Noname manuscript No.
(will be inserted by the editor)

Multi-UAV Trajectory Optimization Utilizing a NURBS-based Terrain Model for an Aerial Imaging Mission

Youngjun Choi · Mengzhen Chen · Younghoon Choi · Simon Briceno · Dimitri Mavris

Received: date / Accepted: date

Abstract Trajectory optimization precisely scanning an irregular terrain is a challenging problem since the trajectory optimizer needs to handle complex geometry topology, vehicle performance, and a sensor specification. To address these problems, this paper introduces a novel framework of a multi-UAV trajectory optimization method for an aerial imaging mission in an irregular terrain environment. The proposed framework consists of terrain modeling and multi-UAV trajectory optimization. The terrain modeling process employs a Non-Uniform Rational B-Spline (NURBS) surface fitting method based on point cloud information resulting from an airborne LiDAR sensor or other sensor systems. The NURBS-based surface model represents a computationally efficient terrain topology. In the trajectory optimization method, the framework introduces a multi-UAV vehicle routing problem enabling UAV to scan an entire area of interest, and obtains feasible trajectories based on given vehicle performance characteristics, and sensor specifications, and the approximated terrain model. The proposed multi-UAV trajectory optimization algorithm is tested by representative numerical simulations in a realistic aerial imaging environment, namely, San Diego and Death Valley, California.

Keywords Multi-UAV trajectory optimization · Aerial imaging · Non-Uniform Rational B-Spline (NURBS) · Terrain modeling · Vehicle Routing Problem

Youngjun Choi (✉) · Mengzhen Chen · Younghoon Choi · Simon Briceno · Dimitri N. Mavris
School of Aerospace Engineering, Georgia Institute of Technology North Avenue, Atlanta, GA 30332
E-mail: ychoi95@gatech.edu

Mengzhen Chen
E-mail: mchen328@gatech.edu

Younghoon Choi
E-mail: younghoon.choi@gatech.edu

Simon Briceno
E-mail: briceno@gatech.edu

Dimitri N. Mavris
E-mail: dimitri.mavris@aerospace.gatech.edu

1 Introduction

Unmanned Aerial Systems (UAS) have been adopted in various aerial missions, such as surveillance, inspection, surveying, and 2D/3D mapping applications because of their endurance capability resulting from high specific energy battery technologies, light weight structures, and high-efficiency propeller/actuator systems. UAS have particularly gained much attention in aerial imaging missions (e.g., three-dimensional mapping, building inspection, and crop-monitoring missions) due to rapid scanning of an area of interest and high-quality sensor systems.

To collect aerial images of an entire area of interest (AOI), solving a coverage path-planning problem is critical. This is because most current off-the-shelf Unmanned Aerial Vehicles (UAVs) depending on their type (i.e., fixed wing, VTOL, and quadcopter), have limited endurance capabilities, with approximately 10 $\sim$ 120 minutes endurance, and payload weight constraints [6]. Even, if a UAV carries higher payload weight, its flight endurance tends to significantly decrease. Therefore, the coverage path planning problem must consider this endurance constraint more precisely to generate a feasible coverage trajectory. The areas for agricultural aerial imaging missions and surveillance missions are often too large to be completed by a single UAV due

to the endurance limitation. Hence, the coverage path-planning problem should also handle multi-UAV operations and generate their coverage paths.

In the trajectory optimization problem, many techniques/methods have been introduced for a real-time or off-line process. These techniques can be classified by five groups [7]: stochastic methods, roadmap methods, potential field methods, geometric methods, and optimization-based methods. For the coverage path planning problem, a common technique is a road map method, which is a grid-based approach, because a grid map can be determined by an image field of view defined from sensor specifications and a flight altitude [12].

One of the classical grid-based approaches is the coverage path-planning using an exact cellular decomposition method that uses a back-and-forth search trajectory [12]. Because the back-and-forth search trajectory cannot generate a full-coverage path in a non-convex AOI, a cellular decomposition method is adopted to obtain multiple small convex regions from a non-convex AOI [1][17][26]. Based on the decomposition results, first it defines the back-and-forth search trajectory in each region, and then it solves the optimal visiting sequence of all the sub-regions. To determine the flight sequence to visit all the sub-regions, a traveling salesman problem has been implemented [12].

Another classical approach is a wavefront-based algorithm that defines a coverage path based on the score of each grid cell, which depends on the locations of the initial point, terminal point, and any restricted areas [8][29]. This method uses a pseudo-gradient that is computed by the neighborhood scores to select the next visit sequence. This method is powerful because it can generate an optimal coverage path in a non-convex AOI. However, the wavefront-based algorithm cannot directly solve a multi-UAV coverage path planning problem.

An alternative method is a traveling salesman problem (TSP)-based routing optimization approach that generates a route visiting all customers' locations from a depot location [9]. The TSP-based approach has been extended to handle more complex problems, commonly referred to as a vehicle routing problem (VRP) that generates an optimal route by minimizing a cost function (e.g., total distance, and total mission time) with constraints. The major benefits of the VRP-based approach are that it easily constructs constraints defined from the concept of operation and vehicle characteristics, it efficiently controls design variables, and it rapidly solves the optimization problem. Hence, a great quantity of literature has applied the VRP to address UAV coverage path-planning problems [2].

In an agriculture or disaster monitoring UAS mission, even though some scanning areas may not have flat terrain, most UAS coverage path-planning problems solve a two-dimensional coverage problem since their main assumption is that a trajectory is determined on Above Ground Level (AGL) [8]. However, without considering the elevation changes of the terrain, the result of the coverage trajectory cannot accurately account for a UAV endurance constraint, which is the critical element to determine how many UAVs are required to complete a mission. Hameed et al. highlights that the AGL assumption is too idealistic since the total distance of the optimal coverage trajectory can be much shorter than the actual coverage distance depending on elevation variations [13]. Moreover, the camera view direction based on a surface shape can minimize image distortion. In other words, to generate a more precise optimal coverage path, terrain topology should be considered in the trajectory optimization problem. The existing coverage path-planning articles using 3D terrain geometry information mostly address inspection missions for buildings or houses [3][15]. Few articles have actually studied the coverage path-planning dealing with terrain geometry features [6][13].

In the aerial imaging mission with a large AOI, a single UAV is unable to scan the entire AOI because of its short endurance caused by the limitation of battery capacity or heavy payload weight. Therefore, some research has proposed multi-UAV VRP formulations to address the multi-UAV scanning problem. Arman Nediati et al. proposed a multi-UAV VRP approach to generate a two-dimensional coverage path for a post-earthquake assessment, which enabled multi-tour and multi-location coverage path-planning [20]. Avellar et al. introduced a multi-UAV coverage VRP formulation for a remote sensing mission. Their approach applied the traditional VRP with practical aspects such as the number of operators and setup time [2].

This paper presents a novel framework of a multi-UAV three-dimensional trajectory optimization algorithm to execute a coverage path-planning mission in a large/irregular terrain environment. This is an extension of the work of a single UAV-based three-dimensional trajectory optimization for an aerial scanning mission [6]. The proposed approach includes the generation of an approximated terrain model, the determination of scanning waypoints, and the VRP-based multi-UAV trajectory optimization. The approximated terrain model applies a Non-Uniform Rational B-Spline (NURBS)-based fitting method to capture the geometry characteristics of a terrain. Using the approximated terrain model, it obtains scanning waypoints depending on the terrain surface topology. Then, the framework adopts a

distance-constrained multi-vehicle routing problem to solve the multi-UAV coverage path-planning problem. The main contributions of this work are:

- NURBS-based terrain surface model that improves computational efficiency
- A framework for multi-UAV three-dimensional trajectory optimization that considers terrain topology characteristics, vehicle performance, and a multi-UAV vehicle routing problem to minimize total mission time.

This paper is organized as follows. The paper introduces the NURBS-based surface fitting method in Section 2.1. Section 2.2 presents the waypoints selection and view angle determination depending on the topology characteristics from the approximated model. Section 2.3 introduces the formulation of a multi-UAV vehicle routing problem to generate optimal trajectories of multiple UAVs. Section 3 demonstrates the proposed multi-UAV trajectory optimization algorithm with realistic example scenarios. The conclusions are summarized in Section 4.

2 Framework for Multi-UAV Trajectory Optimization in an Irregular Terrain Environment

The multi-UAV trajectory optimization problem for an irregular terrain scanning mission is inherently challenging because the terrain is not flat, which requires more sophisticated trajectory based on the terrain topology and has impacts on the required flight endurance. Therefore, the multi-UAV trajectory optimization needs to consider terrain geometry characteristics, satisfy UAV endurance constraints, and account for a sensor specifications. However, typical aerial coverage path-planning problems only deal with vehicle characteristics and sensor specifications, neglecting terrain topological features. To improve the traditional coverage path-planning problem, this paper proposes a novel framework accommodating three major steps summarized in Figure 1. In the first step, the terrain model is generated by the NURBS-based surface fitting method to create a computationally efficient terrain model. In the second step, waypoints are specified by different view angles. This step uses terrain topology information based on the NURBS-based surface model. In the last step, a multi-UAV trajectory optimization is formulated which is based on a distance-constraint and arc-based vehicle routing problem.

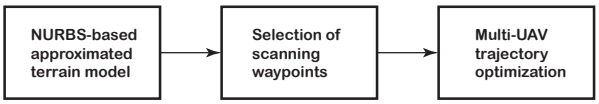


Fig. 1 Framework of Multi-UAV Trajectory Optimization

2.1 NURBS-based Terrain Modeling

Defining a simplified terrain model based on LiDAR or optical sensor information is a difficult task because inherent sensor noise is inevitable, and a typical terrain dataset includes a large amount of information. Therefore, to address these difficulties, many terrain modeling methods have been introduced. Triangulation based on terrain point cloud information is a popular technique due to its simplicity. However, because of the inherent noise and large dataset, the triangulation-based approach leads to a low signal to noise ratio, which is not efficient with respect to accurate modeling and computational perspectives. Therefore, to resolve these drawbacks, a interpolation-based terrain modeling is adopted, which uses proximity points to determine a new point location [13]. Another terrain modeling technique is Gaussian Process (GP)-based approaches [28]. One of the GP-based terrain modeling methods combines with KD-Trees to handle large amounts of terrain information [27]. The other GP-based approach uses sparse GP-based terrain modeling, which uses pseudo-input points and applies a variation learning method to generate a terrain prediction model [6][25]. This sparse-GP based terrain model enables us to generate a scalable terrain model, which reduces the computational cost to $O(n^2m)$ from $O(n^3)$, the typical full-GP computational cost, where n is the number of training data points and m is the number of pseudo-input points. However, the terrain modeling result of the GP-based approach is highly dependent on the formulation of the covariance function. In other words, if the shape of a terrain is highly complex, this GP-based approach may not precisely capture its topology features. The other terrain modeling technique is the NURBS-based surface fitting method. The NURBS function is a well-known method in computer graphics fields, which generates 2D or 3D flexible and versatile models. Thus, the NURBS-based curve and surface modeling method has been adopted in various areas, such as the reconstruction of medical imaging, reverse engineering, and CAD modeling software [4][14]. The NURBS-based fitting method has been successfully implemented to reconstruct a digital terrain model using a triangulated irregular network [30]. Because of its flexibility and computationally efficient characteristics, we adopt the NURBS surface fit-

ting method to reconstruct a terrain model from point cloud information. The fundamental mathematical formulation of NURBS is described in literature written by Piegl et al. [21]. A NURBS curve $C(u)$ with degree p can be defined as a piecewise rational function:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{1}$$

$$= \sum_{i=0}^{n} R_i(u) P_i, \tag{2}$$

where w_i is weight, and P_i is a control point. $N_{i,p}(u)$ defined as pth-degree basis function on knot vector $\mathbf{U}$ can be specified by the Cox-de Boor recursion formulas,

$$N_{i,0}(u) = \begin{cases} 1, & if \ \ u_i \le u < u_{i+1} \\ 0, & otherwise \end{cases} \tag{3}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{4}$$

where u_i is a knot element, ($u_i \in \mathbf{U}$). In a similar way, the pth-by-qth-degree NURBS-based surface model can be expressed by two knot vectors, $\mathbf{U}$ and $\mathbf{V}$, as follows:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} P_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}} \tag{5}$$

$$= \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}(u,v) P_{i,j}, \tag{6}$$

where $N_{i,p}(u)$ and $N_{j,q}(v)$ are basis functions along the knot vectors, $\mathbf{U}$ and $\mathbf{V}$, $w_{i,j}$ is a weight, and $P_{i,j}$ is a control point. Generally, there are two ways to fit a surface: interpolation and approximation. Interpolation constructs a surface model that passes through all the data points [5]. Approximation allows the fitted surface to deviate from the given data within a specified maximum bound [10][19]. Since the data normally generated by LiDAR or optical sensors may contain measurement or computational noise, fitting through interpolation will include all this noise, which is not ideal. On the other hand, approximation with certain constraints only captures the shape and avoids the noise. For these reasons, the global NURBS surface approximation is applied in this paper. In the approximation problem, the number of control points remains unknown and is defined by the desired accuracy. For the determination of the number of control points, we apply a general process proposed by Piegl et al. [21], which is described as follows:

1. Initialize with degree one and interpolate a surface with the maximum number of allowed control points
2. Remove as many control points as possible within a preset error bound
3. If the desired degree hasn't been achieved, elevate the degree of the surface and fit a new surface model
4. Update the information of the deviation and knot vectors and return to step 2
5. Final fit with the reduced-size knot vectors

In step 1, we initialize the knot vector and create a first degree model by using the data points as initial control points. The method to build the initial knot vector is described below:

$$d = \sum_{k=1}^{n} |\mathbf{Q}_k - \mathbf{Q}_{k-1}| \tag{7}$$

$$\bar{u}_0 = 0, \quad \bar{u}_n = 1 \tag{8}$$

$$\bar{u}_k = \bar{u}_{k-1} + \frac{|\mathbf{Q}_k - \mathbf{Q}_{k-1}|}{d} \quad k = 1, \ldots, n-1 \tag{9}$$

$$\mathbf{U} = \{0, 0, \bar{u}_1, \ldots, \bar{u}_{n-1}, 1, 1\} \tag{10}$$

This step measures the 3D distance between the data points, builds a $\bar{u}_k$ vector based on it, and uses Equation(10) to build the initial knot vector.

In step 2, a knot removal algorithm is introduced. This knot removal technique has been widely applied in NURBS-based surface fitting to create a surface fitting model with the minimum number of control points. Kjellander first discussed removing one knot from the knot vector to smooth the curve [16]. Later, Farin discussed locally fairing a B-Spline curve, and Lyche et al. discussed applying a knot removal algorithm for parametric splines and surfaces [11][18]. Recently, Sederberg has applied the knot removal technique to T-Spline simplification and local refinement [22]. This paper employs the knot removal algorithm in the literature written by W. Tiller [24].

Before going deeper into the mathematical details, a brief introduction of the knot removal algorithm will be given here. The knot removal algorithm aims to keep reducing the number of control points as long as the surface model is within some deviation bounds Γ. It is obvious that removing one knot in the knot vector changes the basis functions and thus has an impact on the surface model. Therefore, the quantification of the deviation between the original surface and the surface after removing one knot is required. If the deviation calculated by removing one knot is within the bound, this knot will be removed from the knot vector and the deviation by this process will be saved for later calculation. On the contrary, if the computed deviation for removing one knot exceeds the predetermined deviation bound, that knot will be kept and flagged. If none of the remaining knots are removable, the resulting knot vector

will be the shortest knot vector with bound Γ. A surface model with fewer knots to represent a raw dataset can save quite a lot of computational time. Looking back to the quantification of the deviation resulted from removing one knot, the following equations determine whether one knot is removable or not. Let B_r^j represent the distance between the new control points and the original one in the jth curve if removing the rth knot of the knot vector in the u direction. Tiller [24] proved that the maximum deviation $|\mathbf{S}(u,v)-\hat{\mathbf{S}}(u,v)|$ between the original surface model and the new one satisfies the following conditions:

$$
\begin{aligned}
&if\ (p+s)\ mod\ 2=0,\ set\ k=(p+s)/2\ and\\
&B_r^j=\left|\mathbf{P}_{r-k,j}-\alpha_{r-k}\mathbf{P}^1_{r-k+1,j}-(1-\alpha_{r-k})\mathbf{P}^1_{r-k-1,j}\right|\\
&where\ \alpha_{r-k}=\frac{u_r-u_{r-k}}{u_{r-k+p+1}-u_{r-k}}\\
&Then\ \ |\mathbf{S}(u,v)-\hat{\mathbf{S}}(u,v)|\le N_{r-k,p}(u)\sum_{j=0}^{m}N_{j,q}(v)B_r^j=\varphi_1
\end{aligned}
\tag{11}
$$

$$
\begin{aligned}
&if\ (p+s)\ mod\ 2=1,\ set\ k=(p+s+1)/2\ and\\
&B_r^j=|\mathbf{P}^1_{r-k,j}-\mathbf{P}^1_{r-k+1,j}|\\
&Then\ \ |\mathbf{S}(u,v)-\hat{\mathbf{S}}(u,v)|\\
&\le(1-\alpha_{r-k+1})N_{r-k+1,p}(u)\sum_{j=0}^{m}N_{j,q}(v)B_r^j=\varphi_2\\
&with\ \ \alpha_{r-k+1}=\frac{u_r-u_{r-k+1}}{u_{r-k+p+2}-u_{r-k+1}}
\end{aligned}
\tag{12}
$$

To judge whether the rth knot is removable or not, the process checks two conditions, $\varphi_1 \le \Gamma$ or $\varphi_2 \le \Gamma$. If φ_1 or φ_2 does not exceed the error bound, it removes that knot, and updates the deviation information and the knot vector. The algorithm selects the knot with the minimum deviation and iteratively removes the knots until the accumulated deviation exceeds the bound. The pseudocode in Algorithm 1 will describe this process. Step 3 applies the degree elevation because it reduces the number of control points and captures highly non-linear responses as well, which improves computational efficiency. In general, the rule of the degree elevation starts from a low-degree curve. To create a fitting surface with a given degree of a curve, the typical NURBS-based surface approximation uses the least squares method, which can mathematically be written as:

$$
f=\sum_{k=0}^{r}\sum_{l=0}^{s}|Q_{k,l}-S(\bar{u}_k,\bar{v}_l)|^2, \tag{13}
$$

Algorithm 1 Knot Removal Algorithm

Inputs: $n, m, p, q, \mathbf{U}, \mathbf{V}, \mathbf{P}, \bar{u}, \bar{v}, e_k, E$
Outputs: $e_k, \hat{n}, \hat{m}, \hat{\mathbf{U}}, \hat{\mathbf{V}}, \hat{\mathbf{P}}$
Get the values B_r^j for all distinct interior knots
while 1 **do**
  Find knot with the smallest B_r bound
  if $B_r = \infty$ **then**
    break
  end if
  Use Eq.(11) and Eq.(12) to compute the new error and update e_k
  if knot is removable (all the errors are within the bound) **then**
    Remove the knot and update the knot vectors
    if No more internal knots **then**
      break
    end if
  else
    Set this B_r to ∞
  end if
end while

where $Q_{k,l}$ is the data point and $S(\bar{u}_k, \bar{v}_l)$ is the corresponding point on the fitting model. In the NURBS-based surface approximation problem, solving the above function requires a large amount of computational resources. However, this process can be optimized and improved to be more computationally favorable by using the concepts of curve approximation which applies the least squares technique to each row(column) of data first, then uses the resulting control points to do another curve fitting across each column(row) to obtain the final control points grid. For each curve approximation, the control point vector $\mathbf{P}$ is solved through the following linear equation:

$$
(N^T N)\mathbf{P}=\mathbf{R} \tag{14}
$$

where,

$$
N=\begin{bmatrix} N_{1,p}(\bar{u}_1) & \dots & N_{n-1,p}(\bar{u}_1)\\ \vdots & \ddots & \vdots\\ N_{1,p}(\bar{u}_{m-1}) & \dots & N_{n-1,p}(\bar{u}_{m-1})\end{bmatrix} \tag{15}
$$

$$
R=\begin{bmatrix} N_{1,p}(\bar{u}_1)\mathbf{R}_1+\dots+N_{1,p}(\bar{u}_{m-1})\mathbf{R}_{m-1}\\ \vdots\\ N_{n-1,p}(\bar{u}_1)\mathbf{R}_1+\dots+N_{n-1,p}(\bar{u}_{m-1})\mathbf{R}_{m-1}\end{bmatrix} \tag{16}
$$

$$
\mathbf{R}_k=\mathbf{Q}_k-N_{0,p}(\bar{u}_k)\mathbf{Q}_0-N_{n,p}(\bar{u}_k)\mathbf{Q}_m \tag{17}
$$

After creating a model by surface approximation, the deviation information, mentioned in step 4, needs to be updated. To compute the deviation, the point inversion algorithm, based on Newton-Raphson method [21], that searches for the nearest point on the fitting surface, is

applied. With the projected point, the deviation can be measured through a 3D distance calculation. For each projected point on the surface, one condition should be satisfied: The dot product of the vector, which starts from the projected point on the surface and points to the true data points, and the first derivative at the projected point on the surface should be zero. Therefore, at the projected point, the following dot product functions should be satisfied:

$$\begin{aligned} f(u,v) &= \mathbf{r}(u,v)\cdot\mathbf{S}_u(u,v) = 0 \\ g(u,v) &= \mathbf{r}(u,v)\cdot\mathbf{S}_v(u,v) = 0 \\ \mathbf{r}(u,v) &= \mathbf{S}(u,v) - \mathbf{P} \end{aligned} \tag{18}$$

$\mathbf{P}$ is the point in the point cloud. Given an initial guess, we apply the Newton-Raphson method to iteratively search for the final solution that satisfies Equation (18). The following linear equation is being solved for each iteration and updates the projection point:

$$\begin{aligned} J_i\delta_i &= \kappa_i \\ \delta_i &= \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} u_{i+1}-u_i \\ v_{i+1}-v_i \end{bmatrix} \\ J_i &= \begin{bmatrix} f_u & f_v \\ g_u & g_v \end{bmatrix} \\ &= \begin{bmatrix} |\mathbf{S}_u|^2 + \mathbf{r}\cdot\mathbf{S}_{uu} & \mathbf{S}_u\cdot\mathbf{S}_v + \mathbf{r}\cdot\mathbf{S}_{uv} \\ \mathbf{S}_u\cdot\mathbf{S}_v + \mathbf{r}\cdot\mathbf{S}_{vu} & |\mathbf{S}_v|^2 + \mathbf{r}\cdot\mathbf{S}_{vv} \end{bmatrix} \\ \kappa_i &= -\begin{bmatrix} f(u_i,v_i) \\ g(u_i,v_i) \end{bmatrix} \end{aligned} \tag{19}$$

The first-order gradients $\mathbf{S}_u$ and $\mathbf{S}_v$ and second-order derivatives $\mathbf{S}_{uu}$, $\mathbf{S}_{vv}$ and $\mathbf{S}_{uv}$ are calculated through the finite difference method [23]. The following equations describe how that works for the second-order gradient. Forward difference and backward difference are applied at the boundary while central difference is applied to the rest of the curve.
Central difference:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{20}$$

Forward difference:

$$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} \tag{21}$$

Backward difference:

$$f''(x) \approx \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2} \tag{22}$$

In Equations 20 $\sim$ 22, h is the spacing and should be small enough to achieve an accurate result. The convergence criteria to control whether to stop finding a more accurate projection point are given by:

$$|(u_{i+1}-u_i)\mathbf{S}_u(u_i,v_i) + (v_{i+1}-v_i)\mathbf{S}_v(u_i,v_i)| \le \epsilon_1 \tag{23}$$

$$|\mathbf{S}(u_i,v_i) - \mathbf{P}| \le \epsilon_1 \tag{24}$$

$$\frac{|\mathbf{S}_u(u_i,v_i)\cdot(\mathbf{S}(u_i,v_i)-\mathbf{P})|}{|\mathbf{S}_u(u_i,v_i)||\mathbf{S}(u_i,v_i)-\mathbf{P}|} \le \epsilon_2 \tag{25}$$

$$\frac{|\mathbf{S}_v(u_i,v_i)\cdot(\mathbf{S}(u_i,v_i)-\mathbf{P})|}{|\mathbf{S}_v(u_i,v_i)||\mathbf{S}(u_i,v_i)-\mathbf{P}|} \le \epsilon_2 \tag{26}$$

The algorithm to find the projection point on the surface model can be summarized in Algorithm 2. In summary, the whole approximation process from step 1 to step 5 can be described in Algorithm 3: To demonstrate the surface approximation process, a numerical experiment is conducted using a simple example to show how

Algorithm 2 Point Projection Algorithm

Input: $n, m, p, q, \mathbf{U}, \mathbf{V}, \mathbf{P}$
Output: $\bar{u}, \bar{v}$
$u_0, v_0 \leftarrow$ *initial value*
Check the last three three conditions in the convergence criteria
while One of them not satisfied **do**
  Call Eq. (19) to update the initial guess
  Check with all the conditions in the convergence criteria
  if One of them is satisfied **then**
    return u, v
  end if
end while

Algorithm 3 Surface Approximation Method

Inputs:
r, the number of data points for each row
s, the number of data points for each column
p, the final degree in the row direction
q, the final degree in the column direction
Q, the point cloud
E, the maximum error bound
Outputs: $\mathbf{U}, \mathbf{V}, \mathbf{P}$
Compute $\bar{u}_k$ and $\bar{v}_l$
Use Q_k to interpolate a $1st$ degree surface
for $deg = 1$; $deg \le p$; $deg++$ **do**
  Call the algorithm to remove knots
  if $deg = p$ **then**
    break
  end if
  Increase the surface degree by degree elevation
  Fit a new surface model with the updated knot vectors
  Use point inversion technique to update the error and $\bar{u}_k$,$\bar{v}_l$
end for
if No more knots have been removed in the last knot removal process **then**
  return
end if
Do final fitting and update the knot vectors and error
Make a final call to the knot removal algorithm
return

the knots are removed and compare between the knot vectors with the original point cloud and the knot vectors with point cloud from the approximation model. The simple surface model is

$$f(x, y) = y \sin x - x \cos y \tag{27}$$

From this surface model, we collect 41×41, the number of points, and set the error bound Γ to be 0.1. Figure 2(a) illustrates the point cloud generated from Equation (27), and Figure 2(b) shows the approximation result. Figure 2(c) is the error calculated by projecting all the data points in the point cloud to the surface model, and thus calculates the error distance between the projected point and the original one. Even if the error bound is set to 0.1, only a small portion of the points have an error greater than 0.03, which implies the surface model fits the point cloud well. As the result of the knot removal process, we are left with 23×21 control points. The original knot vector grid represents the initial knot vectors generated by using all the data points as the control points, which has a size of 43×43. (Note that knot vector will have a different size with the control points). The size of the knot vector grid after applying the knot removal algorithm is 26×23. Since degree elevation will duplicate the original knot vectors, some of the final knots will overlap with each other. Therefore, the distinct knots in the knot vectors may not equal the knot vector size. Results reveal that even if the remaining grid is not evenly divided, it still has some pattern in which more control points remain where the surface has a larger curvature. Results also show that the algorithm is capable of the reduction of the number of control points, which is computationally more efficient.

2.2 Determination of Scanning Waypoints

The method for selecting the waypoints uses the approach proposed by Choi et al. [6] in which waypoints which are determined from the center of the grid cells. The size of a grid cell is computed by the Field of View (FOV), the ground coverage of a single image. Its size defines grid cells in the NURBS-based terrain model. Note that the size of each grid cell is defined by sensor specifications, a Ground Sampling Distance (GSD) requirement, and an overlap ratio, $min(\tilde{G}_x, \tilde{G}_y)$, where $\tilde{G}_x$ and $\tilde{G}_y$ are the x and y ground sampling distance. More detailed information can be found in the literature [6].

Once the size of the grid cells is defined, three-dimensional flight waypoints are specified by two different view angle conditions: vertical and normal views. In the vertical view, the vertical offset is considered, which translates the center of all the grid cells, $\mathbf{x}_c$, into the z-direction with the offset distance H. We note that H is computed from the GSD requirement, and sensor specifications, and z-direction is the down direction in the North East Down (NED) coordinate system. Mathematically, UAV scanning waypoints $\mathbf{x}_w$ by the vertical view can be given by

$$\mathbf{x}_w = \mathbf{x}_c + [0 \;\; 0 \;\; H]^T, \tag{28}$$

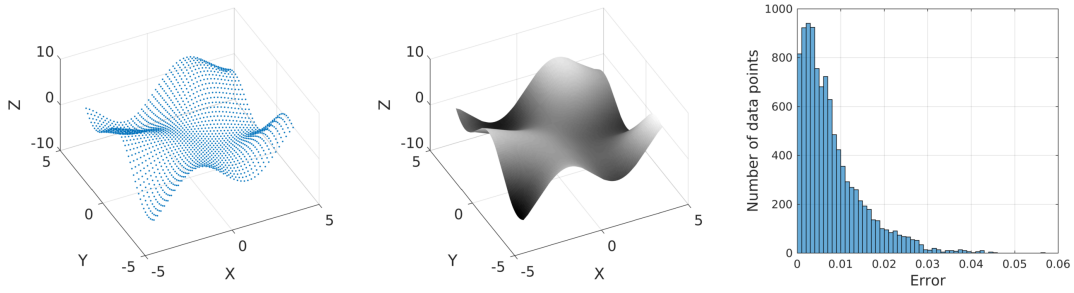
Because the optical sensor points down, the vertical view approach may not be a dense point cloud after imaging processing or generate a distorted image when the surface slope of a terrain is steep. To minimize the impact of the terrain surface slope, Choi et. al suggested the normal offset approach using the gradient information of an approximated terrain model. This paper applies the same approach using the gradient information of the approximated NURBS-based terrain model. The waypoints $\mathbf{x}_w$ with the normal view can be written as

$$\mathbf{x}_w = \mathbf{x}_c + N^T H, \tag{29}$$

Here, N is determined from the NURBS gradient information, which is $N = [\frac{\partial f}{\partial x}, \;\; \frac{\partial f}{\partial y}, \;\; -1]$. The NURBS gradient information is applied to the finite difference method.

2.3 Multi-UAV Trajectory Optimization

The framework of the distance-constrained UAS trajectory optimization proposed by Choi et al. can efficiently scan an irregular terrain to collect aerial images [6]. However, this trajectory optimization algorithm is limited in a large AOI since it solves the trajectory optimization problem for a single UAV. To address a large area of interest that requires multiple UAVs, this paper suggests a multi-UAV trajectory optimization framework that is the extended version of the distance-constrained vehicle routing problem Choi et al. proposed. The multi-UAV trajectory optimization is defined on a graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ indicates a set of nodes, $N = \{0, 1, \cdots, n, n+1\}$, and $\mathcal{A}$ is a set of arcs, $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$ that represents the connectivity between two nodes. The 0th and the $n + 1$th nodes are an initial depot position, and a terminal depot position that is an artificial depot node, respectively. Note that the physical location of the two nodes (initial and terminal depot positions) are the same. We also define a set of waypoints $\mathcal{W} = \{1, \cdots, n\}$ that is a subset of the nodes N, $(\mathcal{W} \in \mathcal{N})$. The optimization problem involves a set of UAVs, $(\mathcal{V} = \{1, \cdots, m\})$. To formulate the multi-UAV vehicle routing problem, there are three main assumptions. All UAVs start their missions from the initial



(a) Point cloud generated from the function (b) Approximation result (c) Error between the point cloud and surface model

Fig. 2 Example to show the surface approximation process

node (0th node) and finish their missions on the terminal nodes ($n+1$th node). Each waypoint must be scanned by a single UAV. Each route of a UAV must satisfy its endurance requirement. Based on these assumptions, the cost function that minimizes the total distance of all the UAVs is defined as:

$$J = \min \sum_{k\in\mathcal{V}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} d_{ij} x_{ijk}, \tag{30}$$

where d_{ij} is the corresponding distance between two nodes (i, j) and x_{ijk} represents the edge connection. If the edge between two nodes (i, j) by the kth vehicle is connected, then x_{ijk} is defined as 1, otherwise, it is defined as 0. The constraint to visit all the waypoints and meet the vehicle endurance constraint can be written as:

$$g_1 = \sum_{k\in\mathcal{V}} \sum_{j\in N} x_{ijk} = 1, \quad (\forall i \in \mathcal{W}) \tag{31}$$

$$g_2 = \sum_{j\in\mathcal{N}} x_{0jk} = 1, \quad (\forall k \in \mathcal{V}) \tag{32}$$

$$g_3 = \sum_{i\in\mathcal{N}} x_{i(n+1)k} = 1 \quad (\forall k \in \mathcal{V}) \tag{33}$$

$$g_4 = \sum_{i\in\mathcal{N}} x_{ihk} - \sum_{j\in\mathcal{N}} x_{hjk} = 0, \quad (\forall h \in \mathcal{W}, \forall k \in \mathcal{V}) \tag{34}$$

$$g_5 = \sum_{j\in\mathcal{N}} y_{ijk} - \sum_{j\in\mathcal{N}} y_{jik} - \sum_{j\in\mathcal{N}} d_{ijk} x_{ijk} = 0, \quad (\forall i \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{35}$$

$$g_6 = y_{0jk} = d_{0jk} x_{0jk}, \quad (\forall j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{36}$$

$$g_7 = y_{ijk} \leq (D - d_{j(n+1)k}) x_{ijk}, \quad (\forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{37}$$

$$g_8 = y_{i(n+1)k} \leq D x_{i(n+1)k}, \quad (\forall i \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{38}$$

$$g_9 = y_{ijk} \geq (d_{0ik} + d_{ijk}) x_{ijk}, \quad (\forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{39}$$

$$g_{10} = x_{ij} \in \{0, 1\}, \quad (\forall (i, j) \in \mathcal{N}), \tag{40}$$

where y_{ijk} is a flow variable that presents the distance between the ith node and the jth node given the kth UAV. The constraints $g_1 \sim g_4$ represent network construction constraints. To be more specific, the constraint g_1 forces all the waypoints $\mathcal{W}$ to be visited exactly once, the constraint g_2 ensures that all the vehicles $\mathcal{V}$ are deployed from the initial depot node (0th node), the constraint g_3 means that all the vehicles $\mathcal{V}$ arrive at the artificial node ($n+1$th node), and the constraint g_4 makes sure that a UAV arrives on the hth node and leaves on the same node (hth node). Constraint g_5 prevents sub-tours, and constraints $g_6 \sim g_9$ allow all the UAVs to satisfy the endurance constraint.

This multi-UAV vehicle routing formulation can minimize the total endurance time of UAVs when UAVs are deployed sequentially, but it may not minimize the total mission time when all the UAVs perform the aerial imaging mission at the same time. This is because minimizing the total endurance time, Equation 30, can result in large variations of mission time between UAVs. In the concurrent operation, ideally, all the UAVs would have similar mission times that can minimize the total mission time. In the vehicle routing problem, we can revise the minimization cost function into a minmax formulation that minimizes the maximum distance of all the UAVs' trajectories. The objective function of the minmax approach can be written as:

$$J = \min \max_{k\in\mathcal{V}} \left(\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} d_{ijk} x_{ijk} \right) \tag{41}$$

subject to

$$Equations(31) - (40)$$

To solve this minmax objective function using the mixed integer programming model, it needs to be converted into a general linear programming formulation that has a linear objective function and linear constraints. The

typical transformation method is introducing an additional decision variable z in the objective function as follows:

$$\bar{J} = \min z \tag{42}$$

In order to impose this decision variable, an additional constraint $\bar{g}_{10}$ that z will be greater than or equal to the total distance of each vehicle should be included with the other constraints, which are:

$$\bar{g}_1 = \sum_{k\in\mathcal{V}} \sum_{j\in N} x_{ijk} = 1, \quad (\forall i \in \mathcal{W}) \tag{43}$$

$$\bar{g}_2 = \sum_{j\in\mathcal{N}} x_{0jk} = 1, \quad (\forall k \in \mathcal{V}) \tag{44}$$

$$\bar{g}_3 = \sum_{i\in\mathcal{N}} x_{i(n+1)k} = 1 \quad (\forall k \in \mathcal{V}) \tag{45}$$

$$\bar{g}_4 = \sum_{i\in\mathcal{N}} x_{ihk} - \sum_{j\in\mathcal{N}} x_{hjk} = 0, \quad (\forall h \in \mathcal{W}, \forall k \in \mathcal{V}) \tag{46}$$

$$\bar{g}_5 = \sum_{j\in\mathcal{N}} y_{ijk} - \sum_{j\in\mathcal{N}} y_{jik} - \sum_{j\in\mathcal{N}} d_{ijk} x_{ijk} = 0, \quad (\forall i \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{47}$$

$$\bar{g}_6 = y_{0jk} = d_{0jk} x_{0jk}, \quad (\forall j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{48}$$

$$\bar{g}_7 = y_{ijk} \leq (D - d_{j(n+1)k}) x_{ijk}, \quad (\forall i, j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{49}$$

$$\bar{g}_8 = y_{i(n+1)k} \leq D x_{i(n+1)k}, \quad (\forall i \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{50}$$

$$\bar{g}_9 = y_{ijk} \geq (d_{0ik} + d_{ijk}) x_{ijk}, \quad (\forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}) \tag{51}$$

$$\bar{g}_{10} = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} d_{ijk} x_{ijk} \leq z, \quad (\forall k \in \mathcal{V}) \tag{52}$$

$$\bar{g}_{11} = x_{ij} \in \{0, 1\}, \quad (\forall (i, j) \in \mathcal{N}), \tag{53}$$

The minmax-based reformulated vehicle routing problem leads to the actual minimum of the total mission time.

3 Numerical Simulation

For the numerical simulation, we assume that the UAS platform is the DJI Matrice 210, and the optical camera sensor is the ZenmuseX5S model summarized in Table 1. The simulation parameters are illustrated in Table 2. We note that the operating altitude is defined from the GSD requirement, sensor specifications, and the overlap ratio. The vehicle endurance is only for an aerial imaging mission segment and does not include the endurance for takeoff and landing segments. As realistic aerial imaging mission areas, San Diego around Point Loma and Death Valley are selected and their point cloud dataset is collected. The simulation results compare four different algorithms: min-vertical view, min-normal view, minmax-vertical view, and minmax-normal view. The 'min-vertical view'indicates that it solves the minimization optimization problem with the vertical view angle, and the 'min-normal view'indicates that it solves the minimization optimization problem with the normal view angle on the terrain surface shape. The 'minmax-vertical view'means that it solves the minmax optimization problem with the vertical view angle, and the 'minmax-normal view'solves the minmax optimization problem with the normal view angle on the terrain surface. Figures 3(a), 3(b), and 3(c) show the image of Point Loma in San Diego, the raw point cloud (262,144 points), and the result of the NURBS-based terrain model. The NURBS-based approximation model visually shows that it can precisely capture the topology features in the given area. To quantify the quality of the NURBS-fitting model, Mean Squared Error (MSE) is computed, which is a common metric to inspect the quality of the terrain approximation model [27][28]. The MSE is 8.966 square meters, which is little high because the area around the ocean changes the slope radically. Figure 4 shows the results of the multi-UAV trajectory optimization based on different objective functions and view angles. All four optimization results cover the entire AOI and require three UAVs to complete the aerial imaging mission. Table 3 summarizes the scanning time of each UAV depending on the optimization problem. As expected, all three UAVs of each method satisfy the endurance constraint. We can also observe that the results from the minimization objective function have a short total mission time that is the sum of all the UAVs' scanning mission times. On the other hand, the result from the minmax objective function has the shortest mission time of all the UAVs. This implies that when we operate UAVs sequentially, the trajectory result from the minimization objective is a better approach, but when we operate UAVs simultaneously, the trajectory result from the minmax objective function is more attractive. Figures 3(d), 3(e) and 3(f) present the Google image of the Death Valley area, the raw point cloud (409,600 points), and the result of the NURBS-based terrain model. The MSE of the NURBS-based approximation model is 0.760 square meter, which is better than the result for San Diego. The reason is that this terrain has more moderate curvature compared to the terrain of San Diego and a larger number of points. Figure 5 represents the trajectory optimization results depending on the optimization formulation. All the trajectories are able to generate the complete scanning paths in the given AOI. This mission requires three UAVs to cover the AOI. Table

4 describes the summary of the experiment results regarding the scanning time of each UAV. The results show that all their scanning times meet the endurance constraint. Like the previous San Diego result, the optimization results with the minimization objective function provides the trajectories with the minimum total mission time. On the other hand, the minmax optimization results lead to the shortest trajectories when UAVs are deployed simultaneously.

Table 1 Specifications of Zenmuse X5S

Sensor width	17.3 (mm)
Sensor height	13 (mm)
Resolution	20.8 (Mpix)
Lens Focal length	9 (mm)

Table 2 Numerical simulation parameters

Vehicle speed	8 (m/s)
Vehicle endurance	10 (minutes)
Ground Sampling Distance (GSD)	0.05 (m)
Overlap ratio	0.4
Operating altitude	150 (m) in AGL

4 Conclusion

The approach proposed in this paper features a multi-UAV optimal scanning trajectory algorithm that consists of a NURBS-based terrain approximation, waypoint selection depending on terrain surface shape and view angle, and a distance-constrained multi-UAV vehicle routing problem. In the framework, we suggested four optimization structures: min-vertical view, min-normal view, minmax-vertical view, and minmax-normal view. These formulations are tested and compared in numerical simulations using realistic aerial imaging scenarios, San Diego and Death Valley. Numerical simulations with four different algorithms are conducted to compare their performances in terms of the total scanning time. Results indicate that the optimization solution with a minimization cost function provides a better solution for the sequential UAVs operation concept, while the minmax optimization solution is a more desirable method for the concurrent UAVs operation concept. In summary, the proposed approach can precisely capture terrain geometry features and generate multi-UAV trajectories based on an area of interest and system constraints. The proposed framework is also a flexible structure since one can easily change the terrain approximation technique and the formulation of the vehicle routing optimization problem with diverse operational constraints and vehicle performance characteristics.

Acknowledgements This paper is a major enhancement of the ICUAS 2018 accepted paper.

References

1. Acar, E.U., Choset, H., Rizzi, A.A., Atkar, P.N., Hull, D.: Morse decompositions for coverage tasks. The International Journal of Robotics Research **21**(4), 331–344 (2002)
2. Avellar, G.S., Pereira, G.A., Pimenta, L.C., Iscold, P.: Multi-UAV routing for area coverage and remote sensing with minimum time. Sensors **15**(11), 27783–27803 (2015)
3. Bircher, A., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., Siegwart, R.: Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 6423–6430. IEEE (2015)
4. Brujic, D., Ainsworth, I., Ristic, M.: Fast and accurate NURBS fitting for reverse engineering. The International Journal of Advanced Manufacturing Technology **54**(5-8), 691–700 (2011)
5. Carr Jonathan C., W.R.F., Beatson., R.K.: Surface interpolation with radial basis functions for medical imaging. IEEE transactions on medical imaging **16**(1), 96–107 (1997)
6. Choi, Y., Choi, Y., Briceno, S., Mavris, D.N.: Three-dimensional UAS trajectory optimization for remote sensing in an irregular terrain environment. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS) (2018)
7. Choi, Y., Jimenez, H., Mavris, D.N.: Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories. Robotics and Autonomous Systems **98**, 158–173 (2017)
8. Choi, Y., Payan, A.P., Briceno, S.I., Mavris, D.N.: A framework for unmanned aerial systems selection and trajectory generation for imaging service missions. 2018 Aviation Technology, Integration, and Operations Conference (2018)
9. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science **6**(1), 80–91 (1959)
10. Dierckx, P.: Curve and surface fitting with splines. Oxford university press (1995)
11. Farin Gerald, e.a.: Fairing cubic b-spline curves. Computer Aided Geometric Design **4**(1-2), 91–103 (1987)
12. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous Systems **61**(12), 1258–1276 (2013)
13. Hameed, I.A., la Cour-Harbo, A., Osen, O.L.: Side-to-side 3D coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths. Robotics and Autonomous Systems **76**, 36–45 (2016)
14. Iglesias, A., Galvez, A., Avila, A.: Immunological approach for full NURBS reconstruction of outline curves from noisy data points in medical imaging. IEEE/ACM transactions on computational biology and bioinformatics (1), 1–1 (2017)

15. Jing, W., Polden, J., Lin, W., Shimada, K.: Sampling-based view planning for 3D visual coverage task with unmanned aerial vehicle. In: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pp. 1808–1815. IEEE (2016)
16. Kjellander, J.A.: Smoothing of cubic parametric splines. Computer-Aided Design **15**(3), 175–179 (1983)
17. Li, Y., Chen, H., Er, M.J., Wang, X.: Coverage path planning for uavs based on enhanced exact cellular decomposition method. Mechatronics **21**(5), 876–885 (2011)
18. Lyche, T., Mrken., K.: Knot removal for parametric b-spline curves and surfaces. Computer Aided Geometric Design **4**(3), 217–230 (1987)
19. Mongus, D., Lukač, N., Žalik, B.: Ground and building extraction from LiDAR data based on differential morphological profiles and locally fitted surfaces. ISPRS Journal of Photogrammetry and Remote Sensing **93**, 145–156 (2014)
20. Nedjati, A., Izbirak, G., Vizvari, B., Arkat, J.: Complete coverage path planning for a multi-UAV response system in post-earthquake assessment. Robotics **5**(4), 26 (2016)
21. Piegl, L., Tiller, W.: The NURBS book. Springer Science & Business Media (2012)
22. Sederberg Thomas W., e.a.: T-spline simplification and local refinement. ACM transactions on graphics **23**(3), 276–283 (2004)
23. Smith, G.D.: Numerical solution of partial differential equations: finite difference methods. Oxford university press (1985)
24. Tiller, W.: Knot-removal algorithms for NURBS curves and surfaces. Computer-Aided Design **24**(8), 445–453 (1992)
25. Titsias, M.: Variational learning of inducing variables in sparse Gaussian processes. In: Artificial Intelligence and Statistics, pp. 567–574 (2009)
26. Torres, M., Pelta, D.A., Verdegay, J.L., Torres, J.C.: Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. Expert Systems with Applications **55**, 441–451 (2016)
27. Vasudevan, S., Ramos, F., Nettleton, E., Durrant-Whyte, H.: Gaussian process modeling of large-scale terrain. Journal of Field Robotics **26**(10), 812–840 (2009)
28. Vasudevan, S., Ramos, F., Nettleton, E., Durrant-Whyte, H.: Non-stationary dependent gaussian processes for data fusion in large-scale terrain modeling. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 1875–1882. IEEE (2011)
29. Zelinsky, A., Jarvis, R.A., Byrne, J., Yuta, S.: Planning paths of complete coverage of an unstructured environment by a mobile robot. In: Proceedings of international conference on advanced robotics, vol. 13, pp. 533–538 (1993)
30. Zhong, D., Liu, J., Li, M., Hao, C.: NURBS reconstruction of digital terrain for hydropower engineering based on tin model. Progress in Natural Science **18**(11), 1409–1415 (2008)

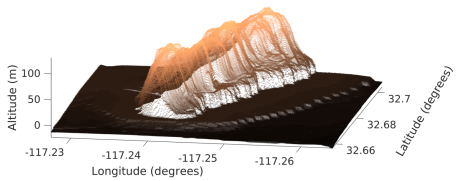
Table 3 Result of multi-UAV trajectory optimization in San Diego

	Scanning mission time (sec)			
	Minimization + Vertical view	Minimization + Normal view	Minmax + Vertical view	Minmax + Normal view
UAV-1	553.34	550.52	573.67	542.17
UAV-2	597.74	403.72	579.28	548.84
UAV-3	526.85	580.67	580.92	548.44

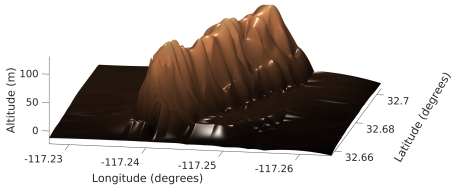
Table 4 Result of multi-UAV trajectory optimization in Death Valley

	Scanning mission time (sec)			
	Minimization + Vertical view	Minimization + Normal view	Minmax + Vertical view	Minmax + Normal view
UAV-1	572.53	459.92	554.64	580.50
UAV-2	593.36	595.25	542.08	581.56
UAV-3	442.92	599.44	556.15	572.09

(a) Point Loma in San Diego (Google Image)

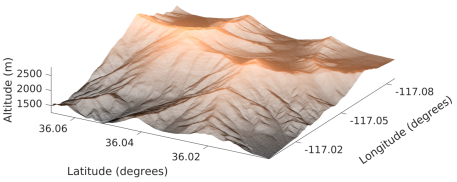


(b) Point cloud of Point Loma in San Diego (262,144 points)

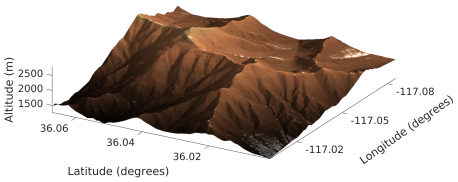


(c) NURBS-based terrain model

(d) Death Valley (Google Image)

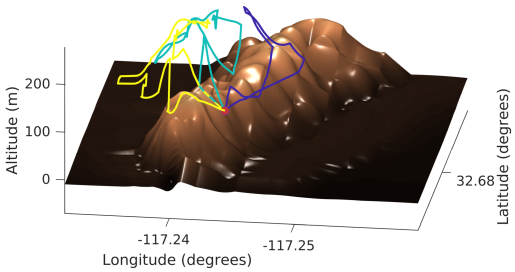


(e) Point cloud of Death Valley (409,600 points)

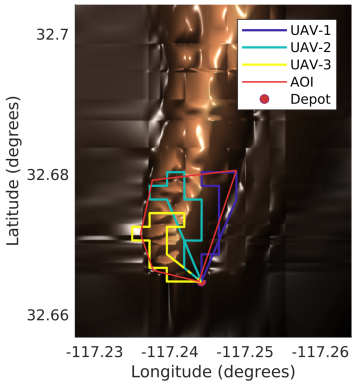


(f) NURBS-based terrain model

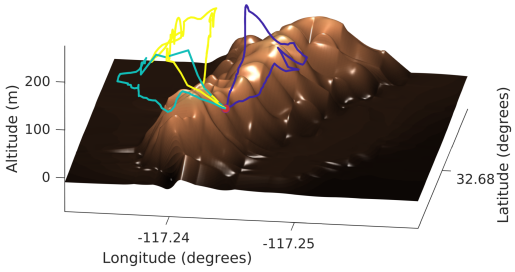
Fig. 3 Results of NURBS-based terrain models in San Diego and Death Valley



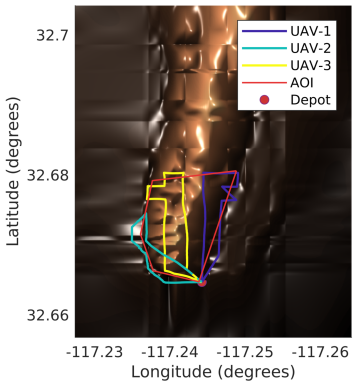
(a) 3D view (Min-vertical view)



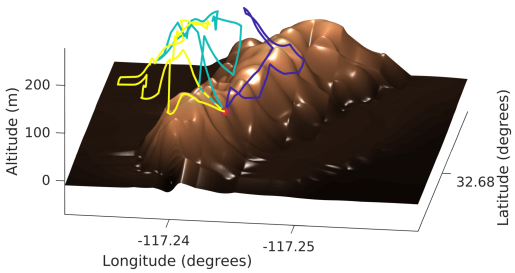
(b) Top view (Min-vertical view)



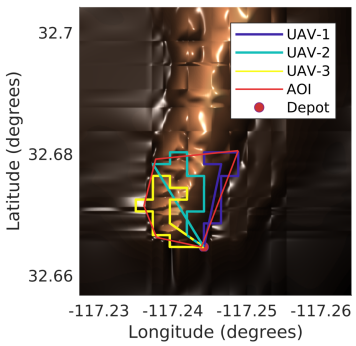
(c) 3D view (Min-normal view)



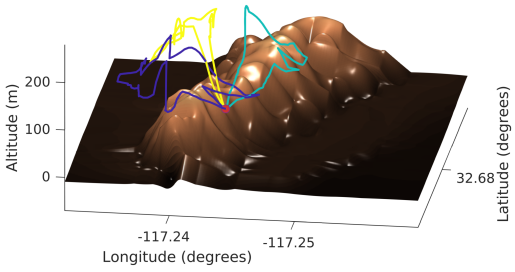
(d) Top view (Min-normal view)



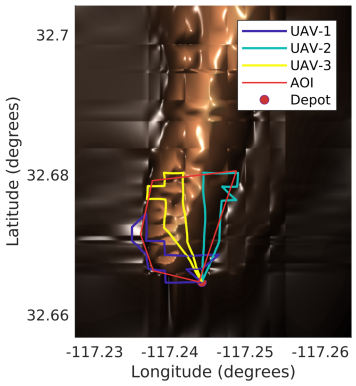
(e) 3D view (Minmax-vertical view)



(f) Top view (Minmax-vertical view)

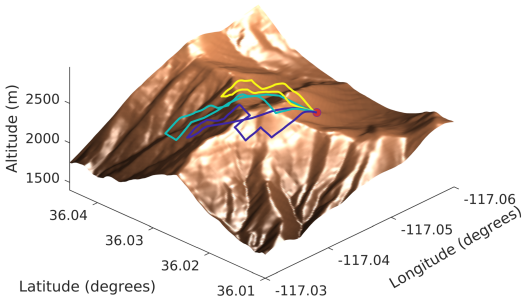


(g) 3D view (Minmax-normal view)

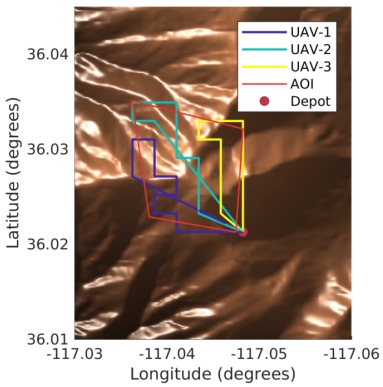


(h) Top view (Minmax-normal view)

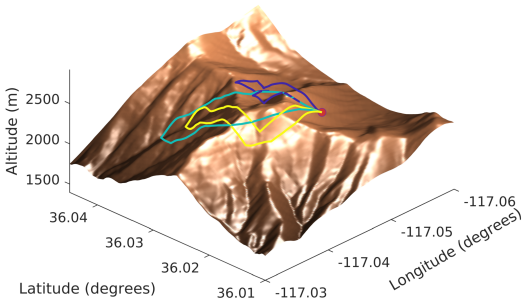
Fig. 4 Results of multi-UAV scanning trajectories in San Diego



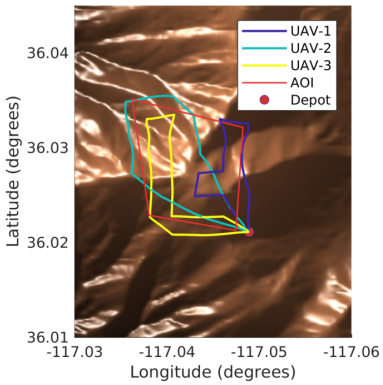
(a) 3D view (Min-vertical view)



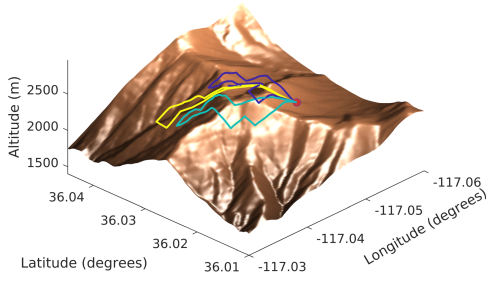
(b) Top view (Min-vertical view)



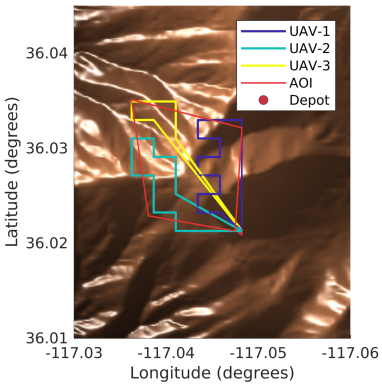
(c) 3D view (Min-normal view)



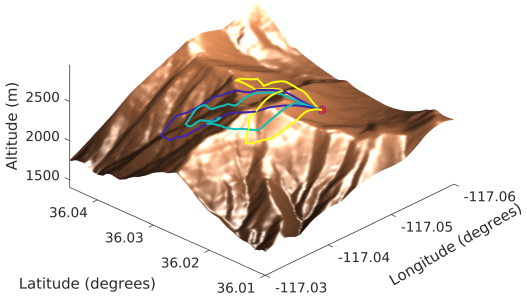
(d) Top view (Min-normal view)



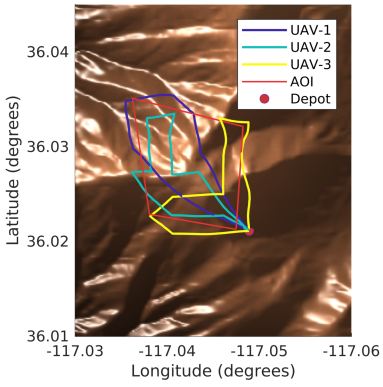
(e) 3D view (Minmax-vertical view)



(f) Top view (Minmax-vertical view)



(g) 3D view (Minmax-normal view)



(h) Top view (Minmax-normal view)

Fig. 5 Results of multi-UAV scanning trajectories in Death Valley