2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Optimal Multi-Robot Coverage Path Planning for Agricultural Fields using Motion Dynamics

Jahid Chowdhury Choton and Pavithra Prabhakar
Kansas State University, Manhattan, Kansas, USA

***Abstract*— Coverage path planning (CPP) is the task of computing an optimal path within a region to completely scan or survey the area of interest by using robotic sensor footprints. In this work, we propose a novel approach to find the multi-robot optimal coverage path of an agricultural field using motion dynamics while minimizing the mission time. Our approach consists of three steps: (i) divide the agricultural field into convex polygonal areas to optimally distribute them among the robots, (ii) generate an optimal coverage path to ensure minimum coverage time for each of the polygonal areas, and (iii) generate the trajectory for each coverage path using Dubins motion dynamics. Several experiments and simulations were performed to check the validity and feasibility of our approach, and the results and limitations are discussed.**

I. Introduction

Multi-robot coverage path planning (CPP) algorithms are used to cover an area of interest using sensor footprints of a group of mobile robots such as aerial, ground, underwater, or hybrid robots. Each type of robot has different motion dynamics that leads to different trajectories for the coverage. In this paper, we formulate the problem of optimal coverage path planning (CPP) that takes into account the robot motion dynamics. Our proposed solution is divided into three steps. The first step consists of the division of the agricultural field into several convex polygonal cells and their assignments to the robots so that each robot is assigned approximately the same area to cover. We used trapezoidal decomposition [10] to generate the polygonal cells and developed a dynamic programming algorithm to assign them to the robots. The second step is to generate an optimal coverage path for each robot to cover its assigned polygonal cells. The third and last step is the trajectory generation of each robot using the motion dynamics and computing the control input for the trajectory. We used Dubins Path [12] to calculate the inputs to generate the trajectories for the robots. We performed several experiments with different polygonal shapes representing the agricultural field as input and report the observations in terms of optimality and feasibility by varying different parameters.

II. Related Works

Coverage path planning (CPP) using a single robot has been vastly developed in the last two decades [4], [5], [11], [17]–[19], [21], [27], [29], [31], [34], [36], [38], [39]. But multi-robot CPP is still a relatively new topic to explore [1], [2], [7]–[9], [14], [16], [22], [24], [25], [28], [32]. The multi-robot trajectory generation [2], [23] and coverage path planning (CPP) [3], [14], [16], [25], [28] have become very popular in recent times due to the growing need of human alternatives while working in remote and inaccessible regions such as large agricultural fields [31], vast ocean [8], areas with wildfire [32], and so on. The dynamics and trajectory generation of a single robot have also been extensively studied [13], [17], [30], [37]. However, most of the works on multi-robot CPP do not consider the motion dynamics while generating the optimal path. In [17], the authors developed a software tool to determine an optimal coverage path using the Dubins path for a single robot. But our work consists of finding optimal trajectories for multiple robots to ensure optimal coverage. Several classifications of recent works in CPP are described in Table I based on robot types, algorithms, decompositions, and so on. It can be clearly seen from Table I that there is not much work done in coverage path planning while considering the motion dynamics [14], [17]. Our work addresses this issue and defines the CPP problem using the dynamics of mobile robots.

TABLE I
Recent works on CPP

Category	Types	Reference
Number of robots	Single robot	[4], [5], [11], [17]–[19], [21], [27], [29], [31], [34], [36], [38], [39]
	Multi-robot	[1], [2], [7]–[9], [14], [16], [22], [24], [25], [28], [32]
Algorithm	Optimal	[2], [8], [11], [17], [19], [28], [38], [39]
	Heuristic	[4], [24], [25], [31], [38]
Decomposition	Exact	[11], [19], [21], [36]
	Others	[1], [8], [18]
Dynamics	Dubins Path	[17]
	Robot dynamics	[14]
	Cubic Spline	[29]
Dimensions	2D	All other papers
	3D	[2], [22], [24], [25], [27], [36]
Aerial Imaging	2D and 3D	[5], [11], [29], [36]
Environment	Known	All other papers
	Unknown	[1], [4], [9], [16], [18], [26], [32], [34]
Survey Papers	All types	[3], [6], [15], [35]

III. Preliminaries

Let $\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{R}^+$ denote the set of natural numbers, integers, reals, and positive reals, respectively. For any $r \in \mathbb{R}$, $\|r\|$, $\lfloor r \rfloor$, and $\lceil r \rceil$ are defined as the absolute value, floor and ceiling of r respectively. The number of elements of a set S is denoted as $|S| \in \mathbb{N}$. A point in n-dimensional space is a tuple $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ where $x_1, \ldots, x_n \in \mathbb{R}$.

Definition 1: Given a finite set of n points $S =$

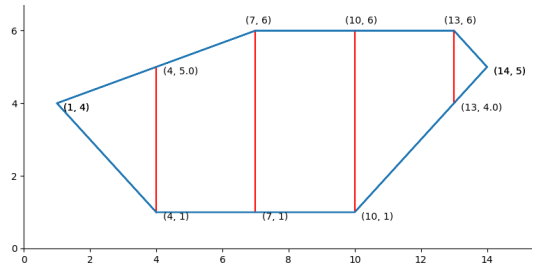


Fig. 1. Trapezoidal decomposition of a convex polygon. For each vertex, a vertical line is drawn inside the polygon which intersects one of its edges. This will divide the polygon into several trapezoids and triangles.

$\{v_1, v_2, \ldots, v_n\}$, the convex hull of set S is defined as,

$$\mathrm{H}(S) = \{\sum_{i=1}^{n} \alpha_i v_i | \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i = 1\}$$

The convex hull of two points v and v' will be written as $\mathrm{H}(v, v')$ instead of $\mathrm{H}(\{v, v'\})$. A set $P \subset \mathbb{R}^n$ is a convex polyhedron if there exists a finite set of points V (set of vertices) such that, $P = \mathrm{H}(V)$. Here P is specified using the set of vertices V. The Euclidean distance between two points $v_1 = (x_1, \ldots, x_n)$ and $v_2 = (y_1, \ldots, y_n)$ is defined as, $d(v_1, v_2) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_n - y_n)^2}$. Given a point $v \in \mathbb{R}^n$ and a real number $\delta > 0$, the ball around v is defined as,

$$\mathrm{B}_\delta(v) = \{v' \in \mathbb{R}^n | d(v, v') \leq \delta\}$$

The open ball for a set of points S is defined as, $\mathrm{B}_\delta(S) = \bigcup_{v \in S} \mathrm{B}_\delta(v)$. A sequence of finite elements is denoted as ρ. The i^{th} element of ρ is denoted as $\rho[i]$ and its length as $|\rho| \in \mathbb{N}$. We define a dynamical system and its solution as below,

Definition 2: A dynamical system is a function $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ which models $\dot{x}(t) = f(x(t), u(t))$. Given an input signal $u : [0, \mathrm{T}] \mapsto \mathbb{R}^m$, initial state $\phi_1 \in \mathbb{R}^n$, and dynamical system $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$, the solution of f is a trajectory $\phi : [0, \mathrm{T}] \mapsto \mathbb{R}^n$ that satisfies,

$$\phi(0) = \phi_1, \quad \frac{d}{dt}(\phi(t)) = f(\phi(t), u(t)), \quad \forall t \in [0, T]$$

IV. Problem Formulation

We consider a polygonal field $F \subset \mathbb{R}^2$ that is specified by the set of its vertices V such that $F = \mathrm{H}(V)$. We assume that the sensor footprint of a robot at v is $\mathrm{B}_\delta(v)$ for a given δ, that is when a robot arrives at a point $v \in \mathbb{R}^2$, the set $\mathrm{B}_\delta(v)$ is said to be covered. We are given k robots that need to collectively cover F, that is, if we consider the union of the footprints of the trajectories of the k-robots, that should include all of F. Finally, we would like to cover the field in minimum time.

Problem 1: Given k robots with dynamical systems $f_1, \ldots, f_k$, coverage parameter δ, and a field $F \subseteq \mathbb{R}^2$, compute the inputs $u_i : [0, \mathrm{T}] \to \mathbb{R}$ such that the solutions of the dynamical systems $\phi_i : [0, T] \mapsto \mathbb{R}^3$ for $i \in \{1, \ldots, k\}$ satisfies $F \subseteq \bigcup_{i=1}^{k} \bigcup_t \mathrm{B}_\delta(\phi(t))$ and time t is minimized.

V. Solution Approach

As mentioned above, our approach to solving this problem consists of three steps as shown in Fig. 3. In the first step, we take field information and the number of robots as inputs and decompose the field into trapezoidal cells. The trapezoidal cells are merged into segmented cells using dynamic programming to distribute them among the robots. Next, we compute the optimal coverage path for each robot using the coverage parameter δ (for footprint). Finally, we compute the input signals to generate the trajectories for the optimal coverage paths using motion dynamics.

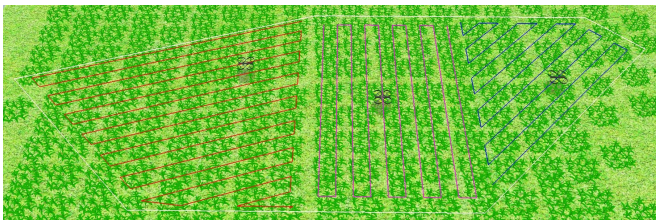


Fig. 2. Simulation of 3 robots in a corn field using CoppeliaSim simulator [33]. The area bounded by the white lines are the area of interest. The red, purple and blue lines are the coverage paths of the robots.

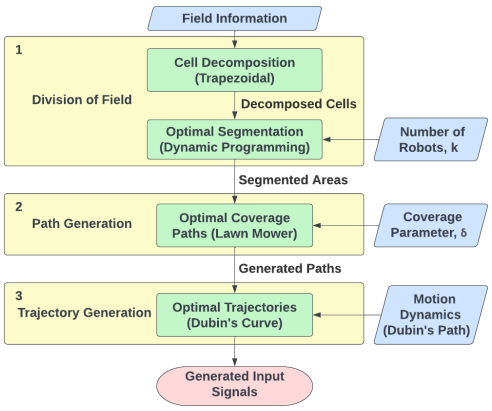


Fig. 3. Framework for the 3 steps of coverage path planning (CPP) using robot dynamics.

We describe our approach by providing an example. Consider a field F with vertices $(10, 1)$, $(14, 5)$, $(13, 6)$, $(7, 6)$, $(1, 4)$, $(4, 1)$; and 3 robots with Dubins dynamics [12] and coverage parameter $\delta = 0.2$. First, we use trapezoidal decomposition [10] to decompose the field as shown in Fig. 1, and compute the sequence of areas of the trapezoidal cells (here the area of the cells are 6, 13.5, 15, 10.5 and 1 units respectfully). We use a dynamic programming algorithm to merge the cells into polygons and distribute them among the 3 robots. Here, one robot gets the first two cells with total area of 19.5 units, another robot gets the next cell with total area of 15 units, and the last robot gets the last two cells with area 11 units. Next, we compute the optimal coverage path for each of the robots using a lawn-mower coverage algorithm [19]. Finally, we compute the input signals to generate the trajectories for each of the 3 robots. The complete coverage paths for the 3 robots are generated using the three steps and illustrated in Fig. 2. We describe each of the three steps in detail in the following sections.

A. Division of field

Our first task is to divide the field into contiguous subfields to be assigned to the robots. Toward this, first, we use trapezoidal decomposition [10] to divide the field F into several trapezoidal cells as shown in Fig. 1. Each trapezoidal cell is either a trapezoid, or a triangle (degenerate trapezoid). We order the trapezoidal cells in a sequence from left to right. We need to assign each robot a contiguous fragment of this sequence such that the sum of the areas of the trapezoids in the fragment is approximately the same for all robots. Before formally writing this problem, we describe several definitions.

A segment $\iota = [m : n]$ refers to the sequence of number $m, m+1, \ldots, n$, where $m, n \in \mathbb{N}$ and $m \leq n$. We denote the starting index m of ι by $\underline{\iota}$ and the ending index n by $\bar{\iota}$. Given a segment $\iota = [m : n]$ and an integer $k \in \mathbb{N}$ where $1 \leq k \leq n$, a k-segmentation of ι is a sequence $\kappa = \iota_1, \ldots, \iota_k$ such that $\underline{\iota_1} = m$, $\bar{\iota_k} = n$ and for all $i = 1, \ldots, k-1$ we have $\bar{\iota_i} + 1 = \underline{\iota_{i+1}}$. The set of all k-segmentations of $[1 : n]$ is denoted as $\mathrm{Seg}(n, k)$. Given a $k-1$-segmentation $\kappa' = \iota_1, \ldots, \iota_{k-1}$ in $\mathrm{Seg}(l, k-1)$ and a segment $[l : n]$, $\kappa'.[l : n]$ refers to the k-segmentation $\iota_1, \ldots, \iota_{k-1}, [l : n]$. We can overload the definition of $\kappa'.[l : n]$ to $S.[l : n]$ for any subset S of $\mathrm{Seg}(l, k-1)$ in the standard manner.

Let $\rho = a_1, a_2, \ldots, a_n$ be a sequence of n elements, where $\rho[i]$ refers to the i^{th} element a_i for any $i = 1, \ldots, n$. The index segment for a sequence ρ is defined as $[1 : n]$ where $|\rho| = n$. Whenever we refer to segments/segmentation of ρ, we refer to segments/segmentation of $[1 : n]$. Given a segment ι of a sequence ρ, where $\bar{\iota} \leq |\rho|$, the cost of the segment is defined as, $\mathrm{Cost}(\rho, \iota) = \sum_{i=\underline{\iota}}^{\bar{\iota}} \rho[i]$. The cost of k-segmentation κ of ρ for some $M \in \mathbb{R}^+$ is defined as,

$$\mathrm{Cost}(\rho, \kappa, M) = \max_{\iota \in \kappa}(\| \mathrm{Cost}(\rho, \iota) - M \|)$$

Let ρ denote the sequence of areas of the trapezoids and k denote the number of robots. Let $\mathrm{MeanArea}(\rho, k)$ denote the average area per robot, that is, $\sum_{i=1}^{|\rho|} \rho[i]/k$. For each segmentation $\mathrm{Cost}(\rho, \kappa, \mathrm{MeanArea}(\rho, k))$ denotes the maximum "deviation" of the area assigned to a robot from the average. Our objective is to minimize this deviation so that all the robots finish the task at the same time. Hence, the mathematical problem corresponding to the assignment of trapezoids to the robots can be formulated as follows:

Problem 2: Given a finite sequence ρ and a parameter k, find a k-segmentation κ such that $\mathrm{Cost}(\rho, \kappa, \mathrm{MeanArea}(\rho, k))$ is minimized.

We solve this problem using a dynamic programming approach. Hence, we first define the sub-problem of calculating the optimal cost of the first i elements of ρ as follows. Let us fix a ρ, k and a parameter $M = \mathrm{MeanArea}(\rho, k)$ for the rest of the section.

Definition 3: Given $1 \leq i, j \leq |\rho|$, the optimal cost for a j-segmentation of $[1 : i]$ of ρ, denoted $\mathrm{Opt}_\rho(i, j, M)$ is defined as,

$$\mathrm{Opt}_\rho(i, j, M) = \min_{\kappa \in \mathrm{Seg}(i,j)} \mathrm{Cost}(\rho, \kappa, M)$$

Note that if we compute $\mathrm{Opt}_\rho(i, j, M)$ for every i, j and the corresponding κ that achieves the minimum, then we have solved our problem, since, we are interested in $\mathrm{Opt}_\rho(n, k, M)$ and the corresponding κ. In the rest of the section, we provide the recursive definition of Opt_ρ and show its correctness, and how to compute $\mathrm{Opt}_\rho(n, k, M)$ and the corresponding κ.

Recall that segments in $\mathrm{Seg}(n, k)$ can be considered as a $k-1$ segmentation followed by a last segment. Using this insight, we obtain the following proposition:

Proposition 1: For a sequence ρ with index segment $[1 : n]$ and an integer $1 \leq k \leq n$ where $k, n \in \mathbb{N}$ we have,

$$\mathrm{Seg}(n, k) = \begin{cases} \bigcup_{l=k}^{n} \mathrm{Seg}(l-1, k-1).[l : n], & \text{for } k > 1 \\ [1 : n], & \text{for } k = 1 \end{cases}$$

Proposition 2: Let $\mathcal{A}$ be a set where each element $A \in \mathcal{A}$ is a set of real numbers, and $b \in \mathbb{R}^+$. Then we have,

$$\min_{A \in \mathcal{A}}(\max(A \cup \{b\})) = \max(\min_{A \in \mathcal{A}} \max\{A\}, b)$$

The proof of the two propositions are omitted here due to lack of space. Now we describe the following lemma for solving the sub-problem,

Lemma 1: The recursive solution to calculate the optimal cost of the sequence ρ with i elements and j-segmentation is,

$$\mathrm{Opt}_\rho(i, j, M) = \min_{j \leq l \leq i}(\max(\mathrm{Opt}_\rho(l-1, j-1, M), \| \mathrm{Cost}(\rho, [l : i]) - M \|))$$

Proof: Here,

$\mathrm{Opt}_\rho(i, j, M)$

$= \min_{\kappa \in \mathrm{Seg}(i,j)}(\mathrm{Cost}(\rho, \kappa, M))$ [Def. of $\mathrm{Opt}_\rho(i, j, M)$]

$= \min\{\mathrm{Cost}(\rho, \kappa, M) \mid \kappa \in \cup_{l=j}^{i} \mathrm{Seg}(l-1, k-1)[l : i]\}$ [From Prop. 1]

$= \min_{j \leq l \leq i}(\min\{\mathrm{Cost}(\rho, \kappa, M) | \kappa \in \mathrm{Seg}(l-1, j-1)[l : i]\})$ [Property of min]

$= \min_{j \leq l \leq i}(\min\{\mathrm{Cost}(\rho, \kappa'[l : i], M) \mid \kappa'[l : i] \in \mathrm{Seg}(l-1, j-1)[l : i]\})$ [$\kappa'[l : i] = \kappa$]

$= \min_{j \leq l \leq i}(\min\{\max_{\iota \in \kappa'[l:i]}(\| \mathrm{Cost}(\rho, \iota) - M \|) \mid \kappa'[l : i] \in \mathrm{Seg}(l-1, j-1)[l : i]\})$ [Def. of $\mathrm{Cost}(\rho, \kappa'[l : i], M)$]

$= \min_{j \leq l \leq i}(\min\{\max(\max_{\iota' \in \kappa'}(\| \mathrm{Cost}(\rho, \iota') - M \|), \| \mathrm{Cost}(\rho, [l : i]) - M \|) \mid \kappa'[l : i] \in \mathrm{Seg}(l-1, j-1)[l : i]\})$ [Separating $[l : i]$ from $\kappa'[l : i]$]

$= \min_{j \leq l \leq i}(\min\{\max(\mathrm{Cost}(\rho, \kappa', M), \| \mathrm{Cost}(\rho, [l : i]) - M \|) \mid \kappa'[l : i] \in \mathrm{Seg}(l-1, j-1)[l : i]\})$ [Def. of $\mathrm{Cost}(\rho, \kappa', M)$]

$= \min_{j \leq l \leq i}(\max(\min\{\mathrm{Cost}(\rho, \kappa', M) \mid \kappa' \in \mathrm{Seg}(l-1, j-1)\}), \| \mathrm{Cost}(\rho, [l : i]) - M \|)$ [From Prop. 2]]

$= \min_{j \leq l \leq i}(\max(\mathrm{Opt}_\rho(l-1, j-1, M), \| \mathrm{Cost}(\rho, [l : i]) - M \|))$ [Def. of $\mathrm{Opt}_\rho(l-1, j-1, M)$]

Let ρ be the sequence of the areas of the trapezoidal

Algorithm 1 Path Generation

Input: A convex polygon C, coverage parameter $\delta > 0$
Output: A coverage path μ
$\mu' \leftarrow$ empty sequence
$e_p \leftarrow \text{SWEEPDIR}(C)$
$L \leftarrow \text{SWEEPLINES}(C, e_p, \delta)$
$L' \leftarrow \text{TRIMLINES}(L, \delta)$
$count = 0$
for each line in L' **do**
  if $count$ is even **then**
    Sort the endpoints based on x values
  else
    Reverse sort the endpoints based on x values
  end if
  Add the sorted endpoints in μ'
  $count \leftarrow count + 1$
end for
$\alpha \leftarrow \text{TRANSFORMANGLE}(e_p)$
$\mu'' \leftarrow \text{TRANSFORM}(\mu', \alpha)$
$\mu \leftarrow \text{ZIG-ZAG}(\mu'')$
return μ

cells of the field F from left to right. We can now compute $\text{Opt}_\rho(n, k, M)$ using the standard dynamic programming algorithm, since $\text{Opt}_\rho(i, j, M)$ only requires computing Opt_ρ for smaller i and j. Further, the κ, the achieves the minimum can be tracked during the solution. We now have a contiguous segment of trapezoids assigned to each robot. In other words, each robot is assigned a polygonal region corresponding to the segment of trapezoids. In the next section, we solve the problem of generating paths for these polygons.

B. Path Generation

In this section, we first describe the problem of generating an optimal coverage path for a convex polygon C. A path μ is a sequence of points $v_1, v_2, ..., v_n$ where $v_i = (x_i, y_i)$ for $1 \leq i \leq n$.

Problem 3: Given a convex polygon C and a coverage parameter $\delta > 0$, generate a path μ such that,

$$C \subseteq \text{B}_\delta(\text{CPath}(\mu))$$

where,

$$\text{CPath}(\mu) = \bigcup_{i=1}^{n-1} \text{H}(\mu[i], \mu[i+1])$$

The optimal lawn-mower coverage problem has been explored before [19], [36]. We summarize Algorithm 1 [36] to generate the path μ to cover the convex polygon C. The function $\text{SWEEPDIR}(C)$ calculates the distance from every vertex to the farthest edge of the polygon, and returns the edge e_p with the shortest such distance which indicates the sweep direction. The $\text{SWEEPLINES}(C, e_p, \delta)$ generate lines that are parallel to e_p within the polygon C with a distance of δ between them and adds them to L in the order of increasing distance to e_p. The $\text{TRIMLINES}(L, \delta)$ trims the endpoints of every line in L by δ along the line and adds it to L'. In Fig. 4, e_2 is the edge with the shortest distance from the farthest vertex v_1 that is stored as e_p. The $\text{SWEEPLINES}(C, e_p, \delta)$ generates the sweeplines $a_1b_1, a_2b_2, a_3b_3, a_4b_4, \ldots$ and appends them in L. The $\text{TRIMLINES}(L, \delta)$ trims the sweeplines by δ into $a'_1b'_1, a'_2b'_2, a'_3b'_3, a'_4b'_4, \ldots$ and appends them in L'. The $\text{TRANSFORMANGLE}(e_p)$ returns the angle α between the edge e_p and the x-axis. Finally, we transform the sequence μ' into a path μ'' by shifting all the points by α degrees such that the sweeplines are parallel with the y-axis in a reference frame (different from the world frame for polygon C). In the final step, we transform μ'' to a zig-zag path μ, by making the y-values the same for the segment that joins two sweeplines. That is, if the point after the turn is "further" away than that before the turn, then we extend the current line. Otherwise, we extend the next line. We formally define a zig-zag path as follows,

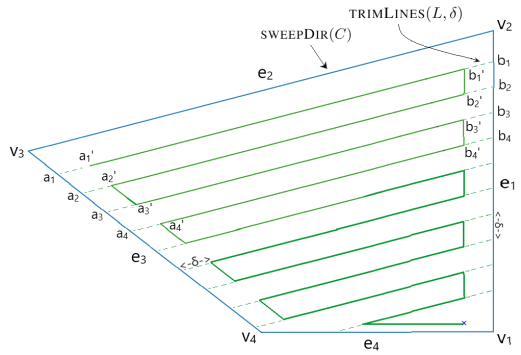


Fig. 4. The sweep lines for optimal coverage path $a_1b_1, a_2b_2, a_3b_3, a_4b_4$ are parallel with the edge e_2 of the polygon. The lines $a'_1b'_1, a'_2b'_2, a'_3b'_3$ and $a'_4b'_4$ are trimmed by δ from the sweep lines.

Definition 4: A path $\mu = (x_1, y_1), \ldots, (x_n, y_n)$ of length n is a *zig-zag* path if, for every pair of points $\mu[i] = (x_i, y_i)$ and $\mu[i+1] = (x_{i+1}, y_{i+1})$ for $i \in \{1, 2, \ldots, n-1\}$ satisfies the following conditions,

1) $x_i + \delta = x_{i+1}$ and $y_{i+1} = y_i$ iff i is even
2) $x_i = x_{i+1}$ and $y_i > y_{i+1}$ iff $i = 4j + 3, j \in \{0, 1, \ldots, \lfloor n/4 \rfloor\}$
3) $x_i = x_{i+1}$ and $y_i < y_{i+1}$ iff $i = 4j + 1, j \in \{0, 1, \ldots, \lfloor n/4 \rfloor\}$

Now we write the following theorem for coverage,

Theorem 1: Given a convex polygon C and a coverage parameter $\delta > 0$, the path μ generated by Algorithm 1 satisfies that μ is a zig-zag path and $C \subseteq \text{B}_\delta(\text{CPath}(\mu))$.

This theorem describes that for a given convex polygon C, Algorithm 1 generates a zig-zag path μ that ensures the coverage of C. The proof of Theorem 1 is omitted due to page limitations.

C. Trajectory Generation

In this section, we describe how to compute the trajectory for a coverage path μ when the robot follows the Dubins motion dynamics [12] as the dynamical system f. The state of the system is (x, y, θ) in $\mathbb{R}^3$, where (x, y) corresponds to the two-dimensional position of the robot and θ is the heading angle (anti-clockwise from y-axis). The input to the system u corresponds to the angular velocity, and v is a

constant corresponding to the linear velocity. The dynamics is given by:

$$\frac{d}{dt}((x,y,\theta)) = f((x,y,\theta),u) = (v\sin\theta, v\cos\theta, u);$$

Let the initial state at time $t = 0$ be (x_1, y_1, θ_1), where $x(0) = x_1$, $y(0) = y_1$ and $\theta(0) = \theta_1$. Hence, we define the position of a state $z = (x, y, \theta)$, denoted, $\mathrm{Pos}(z)$ to be (x, y). In the sequence, we fix the parameter δ and the dynamics f as above. We want to generate the input signal for the Dubin's motion dynamics such that the resulting trajectory closely follows μ. Our broad approach will be to generate a piece-wise constant input signal u to follow the μ. First, we define a piece-wise constant signal as being generated from a sequence of pairs of inputs and times.

Definition 5: Given an input sequence $\eta = (u_1, \tau_1), \ldots, (u_n, \tau_n)$ where u_i are inputs and τ_i are times, the corresponding input signal is $u_\eta : [0, \mathrm{T}] \mapsto \mathbb{R}^m$, where $\mathrm{T} = \sum_{j=1}^{n} \tau_j$ and $\forall t \in [0, \mathrm{T}] : u(t) = u_i$, such that $i = 1$ and $t < \tau$ or $i > 1$ and $\sum_{j=1}^{i-1} \tau_j \leq t < \sum_{j=1}^{i} \tau_j$.

We formally define the trajectory generation problem as follows:

Problem 4: Given a zig-zag path μ, coverage parameter δ, find an input sequence $\eta = (u_1, \tau_1), \ldots, (u_{n-1}, \tau_{n-1})$ such that solution $\phi : [0, \mathrm{T}] \mapsto \mathbb{R}^3$ of the Dubin's dynamics corresponding to input signal $u_\eta : [0, \mathrm{T}] \mapsto \mathbb{R}^m$ satisfies:

1) $\mathrm{Pos}(\phi(0)) = \mu[1]$ and $\mathrm{Pos}(\phi(t_i)) = \mu[i]$ where, $t_i = \sum_{j=1}^{i-1} \tau_j$ for all $i \in \{1, 2, \ldots, n\}$
2) $\mathrm{B}_\delta(\mathrm{CPath}(\mu)) \subseteq \mathrm{B}_\delta(\cup_{t=0}^{T} \mathrm{Pos}(\phi(t)))$

Next, we define the sequence η to solve the problem. Let the state at time t be (x_t, y_t, θ_t), and u be a constant signal in the interval $[t, t + \tau]$. The state at time $t + \tau$ is given by:

$$\theta_{t+\tau} = \begin{cases} \theta_t, & \text{for } u = 0 \\ \theta_t + u \times \tau, & \text{otherwise} \end{cases}$$

$$x_{t+\tau} = \begin{cases} x_t + v\sin\theta_t \times \tau, & \text{for } u = 0 \\ x_t + \frac{v}{u}(\cos\theta_t - \cos(\theta_t + u\tau)), & \text{otherwise} \end{cases}$$

$$y_{t+\tau} = \begin{cases} y_t + v\cos\theta_t \times \tau, & \text{for } u = 0 \\ y_t + \frac{v}{u}(\sin(\theta_t + u\tau) - \sin\theta_t), & \text{otherwise} \end{cases}$$

Consider the zig-zag path $\mu = (x_1, y_1), \ldots, (x_n, y_n)$. We assume that the robot starts at (x_1, y_1, θ_1), where $\theta_1 = 0$. We define sequence of pairs of inputs and times $\eta_\mu = (u_1, \tau_1), \ldots, (u_{n-1}, \tau_{n-1})$ as follows:

$$u_i = \begin{cases} \frac{2v}{\delta}, & \text{if } i = 4j+2,\ j \in \{0, 1, \ldots, \lfloor n/4 \rfloor\} \\ -\frac{2v}{\delta}, & \text{if } i = 4j,\ j \in \{0, 1, \ldots, \lfloor n/4 \rfloor\} \\ 0, & \text{otherwise} \end{cases}$$

$$\tau_i = \begin{cases} \frac{\|y_{i+1} - y_i\|}{v}, & \text{if } i \text{ is odd} \\ \frac{\pi\delta}{2v}, & \text{otherwise} \end{cases}$$

Finally, we have the following theorem for coverage,

Theorem 2: Given a zig-zag path μ, the solution $\phi : [0, \mathrm{T}] \mapsto \mathbb{R}^3$ of f with respect to u_{η_μ} starting from (x_1, y_1, θ_1), where $\mu[0] = (x_1, y_1)$ and $\theta_1 = 0$, satisfies:

1) For every $i \in \{1, 2, \ldots, n\}$, $\phi(t_i) = (x_i, y_i, \theta_i)$, where $t_i = \sum_{j=1}^{i-1} \tau_j$, $(x_i, y_i) = \mu[i]$, and

$$\theta_i = \begin{cases} 0, & \text{if } i = 4j+1 \text{ or } i = 4j+2 \\ \pi, & \text{if } i = 4j \text{ or } i = 4j+3 \end{cases}$$

where, $j \in \{0, 1, \ldots, \lfloor n/4 \rfloor\}$

2) $$\mathrm{B}_\delta(\mathrm{CPath}(\mu)) \subseteq \mathrm{B}_\delta(\cup_{t=0}^{T} \{\mathrm{Pos}(\phi(t))\}) \quad (1)$$

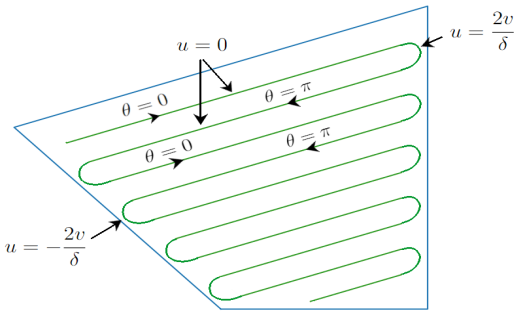


Fig. 5. The heading angle $\theta = 0$ for the odd parallel lines and $\theta = \pi$ for the even parallel lines. The input (angular velocity) $u = 0$ for going straight, $u = \frac{2v}{\delta}$ for turning right, and $u = -\frac{2v}{\delta}$ for turning left.

Recall that a zig-zag path μ is such that $\mu[i]$ to $\mu[i+1]$ is parallel to e_p when i is odd, and joins two adjacent sweeplines when i is even. So, for alternate odd is, that for $i = 4j+1$ and $4j+3$, $\mu[i]$ to $\mu[i+1]$ is parallel to e_p but in opposite directions. So, for $i = 4j+1$ and $4j+3$, u_i should correspond to no change in angular velocity, assuming the heading angle at $\mu[i]$ is along e_p, that is, $\theta_i = 0$ and $\theta_i = \pi$, respectively. Hence, we set $u_i = 0$, and $\tau_i = |y_{i+1} - y_i|/v$ (time taken = distance travelled divided by velocity). Further, this ensures that we reach $\mu[i+1]$ after time τ_i, and $\theta_{i+1} = \theta_i$, and the trajectory is going to follow the path exactly. When $i = 4j$ and $4j+2$, the segment $\mu[i]$ to $\mu[i+1]$ has the same y values, but the x values differ by δ. Further, we need to ensure that the heading angle change by π. The robot traverses a circular path (half-circle of diameter δ) of length $\pi\delta/2$ at a linear velocity v. Hence, it moves to the next line in time $\pi\delta/2v$. Since, the angular velocity u is constant in this interval, it is given by the total angle travelled π (or $-\pi$) divided by time for travel, that is $\pi/(\pi\delta/2v) = 2v/\delta$ (or $-2v/\delta$). Note that this again maintains the invariant that θ_{i+1} is either 0 or π. Further, it is easy to verify that the δ-ball around the curve joining $\mu[i]$ and $\mu[i+1]$ for even i covers the δ-ball around the line joining these points.

Note that each subfield of F assigned to a robot is covered by a path μ according to Theorem 1 which in turn is covered by a trajectory generated by η_μ according to Theorem 2. Hence, we get an input sequence/signal for each robot which covers the whole field. While we don't have optimality, the division of the field given by the dynamic programming algorithm gives a fairly good division as we validate in the experiments.

VI. Experiments

We performed several experiments and simulations to assess the performance and validate our approach in three different polygonal fields F_1, F_2, and F_3 using Windows

TABLE II

EXPERIMENTAL RESULTS FOR THREE POLYGONAL FIELDS F_1, F_2 AND F_3

Experiments of F_1 with vertices $(10,1),(14,5),(13,8),(7,10),(5,10)$ & $(1,2)$

Experiment 1 (Number of Robots, k=1)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[16.89, 17.11, 24.96, 18.0, 2.0]]	2122	499.38
	2	20			274.78
	3	30			199.8
	4	40			167.8
0.3	1	6.67		970	319.87
	2	13.33			182.03
	3	20			123.23
	4	26.67			106.2

Experiment 2 (Number of Robots, k=2)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[16.89, 17.11], [24.96, 18.0, 2.0]]	1227	287.05
	2	20			168.13
	3	30			117.6
	4	40			97.94
0.3	1	6.67		532	177.6
	2	13.33			99.64
	3	20			70.62
	4	26.67			58.62

Experiment 3 (Number of Robots, k=3)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[16.89, 17.11], [24.96], [18.0, 2.0]]	894	209.54
	2	20			126
	3	30			91.04
	4	40			74
0.3	1	6.67		437	148.12
	2	13.33			82.01
	3	20			60.93
	4	26.67			48.49

Experiment 4 (Number of Robots, k=4)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[16.89], [17.11], [24.96], [18.0, 2.0]]	693	164.86
	2	20			97.66
	3	30			74.83
	4	40			63.58
0.3	1	6.67		267	92.53
	2	13.33			50.33
	3	20			37.82
	4	26.67			29.97

Experiments of F_2 with vertices $(10,1),(14,4),(13,6),(7,9),(1,5)$ & $(4,1)$

Experiment 1 (Number of Robots, k=1)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[9.0, 21.0, 21.75, 13.875, 1.375]]	1807	422.18
	2	20			233.29
	3	30			171.86
	4	40			142.9
0.3	1	6.67		839	334.28
	2	13.33			157.17
	3	20			106.79
	4	26.67			92.23

Experiment 2 (Number of Robots, k=2)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[9.0, 21.0], [21.75, 13.875, 1.375]]	1016	238.14
	2	20			137.468
	3	30			98.51
	4	40			80.9
0.3	1	6.67		448	155.79
	2	13.33			84.07
	3	20			59
	4	26.67			49.45

Experiment 3 (Number of Robots, k=3)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[9.0, 21.0], [21.75], [13.875, 1.375]]	790	185.38
	2	20			111.3
	3	30			80.41
	4	40			64.28
0.3	1	6.67		390	133.56
	2	13.33			73.33
	3	20			52.7
	4	26.67			43.21

Experiment 4 (Number of Robots, k=4)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[9.0], [21.0], [21.75], [13.875, 1.375]]	606	143.54
	2	20			91.46
	3	30			64.12
	4	40			52.39
0.3	1	6.67		255	87.98
	2	13.33			48
	3	20			35.89
	4	26.67			28.328

Experiments of F_3 with vertices $(12,1),(14,4),(13,9),(7,8),(1,5)$ & $(4,1)$

Experiment 1 (Number of Robots, k=1)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[8.25, 18.75, 37.08, 7.167, 3.25]]	1984	464.56
	2	20			307.58
	3	30			189.146
	4	40			157.17
0.3	1	6.67		921	304.68
	2	13.33			169.25
	3	20			116.62
	4	26.67			101.51

Experiment 2 (Number of Robots, k=2)

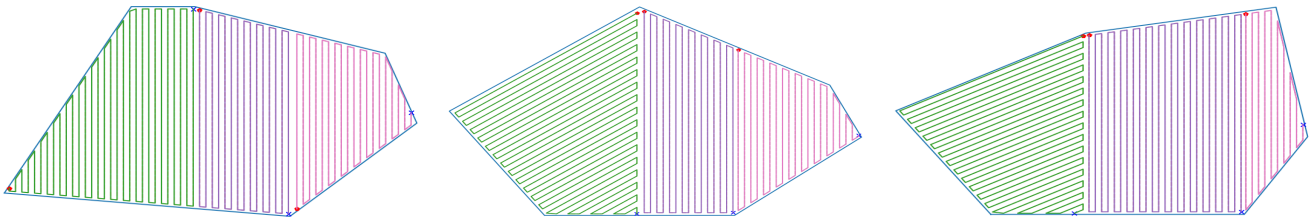
Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[8.25, 18.75], [37.086, 7.167, 3.25]]	1274	299.97
	2	20			168.24
	3	30			121.58
	4	40			102.18
0.3	1	6.67		564	186.21
	2	13.33			106.001
	3	20			73.919
	4	26.67			64.04

Experiment 3 (Number of Robots, k=3)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[8.25, 18.75], [37.086], [7.167, 3.25]]	1000	234.87
	2	20			139.84
	3	30			98.15
	4	40			83.25
0.3	1	6.67		423	143.49
	2	13.33			79.93
	3	20			56.84
	4	26.67			46.96

Experiment 4 (Number of Robots, k=4)

Coverage Parameter, δ	Average Velocity, vs	Input for Curve, u = +2v/δ	Optimal Area Segmentation	Max Path Length	Simulation Runtime (s)
0.2	1	10	[[8.25], [18.75], [37.086], [7.167, 3.25]]	1000	236.62
	2	20			141.92
	3	30			105.29
	4	40			86.31
0.3	1	6.67		423	143.68
	2	13.33			80.31
	3	20			59.57
	4	26.67			47



(a) (b) (c)

Fig. 6. Complete coverage paths for F_1, F_2 and F_3 are shown in different colors in the figures (a), (b) and (c) respectively for $k = 3$ and $\delta = 0.2$.

10 OS, Intel® Core™ i7-4870HQ CPU @2.50GHz, 16GB RAM. We used Matplotlib [20] and Turtle graphics modules in Python 3.10 for running the experiments and simulations. Four experiments were done for each field based on the number of robots, and the results are shown in Table II. We varied the coverage parameter δ from 0.2 to 0.3, and average velocity v over 1, 2, 3, and 4 units, and computed the optimal area segmentation, maximum path length for coverage, and simulation runtime. All the experiments of fields F_1 and F_2 showed the expected behaviors; the simulation runtime decreased when we either increased the velocity, number of robots, or δ and vice-versa. But field F_3 showed some differences. In experiments 3 and 4, the simulation runtime for optimal coverage of F_3 increased instead of decreasing when we increased the number of robots from 3 to 4, which is not optimal. It is because the maximum path length remained the same for both 3 and 4 robots, as the maximum area of a single polygonal cell remains the same even after the number of robots is increased. This is one of the limitations of trapezoidal decomposition. All other experiments showed expected behaviors. One of the experiments are illustrated in Fig. 6 which shows the complete coverage paths for F_1, F_2 and F_3 with number of robot $k = 3$ and $\delta = 0.2$.

VII. CONCLUSION

We have developed a new approach to solve the multi-robot coverage path planning (CPP) problem using simple motion dynamics where the robots move in a 2-D space having 1-D control inputs (Dubins path dynamics). However, our approach can be extended to robots with dynamical systems having n dimensional space and m dimensional control inputs. Moreover, we can use other decomposition methods to divide the field optimally among the robots (e.g., Voronoi regions [10]). Once we finish computing the coverage region of each robot by merging the decomposed cells using the dynamic programming algorithm, we can also use other path planning algorithms to calculate more feasible paths to reduce the overall coverage time. In the future, we would like to extend our work into 3-D space with non-linear aerial dynamics and use 3-D decomposition methods to get the optimal coverage path, simulate it in a 3-D simulator and apply it to real robots.

ACKNOWLEDGMENT

This work was partially supported by NSF CAREER Grant No. 1552668 and NSF Grant No. 2008957.

REFERENCES

[1] Jose Joaquin Acevedo, Begoña C. Arrue, Ivan Maza, and Anibal Ollero. Distributed approach for coverage and patrolling missions with a team of heterogeneous aerial robots under communication constraints. *International Journal of Advanced Robotic Systems*, 10(1):28, 2013.

[2] Nafis Ahmed, Chaitali J. Pawase, and KyungHi Chang. Distributed 3-d path planning for multi-uavs with full area surveillance based on particle swarm optimization. *Applied Sciences*, 11(8), 2021.

[3] Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, and Yahya Zweiri. A survey on multi-robot coverage path planning for model reconstruction and mapping. *SN Applied Sciences*, 1(8):847, Jul 2019.

[4] Y. Bouzid, Y. Bestaoui, and H. Siguerdidjane. Quadrotor-uav optimal coverage path planning in cluttered environment with a limited onboard energy. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 979–984, 2017.

[5] Taua Cabreira, Carmelo Di Franco, and Paulo Ferreira Jr. Energy-aware spiral coverage path planning for uav photogrammetric applications. *IEEE Robotics and Automation Letters*, PP:1–1, 07 2018.

[6] Tauã M. Cabreira, Lisane B. Brisolara, and Paulo R. Ferreira Jr. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, 3(1), 2019.

[7] Giorgio Cannata and Antonio Sgorbissa. A minimalist algorithm for multirobot continuous coverage. *IEEE Transactions on Robotics*, 27(2):297–312, 2011.

[8] Sung-Won Cho, Jin-Hyoung Park, Hyun-Ji Park, and Seongmin Kim. Multi-uav coverage path planning based on hexagonal grid decomposition in maritime search and rescue. *Mathematics*, 10(1), 2022.

[9] Seungyoon Choi, Seunggwan Lee, Hoang Huu Viet, and Taechoong Chung. B-theta*: An efficient online coverage algorithm for autonomous cleaning robots. *J. Intell. Robotics Syst.*, 87(2):265–290, aug 2017.

[10] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

[11] Carmelo Di Franco and Giorgio Buttazzo. Coverage path planning for uavs photogrammetry with energy and resolution constraints. *Journal of Intelligent & Robotic Systems*, 83(3):445–462, Sep 2016.

[12] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[13] Lars-Peter Ellekilde and Henrik I. Christensen. Control of mobile manipulator using the dynamical systems approach. In *2009 IEEE International Conference on Robotics and Automation*, pages 1370–1376, 2009.

[14] Taha Elmokadem. Distributed coverage control of quadrotor multi-uav systems for precision agriculture. *IFAC-PapersOnLine*, 52(30):251–256, 2019. 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019.

[15] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61:1258–1276, 12 2013.

[16] Shuzhi Sam Ge and C. Fua. Complete multi-robot coverage of unknown environments with minimum repeated coverage. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 715–720, 2005.

[17] Ibrahim A. Hameed. Coverage path planning software for autonomous robotic lawn mower using dubins' curve. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 517–522, 2017.

[18] Kuo-Chun Huang, Feng-Li Lian, Chien-Tung Chen, Chung-Hou Wu, and Chao-Cheng Chen. A novel solution with rapid voronoi-based coverage path planning in irregular environment for robotic mowing systems. *International Journal of Intelligent Robotics and Applications*, 5(4):558–575, Dec 2021.

[19] W.H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 1, pages 27–32 vol.1, 2001.

[20] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[21] Yu-Song Jiao, Xin-Min Wang, Hai Chen, and Yan Li. Research on the coverage path planning of uavs for polygon areas. In *2010 5th IEEE Conference on Industrial Electronics and Applications*, pages 1467–1472, 2010.

[22] Wei Jing, Di Deng, Yan Wu, and Kenji Shimada. Multi-uav coverage path planning for the inspection of large and complex structures. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1480–1486, 2020.

[23] Ratan Lal and Pavithra Prabhakar. Time-optimal multi-quadrotor trajectory planning for pesticide spraying. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7965–7971, 2021.

[24] Mickey Li, Arthur Richards, and Mahesh Sooriyabandara. Asynchronous reliability-aware multi-uav coverage path planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10023–10029, 2021.

[25] Mickey Li, Arthur Richards, and Mahesh Sooriyabandara. Reliability-aware multi-uav coverage path planning using a genetic algorithm. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1584–1586, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[26] Huei-Yung Lin and Yi-Chun Huang. Collaborative complete coverage and path planning for multi-robot exploration. *Sensors*, 21(11), 2021.

[27] Abdul Majeed and Sungchang Lee. A new coverage flight path planning algorithm based on footprint sweep fitting for unmanned aerial vehicle navigation in urban environments. *Applied Sciences*, 9(7), 2019.

[28] Hyeun Jeong Min and Nikolaos Papanikolopoulos. The multi-robot coverage problem for optimal coordinated search with an unknown number of robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 2866–2871, 2011.

[29] L. H. Nam, L. Huang, X. J. Li, and J. F. Xu. An approach for coverage path planning for uavs. In *2016 IEEE 14th International Workshop on Advanced Motion Control (AMC)*, pages 411–416, 2016.

[30] Andrea Noonan, Dale Schinstock, Chris Lewis, and Barry Spletzer. Optimal turning path generation for unmanned aerial vehicles. In *Proceedings of the Ninth IASTED International Conference on Control and Applications*, CA '07, page 21–26, USA, 2007. ACTA Press.

[31] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[32] Huy X. Pham, Hung M. La, David Feil-Seifer, and Matthew Deans. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6648–6653, 2017.

[33] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

[34] Junnan Song and Shalabh Gupta. $\varepsilon^\star$: An online coverage path planning algorithm. *IEEE Transactions on Robotics*, 34(2):526–533, 2018.

[35] Chee Sheng Tan, Rosmiwati Mohd-Mokhtar, and Mohd Rizal Arshad. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, 9:119310–119342, 2021.

[36] Marina Torres, David A. Pelta, José L. Verdegay, and Juan C. Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.

[37] Pengcheng Wang, Zhihong Man, Zhenwei Cao, Jinchuan Zheng, and Yong Zhao. Dynamics modelling and linear control of quadcopter. In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 498–503, 2016.

[38] Junfei Xie, Luis Rodolfo Garcia Carrillo, and Lei Jin. Path planning for uav to cover multiple separated convex polygonal regions. *IEEE Access*, 8:51770–51785, 2020.

[39] Liu Yunling, Xu Zili, Li Na, Xu Shuxiang, and Yuan Gang. A path planning algorithm for plant protection uav for avoiding multiple obstruction areas. *IFAC-PapersOnLine*, 51(17):483–488, 2018. 6th IFAC Conference on Bio-Robotics BIOROBOTICS 2018.