2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

A Multi-Resolution Approach for Discovery and 3-D Modeling of Archaeological Sites Using Satellite Imagery and a UAV-borne Camera

Huanyu Ding[1], Eric Cristofalo[2], Joseph Wang[1],
David Castañón[1], Eduardo Montijano[3], Venkatesh Saligrama[1], and Mac Schwager[4]

***Abstract*— This paper proposes a method for discovering new archaeological sites from existing satellite imagery, then building a 3-D computer model of those sites using a controlled UAV with an onboard camera. We use an unmanned vehicle and other remote surveillance sensors, coupled with onboard pattern recognition algorithms, to perform a coarse search, and subsequently a fine search to identify structures of interest. We assume the availability of two sensors. The first sensor is a low resolution camera that sweeps an area of interest, such as an imaging satellite. We process the low-resolution image data to identify tentative locations of interest and to provide confidence estimates with this identification. This information is provided to a control algorithm for an unmanned air vehicle, which plans a trajectory to inspect closely promising objects subject to fuel constraints. These close inspections provide sequences of images that are combined to give 3-D reconstructions of the area of interest, leading to accurate classification of the structure. In this paper, we describe the design of the three principal algorithms in this system: machine learning processing of coarse resolution data, the near-optimal path planning subject to fuel constraints, and the high-resolution 3-D modeling from multiple 2-D views of a site. We illustrate the performance of our system on sample LANDSAT satellite data, and using a quadrotor with an on-board camera in a laboratory environment.**

I. Introduction

The motivating application for this paper is a team of archaeologists who are interested in studying burial mound structures from the ancient Lydian civilizations. These burial mounds lie in the countryside of present day Turkey. Several of these burial mounds have already been excavated, but the country side is simply too vast to send highly-specialized teams of workers to inspect each potential future dig site. Moreover, current satellite imagery of the region is too coarse to accurately predict whether sites are indeed burial mounds or not, so one needs to obtain higher resolution information.

To assist with this problem, we propose a cyberphysical system that consists of a low-resolution imaging sensor (either from a satellite or a high-flying aircraft), an intelligent processing system to identify potential locations of interest, and an unmanned aerial vehicle (UAV) with simple imaging sensors that flies efficiently with limited fuel to collect information to construct high-resolution 3-D images of areas of interest, sufficient to identify the presence of burial mound structures. In this manner, our system provides an automated approach to efficiently locating potential burial mounds.

To handle the large number of potential locations of interest, we utilize the low-resolution satellite imagery to guide our search for locations and deployment of UAVs. In order to efficiently find the locations of burial mounds, we consider the binary classification problem, with a goal of classifying each location as either containing or not containing a burial mound. For each location, we estimate the confidence of classification decision. This estimate of classification confidence allows for many locations with confident classification to be labeled without the need for teams of workers or a UAV to be deployed. The remaining locations with ambiguous classification confidence can then be passed to the UAV path planning algorithm, with a natural goal of maximizing classification performance by collecting additional information to boost classification confidence.

Due to the fuel limitations, the UAV will be unable to collect high resolution information on all the potential site nodes. Instead, we formulate the routing and scheduling problem for the UAV to exploit the information obtained from processing the low resolution imagery, which provides a confidence that a site is a potential burial mound. The path planning problem for the UAV is to design a tour that visits a subset of sites to collect information that maximally allows for correct classification of sites while satisfying fuel limits, and returns to its original launch site.

Tsiligirides [1] first studied a constrained routing problem of this type and named it the *orienteering problem*. He proposed a Monte Carlo heuristic which selects the next node to add to the path by sampling over the normalized desirability measures of unvisited nodes. Golden et al. [2] later developed heuristic procedures to select a subset of locations to visit and a tour. Chao et al. [3] also developed heuristic solutions based on greedy selection of nodes to visit. Applications of the orienteering problem have been studied in different contexts such as persistent monitoring [4].

In this paper, we propose a different approach to path planning that considers an objective function that is closely tied to the information quality obtained by our low resolution image processing, and extends modern techniques for the solution of traveling salesperson problems, such as the Lin-Kernighan-Helsgaun (LKH) algorithm [5]. The resulting algorithm is fast and suitable for real-time path-planning in the context of our system. This algorithm will provide

[1]Department of Electrical and Computer Engineering, Boston University, United States. {hyding, joewang, dac, srv}@bu.edu.
[2]Department of Mechanical Engineering, Boston University, United States. emc73@bu.edu.
[3]Centro Universitario de la Defensa (CUD), Zaragoza, Spain. emonti@unizar.es.
[4]Department of Aeronautics and Astronautics, Stanford University, United States. schwager@stanford.edu.
This work was supported by NSF award CNS-1330008.

routes to the UAV to visit specific locations, where the UAV will collect close-up imagery of a site to compute a high-resolution 3-D reconstruction of the site, in order to determine if there is indeed a burial mound in that location.

There exists a variety of sensors and techniques for determining 3-D structure from an unknown scene, e.g., cameras, laser range scanners or RGB-D sensors. To limit the cost of the system, we have opted for a purely vision-based solution. Standard monocular cameras are cheap, lightweight, and easy to run onboard a mobile UAV while providing dense-information readings at relatively high capture rates. There are numerous ways to generate 3-D reconstructions from a series of images, such as exploiting the simple geometry of corresponding features in multiple images [6], finding solutions that minimize the least squares error between image patches [7], or by representing image depth as a probabilistic map [8]. Alternatively, the robotics community has developed Simultaneous Localization and Mapping (SLAM) algorithms [9], which are also capable of producing accurate representations of unknown environments. In our system, we use a vision-based 3-D reconstruction technique from classic results in Structure from Motion [6] in order to obtain high-fidelity 3-D reconstructions. This technique takes as input a batch of 2-D images to generate the 3-D reconstruction, and is fast enough to be computed quickly after the the images are obtained. The result of the 3-D vision sensor is a high-fidelity 3-D point cloud representation of any environmental structure that is present in the sequence of acquired images.

Below we present the details of our end-to-end system for automating the discovery and 3-D mapping of potential archeological sites. It is expected that this will aid in the currently tedious processes of scouting and surveying potential archeological sites, which today is accomplished in a largely manual fashion. Our system may have other application beyond archaeology, e.g., in the discovery and reconnaissance of military or terrorist threats, or the locating and assessing of forest fire risks, among others. We demonstrate the components of the system on sample data similar to that which will be collected over the areas of interest in Turkey. Specifically, we illustrate 1) the machine learning techniques for burial mound classification and confidence estimation from low-resolution satellite imagery, 2) a path planning algorithm that seeks subsets of locations to visit and finds an optimal tour of this subset considering limits on UAV fuel, flight time, the possible information gain at each site, and the confidence estimates from the classification algorithm, and finally 3) the computer vision algorithms used to generate 3-D reconstructions of a scene given a sequence of 2D images.

This paper is organized into the following sections: in Section II, we formalize the machine learning procedure for the low-resolution data; Section III details the path planning algorithm; Section IV outlines the computer vision pipeline for 3-D vision; Section V shows preliminary results from each of the previous sections; and Section VI draws conclusions about our results and discusses future directions.

II. Machine Learning from Archaeological Data

The two main tasks that arise in learning on low-resolution data are classifying locations and estimating the value of acquiring a 3-D reconstruction of a location. Classification of the locations represents a standard binary learning problem. We assume we are given a set of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ composed of extracted features $x_1, \ldots, x_n \in \mathbf{R}^d$ and labels $y_1, \ldots, y_n \in \{-1, 1\}$, where a label of -1 represents a location that does not contain a burial mound and a label of 1 represents a location that does contain a burial mound. From this training data, our goal is to learn a function $f : \mathbf{R}^d \to \{-1, 1\}$ from a family of functions that map from features to a predicted label. In particular, we consider the family of linear functions $f = sign(w^T x)$ parametrized by the weight vector $w \in \mathbf{R}^d$. To choose a weight vector, an empirical risk minimization problem is posed,

$$w^* = \underset{w \in \mathbf{R}^d}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{y_i w^T x_i \leq 0}, \tag{1}$$

where the goal is to learn a weight vector that minimizes the empirical loss. Unfortunately, this optimization problem cannot be efficiently solved, so we instead use logistic loss as an upper-bounding surrogate for the indicator function,

$$\hat{w} = \underset{w \in \mathbf{R}^d}{\operatorname{argmax}} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)), \tag{2}$$

yielding a convex optimization problem. Minimizing this convex surrogate using gradient descent, we can efficiently learn a classifier on the low-resolution data.

Given the classifier $\hat{w}$, the next problem is estimating the value of acquiring a 3-D model. Previous work has focused on determining when to acquire additional data for classification [10], [11], [12], however these approaches center on independent costs for acquiring additional information for examples. In contrast, the cost (fuel consumption of the UAV) associated with acquiring the data to perform 3-D reconstruction at each location is dependent on the set of locations to be visited and the path chosen. Rather than attempt to jointly solve the problem of selecting locations and planning a path, we instead decouple the problem and first estimate the value associated with adding each node, then maximize this estimated value of the path of the UAV.

To estimate the value of building a 3-dimensional model at a location, we estimate the probability that a location contains a burial mound by using a logistic model,

$$\hat{p}(x_i) = \frac{1}{1 + e^{-\hat{w}^T x_i}}, \tag{3}$$

where $\hat{p}(x_i)$ is an estimate of the conditional probability that example x_i is a burial mound, that is $P(y_i = 1 | x_i)$.

A natural approach to modeling the value of building a 3-D model at a location is the reduction in entropy induced by the model. For each location, we can then define an estimated entropy of the low resolution classifier as $H(\hat{p}(x_i)) = -\hat{p}(x_i) \log(\hat{p}(x_i)) + (\hat{p}(x_i) - 1) \log(1 - \hat{p}(x_i))$. Due to the

accuracy of the 3-D model acquired by the UAV and the ability of human experts to assess the 3-D model, we assume that perfect classification is achievable at any location given the 3-D model. As such, the reward associated with sending a UAV to a location can be modeled as the entropy of the low-resolution classifier. We then model the problem as a path planning problem, with the reward at each location modeled as the estimated entropy of the low-resolution classifier.

III. Path Planning

A. Planning Model

Denote the search area by a complete undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the *node* set with x_k being the extracted feature at v_k and $E = \{(v_i, v_j) : 1 \leq i < j \leq N\}$ is the *edge* set. The graph is complete as we assume our UAV can fly freely among nodes. Let node v_1 be the *home* node of the graph where the UAV starts and ends its flight.

For each node v_k, we associate a *reward* r_k with it. Since we always visit node v_1, without loss of generality, let $r_1 = 0$. For $2 \leq k \leq N$, let r_k be the estimated binary entropy of the low-resolution classifier using the extracted feature x_k.

For each edge (v_i, v_j), we associate an *edge cost* $c(v_i, v_j)$ with it, which is proportional to the Euclidean distance between the two nodes. Thus, edge costs satisfy the triangle inequality: $c(v_i, v_k)+c(v_k, v_j) \geq c(v_i, v_j)$ for all distinct i, j, k. For each node v_k, we also associate a *node cost* $c(v_k)$ with it, which is proportional to the fuel cost required to collect enough information to perform the 3-D reconstruction of the sites. This typically involves flying a raster-scan pattern around the site. $c(v_k)$ is expressed in the same units as the edge costs $c(v_i, v_j)$.

The node cost can be incorporated into the edge cost by defining the *expanded edge cost* c_{ij} for each edge (v_i, v_j) as

$$c_{ij} = c(v_i, v_j) + \frac{c(v_i) + c(v_j)}{2}$$

Through the paper, the cost of any path will be computed using the expanded edge costs. It is easy to see that the expanded edge costs also satisfy the triangle inequality.

Let B denote the budget of the UAV, in units of cost as above. Our goal is to design a closed path to visit a subset of nodes that maximizes the rewards collected, without exceeding the budget constraint. This problem has elements of both the *travelling salesman problem* (TSP) and the *knapsack problem*, each of which is NP-Hard. Due to the budget constraint, not all nodes shall be visited, so the decisions include selecting both a subset of the node set to visit and the edges of that subset to travel. This problem is known as the *orienteering problem* in literature [1].

Define binary variables $\beta_k \in \{0, 1\}$ $(k = 2, \ldots, N)$ to be 1 if node v_k is visited and 0 otherwise. Define binary variables $e_{ij} \in \{0, 1\}$ $(1 \leq i < j \leq N)$ to be 1 if the edge (v_i, v_j) is travelled and 0 otherwise. To handle the case where the solution contains only one edge – from v_1 to some v_j, e_{1j} is also allowed to take value 2 in addition to 0 and 1. Then we have the following integer linear programming formulation:

$$\begin{aligned}
\max \quad & \sum_{k=2}^{N} r_k \beta_k \\
\text{s.t.} \quad & \sum_{j=2}^{N} e_{1j} = 2 \\
& \sum_{i=1}^{k-1} e_{ik} + \sum_{j=k+1}^{N} e_{kj} = 2\beta_k, \quad k = 2, \ldots, N \\
& \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} e_{ij} \leq B \\
& 2 \sum_{v_k \in S} \beta_k \leq |S| \Big(\sum_{v_i \in S, v_j \notin S} e_{ij} + \sum_{v_i \notin S, v_j \in S} e_{ij} \Big) \\
& \qquad\qquad S \subset V \setminus \{v_1\},\ |S| \geq 3 \qquad (4) \\
& e_{1j} \in \{0, 1, 2\}, \quad j = 2, \ldots, N \\
& e_{ij} \in \{0, 1\}, \qquad 2 \leq i < j \leq N \\
& \beta_k \in \{0, 1\}, \qquad k = 2, \ldots, N
\end{aligned}$$

The above formulation modifies the standard TSP by introducing the node selection variables β_k and the budget constraint. In addition, the objective function is to maximize the total rewards of the nodes being visited. The above formulation has an exponential number of subtour elimination constraints, (4), limiting the applicability of direct integer programming approaches. Instead, we propose a new algorithm which does not utilize the integer programming formulation and can effectively find a feasible tour.

B. Planning Algorithm

From the problem formulation, we can see that the optimization procedure includes selecting a node subset. The rewards of a given node subset are fixed. An optimal TSP tour on the subset will spend minimum budget collecting the same rewards. To find optimal TSP tours, we use the *Lin-Kernighan-Helsgaun* (LKH) algorithm [5], [13] as a subroutine. The LKH algorithm has been shown to effectively find the optimal tour for a large number of nontrivial instances including one with 7397 nodes. Given a node subset S of the node set V, let c_{LKH} denote the cost of the Hamiltonian tour on S found by the LKH algorithm.

Our algorithm exploits the following facts from graph optimization problems: Given any spanning tree on S, the cost of an optimal tour on S is no greater than twice the cost of the arcs on the tree, c_{tree}. Furthermore, given a topological order on the nodes in the tree induced by a depth first traversal, the cost of the Hamiltonian tour traversing those nodes in topological order is no greater than twice the cost of the arcs on the tree, because the expanded arc costs satisfy the triangle inequality.

Therefore, our path planning heuristic consists of three steps:

Step 1) Grow a tree of nodes starting from node v_1. Suppose we have selected node subset S $(\ni v_1)$ and the current tree spanning S has a cost c^S $(c^S < \frac{B}{2})$. For each

unselected node $v_k \notin S$, define its *Reward-to-Connection-Cost Ratio* (RCCR) given S as $r_k/c_k^{\overline{S}}$, where the *connection cost* $c_k^{\overline{S}}$ is the minimum cost of connecting v_k to S. Select the node with the biggest RCCR among the nodes that satisfy $c_k^{\overline{S}} + c^S \leq \frac{B}{2}$ and add it to the tree. This continues until all remaining nodes outside of S satisfy $c_k^{\overline{S}} + c^S \leq \frac{B}{2}$.

We add a second round of tree growing by estimating the cost of a tour using the topological order imposed from a depth-first traversal of the tree. Let $c_{k_{tour}}^{\overline{S}}$ denote the cost of this topological tour after adding node k to set S with minimal cost. We then add node $v_k \notin S$ to S in order of biggest RCCR as long as $c_{k_{tour}}^{\overline{S}} \leq B$.

This step is similar to Prim's algorithm for finding minimum spanning trees, except that it is reward-sensitive and budget-constrained.

Step 2) Run the LKH algorithm on the selected node subset S to find a tour for TSP, denoted by l_S, which is guaranteed to satisfy the budget constraint.

Step 3) The tour found by the LKH algorithm in step 2) is likely to have some leftover budget. We try to insert some of the remaining nodes into this tour l_S. For each node $v_k \notin S$, define its *incremental cost* given l_S as $d_k^{\overline{l_S}} = \min_{(v_i,v_j)\in l_S} c_{ik} + c_{jk} - c_{ij}$, and let $(v_{k,1}, v_{k,2})$ denote the argument of the minimum. Define the *Reward-to-Incremental-Cost Ratio* (RICR) of node v_k as $r_k/d_k^{\overline{l_S}}$. Compute the RICR for each unselected node $v_k \notin S$. Select the node v_k^* with the biggest RICR among the nodes that satisfy $c^S + d_k^{\overline{l_S}} \leq B$, then add it to S and replace edge $(v_{k,1}^*, v_{k,2}^*)$ by edges $(v_{k,1}^*, v_k^*)$ and $(v_k^*, v_{k,2}^*)$ in the tour l_S. Proceed until no more nodes can be added in this way with the given budget.

After step 3), if the selected node subset becomes enlarged, the tour we currently have may not be an optimal TSP tour on this enlarged subset. In this case, we repeat step 2) and 3) until no further improvements can be made. In fact, in practice the obtained tour after the initial run of step 3) is often an optimal TSP tour on the selected node subset, then we do not need to repeat the two steps.

IV. Vision-based 3-D Reconstruction

Here we describe the high-resolution sensor of the pipeline that creates a 3-D model of the environment for each of the nodes in the previously computed tour, determining whether or not it contains a burial mound. A standard algorithm using multiple view geometry [6] is used to estimate the 3-D position of features that are visible in multiple images with respect to one of the camera's coordinate frames. The algorithm seeks to choose the relative transformations between camera frames and the individual feature depths that minimize the error from re-projecting estimated 3-D points back onto the original images. The iterative minimization is improved by using an accurate initialization that comes from estimating the relative transformation and feature depth for one pair of images.

Features in this context denote 2-D patches of pixels that have unique, identifying attributes such as colors, gradients, corners, edges, or lines. This technique does not require the images in a set to be sequential, however if we assume small relative camera transformations we may track features rather than perform a more expensive robust matching algorithm using feature descriptors (e.g., SIFT or SURF) and RANSAC. No ground truth, GPS, or camera motion information is required, although it can also be used to initialize the minimization if available.

A. Pinhole Camera Model and 3-D Reconstruction Problem

Suppose the UAV has captured m images, where the k^{th} image is denoted by I_k, $k = 1, \ldots, m$. Without loss of generality, we assume that the reference coordinate frame for the reconstruction is located in the position where the first image is acquired. The burial mound is then represented by a set of n 3-D environment features expressed in this frame, i.e. $\mathbf{P}_1^i \in \mathbb{R}^3$, $i = 1, \ldots, n$. The relationship between the k^{th} camera frame and the reference frame is given by the transformation $(\mathbf{R}_{k1}, \mathbf{t}_k)$, or,

$$\mathbf{P}_k^i = \mathbf{R}_{k1}\mathbf{P}_1^i + \mathbf{t}_k. \tag{5}$$

A point in the k^{th} camera frame, $\mathbf{P}_k^i$, can be transformed into the k^{th} image frame, $\mathbf{p}_k^i \in \mathbb{R}^2$, using the pinhole camera model that assumes the following linear transformation,

$$\lambda_k^i \begin{bmatrix} \mathbf{p}_k^i \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{P}_k^i, \tag{6}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the camera calibration matrix, that is estimated off-line using the OpenCV libraries [14], and λ_k^i is the depth scale factor that is lost when projecting a 3-D scene onto a 2-D image, (Fig. 1).

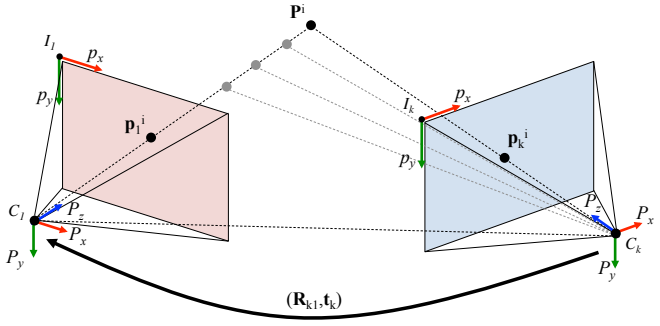


Fig. 1: Image and camera coordinate systems for the image pair (1,k) that observe the environmental object, $\mathbf{P}^i$.

Therefore, given the set of pixel coordinates, $\mathbf{p}_k^i$, observed in the different images, the problem of obtaining the 3-D model of the scene consists of determining the depth scale factors in the first camera's frame, λ_1^i, and the relative transformations $(\mathbf{R}_{k1}, \mathbf{t}_k)$ that minimize the total re-projection error (bundle adjustment), that is,

$$\min_{\mathbf{R}_{k1},\mathbf{t}_k,\lambda_1^i} \sum_{k=1}^{m}\sum_{i=1}^{n} \alpha_{ik} \left\| \lambda_k^i \mathbf{p}_k^i - \mathbf{K}\left(\mathbf{R}_{k1}\left(\lambda_1^i \mathbf{K}^{-1}\mathbf{p}_1^i\right) + \mathbf{t}_k\right)\right\|^2, \tag{7}$$

where $\alpha_{ik} \in \{0, 1\}$ is equal to one if feature $\mathbf{p}_k^i$ was observed in the image I_k. Note that the other scales factors,

λ_k^i, are simply functions of the other estimated parameters. The iterative Levenberg-Marquardt algorithm is a popular choice to solve (7) and tends to reach the global minimum if the initial solution is accurate enough. In the following we review how the initial solution for $(\mathbf{R}_{k1}, \mathbf{t}_k)$ and λ_1^i is obtained.

B. Epipolar Geometry Review

Epipolar geometry describes the geometric relationship between sets of feature matches visible in a pair of images. Intuitively, a feature's 3-D pose can only be determined by adding another unique viewpoint of the same feature (Fig. 1). We arrange this relationship into the epipolar constraint [6] by computing the matrix cross product of translation vector, $\mathbf{t}_k$, which is denoted by the skew-symmetric matrix, $\widehat{\mathbf{t}}_k$. The epipolar constraint is formally written as,

$$\mathbf{P}_k^{i\,T} \widehat{\mathbf{t}}_k \mathbf{R}_{k1} \mathbf{P}_1^i = \mathbf{0}. \tag{8}$$

The relative transformation between these two camera coordinate frames is estimated by minimizing the product in equation (8) for each pair of tracked features. Standard algorithms estimate $(\mathbf{R}_{k1}, \mathbf{t}_k)$ by reformatting the equation (8) into a matrix product involving each of the n tracked features and using least squares estimation. In the 8-point algorithm in OpenCV, for example, at least 8 feature matches between a set of images are required in order to keep sufficient rank on the least squares matrix.

C. Feature Depth Estimation

Recall that the 3-D feature pose with respect to the first camera frame requires the depth scale factor, λ_1^i, for each feature visible in the m images. Unfortunately, this estimation will never provide perfect depth, thus the reconstruction will be accurate up to this scale (in other words, it will be unit-less). However, the final depth scale can be easily set using GPS or altimeter readings.

Starting with equation (5) in image frame coordinates for the same image pair $(1, k)$, we have,

$$\lambda_k^i \mathbf{K}^{-1} \mathbf{p}_k^i = \lambda_1^i \mathbf{R}_{k1} \mathbf{K}^{-1} \mathbf{p}_1^i + \gamma \mathbf{t}_k \tag{9}$$

where γ is a scale factor associated to the length of vector $\mathbf{t}_k$ since the world scale is unknown. Premultiplying equation (9) by the skew-symmetric matrix, $\widehat{\mathbf{p}}_k^i$, allows us to remove the left hand side and write the equation as,

$$\lambda_1^i \widehat{\mathbf{p}}_k^i \mathbf{R}_{k1} \mathbf{K}^{-1} \mathbf{p}_1^i + \gamma \widehat{\mathbf{p}}_k^i \mathbf{t}_k = \mathbf{0}. \tag{10}$$

Finally, the n systems of linear equations may be stacked in the following matrix,

$$\begin{bmatrix} \widehat{\mathbf{p}}_k^1 \mathbf{R}_{k1} \mathbf{K}^{-1} \mathbf{p}_1^1 & \dots & \mathbf{0} & \widehat{\mathbf{p}}_k^1 \mathbf{t}_k \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \dots & \widehat{\mathbf{p}}_k^n \mathbf{R}_{k1} \mathbf{K}^{-1} \mathbf{p}_1^n & \widehat{\mathbf{p}}_k^n \mathbf{t}_k \end{bmatrix} \begin{bmatrix} \lambda_1^1 \\ \vdots \\ \lambda_1^n \\ \gamma \end{bmatrix} = \mathbf{0}, \tag{11}$$

which takes the familiar form of $\mathbf{A}\mathbf{x} = \mathbf{0}$, where $\mathbf{A} \in \mathbb{R}^{3n \times n+1}$ and $\mathbf{x} \in \mathbb{R}^{n+1 \times 1}$. The non-trivial solution for vector $\mathbf{x}$ may be recovered using the Singular Value Decomposition (SVD) of matrix $\mathbf{A}$, where the solution is equivalent to the column of $\mathbf{V}$ that corresponds to the smallest eigenvalue, i.e., the last column [6].

We use the remaining $N - 2$ images in the sequence to refine the depth scales with respect to I_1 as well as estimate the relative transformations between I_1 and each subsequent frame, thus obtaining a sufficient initial estimate for all the parameters involved in (7).

V. Results

For the low resolution sensor, we use LANDSAT data as the baseline for the learning phase. We demonstrate performance of our system on the commonly used Statlog LANDSAT data set from the UCI Machine Learning Repository [15]. This data set is composed of multi-spectral satellite images with 4435 training examples and 2000 test examples. Each example is composed of 3×3 images in 4 different spectral bands and a label corresponding to one of six classes, (red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, and very damp grey soil).

We choose the cotton crop class as our target class and use measurements from a single spectral band as the low-resolution data. On this single spectral band, we learn a binary classifier on the training data as described in Section II. To simulate a field of examples at each location, we assign negative test data to a 10×10 image field (100 images in total), with 10 x and y coordinates uniformly spaced between 1 and 10. In this field of 100 images, we replace 3 to 5 images with a positive test image. Using this process, we generate ten random realizations of image fields to use in our evaluations.

We train a binary classifier using the training data, and use it to estimate the reward for each location in the field. This information is passed to the path planning algorithm.

We compare our path planning heuristic (hereinafter referred to as "NewAlg") with a greedy benchmark approach (referred to as "BenAlg"): We grow the tour incrementally by growing a path from the source node, and adding nodes not already in the path. We compute the reward per incremental cost of connecting to the most recently added node for nodes not already in the path, and choose to extend the path by selecting the node with highest marginal reward among those nodes that can be added while leaving enough budget to return to the source node. This construction maintains a path. If the given budget does not allow us to visit one more node before returning to the original node, then return and close the loop.

In addition, we compare our heuristic with the algorithms proposed by Tsiligirides in [1] (referred to as "TsiAlg") and by Golden et al. in [2] (referred to as "GLVAlg").

To evaluate tours, we use missed detection and false alarm rates. Since the 3-D reconstruction allows us to make accurate classification, nodes that have been visited and scanned will have no prediction error. The nodes that are not included in the flight trajectory will cause prediction errors as only low-resolution data are accessible. Thus, we use the

Budget		40	80	120	160	200
Avg. Rwd.	NewAlg	7.89	16.63	25.69	34.16	42.36
	BenAlg	7.64	15.83	24.24	32.34	39.87
	TsiAlg	8.40	16.88	25.10	32.93	39.71
	GLVAlg	7.25	14.19	20.85	28.23	33.03
ϵ_{MD}	NewAlg	0.649	0.486	0.378	0.243	0.135
	BenAlg	0.595	0.514	0.459	0.297	0.162
	TsiAlg	0.622	0.595	0.432	0.216	0.162
	GLVAlg	0.622	0.514	0.432	0.351	0.324
ϵ_{FA}	NewAlg	0.064	0.046	0.028	0.010	0.000
	BenAlg	0.063	0.042	0.024	0.012	0.001
	TsiAlg	0.061	0.034	0.020	0.005	0.000
	GLVAlg	0.065	0.049	0.025	0.006	0.002
Time (seconds)	NewAlg	5.24	7.12	10.27	12.03	16.41
	BenAlg	0.05	0.10	0.14	0.18	0.24
	TsiAlg	201.9	426.1	625.4	800.2	990.7
	GLVAlg	0.11	0.29	10.50	16.02	135.0
Bgt. Lft.	NewAlg	1.415	1.532	1.148	1.372	1.045
	BenAlg	1.233	1.160	1.760	1.859	1.200
	TsiAlg	0.407	0.478	0.603	0.787	1.063
	GLVAlg	1.602	0.783	1.258	0.976	1.550

TABLE I: Average reward (Avg. Rwd.), average missed detection rate (ϵ_{MD}), average false alarm rate (ϵ_{FA}), average computation time (Time) and average unused budget (Bgt. Ltf.) over ten scenarios for different algorithms.

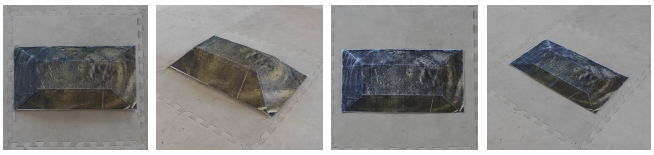
following formulas to compute the two error rates:

$$\epsilon_{MD} = \frac{\sum_{v_k \notin S} \mathbf{1}_{\{y_k=1, \hat{y}_k=0\}}}{\sum_{v_k \in V} \mathbf{1}_{\{y_k=1\}}}$$
$$\epsilon_{FA} = \frac{\sum_{v_k \notin S} \mathbf{1}_{\{y_k=0, \hat{y}_k=1\}}}{\sum_{v_k \in V} \mathbf{1}_{\{y_k=0\}}}$$

where ϵ_{MD} and ϵ_{FA} are the missed detection and false alarm rates, V and S are the total node set and the visited node subset, and y_k and $\hat{y}_k$ are the true and prediction labels.

We conduct the simulation using MATLAB 2014 on a laptop computer with Intel i7-4600M processor and 8GB RAM. We use the ten fields of 100 images generated previously, with an average total reward being 61.37. Given a fixed budget, we compute the average collected reward, the average missed detection rate and the average false alarm rate over the ten scenarios, for all the four algorithms. We report the results under five different budgets ranging from 40 to 200 units in Table I, where 40 represents a tight budget under which the UAV only collects a small portion of reward and 200 represents a generous budget under which the UAV collects over half of the reward. In terms of reward collected, our algorithm performs better than the benchmark algorithm and the Golden-Levy-Vohra algorithm universally, and also outperforms the Tsiligirides algorithm especially in the higher budget domain. It is noteworthy that our algorithm is 23 to 70 times faster than the Tsiligirides algorithm (the ratio tends to increase as budget increases), and has more unused budget, which implies more robustness against unexpected resource consumptions. The result also shows that our algorithm has lower missed detection rates than the other algorithms for different budgets, as well as competitive false alarm rates.

When the UAV visits a site, it conducts a sweep of the site collecting 2-D images in order to form a 3-D reconstruction. To test our algorithms, we controlled a quadrotor over a



(a) (b) (c) (d)

Fig. 2: Low-resolution imagery for the correct model with 3-D topography (a-b), and incorrect model without 3-D topography (c-d).

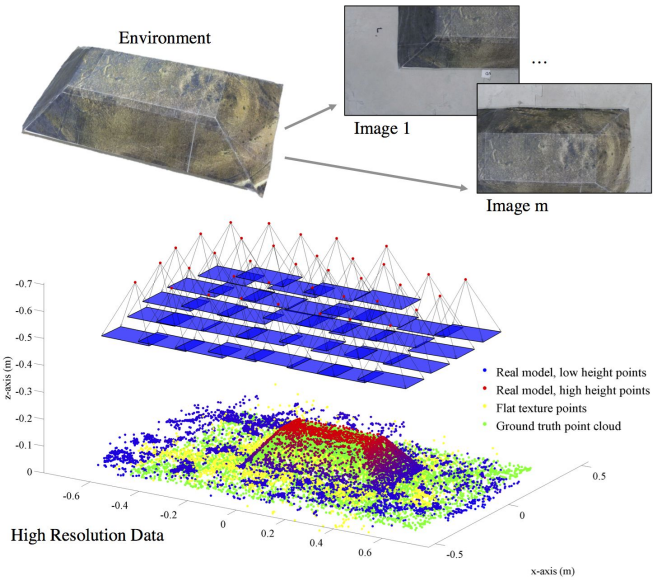


Fig. 3: Example 3-D reconstruction from a sequence of aerial images above an object. The point cloud of the physical model is colored to denote high elevation (red) and low elevation (blue). Each of the 37 aerial images used in this particular reconstruction are displayed as blue rectangles hovering above the point cloud. The point cloud of the flat object is colored yellow. The ground truth point cloud is colored green.

physical model in a laboratory setting. The model is a scale representation of a specific burial mound in Turkey. The 3-D reconstruction of this model was compared to the reconstruction of a visually similar scene, but with no topographical relief (Fig. 2(a) and 2(c)), to show the depth recognition capabilities of our algorithm. The ground truth 3-D point cloud of the correct model was generated from a geometrically equivalent 3-D Computer Aided Design model of the burial mound. The reconstruction of both models was created using a commercially available photogrammetry software, PhotoScan [16] and the results were post processed in an open-source software, CloudCompare [17] (see Fig. 3). Both reconstructions were compared to the ground truth point cloud using the Iterative Closest Point algorithm that estimates the relative transformation and scale between two point clouds that minimize the root mean squared error for each point in the reference point cloud. The distances from the ground truth point cloud for each point were fit to a normal distribution and are given as: 1) correct model $\sim N(0.0755, 0.0689)$ meters and 2) incorrect model

$\sim N(0.1769, 0.1578)$ meters. Therefore, a threshold would allow us to distinguish the two models and make a correct decision.

VI. Conclusion and Future Work

In this paper, we presented the methodology to aid a team of archaeologists in the mission to understand more about an important ancient civilization. We presented the machine learning framework for low-resolution data and validated it on a standard dataset. The results from the learning phase were integrated into the path planning algorithm that produces tours for a vehicle to gain more knowledge of the environment. Finally, the 3-D vision sensor was demonstrated on a simple, real-world environmental structure.

The proposed system will be used to explore large geographical expanses where potential burial mounds are located in rural Turkey. We are currently validating the suite of algorithms using indoor air vehicles such as quadrotors, and indoor environments built to scale.

We are also considering extensions of our approach to using active strategies while acquiring high-resolution images. Several of these directions include control of the high-resolution search pattern to minimize the estimation error in the 3-D reconstruction, and developing optimal strategies for stopping the high resolution 3-D search when enough information has been collected to determine the presence or absence of a burial mound in the area. These techniques will enable more sites to be visited with fixed budgets, reducing the system probability of error in detecting potential sites for burial mounds.

References

[1] T. Tsiligirides, "Heuristic methods applied to orienteering," *Journal of the Operational Research Society*, pp. 797–809, 1984.

[2] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[3] I.-M. Chao, B. L. Golden, and E. A. Wasil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, 1996.

[4] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 342–349.

[5] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[6] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3D Vision: From Images to Geometric Models*. Springer, 2004.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[8] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 2609–2616.

[9] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*. Springer, 2008, pp. 871–889.

[10] T. Gao and D. Koller, "Active classification based on value of classifier." in *Advances in Neural Information Processing Systems (NIPS)*, vol. 24, 2011, pp. 1062–1070.

[11] Z. Xu, M. Kusner, M. Chen, and K. Weinberger, "Cost-sensitive tree of classifiers," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2013, pp. 133–141.

[12] J. Wang, K. Trapeznikov, and V. Saligrama, "An LP for sequential learning under budgets," in *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2014, pp. 987–995.

[13] K. Helsgaun, "Implementation of the LKH algorithm." [Online]. Available: http://www.akira.ruc.dk/ keld/research/LKH/

[14] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[15] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[16] 2015 Agisoft LLC., "Agisoft photoscan." [Online]. Available: http://www.agisoft.com

[17] CloudComplare, "Cloudcompare: Open source 3d point cloud and mesh processing software." [Online]. Available: http://www.danielgm.net/cc/