A Bounded Suboptimal Environmental Monitoring Algorithm

Elizabeth A. Ricci[1] and Ross A. Knepper[1]

***Abstract*— Autonomous robot monitoring tasks require the robot to traverse a map over an extended period of time for the purpose of monitoring some value of interest. The robot must choose paths that efficiently move through the environment and perform observations, such that the maximum amount of information is gained. The robot uses these observations to predict the state of the entire environment, including unobserved portions, using a low-rank model built from prior recorded data. We introduce a path planner that selects a set of points in the map that maximize the entropy subject to a cost constraint. We evaluate our method in simulation on the problem of predicting fall foliage colors using a satellite image dataset and a UAV traversal of the planned path.**

I. Introduction

Most robots collect information about their environment but are constrained to sense only a limited area around the location of the robot. In many problems, like environmental monitoring [5, 7, 9] or persistent monitoring [9, 11, 15], the mission involves sensing broadly in order to build up a map of some quantity throughout a larger region. Consequently, the robot must navigate a path to maximize information gain. In the absence of any structure in the data, building a map amounts to a coverage problem. Often though, there is structure in the data that can be exploited to at least estimate a complete map from limited observations. In this paper, we seek to plan a path for a robot through a region that maximizes the information gathered while adhering to cost constraints.

The measured quantity can have structural correlations over space and time, such as recurring patterns over separate regions or cyclic seasonal repetition. Example measured quantities include vegetation health [14], ocean properties [7, 9], hurricane predictors [5], survivor locations [8], and wireless signal strength [2]. A key insight of this work is that not all areas of a region hold as much uncertainty as others, thus providing less information. Consequently, most of the map may be reconstructed by observing a minority of the locations within it. To make initial decisions about where to visit, we consider the case that there is some prior data about the region to be monitored that enables entropy computations. Our method maximizes information gain by sampling points that have the greatest joint entropy conditioned on prior observations.

A. Problem Statement

A robot travels around a map comprising a finite set of p locations that constitute possible sampling points. Each point in the set holds a scalar-valued, time-varying parameter. At the time the robot makes each observation, it learns the true value of the parameter at its location at that time. We assume the time-varying rate of the quantity is negligible compared with the time elapsed in traveling around the map to make observations. A cost function encodes any desired path qualities, such as end point, and travel cost from one location to another.

In addition to the map structure, the robot is also given some prior map data from t earlier times. The prior data consists of snapshots of the map (or subsets thereof) each at a single point in time. From these observation, one can estimate a joint probability density function for any subset of the random variables corresponding to those observations.

The *offline problem* is to plan a path through a subset of locations that maximizes expected information gain overall. The online planning problem is not addressed in this paper.

In order to illustrate the techniques of our contribution, we focus on using prior satellite data of eastern North America to predict fall foliage colors. Some areas in each prior map are occluded by cloud cover. We model the robot as a UAV that travels around the region captured by the map and makes observations about the color of observed pixels in the map. The cost of a traversal is path length, measured as the sum of Euclidean distances between consecutive observations.

B. Related Work

A variety of approaches to and variants of this problem have been studied. Hollinger and Sukhatme [2] introduce anytime, sampling based planners that use trees and graphs to build paths that consider information gain and budget constraints. A rapidly exploring random tree and a genetic algorithm are used by Kwak and Scerri [4] to maximize information gain along a path given a function of the expected value of the information gain. For applications where the sampling region can be modeled as a Gaussian process, Low et al. [6] present a method that uses a Markov planner to sample the region. Yu et al. [15] model the problem of persistent monitoring as an orienteering problem and provide a solution using mixed integer quadratic programming. Smith et al. [11] use linear programming to control the speed robots with limited sensing such that they can persistently monitor an environment. Multi-agent variants of persistent monitoring have also been studied [8, 10, 12].

C. Contributions

We introduce an offline planner which uses the submodularity of entropy to choose a set of sampling points that approximately maximizes the joint entropy while remaining

*This work was supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0109.

[1] The authors are with the Department of Computer Science, Cornell University, Ithaca, NY, USA {ericci,rak}@cs.cornell.edu

within a given cost budget. We present an algorithm for sampling observations subject to a cost constraint on path traversal. We show that the joint entropy of the path returned by our algorithm is at most $\frac{1}{2}(1-e^{-1})$ times the optimal solution.

In order to demonstrate the utility of our general method for predicting the entire region using a small number of samples, we propose the use of a low-rank model to predict the values of unobserved sampling points using the observed points.

In the remainder of this paper, we present our method for computing entropy in Section II and our algorithm that finds an optimal information-gain path in Section III. We show how the information gained by traversing this path can be used to estimate the complete map in Section IV. Then we evaluate the algorithm in Section V and conclude in Section VI.

II. Computing Entropy

Our method requires a submodular set function f that yields differential entropy given a set of observations of Gaussian random variables. For the following methods we assume that the sample space S is a finite set of random variables and that each random variable has a set of prior values and a Gaussian distribution. S is drawn from the map locations indexed as $\{1,\dots,p\}$.

A. Joint Differential Entropy

Joint differential entropy (hereafter called joint entropy) is a measure of the uncertainty of a set of continuous random variables and can be computed from a covariance matrix. Let Σ be a covariance matrix of the random variables in the sample space S and let s' be a subset of the random variables in S. A square submatrix of Σ representing the covariance matrix of s', called $\Sigma_{s'}$, is composed of the rows and columns of Σ corresponding to the random variables in s'. Then the joint entropy of the samples in s' is equal to $\log\det(\Sigma_{s'})$ [3]. Since the covariance matrix is a symmetric positive definite square matrix, the function $f=\log\det(\Sigma_{s'})$ is a submodular function.

III. Algorithm

In this section we introduce our offline planner, which is based on the generalized cost-benefit algorithm by Zhang and Vorobeychik [16].

A. Cost function

The cost function will be used to compute the cost of the shortest path between a set of points and can be constructed to enforce any desired path properties. In many cases, such as when a tour is desired, the cost function is an NP-hard problem. In order to make the planner tractable we allow for approximate cost functions. In order to make an optimality guarantee about the planner the approximate cost function should be a $v(n)$-approximation where n is the number of points in the set. A cost function is a $v(n)$-approximation when it is at most $v(n)$ times greater than the true cost. In addition the cost function should be α-submodular, that is when x, A, B are minimized and $A \subset B$ the quotient $\frac{cost(A\cup x)-cost(A)}{cost(B\cup x)-cost(B)}$ is equal to α [16].

B. Offline Planner

Given the covariance matrix of the sampling points, and a starting point and a budget, the offline planner returns a set of points to sample such that the joint entropy of the points is approximately maximal and that the approximate cost of a path between the points is within the budget.

The algorithm used to select the sampling points is based on a method introduced by Zhang and Vorobeychik [16], which has the following optimality guarantee: the joint entropy of the points selected is within $\frac{1}{2}(1-e^{-1})$ times the maximum joint entropy possible within the cost constraints. We modified the original algorithm to improve its efficiency and supplied joint entropy as the submodular function. The cost constraint was expressed in terms of nearest neighbor distances. Our method is shown in Algorithm 1.

IV. Fall Foliage Mapping Problem

A. Data

We evaluate our methods on a fall foliage dataset, containing 186 satellite images of the northeast region of North America collected from the fall months over the course of three years collected using NASA Worldview.[1] More specifically there is one 1629×1076 pixel image for every day between September 15 and November 15 for the years 2014, 2015 and 2016. Over the course of the three months, the hue of portions of the land in the images changes from green to orange, yellow and brown, as a result of the foliage changes. The original satellite images contain large areas of clouds which obfuscate the hue of the land below. As a result, we model pixels containing clouds as missing data. A support vector machine was used to identify pixels containing clouds. In addition, due to the length of time it takes to capture a satellite image, many of the images have obvious hue differences due to lighting changes. We negate the effects of these inconsistencies by splitting affected images into two images on the line of hue shift, where one portion of the image was the section of the original image and the remainder is black representing missing data. The splitting line was identified using Hough transform on the edges of the image. The edges were identified using Canny edge detection. After this processing we have 298 processed images, all containing missing data.

For use in the planning methods, the sampling set of this data set are the set of all pixels in an image. Due to space constraints, we only consider about 100 by 100 pixel subsets of the image. With the path lengths we have chosen to evaluate, this does not limit the planners.

B. Cost Function

For our particular dataset we approximate the shortest path using the nearest neighbor algorithm, which is a known-greedy approximation for shortest path problems [1]. Starting

[1] worldview.earthdata.nasa.gov

```
Algorithm 1: Offline Planner
Data: covariance_matrix, start_point, all_points, budget
Result: sample_points
begin
    sample_points = [start_point]
    current_entropy = entropy(sample_points, covariance_matrix)
    current_cost = 0
    points = [] # only consider points that are close enough to the start
    for $point \in all\_points$ do
        if $cost(point, sample\_points) \leq budget$ then
            points = points $\cup$ point
    while $points \neq \emptyset$ AND $current\_cost < budget$ do
        for $point \in points$ do
            cost_change[point] = cost(point, sample_points) - current_cost
            entropy_change[point] = entropy(point, sample_points, covariance_matrix) - current_entropy
            # if the cost of sampling a point is ever over budget, remove it from consideration
            if $cost(point, sample\_points) + current\_cost > budget$ then
                points = points $\setminus$ point
        # add the point that maximizes the ratio of entropy change to cost change to the list of sample points
        new_point = max(entropy_change / cost_change)
        sample_points = sample_points $\cup$ new_point
        points = points $\setminus$ new_point
        current_cost = cost(sample_points)
        current_entropy = entropy(sample_points, covariance_matrix)
    return sample_points
```

with a given point, the algorithm adds points to the path based on proximity to the current point. That is, at each timestep the point to be traveled to is the one that is closest to the current point. Given a set of points $\{v_1, \ldots, v_n\}$, the nearest neighbor algorithm returns a travel ordering of these points, starting at v_1, the ordering is notated $\{v'_1, \ldots, v'_n\}$. The cost of the path is:

$$\sum_{i=1}^{n-1} d(v'_i, v'_{i+1}) \tag{1}$$

where $d(a, b)$ is the Euclidean distance between the points a and b. This cost function is a $v(x)$-approximation and α-submodular by [1].

C. Generalized Low Rank Model

The planned path returned by the algorithm in Section III is designed to gain as much information as possible about the overall map within a limited budget. Having collected those observations, we need a technique to complete the map through extrapolation. We construct a low-rank model to predict the full map. In order to compute the model we first form a data matrix D of t rows corresponding to the time snapshots and p columns containing the observed values at each location at the specified time. A corresponding mask matrix identifies entries in D that were not observed. The problem of constructing a low-rank model is to find two matrices X and Y that minimize the quadratic loss function, $(X^{\mathsf{T}}Y - D)^2$, subject to a non-negative constraint.

We use a Julia implementation of the generalized low rank model of Udell et al. [13] to factor the data matrix while ignoring the missing data generated by in the processing stage. More specifically, the model takes the form of two matrices, the X matrix has a column for each image and the number of rows is equal to the rank of the model. The Y matrix has a column for each pixel and the number of rows is again the rank of the model. The columns of the Y matrix are the principal components of the data set. A set of principal components for a rank three model can be seen in Fig. 2. The product $X^{\mathsf{T}}Y$ is an approximation of the original data matrix and provides an approximate value for each missing data point.

D. Map Completion

Given a vector d_{path} containing the values of the observed pixels in an image, the Y matrix can be used to estimate the values of the pixels not observed. To do so, we take the set of observed pixels situated in a row vector of length p that includes unobserved pixels. We compute an "image digest" vector x_{path} of dimension equal to the model rank by

$$x_{\text{path}} = (Y^{\mathsf{T}})^{+} d_{\text{path}},$$

where $(Y^{\mathsf{T}})^{+}$ represents the pseudoinverse of Y^{T}. We then reconstruct a dense image by

$$\tilde{d}_{\text{dense}} = Y^{\mathsf{T}} x_{\text{path}}.$$

E. Rank Selection

There is no straightforward method for determining the optimal rank for a given data set, so we considered a number of ranks and choose the one that minimized the reconstruction error for known images. That is, for the approximated data matrix, the product XY^{T}, we computed the reconstruction error by summing the difference in the original data and the corresponding values in the approximated data matrix. The graph in Fig. 1 shows the reconstruction error per pixel for six different models computed of ranks 3, 5 and 11. These three ranks were chosen after a preliminary study found that these ranks resulted in the lowest reconstruction error out of models from ranks 1 to 13. From this study we found that one of the rank 5 models minimized the reconstruction error, this model is used in the evaluation section. We note that the function used to compute these models is non-deterministic, as the model is initialized with random matrices.

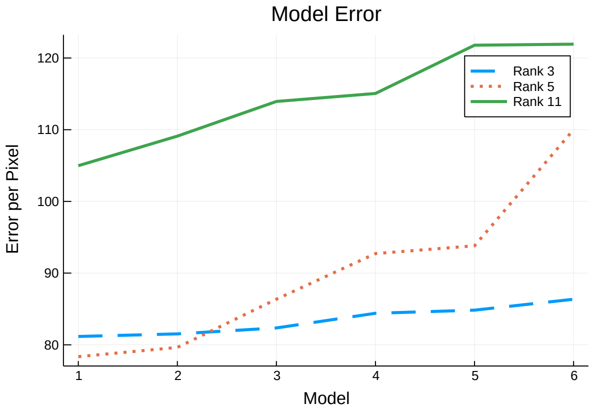

Fig. 1: A graph of the error per pixel of the computed generalized low rank models for ranks 3, 5, and 11. Each unit on the horizontal axis represents one trial of factorizing the D matrix with the chosen rank. The trials are sorted from lowest to highest.

V. Evaluation

In this section, we detail our preliminary simulated experimental results.

A. Baselines

In this subsection we introduce three baseline planners that will be compared with our planner.

1) Random Baseline: Given a budget of n, the random baseline planner randomly selects n points from the sample point set to visit. This baseline has no path constraint, that is the cost to visit all of the sample points is not considered.

2) Straight-Path Baseline: The straight-path planner assumes that the sampling points have a location-based ordering. For the fall foliage dataset the sampling points are pixels and are in raster-scan order starting at the top left corner of the image. Given a starting point with ordering index s and a budget of n, the straight-path planner selects the points with indices s to $s+n-1$ to visit.

3) Sampling-based Baseline: The sampling-based planner is an any-time rapidly exploring tree based planner, called rapidly exploring information gathering tree (RIG-tree) introduced by Hollinger and Sukhatme [2]. The RIG-tree planner builds a tree from a start node through sampling the sampling point space, adding nodes and edges that are within some distance of each other. More precisely, after a sample is taken, the point that are closest to it and within some delta of the nearest existing vertex is found. This point is then used as a center point of a circle of some fixed radius. All vertices within the circle are extended with an edge to a new vertex that is as close to the point as possible, while being at most delta away from the vertex. Since this is an anytime method, it can be run for any amount of time with results improving with diminishing returns as the time increases.

For our evaluation, the sampling-based baseline method will be run with a delta value of 5 and a radius of 10 for 100 iterations.

B. Memory Limitations

A primary limitation of our method is the memory cost associated with the covariance matrices necessary to compute the joint entropy of a set of points. Given a cost budget of n, both our offline planner and the sampling-based baseline planner need the covariance matrix associated with all the sampling points within an n radius of the starting point. That is, $O(n^2)$ points will be considered, and the covariance matrix of the set of points is $O(n^4)$. Thus, for both planners the theoretical upper bound on memory is $O(n^4)$. In the following subsection we show that, on our dataset, the performance of our planner reaches a near optimal reconstruction before the memory needs of the planner become intractable.

C. Results

To evaluate our method we consider both the joint entropy of the samples selected and reconstruction error. We note that for our specific dataset, our planner is tractable for budgets up to 100.

To compare the joint entropy of the sample sets produced by each method we plan paths with a budget of 50 starting from eight different locations and the planning space is limited to a 100 by 100 pixel subregion. For the random planner, no starting point is given, and 50 points are randomly selected from the full image. The average joint entropy values are shown in Table I.

TABLE I: Joint entropy results for paths of length 50.

Method	Average Joint Entropy
Random	379.77
Straight	182.32
Ours	200.52
RIG-Tree	72.81

The random planner has the highest joint entropy value, this is likely due to the lack of path constraints and that it is sampling from the complete image, instead of a subregion. Our method performs the best out of the three path constrained methods. We note that the paths produced by our planner and the straight-baseline planner are sample dense, that is the number of samples is close to the cost budget. The path produced by the RIG-tree is not sample dense, so there are fewer samples used in the joint entropy calculations which may be contributing to its low value.

To evaluate the planners' performance in terms of reconstruction error, we plan paths of a variety of lengths starting at eight different starting points dispersed throughout the region. Then, for nine different training images that were not used to build the model, we sample the points chosen by the planner and reconstruct the full image. We compute the reconstruction error for a given cost budget and planner by averaging the reconstruction errors for each start point and test image combination. Fig. 3 shows the average reconstruction error associated with three different planners for budgets that are multiples of between 10 and 100. The lower bound shown in the graph is the average reconstruction

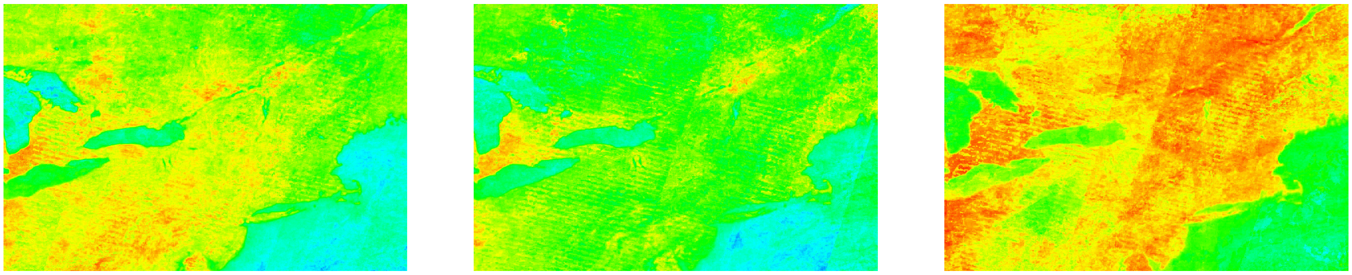

Fig. 2: Principal components of a rank 3 model.

error per pixel when all of the data points in the test images are used to reconstruct.

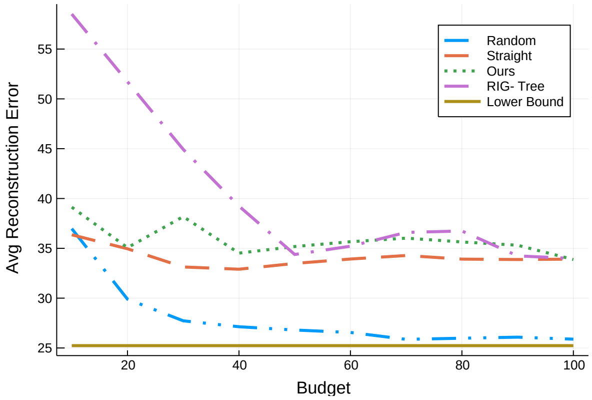

Fig. 3: The relationship between path length and the average reconstruction error per known pixel.

We note that there is no formal relationship between joint entropy and reconstruction error, which explains the non-monotonality of our planner's performance. The random baseline planner reaches the same average reconstruction error at budget 70 as sampling every pixel in the test images would produce. We observe that even though our planner does not reach this minimum reconstruction error, it does come fairly close by budget 100.

While it is not feasible to compute the reconstruction error for the baseline planner beyond path length 100, we found that the reconstruction error for the straight path baseline did not reach the minimum reconstruction error until the budget reached 10,000.

VI. Conclusions and Future Work

In this paper we have considered the problem of sampling regions of a space to learn about the global state of the space by exploiting its structure. To this end, we have introduced a planner for planning cost-constrained paths through a space such that the entropy of the sampled points is bounded suboptimal. In addition we have proposed the use of generalized low-rank models for predicting the global state of a space after observations along a path have been collected.

A main limitation of our planner are memory costs associated with computing the joint entropy using a covariance matrix. We plan to explore online planning algorithms that can more efficiently exploit structure that is discovered in the space during exploration.

Currently, our evaluation is limited to a single dataset, which has a high amount of structure. When we assume Euclidean-distance path lengths and uniformly-dispersed sampling points, it has few limitations. Future work will include evaluation on some other, more challenging datasets.

References

[1] Bruce Golden, Lawrence Bodin, T Doyle, and W Stewart Jr. Approximate traveling salesman algorithms. *Operations research*, 28(3-part-ii):694–711, 1980.

[2] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[3] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

[4] Jun-young Kwak and Paul Scerri. Path planning for autonomous information collecting vehicles. In *2008 11th International Conference on Information Fusion*, pages 1–8. IEEE, 2008.

[5] John S Langford and Kerry A Emanuel. An unmanned aircraft for dropwindsonde deployment and hurricane reconnaissance. *Bulletin of the American Meteorological Society*, 74(3):367–376, 1993.

[6] Kian Hsiang Low, John M Dolan, and Pradeep Khosla. Active markov information-theoretic path planning for robotic environmental sensing. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 753–760. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[7] Kai-Chieh Ma, Lantao Liu, and Gaurav S Sukhatme. An information-driven and disturbance-aware planning method for long-term ocean monitoring. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2102–2108. IEEE, 2016.

[8] Amarjeet Singh, Andreas Krause, and William J Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[9] Ryan N Smith, Mac Schwager, Stephen L Smith, Burton H Jones, Daniela Rus, and Gaurav S Sukhatme. Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics*, 28(5):714–741, 2011.

[10] Stephen L Smith and Daniela Rus. Multi-robot monitoring in dynamic environments with guaranteed currency of observations. In *49th IEEE conference on decision and control (CDC)*, pages 514–521. IEEE, 2010.

[11] Stephen L Smith, Mac Schwager, and Daniela Rus. Persistent monitoring of changing environments using a robot with limited range sensing. In *2011 IEEE International Conference on Robotics and Automation*, pages 5448–5455. IEEE, 2011.

[12] Ruben Stranders, E Munoz De Cote, Alex Rogers, and Nicholas R Jennings. Near-optimal continuous patrolling with teams of mobile information gathering agents. *Artificial intelligence*, 195:63–105, 2013.

[13] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

[14] Junming Wang, Theodore W Sammis, Vincent P Gutschick, Mekonnen Gebremichael, Sam O Dennis, and Robert E Harrison. Review of satellite remote sensing use in forest health studies. *The Open Geography Journal*, 3(1), 2010.

[15] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 342–349. IEEE, 2014.

[16] Haifeng Zhang and Yevgeniy Vorobeychik. Submodular optimization with routing constraints. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.