Minimizing Turns and Altitude Changes in a 3D Sweeping Coverage Problem for UAVs

Yingzi Zhang
Dept. of Mechanical and
Mechatronics Engineering
University of Waterloo
Waterloo, ON N2L 3G1
Email: y689zhan@uwaterloo.ca

Stephen L. Smith
Dept. of Electrical and
Computer Engineering
University of Waterloo
Waterloo, ON N2L 3G1
Email: stephen.smith@uwaterloo.ca

Abstract*—The sweeping coverage problem has been looked at extensively in literature, particularly in scenarios where UAVs must capture photographs of a large 2D area. We attempt to look at this problem where the UAV can move in 3D instead of being conventionally restricted on a plane. It is important for the UAV to be able to move in 3D when the area to be covered does not have a uniformly-defined image-resolution requirement, but instead, has several sub-areas that need to be captured in higher or lower resolutions. In this paper, we attempt to create an algorithm that allows the UAV to autonomously take images of the coverage area at the correct minimum-required resolutions. While there are some existing algorithms that can do this that minimize the total travel distance of the UAV, there does not seem to be one that minimizes the number of turns the UAV conducts, despite the fact that monitoring applications can benefit from minimal UAV turns in terms of image quality and in time. There is also benefit in minimizing the total number of altitude changes the UAV makes as well in order to minimize image distortion. Thus, we attempt to create a 3D path planning algorithm for the UAV to traverse such that we minimize both the number of turns and the number of total altitude changes. This is done using: a clustering algorithm that minimizes the number of altitude changes by grouping regions of similar image-resolution values under one uniform image-resolution, and a CGTSP tour of the clusters such that rotations are minimized. In addition, we provide a focal length controller to take advantage of the zoom capability of newer UAV cameras to further minimize the number of altitude changes, and prove that it is an optimal algorithm.

I. Introduction

The sweeping coverage problem is a problem in which a robot must plan a path over a certain area such that most or all points in this area are successfully covered by the footprint of the robot. There have been many different types of sweeping coverage algorithms devised and solved for specific types of applications, or to optimize specific attributes, covered in seminal surveys [1] [2]. These types of algorithms see many applications in surveillance [3], photogrammetry [4], demining [5], archaeology [6], search and rescue [7], floor cleaning [8], agriculture [9], and many others. However, many of these coverage path planning algorithms are constricted to a 2D plane, and do not take into consideration of a robot moving in 3D. Consideration of the 3D case is useful in applications such as photogrammetry, where some regions of the coverage area need to be photographed in higher quality than others.

Both UAV rotations and altitude changes can create undesirable effects on path quality [10] and image quality [11], which is why the objective is to minimize the two. In regards to UAV rotations, Huang [10] and Bochkarev and Smith [12] both explain that minimizing the number of turns is better for time and image quality. Some UAVs take a considerable amount of time turning compared to simply moving in a straight direction. Others have no camera stabilizers, so distortions of its images and the field-of-view will arise when capturing photos while turning with a slight roll angle. In regards to UAV altitude changes, constantly changing altitude also degrades image quality, as the change of altitude over a short distance will affect the UAV's pitch angle, which will distort the images. In addition, having the UAV change altitude at every time step would be not energy-efficient, and will affect the size of the UAV footprint on the ground as well, thereby necessitating more rotations to ensure full coverage.

Finally, due to the introduction of optimal zoom capabilities in recently developed commercial UAV cameras, a focal length controller algorithm can be devised to take advantage of this new feature in order to further minimize the number of altitude changes for UAVs with such cameras.

A. Contributions

There are four main contributions in this paper. First, the problem of having a 2D area to cover where each discrete point in the area requires the UAV to be at a different altitude appears to not yet be covered in literature. However, this does not mean there are no existing algorithms that can solve such a problem, especially when the objective is to minimize the total travel distance. These algorithms will be discussed in the II section.

Our second contribution is that our algorithms attempt to look at this 3D coverage problem with the objectives of minimizing the number of rotations and the number of altitude changes. This is done by clustering patches of area with similar resolution values together, so that the UAV can fly at a constant altitude within each cluster. After that, each cluster is populated with a minimal number of parallel lines that represents the robot's traversal path. We model it this way, similar to Huang's algorithm Huang [10], since a minimal number of straight traversal lines corresponds to a minimal number of rotations. Then, we compute a tour of these lines.

Computing a tour of these lines involves us solving what we call a "clustered generalized travelling salesman problem" (CGTSP). Both GTSP and CTSP are well known in literature and have approximate solutions. However, the CGTSP problem does not seem to appear in known literature. Our third contribution not only introduces this problem, but we also find a reduction of this problem into the familiar GTSP.

The final contribution of this paper is a greedy focal length controller. We provide proof that this controller minimizes the number of altitude changes and total distance travelled in this third dimension, given a 2D path.

II. Related Works

Plonski and Volkan [unpublished] created a 3D path planning algorithm which attempts to minimize the total distance travelled, where the UAV is required to visit a set of upside-down cones where each cone's apex represents a point on the ground. The volume of each cone represents the 3D space where a UAV is able to be in order to take an image with enough resolution of the point on the ground at the cone's apex. Their problem can be extended to ours by representing every image-resolution point in the coverage area with a cone. However, as previously mentioned, their solution will only minimize the total distance travelled.

Sadat et al. [13] considered a similar problem in regards to covering an area that does not have uniform minimum-required resolution. Instead of using a lawnmower-type covering algorithm which we are doing, they use a tree pattern to cover the entire area, where the root node symbolizes the position at which the UAV can take a low-resolution image of the entirety of the coverage area, and where child-nodes can take images of a smaller sub-region with higher resolution. This tree pattern for area coverage is also used in Carpin et al. [14]. They use various traversal algorithms, such as breadth-first, depth-first, and a strategy they devised called the "shortcut heuristic". However, as they mentioned, their algorithm causes the UAV to cover regions more than once, which lawnmower-type algorithms avoid doing. Finally, from simulations, they discovered that using a lawnmower-type algorithm at a constant altitude corresponding to the maximum resolution value in the coverage area actually performs better in terms of minimizing travel distance when there are many of these regions of high resolution.

Galceran and Carreras [15] created a new coverage algorithm that minimizes elevation changes above the seafloor for an AUV, since moving in the vertical axis is more expensive for most AUVs due to the shape of their chassis being designed for horizontal travel across the seafloor rather than vertical movement. They propose the AUV to first cover contours of the terrain of constant elevation before moving to the next parts of the terrain with a different elevation, to reduce the amount of depth-changing motions the AUV conducts while sweeping the area. As such, their algorithm only minimizes the total number of altitude changes.

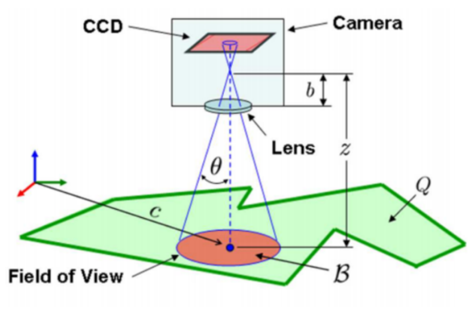

Fig. 1. Diagram of Major Parameters (from MIT Paper)

Schwager et al. [16] utilized altitude by deploying an optimal configuration of multiple non-moving robots to maximize coverage of an area, where the robots can have different altitudes from each other.

Huang [10] attempted to minimize the number of rotations a robot makes in the 2D sweeping coverage problem with an algorithm called "MSA". The MSA algorithm is where area Q is decomposed into the optimal set of polygons such that the minimum number of turns is achieved. This is done by covering each polygon with the minimum number of parallel lines equidistant from each other. These lines represent the boustrophedron traversal path of the UAV, and thus, minimizing the number of lines minimizes the number of rotations. To minimize both, the MSA algorithm orients the lines perpendicular to the minimum height of the polygon. We will be using a variant of the MSA algorithm in this paper.

Finally, we also plan to use a CGTSP solver that is similar to Bochkarev and Smith's GTSP solver [12] to obtain the order in which the parallel lines described in Huang's work are traversed. Note that this GTSP solver solves the order of traversal in a way where the UAV does not necessarily have to completely cover one polygon before going to the next.

III. Preliminaries

Let Q be the area we are trying to cover. Let q be a location (x, y) where $q \in Q$. A set of minimum-required resolution-per-area values over Q is provided, called Ψ. The ith element in this set is defined by $\psi_i(q)$.

Let f be the focal length of the lens. As with most cameras with optical zoom, we assume that there will be a maximum threshold focal length and a minimum threshold focal length of the camera, b_{max} and b_{min}. Since most cameras lose image quality at the extremes of their focal length range, b_{max} and b_{min} should be set to be values within the actual maximum and minimum focal lengths of the camera.

Let s be the length of the image sensor of the camera.

Let z be the distance from the UAV to the ground. For the remainder of this paper, z will be called 'altitude'.

Let b be the radius of the footprint of the camera. Note that for our application, we will assume the footprint is circular, though it can easily be changed into a rectangle.

IV. Minimizing the Number of Turns and Number of Altitude Changes

The proposed algorithm to minimize both the number of turns and altitude changes is based on Huang's MSA algorithm [10]. Essentially, Q is discretized into a set of disjoint polygons by clustering regions of similar $\psi(q)$ values together. Within each polygon, the UAV will fly under a constant altitude and can change its focal length if need be. In each polygon, the range in $\psi(q)$ values corresponds to no more than a $p\%$ change in footprint size. This algorithm can be seen in Algorithm 1.

The reasons why we chose this algorithm are multifold. First, by discretizing the regions into clusters of similar $\psi(q)$ values, we can lower the number of altitude changes. Secondly, the $p\%$ threshold in footprint size difference helps minimize the number of turns as it allows the UAV to travel linearly in a boustrophedon format within each polygon. If p were too big, then the footprint size would vary too much, and thus the traversal path of the UAV cannot be modelled by straight parallel lines. Instead, the UAV would have to turn more often to fully cover a given area. Discretizing the regions also helps to prevent the UAV from flying with an unnecessarily small footprint size in regions that have low $\psi(q)$ values, as small footprint sizes translates into the need for more parallel lines, and thus a larger number of turns.

Algorithm 1 MSA Algorithm

```
1: $K$ = ClusteringAlgorithm() {Refer to Section V}
2: for $k \in K$ do
3:     MSA($k$) {MSA($k$) fills polygon $k$ with parallel lines
       oriented to minimize the number of these lines.}
4: end for
```

V. Discretizing Q via a Clustering Algorithm

While there are numerous heuristic clustering algorithms that exist in literature [17], it is evident that for our problem, a clustering algorithm that both minimizes the number of clusters and intra-cluster $\psi(q)$ variance must be used in order to minimize the number of altitude changes and rotations. Because of the difficulty in knowing how small the variance should be within each cluster, a defined constant percentage threshold p in footprint size can be used. As mentioned before, minimizing the footprint size in each polygon is paramount, as it allows the UAV to travel in straight lines, thereby reducing the number of rotations.

Our objective for the clustering algorithm can be defined as follows.

Given $G = (V, E)$ as an unweighted connected graph, where the vertices V are all $\psi(q)$ values in Q, and E is the set of edges that represent the adjacency of the $\psi(q)$ values in Q, minimize K where $V_1, V_2, ..., V_K$ are connected and disjoint subsets of V. In addition, if we define B_i as the set of footprint sizes corresponding to each $\psi(q)$ value in V_i, $\frac{B_{i_{max}} - B_{i_{min}}}{B_{i_{min}}} \leq p$ for $i = 1...K$.

A. Hierarchical Clustering via Ward's Method

Because there are such a large number of clustering algorithms, only two were chosen in the scope of this research. Of course, further research should be done in determining the best clustering algorithm.

One of the two clustering algorithms chosen is the well-known agglomerative hierarchical clustering algorithm. Agglomerative hierarchical clustering is where individual $\Psi(q)$ values are initialized as clusters of their own, and with each iteration of the algorithm, smaller clusters form into larger clusters, until a stopping point is reached. While there are several methods in choosing the pair of clusters to merge at each iteration, Ward's method was chosen for our problem, as it fits our objective of minimizing intra-cluster variance of both ψ and of footprint size [18]. Only the two clusters with the minimum increase in sum-of-squared errors will be merged at each iteration. To put it more formally, a merge of two clusters C_i and C_j only occurs when the pair, C_i and C_j, specifically results in the lowest value of the following objective function out of all cluster-pair combinations:

$$\Delta(C_i, C_j) = \sum_{k \in C_i \cup C_j} (\psi_k - \mu_{C_i \cup C_j})^2 - \Big(\sum_{k \in C_i} (\psi_k - \mu_{C_i})^2 + \sum_{k \in C_j} (\psi_k - \mu_{C_j})^2 \Big) \quad (1)$$

$$\Delta(C_i, C_j) = \frac{n_{C_i} n_{C_j}}{n_{C_i} + n_{C_j}} (\mu_{C_i} - \mu_{C_j})^2 \quad (2)$$

Where μ_{C_i} corresponds to the average Ψ value in cluster C_i.

Agglomerative hierarchical clustering algorithms need a defined stopping point or else the algorithm will continue until all clusters are merged into one. For our application, we can determine this stopping point by using the user-defined p value. As soon as a merged cluster crosses the p variance threshold, the algorithm ceases.

The clustering algorithm via Ward's method for our specific application is shown in Algorithm 2. Note that M is a symmetric matrix that stores and keeps track of $\Delta(C_i, C_j)$ for every cluster-pair combination. Also note that if two clusters are not adjacent to each other, then $\Delta(C_i, C_j) = \infty$. The time complexity for Algorithm 2 is $O(n^2)$.

B. Clustering Algorithm Lower Bound

It is evident that we cannot find a guarantee to this clustering algorithm without making heavy assumptions on the connectivity of the Ψ graph in Q. Therefore, we attempt to output a lower bound for K by relaxing the connectivity constraint, and only cluster similar Ψ values together, regardless of whether or not two similar Ψ values are necessarily adjacent in Q. To do this, J and all ψ points in Q, Ψ, must be given. This lower bound essentially is the specific scenario when all Ψ values that are very similar to each other are all coincidentally adjacent to each other. There is no guarantee that our algorithm will give a value of K close to this lower bound. However, outputting this lower bound and comparing it with the outputted K from our algorithm can be of use, and with experimental evidence,

Algorithm 2 ClusteringAlgorithm()

```
1: Initialize each $\psi(q) \in \Psi(q)$ as its own cluster $c \in C$.
2: Initialize graph $G$. {G holds every $\Psi(q)$ value and its adjacent relationships to other $\Psi(q)$ values.}
3: Initialize $p$. {$p$ is the percentage footprint size threshold.}
4: Initialize $currMaxVar \leftarrow 0$.
5: while $C.size > 1$ do
6:     Initialize $minDelta \leftarrow \infty$.
7:     Initialize $cluster_1 \leftarrow \emptyset$ and $cluster_2 \leftarrow \emptyset$.
8:     for $i = 1$ to $G.Edges.Size$ do
9:         $M \leftarrow \Delta(G.Edges[i].cluster1, G.Edges[i].cluster2)$ {refer to Equation 1 for delta.}
10:        if $M \leq minDelta$ then
11:            $minDelta \leftarrow M(i,j)$
12:            $cluster_1 \leftarrow G.Edges[i].cluster1$
13:            $cluster_2 \leftarrow G.Edges[i].cluster2$
14:        end if
15:    end for
16:    $b_{min} \leftarrow getMinFP(cluster_1, cluster_2)$
17:    $b_{max} \leftarrow getMaxFP(cluster_1, cluster_2)$
18:    if $\frac{b_{max}-b_{min}}{b_{min}} \geq p$ then
19:        return {Done merging clusters.}
20:    else
21:        mergeClusters($cluster_1$, $cluster_2$)
22:    end if
23: end while
```

we can see if our clustering algorithm can be within a small constant factor from the lower bound.

This type lower bound on K can be found by Algorithm 3. Essentially, this algorithm goes through the sorted list of Ψ from beginning to end and greedily clusters the Ψ values. Note that the Ψ-range of every cluster is the same and must be equal to J.

Algorithm 3 LowerBound(J, Ψ)

```
1: $tempPsi \leftarrow \Psi[1]$
2: $lowerBoundK \leftarrow 1$
3: for $\psi \in \Psi$ do
4:     if $\psi > tempPsi + p$ then
5:         $tempPsi \leftarrow \psi$
6:         $lowerBoundK \leftarrow lowerBoundK + 1$
7:     end if
8: end for
9: return $lowerBoundK$
```

1) Proof of Correctness for Lower Bound: The logic that we use is very similar to the proof of correctness we use for our greedy focal length controller.

To prove the optimality of our algorithm, we must first prove that our algorithm exhibits both the greedy choice property and the optimal substructure property.

Lemma V.1. *Let us define the first step of our algorithm, d_1. d_1 finds the first interval, $[\psi_1, \psi_i]$, that represents the first cluster we greedily find based off of the first Ψ value in the sorted Ψ list, ψ_1. Then, there always exists an optimal solution S^* in which d_1 is its first step.*

The following proves this greedy choice property. Let R be defined as one optimal solution. If R already includes d_1, then the property holds and is trivial. If R does not include d_1, then there is only one possibility: the first step in R starts at a value of Ψ less than ψ_1. The first cluster cannot start at a Ψ value greater than ψ_1 or else ψ_1 will not be in any cluster at all. Let us call this first step of R as d_{R_1}.

We can exchange d_{R_1} that was originally from R with our first step, d_1, very easily, even if the second cluster in R, d_{R_2}, is directly adjacent to d_{R_1}. We can simply remove d_{R_1}, shorten d_{R_2} such that its interval starts right after ψ_i, where ψ_i is the value d_1 ends, and then finally place d_1 into R. Thus, the following holds true: d_{R_1} can be exchanged with d_1 in R, to produce another solution, S^*, which is still optimal.

Lemma V.2. *Let S' denote the sequence of a minimum number of disjoint clusters, where the total range of all the clusters is between $(\psi_i, \psi_n]$, where ψ_i is the last point in the first cluster (whatever the first cluster may be), and where ψ_n is the last Ψ value in the sorted list. Then, $S = \langle d_1, S' \rangle$ must be an optimal solution.*

We now prove this optimal substructure property. We know that none of the clusters in sequence S' overlap with d_1. Thus, $S = \langle d_1, S' \rangle$ is a feasible solution. We also know from the last lemma, there exists an optimal sequence S^* in which d_1 is its first cluster. It is apparent that $S^* - d_1$ is also compatible with d_1. Since S' was defined to be an optimal sequence for $[q_i, q_n]$, then the following inequality holds:

$$|S'| \leq |S^* - d_1|$$

Thus, we can apply the following logic:

$$|S^* - d_1| = |S^*| - 1$$
$$|S'| \leq |S^*| - 1$$
$$|S| = |S'| + 1$$
$$|S| - 1 = |S'|$$
$$|S| - 1 \leq |S^*| - 1$$
$$|S| \leq |S^*|$$

Thus, S is an optimal solution.

Finally, we can prove the correctness of this algorithm by induction:

1) Basis: If $|S| = 1$, the algorithm gives us an optimal solution, as d_1 is already optimal.
2) Induction Step: Assume our algorithm can give us optimal solutions for all sequences where the number of clusters is z, for some $z \geq 1$. We show that our algorithm also provides an optimal solution for sequences of clusters of size $z + 1$. Consider a coverage path where the optimal solution results in $z + 1$ intervals. By the greedy choice property, d_1, obtained from our algorithm, is part of some optimal solution for this problem. By the optimal

substructure property, after d_1, there is an S' that can be combined with d_1 to give us an optimal solution, $S = \langle d_1, S' \rangle$. S' has to consist of z steps. We already assumed we can solve problems of size z, and thus, using the logic above, we can also solve S, or more generally, problems of size $z + 1$.

C. *User-Defined Polygonal Decomposition of* $\Psi(q)$

Instead of the above two methods, we could disregard the concept $\Psi(q)$ entirely and instead use a completely different way to describe the minimum required resolution over sub-regions in area Q. We could instead have the end-user manually tell us convex polygons over Q in which there is a uniform minimum resolution required. For example, using a Google Maps-like application, the user can draw polygons over sub-regions of Q where there needs to be better or worse resolution than the rest of Q. If the user requests a concave polygon shape to have a certain $\Psi(q)$ value, we can break the concave polygon into multiple convex polygons in order to get a smaller amount of lines, and thus a smaller number of rotations, than if we were to keep the concave polygon intact—similar to Bochkarev and Smith [12].

VI. Clustered GTSP (CGTSP) Tour Generation

In Bochkarev and Smith [12], a GTSP (generalized travelling salesman problem) solver was used in order to plan a coverage path of disjoint polygons, where within each polygon, straight parallel lines oriented along the polygon's longest height are traversed. For graph $G = (V, E)$ where $V = \{v_i ... v_n\}$ is the set of all vertices and $E = \{(v_i, v_j); v_i, v_j \in V, i \neq j\}$ is the set of all edges, the GTSP is defined such that the set V is further subdivided into subsets $V_1 ... V_k$, where k is the number of subsets, $V_1 \cup V_2 \cup ... V_k = V$, and that for all subsets V_i and V_j, where $i \neq j$, $V_i \cap V_j = \emptyset$. Solving the GTSP requires finding the shortest path such that we visit only one node for every subset. For both Bochkarev and Smith's problem and our problem, each straight line can be traversed in two possible ways. Thus, every vertex should represent one way of traversal for one line, and every subset has a size of 2, and represents the two possible traversal paths of a line.

In contrast to the GTSP, a clustered TSP (CTSP) solver is a bit different. In a CTSP, every time the traveler goes into a cluster, they must travel to every node within the same cluster before moving onto another cluster.

A CGTSP is a combination of the two above problems, and must be solved in our specific problem. To clarify, a CGTSP is where the subsets from a GTSP are clustered such that the travelling salesman needs to travel to every subset within the same cluster of subsets before going onto the next cluster of subsets.

Therefore, our graph $G = (V, E)$ is defined such that each $v_i \in V$ represents a line that the UAV traverses in one of two possible directions. Similar to Bochkarev and Smith's GTSP, each subset in our problem consists of only two nodes, representing the two possible traversal directions. The weight of each edge in E represents the cost of travel for the UAV from one line to another. In our simulations, we utilize Dubins' car model to calculate these edge weights. Every cluster of subsets in this CGTSP represents a polygon obtained from the clustering algorithm mentioned in Section V. These clusters of polygons of similar altitudes need to be consecutively travelled one at a time in order to minimize the number of altitude changes, which is why we could not use the simple GTSP solver from Bochkarev and Smith.

A. *Reduction of CGTSP into GTSP*

No solver for CGTSP exists in literature as of yet. However, the previous two types of problems, GTSP and CTSP, have been solved by reducing the problems to a simple ATSP [19] and TSP problem [20], respectively. The CGTSP solver can be solved in a similar way to these two, by reducing it into a GTSP. The transformation, similar to that of Helsgaun's [20], is described below where V' is the transformed vertex set, V_i' is the ith transformed subset of vertices, and c_{ij}' represents the new transformed cost between vertices v_i and v_j:

(a) $V = V'$
(b) $\{V_1 ... V_k\} = \{V_1' ... V_k'\}$
(c) Define $c_{ij}' = c_{ij}$, when v_i and v_j belong to the same cluster.
(d) Define $c_{ij}' = c_{ij} + M$, when v_i and v_j belong to two different clusters. M is a large constant, where $\sum_{(i,j) \in V} c_{ij} < M < \infty$ [21].

When entering a cluster at a vertex v_i, an optimal GTSP tour will always visit all other vertex sets of the cluster before moving to the next cluster due to the added M term for inter-cluster edges. If we define m as the number of inter-cluster edges the GTSP tour will have, then cost of the tour for the actual CGTSP is the cost of the above CGTSP-to-GTSP tour subtract mM.

VII. Focal Length Controller

We have made several assumptions for the focal length controller. First, the camera uses parfocal lens rather than varifocal lens. Varifocal lens are used by most standard cameras, where they take a set amount of time to be refocused. Parfocal lens, on the other hand, stays in focus even when the focal length is changed. Thus, parfocal lens is more practical in aerial surveillance for a fixed-wing UAV since there will not be any waiting duration for the camera to refocus while the aircraft is flying over the coverage area. The second assumption we made is that the UAV can accurately estimate its altitude over Q. There are various papers that cover altitude estimation for UAVs, including [22] and [23]. Thirdly, we do not consider digital zoom at all. This section entirely deals with optical zoom. Digital zoom is undesirable as it is analogous to merely cropping the original image and enlarging it, which severely affects resolution.

In addition to these assumptions, an equation relating ψ with the altitude of the UAV, z, and the UAV's focal length, f, must be derived.

Given that n is the total number of pixels for the camera and b is the footprint size:

$$\psi = \frac{n}{\pi b^2} \tag{3}$$

We can also define b in terms of the UAV's altitude, z, the size of the sensor, s, and the focal length, f by understanding that the relationship between them comes from similar triangles:

$$b = z\frac{s}{2f} \tag{4}$$

Note that half the length of the sensor is used, $\frac{s}{2}$, due to the fact that b is only the radius of the footprint and not the diameter. Combining the two above equations together results in the following relationship:

$$\psi = \frac{n}{\pi\left(\frac{zs}{2f}\right)^2} \tag{5}$$

A. Greedy Focal Length Controller

An offline greedy focal length controller was devised using Equation 5 and by minimizing the following cost function in order to minimize the number of altitude changes.

Let us define a function $f(x)$ where $x \in [0, x_n]$. Let us also define a set $f_1, f_2, ...f_n$ where:

$$f_1(x) = f(x) : x \in [0, x_1] \tag{6}$$
$$f_2(x) = f(x) : x \in (x_1, x_2] \tag{7}$$
$$... \tag{8}$$
$$f_n(x) = f(x) : x \in (x_{n-1}, x_n] \tag{9}$$

so that:

$$f(x) = \begin{cases} f_1(x) & \text{for } [0, x_1] \\ f_2(x) & \text{for } (x_1, x_2] \\ ... & \\ f_n(x) & \text{for } (x_{n-1}, x_n] \end{cases}$$

Also define $g_i = max(f_i(x))$ and $h_i = min(f_i(x))$. Our objective function is to minimize n:

$$C_{focal} = n \qquad \text{such that} \quad g_i/h_i \leq F \text{ for } \quad i = 1...n$$

Where C_{focal} is the cost function, F is a constant, and wherever the condition $g_i/h_i \leq F$ breaks is where f_{i+1} starts.

In our specific application, $f(x)$ is $\Psi(q)$, and F is the range of possible resolutions that can be achieved by solely changing the focal length of the camera, without the altitude. This range can be derived from the following:

$$g_i = \frac{n}{\pi\left(\frac{z_i s}{2f_{max}}\right)^2} \tag{10}$$

$$h_i = \frac{n}{\pi\left(\frac{z_i s}{2f_{min}}\right)^2} \tag{11}$$

$$\frac{g_i}{h_i} = \left(\frac{f_{max}}{f_{min}}\right)^2 \tag{12}$$

$$F = \left(\frac{f_{max}}{f_{min}}\right)^2 \tag{13}$$

With the above equation, we can figure out where along the traversal path the intervals should be. The next step is to figure out what altitude the UAV should fly at for each interval. The following derives the correct altitude value, z_i, for $(x_i, x_{i+1}]$:

$$z_i = \frac{2f_{max}}{s}\sqrt{\frac{n}{\pi g_i}} \tag{14}$$

$$z_i = \frac{2f_{min}}{s}\sqrt{\frac{n}{\pi h_i}} \tag{15}$$

By adding Equations 14 and 15 together and re-arranging, we get z_i:

$$z_i = \frac{\sqrt{\frac{n}{\pi}}\left(\frac{f_{max}}{\sqrt{g}} + \frac{f_{min}}{\sqrt{h}}\right)}{s} \tag{16}$$

Algorithm 4 uses Equations 13 and 16 to find the optimal altitudes for the UAV change throughout the coverage path in order to minimize the cost function. Note that this algorithm assumes we already have a known traversal path to cover area Q from the CGTSP. Also note that this algorithm is run strictly before the UAV actually travels along the coverage path. The algorithm returns the tuple-list object, P, which is used to figure out at what points along the coverage path from q_i to q_m, called CP, the altitude must change, and also the exact value the altitude of the UAV must be changed to. Algorithm 5 is run while the UAV is operating. This algorithm uses the tuple list obtained from Algorithm 4 in order to change the altitude of the UAV as the UAV travels along the coverage path.

Finally, the time complexity of both algorithms 4 and 5 is $O(n)$, where n in this case is the number of discrete steps in the coverage path CP.

1) Proof of Correctness: This section establishes the correctness of Algorithms 4 and 5 and proves why this greedy controller produces an optimal number of altitude changes for a given coverage path.

To prove the optimality of our algorithm, we must first prove that our algorithm exhibits both the greedy choice property and the optimal substructure property.

Lemma VII.1. *Let us define the first step of our algorithm, d_1. d_1 finds the first interval, $[q_0, q_i]$, that greedily maximizes the distance travelled until the UAV must change altitude. Then, there always exists an optimal solution S^* in which d_1 is its first step.*

The following proves this greedy choice property. Let R be defined as one optimal solution. If R already includes d_1, then the property holds and is trivial. If R does not include d_1, and instead has its first step as d_{R_1} with an interval $[q_{R_0}, q_{R_i}]$, where $d_{R_1} \neq d$, then $q_{R_i} < q_i$, since d_1 was already defined as the maximum interval possible before the maximum camera resolution range condition breaks. From this, we can exchange the original interval $[q_{R_0}, q_{R_i}]$ in R with $[q_0, q_i]$. Note that in order to do so, the interval of difference between d_1 and d_{R_1}, $[q_{R_i}, q_i]$, that was originally in the second step interval in R, d_{R_2}, will move into the first step interval. This transferal process is valid as it does not violate the maximum camera

Algorithm 4 Offline Greedy Function to Determine When Altitude Must be Changed

```
1: $P \leftarrow \emptyset$ {where $P$ is a list of tuples. Each tuple has one position value of $q \in Q$, and one altitude value $z$.}
2: $g \leftarrow \Psi(q_1)$ {where $q_1$ is the first area the UAV will be while traversing along the coverage path.}
3: $h \leftarrow \Psi(q_1)$
4: for $i = 2$ to $m$ do {where m is the number of ordered elements in $Q$ that make up the traversal path of the UAV, $CP$}
5:     if $\Psi(q_i) > g$ then
6:         $g \leftarrow \Psi(q_i)$
7:     end if
8:     if $\Psi(q_i) < h$ then
9:         $h \leftarrow \Psi(q_i)$
10:    end if
11:    if $\frac{g}{h} \geq \left(\frac{f_{max}}{f_{min}}\right)^2$ then
12:        $z_i = \frac{\sqrt{\frac{n}{\pi}}\left(\frac{f_{max}}{\sqrt{g}} + \frac{f_{min}}{\sqrt{h}}\right)}{s}$ {from Equation 16}
13:        $p \leftarrow (q_i, z_i)$
14:        $P.append(p)$
15:        $g \leftarrow \Psi(q_i)$
16:        $h \leftarrow \Psi(q_i)$
17:    end if
18: end for
19: return P
```

Algorithm 5 Greedy Focal Length Controller

```
1: Let $f$ be the current focal length, and $z$ be the current altitude. Let $P$ be the list of tuples obtained from Algorithm 4.
2: for $i = 1$ to $m$ do
3:     if $q_i = P[0].q$ then
4:         $z \leftarrow P[0].z$
5:         Pop $P[0]$ from $P$.
6:     end if
7:     $f \leftarrow \frac{zs}{2}\sqrt{\frac{\pi\psi}{n}}$
8: end for
```

resolution range condition for the second step interval, d_{R_2}. Thus, the following holds true: d_{R_1} can be exchanged with d_1 in R, to produce another solution, S^*, which is still optimal. The only difference now between R and S^* is that the altitude value at which the UAV must fly in the first interval will change, though this does not affect the proof in any way.

Lemma VII.2. *Let S' denote an optimal sequence of intervals between $(q_i, q_n]$, where q_i is the last point in the first interval (whatever the first interval may be), and q_n is the last point in the coverage path. Then, $S = \langle d_1, S' \rangle$ must be an optimal solution.*

We now prove this optimal substructure property. We know that none of the intervals in sequence S' overlap with d_1. Thus, $S = \langle d_1, S' \rangle$ is a feasible solution. We also know from the last lemma, there exists an optimal sequence S^* in which d_1 is its first interval step. It is apparent that $S^* - d_1$ is also compatible with d_1. Since S' was defined to be an optimal sequence for $[q_i, q_n]$, then the following inequality holds:

$$|S'| \leq |S^* - d_1|$$

Thus, we can apply the following logic:

$$|S^* - d_1| = |S^*| - 1$$
$$|S'| \leq |S^*| - 1$$
$$|S| = |S'| + 1$$
$$|S| - 1 = |S'|$$
$$|S| - 1 \leq |S^*| - 1$$
$$|S| \leq |S^*|$$

Thus, S is an optimal solution.

Finally, we can prove the correctness of this algorithm by induction:

1) Basis: If $|S| = 1$, the algorithm gives us an optimal solution, as d_1 is already optimal.
2) Induction Step: Assume our algorithm can give us optimal solutions for all sequences where the number of intervals is z, for some $z \geq 1$. We show that our algorithm also provides an optimal solution for sequences of size $z + 1$. Consider a coverage path where the optimal solution results in $z + 1$ intervals. By the greedy choice property, d_1 is part of some optimal solution for this problem. By the optimal substructure property, after d_1, there is an S' that can be combined with d_1 to give us an optimal solution, $S = \langle d_1, S' \rangle$. S' has to consist of z steps. We already assumed we can solve problems of size z, and thus, using the logic above, we can also solve S, or more generally, problems of size $z + 1$.

2) Minimizing Total Distance in Change of Altitude: Not only does this algorithm minimize the number of altitude changes, but it also minimizes the total distance of change in altitude.

Claim VII.3. *The greedy focal length controller algorithm minimizes the total distance in change of altitude for the UAV.*

We prove this via proof of contradiction. We know that our algorithm produces the minimum number of intervals for path CP from the proof of correctness in the previous section, which directly corresponds to a minimum number of altitude shifts. Let this minimum number of altitude shifts be w. Thus, the only case we have to consider is the case in which there are more than w altitude shifts.

Take some arbitrary interval from point q_i to some point q_j, where $q_j > q_i$, such that our algorithm will have one altitude shift. Define this altitude shift's distance as H. Let us assume there exists another algorithm that gives this interval $[q_i, q_j]$ a larger number of altitude shifts, n, where $n \geq 2$, and where the total distance in altitude change is defined as $H_* = h_1 + h_2 + ... + h_n$. Additionally assume that $H_* < H$, which means $h_1 + h_2 + ... + h_n < H$. However, if this inequality holds true, then there must exist at least one $\Psi(q)$ point along the coverage path that cannot be reached purely changing the focal length of the camera, since we have shortened the total altitude distance to H_*. To compensate for this, we must add more altitude shifts. However, by then, $h_1 + h_2 + ... + h_n + h_{n+1} + ... \geq H$. Thus, by proof of contradiction, there is no such total altitude distance H_*, where $H_* < H$, that can fully cover the desired minimum-required resolutions without changing altitude at another q point between $[q_i, q_j]$. This reasoning can be extended to the entire coverage path, CP, where the following inequality must be true: $H(w+n) \geq H$, where $H(w+n)$ is the total altitude distance when there are $w+n$ number of altitude shifts, and where $n \geq 1, n \in \mathbb{Z}$. Therefore, our algorithm does indeed minimize the total distance in change of altitude.

To summarize, once you decrease the value of H from our algorithm to a smaller value, there will exist $\Psi(q)$ points that cannot be reached without creating additional altitude shifts to compensate. The total H_*, including the additional altitude shifts, will be at least equal to our original H. Thus, our H must be the minimum.

VIII. Simulations

The simulation results goes here.

IX. Conclusion

The conclusion goes here.

References

[1] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[2] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[3] A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar, "An optimization-based approach to time-critical cooperative surveillance and coverage with uavs," in *Experimental Robotics*. Springer, 2008, pp. 491–500.

[4] C. Di Franco and G. Buttazzo, "Coverage path planning for uavs photogrammetry with energy and resolution constraints," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2016.

[5] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *The International journal of robotics research*, vol. 22, no. 7-8, pp. 441–466, 2003.

[6] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles," *Journal of Field Robotics*, vol. 32, no. 7, pp. 952–983, 2015.

[7] L. Lin and M. A. Goodrich, "Uav intelligent path planning for wilderness search and rescue," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 709–714.

[8] J. Hess, M. Beinhofer, and W. Burgard, "A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5600–5605.

[9] I. Hameed, D. Bochtis, and C. A. G. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," *International Journal of Advanced Robotic Systems*, vol. 10, no. 231, pp. 1–9, 2013.

[10] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 27–32.

[11] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based road-following using a small autonomous aircraft," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol. 5. IEEE, 2004, pp. 3006–3015.

[12] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning."

[13] S. A. Sadat, J. Wawerla, and R. T. Vaughan, "Recursive non-uniform coverage of unknown terrains for uavs," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1742–1747.

[14] S. Carpin, D. Burch, N. Basilico, T. H. Chung, and M. Kölsch, "Variable resolution search with quadrotors: Theory and practice," *Journal of Field Robotics*, vol. 30, no. 5, pp. 685–701, 2013.

[15] E. Galceran and M. Carreras, "Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4159–4164.

[16] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[17] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[18] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[19] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm," *Mathematical Programming Computation*, vol. 7, no. 3, pp. 269–287, 2015.

[20] ——, "Solving the clustered traveling salesman problem using the lin-kernighan-helsgaun algorithm," *Computer Science Research Report*, no. 142, pp. 1–16, 2011.

[21] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 31, no. 1, pp. 39–44, 1993.

[22] D. Eynard, P. Vasseur, C. Demonceaux, and V. Frémont, "Uav altitude estimation by mixed stereoscopic vision," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 646–651.

[23] A. Cherian, J. Andersh, V. Morellas, N. Papanikolopoulos, and B. Mettler, "Autonomous altitude estimation of a uav using a single onboard camera," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3900–3905.