2016 IEEE Conference on Control Applications (CCA)
Part of 2016 IEEE Multi-Conference on Systems and Control
September 19-22, 2016. Buenos Aires, Argentina

Automated UAV Tasks for Search and Surveillance

Derek Kingston[1], Steven Rasmussen[2], Laura Humphrey[1]

Abstract— **There is growing interest in using unmanned aerial vehicles to assist in operations that require search and surveillance. However, a challenge remains in providing the right level of automation. Manual teleoperation and waypoint-based planning provide a great deal of flexibility but require significant human effort, whereas high levels of automation reduce the level of human effort but at the cost of flexibility and human judgement in uncertain and dynamic environments. Here, we seek to provide an intermediate level of automation through a set of parameterized tasks that implement common UAV search and surveillance patterns, taking into account low-level details such as dynamics of the UAV, geometry of the sensor footprint, and quality of the sensor imagery. We describe implementation of these tasks and discuss how they can be used by either human operators or by higher levels of automation to plan search and surveillance missions.**

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have the potential to save time, money, and even lives across a variety of domains that require search and surveillance. As an example, the US Coast Guard estimates that while only 10 percent of their cases involve searches, these searches have an annual cost of approximately $50 million [1]. Not only that, but the condition of those in distress degrades and the probability of their survival decreases the longer the search takes. Similar concerns exist for wilderness search and rescue operations, which may require thousands of hours of search over large, rugged, and remote terrains [2], and for forest fire suppression, where persistent surveillance and monitoring is needed to guide fire fighting efforts and ensure fire fighters do not unknowingly enter high risk regions [3].

While UAVs can in theory decrease the cost and increase the efficiency of search and surveillance in domains such as these without the need for risking manned assets, a challenge remains in providing the right level of automation. Manual teleoperation provides a great deal of flexibility but is extremely labor intensive, requiring multiple human operators per vehicle [4]. Highly autonomous UAV controllers have been developed that allow a single human operator to supervise multiple vehicles, but they are currently less flexible and more fault-prone in complex and uncertain environments. They also suffer from issues that arise when humans interact with high levels of automation, including loss of situational awareness, mis-calibrated trust in automation, and increased workload when automation performs poorly [5]. Emphasis has therefore been placed on providing functionality at different levels of automation [6], [7].

Here, we seek to provide an intermediate level of automation for UAV control by formulating a set of automated search tasks that implement common search and surveillance patterns. These automated search tasks handle the low-level details of waypoint-based path planning and sensor steering, with parameters that can be used to tailor the search pattern to meet the needs of the current situation. In what follows, Section II provides background on details accounted for in the search tasks, Section III describes the search tasks themselves, and Section IV describes a flexible method for mission planning based on the tasks. Section V concludes with a discussion of ongoing improvements to these tasks and how they fit into the hierarchy of automation for multi-UAV control.

II. BACKGROUND

The automated tasks we will develop in Section III are motivated in part by the need to quickly and precisely account for factors that affect imagery collected by a UAV. While modern ground control stations allow users to pre-specify a UAV flight plan as a series of waypoint and sensor gimbal commands [8], the burden is generally on the user to ensure the resulting trajectory will meet all image collection requirements for the mission. When requirements include – e.g., that an image of a target must be taken from a particular angle of approach, that a target must be visible in a video feed for a specific amount of time, or that a video must capture the entirety of a specified region – planning of the necessary commands must take into account low-level implementation details such as dynamics of the UAV, geometry of the sensor footprint, and quality of the obtained imagery. Here, we will address these implementation details by assuming Dubins vehicle dynamics for the UAVs and computing the sensor footprint under a flat earth assumption, though we note the proposed framework is readily extensible to more elaborate models.

A. Dubins Vehicles

Let a UAV's position relative to the Earth's inertial frame in North-East-Down (NED) coordinates be denoted

$$p = (x_p, y_p, z_p), \tag{1}$$

where x_p is the position along an axis pointing toward inertial North, y_p is the position along an axis pointing toward inertial East, z_p is the position along an axis pointing

This work was supported by the OSD Autonomy Pilot Research Initiative and AFOSR grant #13RQ03COR.

[1]D. Kingston and L. Humphrey are with the Autonomous Control Branch of the Aerospace Systems Directorate, Air Force Research Laboratory, Dayton, OH 45433, USA. {derek.kingston, laura.humphrey}@us.af.mil

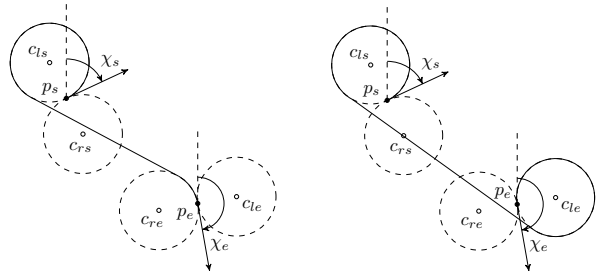
[2]S. Rasmussen is with Miami Valley Aerospace LLC, working for the Air Force Research Laboratory, Dayton, OH 45433, USA. steven.rasmussen.5.ctr@us.af.mil

into the Earth, and $p_0 = (0, 0, 0)$ is some fixed point on the Earth's surface. Note that since z_p is negative above the Earth's surface, we define $h_{alt} = -z_p$ to be the UAV's altitude.

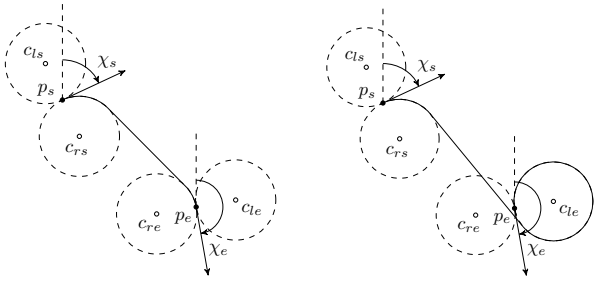
Assume that each UAV flies at a constant speed and altitude and that the Earth's surface is approximated as flat. We then model each UAV as a Dubins vehicle [9], with dynamics in the North-East plane given by

$$\begin{aligned}\dot{x}_p &= V\cos(\psi)\\ \dot{y}_p &= V\sin(\psi)\\ \dot{\psi} &= u(t),\end{aligned}$$

where V is a constant and $u(t) \in [-\bar{u}, \bar{u}]$ for all time t. Intuitively, these equations describe a vehicle with a minimum turn radius $r_{min} = V/\bar{u}$, where turns can be clockwise or counterclockwise. Suppose we have a starting configuration $\nu_s = (p_s, \chi_s)$, where p_s and χ_s are the starting position and heading respectively, and a similarly defined desired ending configuration $\nu_e = (p_e, \chi_e)$. It was shown in [10] that for a vehicle with these dynamics, the time-optimal feasible path between two such configurations consists of a circular arc followed by a line followed by another circular arc. This path can be computed by placing two circles of radius r_{min} on p_s, a "left" and "right" circle representing the two possible turns from ν_s, with centers c_{ls} and c_{rs} at distance r_{min} from p_s and 90° counterclockwise and clockwise from heading χ_s. Similarly, two circles with centers c_{le} and c_{re} are placed on p_e. The shortest path can be found by checking each possible combination of starting and ending turns, with a straight line tangent to the corresponding circles connecting them, as shown in Figure 1.



(a) Left-straight-right Dubins path (b) Left-straight-left Dubins path



(c) Right-straight-right Dubins path (d) Right-straight-left Dubins path

Fig. 1. Dubins paths that transition a UAV from a starting configuration (p_s, χ_s) to an ending configuration (p_e, χ_e). Adapted from [9]. In this case, the right-straight-right Dubins path is optimal.

B. Sensor Footprint

We would generally like to select the UAV's starting and ending configurations for a task such that the task's target is in front of the leading edge of the sensor footprint at the task's start and behind the trailing edge of the sensor footprint at the task's end. Assuming the UAV's roll and pitch are 0° throughout the task, the geometry of the sensor footprint depends on the sensor's gimbal elevation angle θ_g, its vertical field of view η_v, its horizontal field of view η_h, and the UAV's altitude h_{alt}, as shown in Figure 2. We can then compute the distance from the UAV's ground position (x_p, y_p) to the sensor footprint's leading edge d_l, the distance to its center d_c, the distance to its trailing edge d_t, and its width d_w as

$$\begin{aligned}d_l &= \frac{h_{alt}}{\tan(\theta_g - \eta_v/2)}\\ d_c &= \frac{h_{alt}}{\tan(\theta_g)}\\ d_t &= \frac{h_{alt}}{\tan(\theta_g + \eta_v/2)}\\ d_w &= 2h_c\tan(\eta_h/2) = 2\frac{h_{alt}}{\cos(\theta_g)}\tan(\eta_h/2).\end{aligned}$$

Note that if we allow the gimbal azimuth angle ψ_g to change, the sensor footprint is rotated clockwise by ψ_g about the UAV, but its shape remains the same. Note $\psi_g = 0^\circ$ directly in front of the UAV.

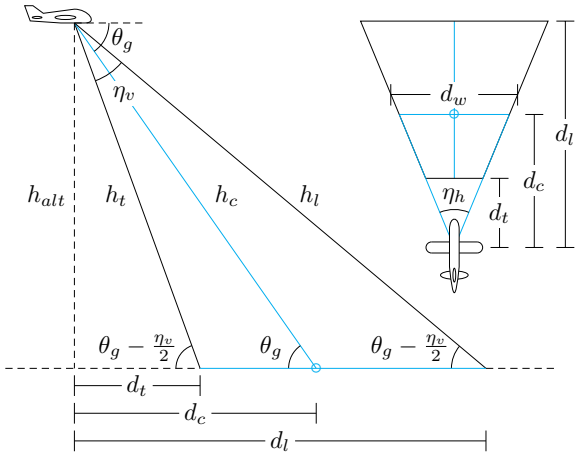


Fig. 2. When UAV roll and pitch are both 0°, geometry of the sensor footprint is determined by altitude h_{alt} as well as the sensor's gimbal elevation θ_g, vertical field of view η_v, and horizontal field of view η_h.

C. Ground Sample Distance

For many of our tasks, we would like to set a desired ground sample distance (GSD), i.e. number of meters per pixel in the sensor image. Note that GSD is not uniform throughout the image, so we define GSD based on the maximum number of meters per pixel at the sensor footprint's center as measured along the vertical and horizontal. Let r_h and r_v be the horizontal and vertical resolution of the sensor

in pixels. Then we define gsd as

$$gsd = \max\left(\frac{d_w}{r_h}, \frac{d_l - d_t}{r_v}\right).$$

Note that if we fix the UAV's altitude and the sensor's field of view, gsd depends only on gimbal elevation θ_g. Also note that smaller values of gsd correspond to more detailed images, and gsd is smallest when $\theta_g = 90°$. However, smaller values for gsd are not always desirable, e.g. if the UAV is flying quickly and we want a target to remain in the sensor footprint longer. For most tasks, we then select a target value for gsd and set the gimbal elevation θ_g to the value that achieves it. If the target value cannot be achieved, θ_g is set to 90°.

III. TASKS

Each of the automated tasks we describe in this section plans waypoint-based paths and sensor steering commands according to the task's type and the value of its customizable parameters. For some tasks, the resulting task path is simply a starting configuration ν_s and ending configuration ν_e, with waypoints placed according to the time-optimal Dubins path between them. For other tasks, the task path is decomposed into a sequence of starting and ending configurations $(\nu_s(1), \nu_e(1)), \ldots (\nu_s(n), \nu_e(n))$ that are then connected by Dubins paths. For yet other tasks, the task path is computed based on a function or more complicated algorithm that does not explicitly make use of Dubins paths but returns a task path that is approximately flyable. In such a case, the UAV's autopilot will make adjustments as it flies between waypoints, and there may be some deviations between the planned task path and the actual path flown by the UAV. For sensor steering, some tasks use a single fixed sensor orientation throughout, while others require re-positioning the sensor at certain points on the task path.

The following sections describe planning approaches and parameters for five tasks: *Point Inspect*, *Line Search*, *Area Search*, *Spiral Search*, and *Sector Search*. Note that while each task has its own unique parameters, all tasks have a parameter for desired gsd. This sets the sensor gimbal elevation angle θ_g as described in Section II-C.

A. Point Inspect

The *Point Inspect* task is motivated by the need to image a target at a known location, possibly from a particular angle of approach and standoff distance. Parameters for this task are given in Table I. As shown in Figure 3, for a specified point p, this task is by default configured to start and end with the center of the leading and trailing edge of the sensor footprint on p. The angle of approach χ is then chosen from a set of discrete angles about p such that the path from the previous task ending configuration $\bar{\nu}_e$ is minimized. The starting and ending configurations $\nu_s = (p_s, \chi_s)$ and $\nu_e = (p_e, \chi_e)$ are then computed according to

$$\begin{aligned} p_s &= p + (d_l \cos(\chi), d_l \sin(\chi), z_p) \\ p_e &= p + (d_t \cos(\chi), d_t \sin(\chi), z_p) \\ \chi_s = \chi_e &= \chi + 180°. \end{aligned} \tag{2}$$

It should be noted that when searching over the discrete set of angles, we can also apply other metrics for selecting χ, e.g. to include a hard constraint for avoiding no-fly zones or to optimize some metric other than distance.

TABLE I
PARAMETERS FOR THE POINT INSPECT TASK

Name	Description
Point p	Point to inspect.
Approach angle range (χ_c, χ_r)	Range of allowable angles of approach to p measured clockwise from inertial North, with $\chi_s = \chi_e \in \left(\left(\chi_c \pm \frac{\chi_r}{2}\right) + 180°\right)$. Default = null.
Standoff distance s	Distance from which to approach p. Default = d_l.

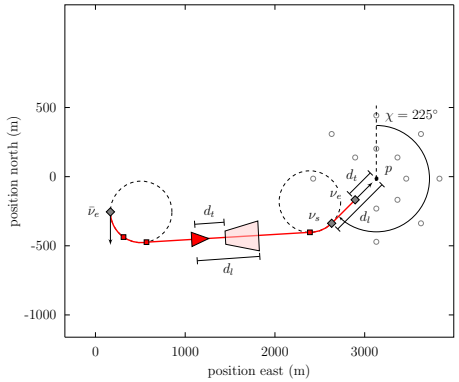


Fig. 3. A Point Inspect task on point p. Since no angle of approach range is specified, a search is performed over a discrete set of angles about p, and angle of approach χ is chosen to minimize the path.

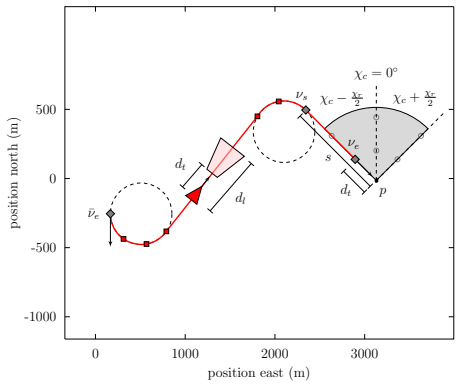


Fig. 4. A Point Inspect task on point p when an angle of approach range (χ_c, χ_r) and standoff distance s are specified.

If a particular view angle on a target at p is desired, the user can specify a range of approach angles through parameters χ_c and χ_r. The task will then use the value $\chi \in [\chi_c \pm \frac{\chi_r}{2}]$ that minimizes the path or directly set $\chi = \chi_c$ if $\chi_r = 0°$. If the user would like to allow for more time to approach p or see more of the region leading up to p, the

user can set a standoff distance s for the start of the task. Eq. (2) for the starting point p_s is then modified to

$$p_s = p + (s\cos(\chi), s\sin(\chi), z_p),$$

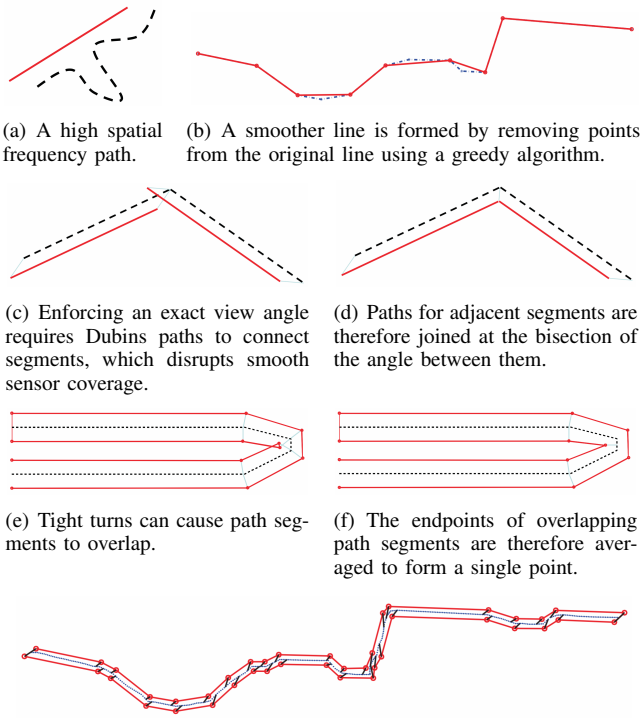
which may also affect the angle of approach χ. An example with values chosen for χ_c, χ_r, and s is shown in Figure 4.

B. Line Search

The *Line Search* task is motivated by the need to image entities that can be modeled as lines – e.g., roads and area perimeters – using a particular view angle. The task parameters are given in Table II, with line l described by a sequence of points $l = l(1), \ldots, l(n)$ and the list of allowable view angles as $\chi = \chi(1), \ldots, \chi(n)$. A boolean flag b indicates whether these view angles specify that the gimbal azimuth angle should be set to $\psi_g = \chi$ for the task (when $b =$ false) or that path waypoints should be placed relative to line l at distance d_c and angle χ measured clockwise from inertial North (when $b =$ true).

In practice, the line described by l may consist of a great many points, e.g. if it corresponds to a road whose segments are derived from a high fidelity map. This can be problematic, both because UAV autopilots have a limit on the number of waypoints that can be uploaded at a time and because lines may have high spatial frequency components that cannot be imaged at a constant desired view angle due to the dynamics of the UAV, as shown in Figure 5(a). We therefore implement the Line Search task using the approach described in [11]. The steps for computing the task path are as follows:

1) Apply a greedy algorithm to line l that removes points to form a smoother line l'. This algorithm starts with $l' = l$ and at each iteration, it removes the point in l' that results in the smallest area between l' and l. The algorithm stops when only a specified number of points remain, as demonstrated by a simple example in Figure 5(b).
2) Form a preliminary flight path p' for the UAV by creating path segments for each segment in l' such that the center of the sensor footprint covers the corresponding line segment at an acceptable view angle χ. However, as shown in Figure 5(c), this leads to discontinuities between path segments that would have to be handled with Dubins paths, which would preclude smooth sensor coverage of the line. Rather than strictly enforce the view angle, adjacent segments are joined based on the bisection of the angle between them, as shown in Figure 5(d).
3) There might now be segments in p' that overlap, e.g. due to tight turns, as shown in Figure 5(e). For such segments, we average their endpoints to form a single point, as shown in Figure 5(f).
4) As the UAV flies the path p' derived from l', the center of the sensor footprint is steered to points on l such that the proportion of l that has been covered by the sensor is equal to the proportion of p' that has been flown by the UAV. A sampling of view angles is shown in Figure 5(g), where the UAV searches the line in both directions.



(a) A high spatial frequency path.

(b) A smoother line is formed by removing points from the original line using a greedy algorithm.

(c) Enforcing an exact view angle requires Dubins paths to connect segments, which disrupts smooth sensor coverage.

(d) Paths for adjacent segments are therefore joined at the bisection of the angle between them.

(e) Tight turns can cause path segments to overlap.

(f) The endpoints of overlapping path segments are therefore averaged to form a single point.

(g) During the Line Search task, the actual view angle is chosen based on percent progress over the path and the percent of the line that has been imaged.

Fig. 5. Implementing the Line Search task requires smoothing line l to form l', forming p' from l' according to an acceptable view angle χ, connecting adjacent path segments in p', removing overlapping path segments in p', and making a best effort to match view angle χ throughout the task. Figures adapted from [11].

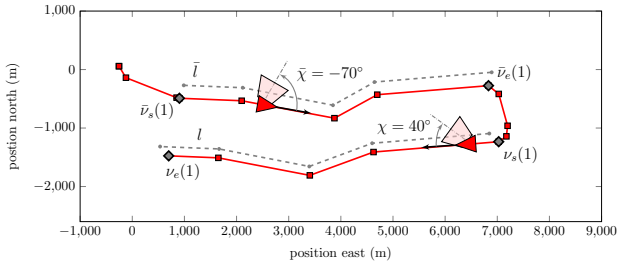


Fig. 6. Two Line Search tasks, the first on line $\bar{l}$ with $\bar{\chi} = -70°$ and the second on line l with $\chi = 40°$. For both tasks, $b =$ false and $\theta_g = 50°$.

Two consecutive line search tasks are shown in Figure 6, where the first line is $\bar{l}$ and the second is l. For both tasks, gimbal elevation angle $\theta_g = 50°$ so that the shape of the sensor footprint is the same. For both tasks, $b =$ false so that the gimbal azimuth angle ψ_g is set to the view angle; however, the first task has view angle $\bar{\chi} = -70°$, and the second has view angle $\chi = 40°$. This results in the UAV flying closer to line l than it does to line $\bar{l}$, and the sensor points to the UAV's right rather than its left.

TABLE II

PARAMETERS FOR THE LINE SEARCH TASK

Name	Description
Line l	Line to search, defined by points $l = l(1), \ldots, l(n)$.
View angles χ	List of allowable view angles $\chi = \chi(1), \ldots, \chi(n)$. Default = $0°$.
Inertial angle flag b	If b = true, interpret view angles as placing waypoints relative to line l at distance d_c and angle χ measured clockwise from inertial North; otherwise, use as the gimbal azimuth ψ_g. Default = true.

C. Area Search

The *Area Search* task is motivated by the need to image entire regions, possibly using a particular sweep angle and after visiting a particular point first. The task parameters are given in Table III. The area a is described by vertices $a(1), \ldots, a(n)$, with the option to have the task generate vertices that approximate a circle with a specified radius and center. By default, the task sweeps the area by creating "lane" segments that run at angle χ measured clockwise from inertial North, with $\chi = 0°$ as the default. The user can also optionally specify a Point Inspect task to perform before the Area Search, which can be convenient, e.g. if the area has an associated station or facility that should be inspected before the area is searched. Given values for these parameters, the steps for computing the search path are as follows:

1) If a Point Inspect task is requested, compute the starting and ending configurations $\nu_s(p)$ and $\nu_e(p)$ as in Section III-A. When computing the subsequent Area Search, $\nu_e(p)$ is then taken as the previous task's ending configuration.
2) Form the convex hull a' of the polygon a. If a is already convex, $a' = a$.
3) Rotate a' about its center by $-\chi$.
4) Find the westernmost point $y_{wa'}$ of the polygon described by a', and set the starting and ending y-coordinate for the first search lane to $y(1) = y_{wa'} + \frac{\alpha d_w}{2}$. At $y = y(1)$, find the southernmost and northernmost points $x_{sa'}$ and $x_{na'}$ of a', and account for the sensor footprint at the start and end of the lane by setting $x_s(1) = x_{sa'} - d_l$ and $x_e(1) = x_{na'} - d_t$. The starting and ending configurations for the first lane are then $\nu_s(1) = ((x_s(1), y(1), z_p), 0°)$ and $\nu_e(1) = ((x_e(1), y(1), z_p), 0°)$.
5) For each subsequent lane $i > 1$, set $y(i) = y(i-1) + \alpha d_w$. At $y = y(i)$, find the northernmost and southernmost points $x_{na'}$ and $x_{sa'}$ of a'. If i is odd, set $x_s(i) = x_{sa'} - d_l$, $x_e(i) = x_{na'} - d_t$, and $\chi(i) = 0°$. If i is even, set $x_s(i) = x_{na'} + d_l$, $x_e(i) = x_{sa'} + d_t$, and $\chi(i) = 180°$. The starting and ending configurations for the lane are then $\nu_s(i) = ((x_s(i), y(i), z_p), \chi(i))$ and $\nu_e(i) = ((x_e(i), y(i), z_p), \chi(i))$.
6) Repeat Step 5 until $y(i)$ is less than $\frac{\alpha d_w}{2}$ from the easternmost point in a'.
7) Rotate the search lanes about the center of a' by χ.
8) Connect the search lanes with Dubins paths.

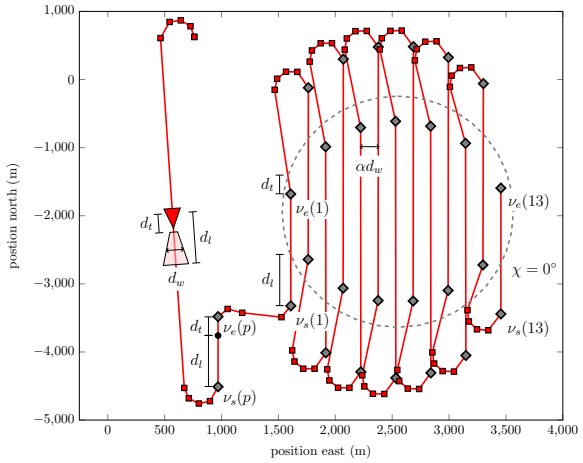


Fig. 7. An Area Search over a circle with $\chi = 0°$, $\alpha = .9$, $\theta_g = 60°$, and a Point Inspect task.

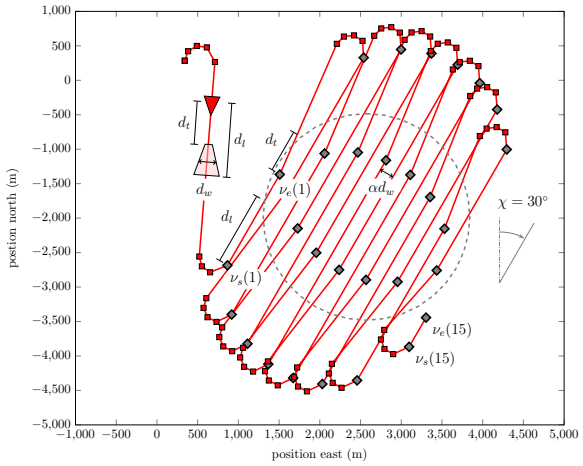


Fig. 8. An Area Search over the same circle as in Figure 7, but with $\chi = 30°$, $\alpha = .6$, $\theta_g = 20°$, and no Point Inspect task.

Figure 7 shows an example Area Search for a circle when $\chi = 0°$, $\alpha = .9$, $\theta_g = 60°$, and there is a Point Inspect task. Figure 8 shows an example for the same circle when $\chi = 30°$, $\alpha = .6$, $\theta_g = 20°$, and there is no Point Inspect task. Comparing Figure 7 and Figure 8, note that because θ_g is larger in the former, the leading and trailing edges of the sensor footprint are closer to the UAV and as a result, the search lanes start and end much closer to the boundary of the circle. In the latter, since α is smaller, the search lanes are closer together and as a result, more search lanes are required to complete the task.

D. Spiral Search

The *Spiral Search* task is motivated by the need to search for an entity from a last known location. As the name suggests, the search is performed by following a spiral path

TABLE III

PARAMETERS FOR THE AREA SEARCH TASK

Name	Description
Area a	Polygon defined by points $a = a(1), \ldots, a(n)$.
Sweep Angle χ	Angle of the search lanes. Default = $0°$.
Lane Spacing Factor α	Lanes are separated by distance αd_w. Default = 0.9.
Point Inspect P_t	Point Inspect task to perform before Area Search. Default = null.

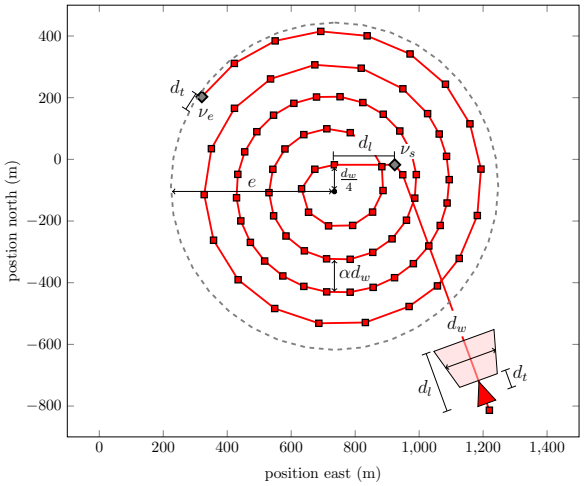


Fig. 9. A Spiral Search task with an adjustable extent e and with rotations of the spiral separated by adjustable distance αd_w.

from this location, an efficient strategy promoted in domains such as maritime search and rescue [1]. Parameters for this task are given in Table IV. The center of the spiral is set to a specified location p, and the starting configuration of the path is set to $\nu_s = (p + (\frac{d_w}{4}, d_l, z_p), 270°)$. The rest of the path for the search is then formed by sampling points from an Archimedian spiral described in polar coordinates (r, θ) by the equation

$$r = \frac{d_w}{4} + \alpha d_w \theta \tag{3}$$

starting from $\theta = 0°$ with θ measured counterclockwise. This sets the start of the spiral so that the region near the center is covered by the sensor footprint during the first rotation. Subsequent rotations cover points at distance αd_w from points in the previous rotation, with the default $\alpha = 0.9$ ensuring there is a slight overlap in the sensor footprint between rotations so that the whole region is imaged. The task ends when the center of the trailing edge of the sensor footprint intersects the circle centered at p and with radius equal to the specified extent e. An example Spiral Search task is shown in Figure 9.

E. Sector Search

The *Sector Search* task is motivated by the need to image a target from many different view angles. Task parameters are

TABLE IV

PARAMETERS FOR THE SPIRAL SEARCH TASK

Name	Description
Point p	The last known position of a target.
Extent e	The radius of the search area.
Lane Spacing Factor α	The distance between points on the spiral that lie at the same angle from the center is αd_w. Default = 0.9.

given in Table V, with an example search shown in Figure 11. At a high level, a Sector Search iteratively approaches a target at point p from a standoff distance determined by the search extent e, changing the angle of approach for each pass by a fixed offset at every iteration. Approaches are made until the circular region of radius e around p has been imaged. Given that the circumference of the region is $2\pi e$, the approximate number of passes needed is $\frac{2\pi e}{\alpha d_w}$. Then amount to offset the angle of approach between passes is $360°(\frac{\alpha d_w}{2\pi e} - \frac{1}{2})$, where the default of $\alpha = 0.9$ results in full coverage of the region. Steps for computing the path for a Sector Search task are as follows:

1) Set the heading for the first approach to $90°$, the point for the starting configuration such that the center of the leading edge of the sensor footprint is on closest point of the extent boundary around p at that heading, and the point for the ending configuration such that the leading edge of the sensor footprint is on the farthest point of the extent boundary at that heading. In addition, create an "intermediate" point at two times the UAV's turn radius below the starting configuration with heading $270°$. The corresponding intermediate, starting, and ending configurations are then

$$\begin{aligned}\nu_i(1) &= (p + (-2r, -e - d_l, z_p), 270°)\\ \nu_s(1) &= (p + (0, -e - d_l, z_p), 90°)\\ \nu_e(1) &= (p + (0, e - d_l, z_p), 90°),\end{aligned}$$

   where the goal of the "intermediate" configuration is to enforce a consistent shape for each pass.
2) For each pass $i > 1$, draw a line l from $\nu_e(i-1)$ at angle $\chi(i-1) - 360°(\frac{\alpha d_w}{2\pi e})$ and a parallel line l' with points at distance $2r$ and $90°$ clockwise from points on l. Set the heading for $\nu_s(i)$ and $\nu_e(i)$ to $\chi(i) = \chi(i-1) - 360°(\frac{\alpha d_w}{2\pi e} - \frac{1}{2})$, place the point for the starting configuration on l' such that the leading edge of the sensor footprint is on the closest point of the extent boundary along this heading, and place the point for the ending configuration such that the leading edge of the sensor footprint is on the farthest point. Set the heading of the intermediate configuration $\nu_i(i)$ to $\chi(i) - 180°$ and the point on l perpendicular to $\nu_s(i)$ on l'.
3) Repeat step 2 until $N \geq \frac{2\pi e}{\alpha d_w}$ passes are made.

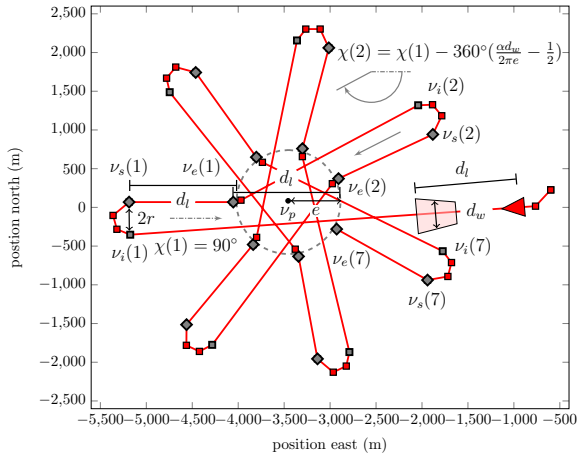


Fig. 10. A Sector Search task for point p with an extent $e = 350$ m. Note that in this case, the leading edge of the sensor footprint d_l is approximately equal to $2e$, so that approaches end near the extent boundary.

TABLE V
PARAMETERS FOR THE SECTOR SEARCH TASK

Name	Description
Point p	The position of the target of interest.
Extent e	The radial distance to extend the search with respect to p.
Lane Spacing Factor α	Spacing between arms of the sector search is a function of αd_w. Default = 0.9.

IV. PROCESS ALGEBRA FOR TASK-BASED MISSION PLANNING

Just as each task described in Section III provides a fast and flexible way to quickly accomplish a single search objective, it is useful to have a similarly fast and flexible way to compose tasks into a mission plan. A major advantage of these tasks is that they can easily be used in conjunction with a wide variety of planning approaches. Here, we demonstrate an approach based on process algebra [12].

Suppose a and b are two search tasks. Process algebra allows one to specify the relationship between task assignments in a mission plan through three composition operators: $a + b$, $a \cdot b$, and $a \parallel b$, which are called the *alternative*, *sequential*, and *parallel* compositions, respectively. The mission plan formed by alternatively composing $a + b$ assigns either task a or task b, but not both; the sequential composition $a \cdot b$ assigns task a followed by task b; and the parallel composition $a \parallel b$ assigns tasks a and b in an interleaved manner, i.e. it assigns both task a and task b but in no particular order. A process algebra specification then describes all possible ways in which tasks can be assigned. We emphasize that in this work, assignment refers to the allocation of a task to a UAV, and that the order in which tasks are allocated are not necessarily the same as the order in which they are actually executed during the mission if they are assigned to different UAVs. For example, let us annotate a task a with a subscript ui if it is eligible to be performed by UAV i. Then for instance, if tasks a and b could both be performed by either UAV $u1$ or $u2$ and we want to assign task a before task b, we could specify this through the process algebra string $(a_{u1} + a_{u2}) \cdot (b_{u1} + b_{u2})$. Task a could then be assigned to UAV 1, followed by task b being assigned to UAV 2. However, if UAV 2 is closer to task b, task b might actually start and complete first.

Recall that most tasks have multiple options. For instance, the Point Inspect, Line Search, and Area Search tasks may each be associated with several possible angles. If our mission includes multiple UAVs, then each task could also be associated with a list of eligible vehicles. Suppose we have an Area Search task $AS50$, with three eligible UAVs $u1$, $u2$, and $u3$ and 6 possible sweep angles $\chi = [0°, 30°, \ldots, 150°]$. Then we could represent these options in process algebra as

$$AS50 = (AS50_{u1,0} + AS50_{u2,0} + AS50_{u3,0} + \ldots + AS50_{u1,150} + AS50_{u2,150} + AS50_{u3,150}).$$

We can similarly express the options for several Point Search tasks $PS10$, $PS11$, $PS12$, and $PS13$; Line Search tasks $L10$, $L20$, $L30$; and a Sector Search task $SS61$. If we do not have any strict constraints on the order in which these tasks are assigned, we can write a mission specification in process algebra using the $\parallel$ composition as

$$PS10 \parallel PS11 \parallel PS12 \parallel PS13 \parallel L10 \parallel L20 \parallel L30 \parallel AS50 \parallel SS61. \quad (4)$$

The set of possible task assignments permitted by (4) can be represented as a tree [12]. We then use a depth-first search with a limited search depth after the first found solution to generate mission plans that meet additional constraints beyond those encoded in the process algebra string, e.g. that keep task paths within "Keep In" zones and out of "Keep Out" zones, and that minimize a cost function, e.g. maximum distance traveled by any of the UAVs. A solution for this example is shown in Figure 11, whose computation time is on the order of a few seconds.

V. CONCLUSIONS

In this paper, we have described the development of a set of UAV search tasks that automate planning of waypoint-based paths and sensor steering commands for common search and surveillance patterns, taking into account factors such as dynamics of the UAV, geometry of the sensor footprint, and quality of the sensor imagery. Each task has certain parameters that can be used to tailor the generated path, e.g. to use a particular angle of approach, view angle, standoff distance, shape, or search extent. These search tasks provide a level of automation that is higher than that found in most UAV ground control stations, which will guide a UAV between waypoints but generally require users to generate waypoint-based flight plans themselves.

We briefly mention that development of a user-friendly interface to provide fast and easy access to these tasks is underway [13]. The general concept starts with an interface

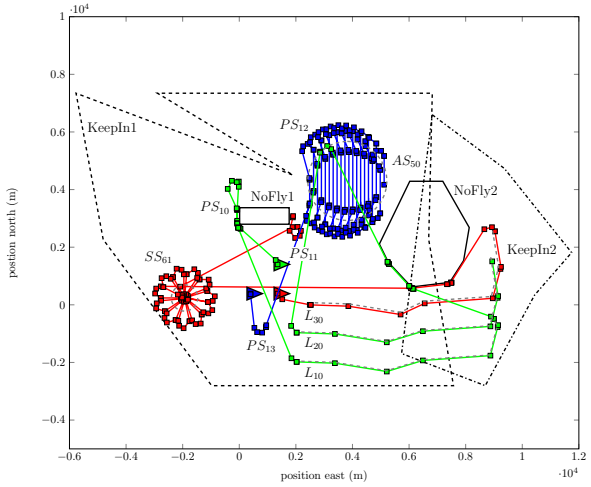


Fig. 11. A mission plan derived from a process algebra string. The solution algorithm attempts to minimize the maximum distance traveled by the UAVs while constraining paths to remain inside the "Keep In" zones and stay out of the "Keep Out" zones.

similar to those found on standard UAV ground control stations – one that shows an overhead view of the map, current positions of the UAVs, planned UAV paths, waypoints, labeled zones, etc. – and adds icons for each task so that a user can select a task along with a point, line, or area to search and a UAV to execute it. Additional menus provide text fields for entering specific parameters for the tasks if desired.

In the future, we plan to implement and test additional tasks. Consider a task for tracking a moving target. Though it was not discussed in Section III, we have developed a task to track a target using a desired view angle and standoff distance that works by placing a continuously updated waypoint at the appropriate point relative to the target. However, this implementation does not account for certain factors, e.g. differences in speeds between the two vehicles, which can lead to undesirable curves in the UAV's path; more sophisticated approaches exist that would have better performance [14]. Similarly, we have developed a task that uses one UAV as a relay to extend the effective communication range of another; however, the current implementation is based on the assumption that the communication range can be simply modeled as a circle with a particular radius around each UAV. This is not an accurate assumption in practice, and a more realistic approach is needed [15]. Other multi-vehicle tasks could also be developed, e.g. to implement formation flying [16].

We are also interested in using these tasks as a basis for higher levels of autonomy. As an example, we are working to design slightly higher-level automated tasks that are reactive, i.e. they can switch between lower-level tasks in reponse to specific events events. For instance, a higher-level automated task might switch from a search task to a target tracking task when a moving target is detected in the sensor footprint [17]. These tasks have also been used in conjunction with an "Intelligent Agent" that chooses which UAV should be assigned to a task by considering factors such as a UAV's sensor type, stealth level, speed, and fuel efficiency in the context of the needs of the mission [14]. These tasks could be used in a similar manner by any number of classic planning and scheduling approaches to implement higher levels of autonomy [18].

References

[1] C. B. Thomas, "U.S. coast guard addendum to the U.S. NSS to the IAMSAR," United States Coast Guard, Tech. Rep. COMDTINST M16130.2F, 2013.

[2] J. A. Adams, C. M. Humphrey, M. A. Goodrich, J. L. Cooper, B. S. Morse, C. Engh, and N. Rasmussen, "Cognitive task analysis for developing unmanned aerial vehicle wilderness search support," *J. Cognitive Eng. and Decision Making*, vol. 3, no. 1, 2009.

[3] D. Casbeer, D. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *Int. J. Syst. Science*, vol. 37, no. 6, pp. 351–360, 2006.

[4] N. J. Cooke and H. K. Pedersen, "Unmanned aerial vehicles," in *Handbook of Aviation Human Factors*. CRC Press, 2009.

[5] M. L. Cummings, S. Bruni, S. Mercier, and P. J. Mitchell, "Automation architecture for single operator, multiple UAV command and control," *Inter. C2 J.*, vol. 1, no. 2, 2007.

[6] R. Parasuraman and C. D. Wickens, "Humans: Still vital after all these years of automation," *Human Factors: J. Human Factors and Ergonomics Society*, vol. 50, no. 3, pp. 511–520, 2008.

[7] C. A. Miller, M. Draper, J. D. Hammell, G. Calhoun, T. Barry, and H. Ruff, "Enabling dynamic delegation interactions with multiple unmanned vehicles; flexibility from top to bottom," in *Engineering Psychology and Cognitive Ergonomics: Applications and Services*. Spring Berlin Heidelberg, 2013.

[8] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: a survey," *Int. J. Control, Automation and Systems*, vol. 8, no. 1, pp. 36 – 44, 2010.

[9] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.

[10] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American J. Mathematics*, vol. 79, pp. 497–516, 1957.

[11] D. Kingston, "Road surveillance using a team of small UAVs," in *Proc. SPIE Defense, Security, and Sensing*, 2009.

[12] S. Karaman, S. Rasmussen, D. Kingston, and E. Frazzoli, "Specification and planning of UAV missions: a process algebra approach," in *American Control Conf.*, 2009.

[13] J. E. Mercado, M. A. Rupp, J. Chen, M. J. Barnes, D. Barber, and K. Procci, "Intelligent agent transparency in human-agent teaming for multi-UxV management," *Human Factors*, vol. 58, no. 3, pp. 401–415, 2016.

[14] R. Anderson and D. Milutinović, "A stochastic approach to Dubins vehicle tracking problems," *IEEE Trans. Automatic Control*, vol. 59, no. 10, pp. 2801–2806, 2014.

[15] P. Zhan, K. Yu, and A. L. Swindlehurst, "Wireless relay communications with unmanned aerial vehicles: Performance and optimization," *IEEE Trans. Aerospace and Electronic Syst.*, vol. 47, no. 3, pp. 2068–2085, 2011.

[16] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick, "An overview of emerging results in cooperative UAV control," in *IEEE 43rd Annu. Conf. Decision and Control*, 2004.

[17] T. Apker, B. Johnson, and L. Humphrey, "LTL templates for play-calling supervisory control," in *Proc. AIAA SciTech Conf.*, 2016.

[18] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: Theory & practice*. Elsevier, 2004.