leq x_e \leq VT \cos \theta_1 + x_1 \\
x_2 &\leq x_e \leq VT \cos \theta_2 + x_2 \\
\text{where} &
\end{align*}
\]

\[
x_e = \frac{y_2 - x_2 \tan \theta_2 - y_1 + \tan \theta_1}{\tan \theta_1 - \tan \theta_2}
\]

(12)

**Internal structure of collision avoidance mode**

It has been shown how to calculate the cost of a collision avoidance maneuver and how to evaluate its safety. The internal structure of the collision avoidance mode uses both elements to efficiently select the safe maneuver with the lowest cost. It can be seen from the previous sections that it is more computationally expensive to evaluate a maneuver’s safety than its cost. Therefore, the set of nine possible collision avoidance maneuvers will first be sorted according to cost. The least costly maneuver will then be evaluated for safety, and thus either selected or eliminated. If it is unsafe, the next least costly maneuver will be evaluated, and so on.

**Proof of safety**

Aircraft safety is a strict requirement, and therefore must be guaranteed. First it must be shown that the conflict detection algorithm functions as desired, specifically, that UAVs will transition to collision avoidance mode \( T \) seconds before a collision can take place. Then, it must be shown that the collision avoidance mode safely resolves the conflict.

To show that the conflict detection algorithm functions as desired, it must be shown that the rectangular approximations of the UAVs’ reach sets will intersect \( T \) seconds before a collision is possible. It was explained previously that if a UAV’s reach set is calculated at time \( t \), the UAV must be located within the rectangular approximation at time \( t + T \). Assume that the two UAVs collide at time \( t_c \). Then at time \( t_c - T \), each UAV’s rectangular reach set approximation must have contained the collision point. Therefore the rectangular approximations of the UAVs’ reach sets intersected \( T \) seconds before the collision, at which time the conflict was detected.

Having shown that a potential collision will be detected \( T \) seconds in advance, it must be shown that the collision avoidance controller will safely resolve the conflict. This requires two conditions. First, at least one of the nine combined collision avoidance maneuvers must be safe. Second, execution of the selected safe collision avoidance maneuver must resolve the conflict.

The collision avoidance actions (L) and (R) represent the boundaries of a UAV’s reach set. Therefore, if there does not exist a safe collision avoidance maneuver, then both boundaries of both UAVs’ reach sets must intersect as shown in Fig. 6. However, the turn rate constraint implies that the UAVs cannot make sudden heading changes, so the edges of the reach sets must intersect long before such a configuration can occur. This means that the conflict will always be detected before all safe maneuvers are eliminated.

Having guaranteed that there will exist a safe collision avoidance maneuver, it must be shown that executing this maneuver will resolve the conflict. For the search and rescue application, the UAVs are tracking waypoints that are separated by a fixed distance. A conflict is introduced when the UAVs must cross paths in order to exchange waypoints. Therefore, the conflict is resolved when each UAV can follow waypoint control to its assigned waypoint without reactivating the collision avoidance mode. Because the maneuver cost function penalizes maneuvers that direct UAVs away from their waypoints, a maneuver
that resolves the conflict will be selected rather than a maneuver that simply prevents collision.

SIMULATION RESULTS

The hybrid search and rescue controller was developed using Hyvisual, a hybrid systems simulation environment developed at the University of California, Berkeley [5]. This graphical environment was useful for modular development, testing and integration of individual software components. The complete simulation environment of UAVs, helicopter, and mode-switching controllers was too large to compile in the graphical environment, and was therefore simulated using C++. The aircraft paths resulting from one such simulation are shown in Fig. 7. A simulation in which waypoint swapping is disabled is shown for comparison in Fig. 8. At each corner, the inside UAV overshoots the desired path and the outside UAV cuts the corner and falls behind.

The UAVs are represented by a standard kinematic model in two dimensions as in Eqn. (14), with constant velocity $V$ and maximum turn rate $\psi_{max}$. The control input is $\psi$.

$$
\dot{x} = V \cos \psi \\
\dot{y} = V \sin \psi \\
|\psi| \leq \psi_{max}
$$

The parameter values below come from the University of California Berkeley Sig Rascal experimental platform. The altitude falls within the range suggested by USCG procedure [1] for searching for a human or small raft.

$$
V = 20 \text{ m/s} \\
\psi_{max} = 0.2 \text{ r/s} \\
\text{Altitude} = 150 \text{ m}
$$

The helicopter model is identical, except the maximum turn rate is twice as large.

When an aircraft flies at altitude $A$, its sensor footprint is a circle centered below its location with diameter $D$ given by Eqn. (14). $\beta$ is the field of view angle of the sensor. The sensor is modeled with a field of view angle of 80 degrees, which is typical for small weatherproof infrared cameras that may be used for this
The aircrafts’ sensor footprints can be plotted over their positions, and analysis of the resulting image results in the coverage ratio of the searched area. In Fig. 9, each aircraft’s sensor footprint is plotted over its path, and in Fig. 10, the image has been converted for analysis. Black pixels represent area that has been imaged by an aircraft’s sensor. The ratio of black to total pixels gives the percent coverage for a test.

\[ D = 2A \tan \frac{\beta}{2} \]  

(14)

Ground coverage

A number of simulations of the type described in the previous section were performed in order to examine the resulting sensor coverage under varying conditions. The two varied parameters were the separation distance \( d_{\text{span}} \) and the effect of waypoint swapping, which could be enabled or disabled. When \( d_{\text{span}} \) is reduced by some fraction, the aircrafts’ sensor footprints will overlap with the same percentage redundancy. Decreasing the span distance reduces gaps in sensor coverage, but it also decreases the rate of area searched. These factors must be balanced to produce an accurate search that can be completed in a reasonable time. The sensor coverage results from these tests are compared in Fig. 11. The x-axis shows the percent redundancy in the sensor coverage. With position swapping enabled, ground coverage between 91 and 99 percent was achieved. Without position swapping enabled, the ground coverage was inferior in all cases, ranging from 85 to 97 percent.

Collision avoidance

In the series of simulations discussed previously, the collision avoidance mode was never activated because the UAVs’ finite time reach sets did not intersect. Although in these cases aircraft safety was provided by the waypoint controllers, this may not always be the case, especially in the presence of disturbances. Therefore, the collision avoidance mode was tested by overriding the helicopter tracking components, and instead assigning fixed waypoints. These were assigned such that the UAVs would collide unless evasive action was taken.

In Fig. 12 the UAVs would be expected to cross paths at a relative angle of 90 degrees and collide. It can be seen that instead of flying directly toward their waypoints, the UAVs deviate from their direct trajectories. This causes one to reach the point where their paths intersect well before the other, preventing col-
CONCLUSIONS

We have developed a distributed hybrid controller to allow two UAVs to track a helicopter performing sharp turns. The intent is to augment U.S. Coast Guard search and rescue procedures by placing sensors on the UAVs and thereby increasing the search rate. One UAV is assigned to fly on each side of the helicopter, being controlled by the appropriate tracking control mode. The UAVs swap their side assignments when a heuristic cost function indicates that doing so would provide better sensor coverage and tracking error. At all times, a conflict detection algorithm checks for the intersection of the UAVs’ reach sets. If a conflict is detected, the UAVs switch into collision avoidance mode. The collision avoidance mode commands a safe trajectory, favoring trajectories that direct the UAVs toward their assigned waypoints.

Simulations showed that the mode-switching tracker provided more complete sensor coverage of the search area than fixing a particular UAV on each side of the helicopter. This control had the added benefit that the UAVs never came into conflict, and so the collision avoidance mode was never activated. In separate tests, the collision avoidance algorithm caused the UAVs to safely arrive at their assigned waypoints in situations that otherwise would result in collisions. Overall, the hybrid controller performed very well in simulations with kinematic aircraft models and perfect communication. Future work will involve development and implementation in a distributed environment with limited communication and more complex aircraft models.

REFERENCES