Coupled Sensor Configuration and Path-Planning in Unknown Environments with Adaptive Cluster Analysis

Chase St. Laurent[†] and Raghvendra V. Cowlagi,[*] *Senior Member, IEEE*

***Abstract*— We present an adaptive fast-approximation for sensor configuration which finds near-optimal placements and sensor field of views (FoV). The fast-approximation, either via partition-based or density-based cluster analysis, adapts based on the relation between statistical uncertainty of the path plan and environmental uncertainty. The sensor configurations are performed over regions of interest which most directly influence the path-planning efforts. These regions of interest can include exploratory paths by sampling the probabilistic environment model. The path-planning efforts aim to decide upon a path which minimizes an agent's exposure to threats in an unknown static environment. The noisy sensor network observations are used to construct a threat field estimate using Gaussian Process Regression each iteration with a stationary kernel and heteroscedastic gaussian likelihood. The optimization of a task-driven information gain determines optimal sensor configurations when maximized. The numerical performance of the direct optimization and the adaptive cluster analysis method is presented. Finally, we show that the cluster centers can be utilized as a dimensionality reduction technique for FoV optimization whereby we only optimize FoV radial coverage.**

I. Introduction

Path-planning for an autonomous agent is typically performed subsequent to the observation and modeling of it's surrounding environment. When the agent is "blind" to it's surroundings, it relies on extroceptive sensors to map and realize the environment. The separation between the path planning and sensor configuration efforts leads to unnecessary and excess exploration of the environment prior to deciding upon a best path. The goal of this paper is to *couple* the sensor configuration and path-planning objectives to minimize unnecessary exploration of the environment. We devise a centralized framework which deploys extroceptive mobile sensors to observe regions of direct relevance to the agent's path planning efforts. Throughout this paper, we use the environment, as shown in Fig 1, as an example for planning an optimal path which matches the true optimal path as shown in green.

Related Work: Path-planning is well studied in literature, with notable approaches being probabilistic roadmaps, cell decomposition, Dijkstra's algorithm, and A* [1]–[3]. These approaches typically aim to minimize path length, maximize path utility, or avoid obstacles [4].

Bayesian methods are used to develop probabilistic environment models. A Gaussian Process (GP) model is one such approach in which field estimation is performed via Bayesian updates using a mean function and kernel [5], [6].

[†]Mechanical Engineering Department.
[*]Aerospace Engineering Department, Worcester Polytechnic Institute, Worcester, MA, USA. {clstlaurent, rvcowlagi}@wpi.edu.

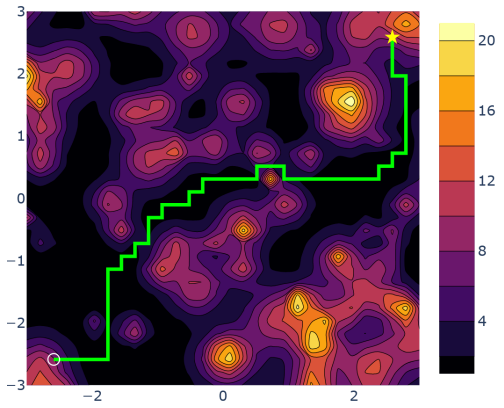

Fig. 1. Example field with goal (Star), start (Circle), and threat minimized optimal path (Green Line).

Variational approaches to ease the computational burden of matrix inversion are detailed in [7]–[9]. Optimization of the kernel structure and hyperparameters is studied in [10], [11].

Sensor placement aims to minimize uncertainty and spatial coverage of an observable domain [12] with performance measures such as information entropy and mutual information [13]. Optimal sensor placements in GP estimated fields are studied in [14]. Clustering algorithms have also been used to find sensor configurations, namely [15] uses K-Means for detecting degrees of freedom in frequency response functions for catching placement redundancies and in [16] density-based clustering was used for optimal placement in office spaces. Evidential c-means clustering is utilized to find the minimum number of sensors for water leak monitoring [17]. An interactive planning and sensing scheme which attempts to minimize the number of iterations to plan a path with an optimality guarantee is studied in [18].

Statement of Contributions: We detail a coupled sensor configuration and path-planning algorithm which iteratively configures extroceptive mobile sensors to observe regions of direct relevance to the path-planning objective. The method relies on a statistical environment model based on Gaussian Process Regression. As optimization of the sensor configuration is computationally expensive, we develop an adaptive strategy which switches between partition and density-based cluster analysis. This switching enables sensor network observations which either attempt to maximize coverage quantity or maximize coverage quality. The clustering approaches enable a significant reduction in computational complexity while still achieving performance from direct sensor configuration optimization. Additionally, we show that the discovery of cluster centers can be utilized as a dimen-

Symbol	Definition	Description
$\mathcal{E}$	$\mathcal{E} \subset \mathbb{R}^2$	Environment
$\mathcal{W}$	$\mathcal{W} \subset \mathcal{E} \subset \mathbb{R}^2$	Workspace
N_{g}	$i = 1, 2, \ldots, N_{\mathrm{g}}$.	Number of Grid Points in $\mathcal{W}$
V	$V = [N_{\mathrm{g}}]$	Vertices
E	$E = \{V_i, V_j\}\, i \neq j\ \forall i, j \in N_{\mathrm{g}}$	Edges of adjacent grid points
$\mathcal{G}$	$\mathcal{G} = (V, E)$	Graph
$\boldsymbol{p}_i$	$\boldsymbol{p}_i = (p_{ix}, p_{iy})$	Coordinate of the i^{th} grid point
Δp	$\Delta p \in \mathbb{R}$	Distance of adjacent vertices
$c(\cdot)$	$c : \mathcal{E} \to \mathbb{R}_{>0}$	Strictly positive temporally static threat function
$\boldsymbol{f}$	$\boldsymbol{f} \in \mathbb{R}^{N_{\mathrm{g}}}$	Threat estimate vector
P	$P \in \mathbb{R}^{N_{\mathrm{g}} \times N_{\mathrm{g}}}$.	Estimate covariance matrix
$\boldsymbol{\pi}$	$\boldsymbol{\pi} = (\boldsymbol{\pi}[0], \boldsymbol{\pi}[1], \ldots, \boldsymbol{\pi}[A])$	Sequential path, no repetition, between $i_{\mathrm{start}}, i_{\mathrm{goal}} \in V$
$\boldsymbol{v}_{\boldsymbol{\pi}}$	$\boldsymbol{v}_{\boldsymbol{\pi}}[i] = 1$ if $i = \boldsymbol{\pi}[j]$ for $j \in [A] \backslash 0$ else $\boldsymbol{v}_{\boldsymbol{\pi}}[i] = 0$	Path incidence vector
$\mathcal{J}(\boldsymbol{\pi})$	$\mathcal{J}(\boldsymbol{\pi}) := \Delta p \sum_{j=1}^{A} c(\boldsymbol{p}_{\boldsymbol{\pi}_j})$	Path cost
N_{s}	$N_{\mathrm{s}} \in \mathbb{N}$	Number of sensors available
$\boldsymbol{s}_k$	$\boldsymbol{s}_k \in \mathcal{W}$	Sensor center for k^{th} sensor
ϱ_k	$\varrho_k \in \mathbb{R}_{>0}$	Sensor radius for k^{th} sensor
$\mathcal{S}_k$	$\mathcal{S}_k \subset \mathcal{E}$	Circular FoV for k^{th} sensor
$\boldsymbol{C}$	$\boldsymbol{C} = \{\boldsymbol{s}_1, \varrho_1, \boldsymbol{s}_2, \ldots, \varrho_{N_{\mathrm{s}}}\}$	Sensor Network Configuration
$V_{\mathcal{S}_k}$	$V_{\mathcal{S}_k} := \{\mathcal{S}_k \cap V\}$	Vertices in network FoV
$\boldsymbol{\nu}_k$	$\boldsymbol{\nu}_k[i] = 1$ if $i \in V_{\mathcal{S}_k}$ else $\boldsymbol{\nu}_k[i] = 0$	Cover incidence for k^{th} sensor
$\boldsymbol{\nu}$	$\boldsymbol{\nu} := (\boldsymbol{\nu}_1 \vee \boldsymbol{\nu}_2 \vee \ldots \vee \boldsymbol{\nu}_k)$	Sensor network cover incidence
M_k	$M_k = \lvert\boldsymbol{\nu}_k\rvert \in \mathbb{N}$	Number of measurements made by k^{th} sensor
η_{km}	$\eta_{km} \sim \mathcal{N}(0, \sigma_k^2)$	The i.i.d measurement error
z_{km}	$z_{km} = c(\boldsymbol{x}_{km}) + \eta_{km}$	Sensor measurement of k^{th} sensor at m^{th} point
$\boldsymbol{z}$	$\boldsymbol{z} = [z_{11} \ldots z_{1M_1} \ldots z_{N_{\mathrm{s}}M_{N_{\mathrm{s}}}}]^{\intercal}$	All sensor measurements
ε	$\varepsilon \in \mathbb{R}_{>0}$	Termination threshold

TABLE I - Table of Common Notation

sionality reduction for the direct configuration optimization.

This paper is organized as follows. Section II presents the problem background and formulation. Section III describes the CSCP algorithm with adaptive clustering. Section IV presents the numerical studies and results. Section V concludes the paper with a summary of the findings.

II. Problem Formulation

We denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively, and by $[N]$ the set $\{1, 2, \ldots, N\}$ for any $N \in \mathbb{N}$. For any $\boldsymbol{a} \in \mathbb{R}^N$, $\boldsymbol{a}[i]$ is the i^{th} element of $\boldsymbol{a}$ and $\mathrm{diag}(\boldsymbol{a})$ denotes the $N \times N$ diagonal matrix with the elements of $\boldsymbol{a}$ on the principal diagonal. For any matrix $A \in \mathbb{R}^{M \times N}$, $A[i, j]$ is the element in the i^{th} row and j^{th} column. $\boldsymbol{I}_{(N)}$ denotes the identity matrix of size N.

The main elements of the problem are (1) the acting agent, (2) the environment, and (3) the sensor network. The acting agent is *passive* and has only the requirement to follow the path provided upon termination of the algorithm. The environment, $\mathcal{E}$, contains a workspace $\mathcal{W}$, which is a closed square region the acting agent sequentially traverses. It utilizes a graph structure where vertices V are uniquely associated with grid points in $\mathcal{W}$ and edges E are the sequential connections along it's path. The acting agent's path within the environment aims to minimize the cumulative exposure to threats c. To develop this path plan, we require an estimate of the threat field $\boldsymbol{f}$, which is accomplished by iteratively determining a configuration, $\boldsymbol{C}$, of the sensor network to locations of highest relevance to the path-planning.

Coupled Sensor Configuration and Path-Planning

Let $\ell = 0$, $\boldsymbol{f}_0 = \boldsymbol{0}$, $P_0 = \chi \boldsymbol{I}$, $\boldsymbol{\mathcal{I}} = \emptyset$
Solve for $\boldsymbol{\pi}_0^*$
while $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) > \varepsilon$ **do**
  Perform a Sensor Configuration Strategy
  Record measurements $\boldsymbol{z}$
  Increment iteration counter $\ell := \ell + 1$
  Find GPR-based threat field estimate $\boldsymbol{f}_\ell$ and error covariance P_ℓ
  Use Dijkstra's algorithm to find path $\boldsymbol{\pi}_\ell^*$ with minimum expected cost $\overline{\mathcal{J}}_\ell(\boldsymbol{\pi}_\ell^*)$
end while

Fig. 2. Pseudocode for an iterative algorithm to solve Problem 1.

A. Problem Formulation

We aim to drive the uncertainty of the estimated path cost to converge below a termination threshold, ε. The iterative sensor configurations and their observations develop the threat field estimate and in turn enable an estimated optimal path, $\boldsymbol{\pi}^*$, to be discovered. The main problem we wish to solve can then be written as follows:

Problem 1. *Over a finite number of iterations* $\ell = 0, 1, \ldots, L$, *find sensor configurations* $\boldsymbol{C}_\ell$ *and a path* $\boldsymbol{\pi}^*$ *of minimum expected cost* $\overline{\mathcal{J}}^* := \mathbb{E}[\mathcal{J}(\boldsymbol{\pi}^*)]$ *that satisfies* $\mathbb{E}[(\mathcal{J}(\boldsymbol{\pi}^*) - \overline{\mathcal{J}}^*)^2] \leqslant \varepsilon$.

III. Coupled Sensor Configuration and Path-Planning

Herein, we detail a coupled sensor configuration and path-planning (CSCP) algorithm which aims to solve problem 1 in minimal iterations. The algorithm iteratively finds optimal sensor locations (location and FoV), updates the threat field estimate and error covariance matrix, and finds a candidate optimal path $\boldsymbol{\pi}_\ell^*$ which is evaluated against a termination threshold, ε. This path is computed using Dijkstra's algorithm [19] using the current iteration's estimated mean threat $\boldsymbol{f}_\ell$ at each grid point, with an offset of an arbitrarily small constant applied to each vertices' euclidean distance to the goal. This offset aims to reject long paths to goal when an equally good path that is shorter exists. However, we do not include this in the expected cost of the path due to its equivalency in both the true and estimated field costs. The expected cost of a path can be calculated as:

$$\overline{\mathcal{J}}_\ell(\boldsymbol{\pi}) = \mathbb{E}[\mathcal{J}(\boldsymbol{\pi})] = \Delta p\ \boldsymbol{f}_\ell^{\intercal} \boldsymbol{v}_{\boldsymbol{\pi}}. \tag{1}$$

The termination threshold is compared against the current iteration path cost variance, which is written as:

$$\mathrm{Var}_\ell(\boldsymbol{\pi}) := \mathbb{E}[(\mathcal{J}(\boldsymbol{\pi}) - \overline{\mathcal{J}}_\ell(\boldsymbol{\pi}))^2] = (\Delta p)^2 \boldsymbol{v}_{\boldsymbol{\pi}}^{\intercal} P_\ell \boldsymbol{v}_{\boldsymbol{\pi}}. \tag{2}$$

The primary step which most directly affects iterations until convergence, as we describe in more detail in III-D, is the sensor configuration method. The procedure we present aims to minimize the required iterations to find a near-optimal path that satisfies convergence criteria. An outline of the algorithm is shown in Fig. 2.

A. Algorithm Initialization

The algorithm is initialized with an optimistic field estimate, $\boldsymbol{f}_0 = \boldsymbol{0}$, a low confidence threat covariance matrix

with a vertex independence assumption, $P_0 = \chi \boldsymbol{I}_{(N_g)}$ where $\chi \gg 1$ is an arbitrary constant, and an empty identified vertex set $\boldsymbol{\mathcal{I}} = \emptyset$ at iteration $\ell = 0$. Given this initialization of parameters, the initial candidate path $\boldsymbol{\pi}_0^*$ is of minimum length. After initialization, the algorithm proceeds to iteratively perform sensor configuration, update the probabilistic field estimate, and finds a new candidate path until the path cost variance converges below the termination threshold ε.

B. Nonparametric Statistical Environment Modeling

Before detailing the iterative portion of the CSCP algorithm, we first introduce the environment modeling stage as it is the backbone to the entire procedure. To determine the threat field estimate $\boldsymbol{f}$ and related covariance matrix P, we utilize Gaussian Process Regression (GPR) which is a nonparametric method for statistical modeling. GPR is a supervised learning technique which makes use of a kernel $K = \kappa(\boldsymbol{x}, \boldsymbol{x}')$ to define model shape and structure. Due to space constraints, we omit the full formulation and kernel definition, but refer the reader to [20] and highlight the key points. In this work, we utilize the squared-exponential (SE) kernel, also known as the gaussian or radial basis function kernel, which has the property $K \to 0$ as $\|\boldsymbol{x} - \boldsymbol{x}'\| \to \infty$ as it is a stationary kernel. This property combined with an assumption of mean 0 enforces optimistic field estimates in regions away from the training data, preventing the search from being trapped in suboptimal regions. This kernel requires that we find a length scale hyperparameter λ and signal height variation hyperparameter σ_f^2 by maximizing the marginal log-likelihood equation. The GPR is trained using the observations $\boldsymbol{z}$ at their respective locations. We combine observations at identical locations with a weighted update procedure as performed in [21], according to the current set of identified vertices, $\boldsymbol{\mathcal{I}}$. Finally, we determine the field estimate as $\boldsymbol{f}_\ell = K_*^\intercal K_{\boldsymbol{z}}^{-1} \boldsymbol{z}$, and it's covariance matrix $P_\ell = K_{**} - K_*^\intercal K_{\boldsymbol{z}}^{-1} K_*$, where K_* is the cross covariance of the training and test data, K_{**} is the covariance matrix of the test data, and $K_z = K + R$, where R is a diagonal of each k^{th} sensor's noise, is the training data covariance matrix with additive heteroscedastic noise from the sensor observations applied to the diagonal.

C. Path Plan Region of Interest

Prior to sensor configuration each iteration, we must define the path planning region of interest incidence vector $\boldsymbol{\tau} := [N_g]$. The statistical model of III-B provides a method for generating statistically feasible samples of the threat field. Since a GPR follows the formulation of a multivariate normal distribution, we may generate each i^{th} sample as:

$$\hat{\boldsymbol{f}}^{(i)} = \boldsymbol{f}_\ell + \boldsymbol{A}\boldsymbol{g}_i \tag{3}$$

The matrix $\boldsymbol{A}$ can be computed from the threat covariance matrix using cholesky decomposition as $\boldsymbol{A}\boldsymbol{A}^\intercal = P_\ell$. The vector $\boldsymbol{g}_i$ is therefore the i^{th} sample vector's independent normal variates. By introducing generated threat field estimates using the statistical model, we may find alternate potential path plans by recomputing Dijkstra's algorithm. This introduces a spatially exploratory element and the benefits are explored numerically in the results section. We may combine these alternate paths with the candidate optimal path and compute $\boldsymbol{\tau}$ as the region of interest incidence vector. Fig. 3 shows the initial and 10^{th} iterations for the CSCP method, the CSCP method with alternative paths, CSCP which considers the entire environment, and the adaptive cluster analysis (CLAN) method we discuss in III-D.

D. Sensor Network Configuration

Each iteration, after determining the path plan region of interest, we perform sensor configuration. This stage is responsible for configuring sensor locations and FoVs which best observe $\boldsymbol{\tau}$. To quantify this, we utilize a task-driven information gain (TDIG) metric which represents the information gain via variance reduction of $\boldsymbol{\tau}$:

$$h(\boldsymbol{C}_\ell, \boldsymbol{\tau}) := \boldsymbol{\tau}^\intercal (P_\ell - P_{\ell+1}) \boldsymbol{\tau} \tag{4}$$

1) Direct Optimization of the TDIG metric: Optimization of the TDIG metric is constrained to $\boldsymbol{s}_k \in \mathcal{E}, \varrho^{\min} \leqslant \varrho_k \leqslant \varrho^{\max}$, for each $k \in [N_s]$. The difficulty in optimization is the dependency on the posterior threat covariance matrix of a future iteration $P_{\ell+1}$, which cannot be directly determined during optimization by the sensor configuration $\boldsymbol{C}_\ell$ at iteration ℓ. We now describe two approaches to approximate the posterior to enable optimization of the TDIG metric.

Fixed Correlations: The first approach to computing the approximate posterior threat covariance matrix $\hat{P}_{\ell+1}$ is to fix the correlations between vertices. The proportional relationship between correlation and variance is used as $\hat{P}_{\ell+1} := Q\Omega_\ell Q$, where Q is the weighted update of variance given sensor cover incidence and current variance and Ω_ℓ is the current iteration threat correlation matrix. The current iteration correlation matrix can be found almost directly from the trained GPR output. We compute the correlation matrix Ω_ℓ by directly dividing the covariance matrix by the signal variance coefficient σ_f^2. We compute $Q = diag(\boldsymbol{q})$, where $\boldsymbol{q}$ is a reduction factor for each point in the workspace, as:

$$\boldsymbol{q} := (P_\ell[i,i]^{-1} + \textstyle\sum_{k=1}^{N_s} \boldsymbol{\nu}_k / \sigma_k^2)^{-1} \tag{5}$$

We note that a fixed correlation strategy is non-admissible due to the potential to overestimate the reduction given correlations. If the average posterior correlation between observed vertices decreases, we have overestimated the TDIG metric.

Independence Assumption: Alternatively, we may ignore the correlation and assume independence. In contrast, this is an admissible approach as we do not overestimate the posterior covariance matrix. We utilize this approach herein as it is directly comparable to the cluster analysis approach. We can therefore take $\hat{P}_{\ell+1} = Q$. This approximation can then be used to find a sensor network configuration $\boldsymbol{C}_\ell^*$ that maximizes the TDIG along $\boldsymbol{\tau}$ and subject to the constraints.

2) Cluster Analysis for Sensor Configuration: The TDIG objective function is non-convex nor submodular, leading to a quick locally optimal solution or a computationally expensive near global optimal solution. The training surface is non-differentiable due to the presence of corners induced by

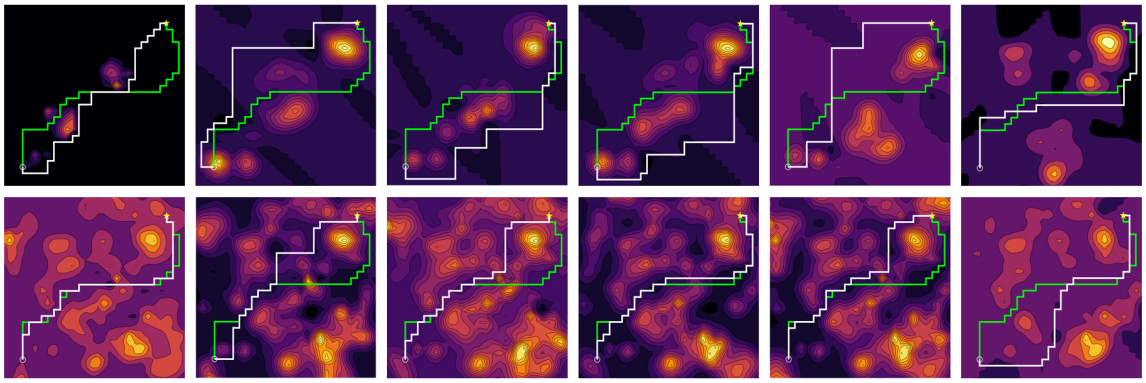

Fig. 3. Initial (Top Row) and 10^{th} Iteration (Bottom Row) field estimates and estimated optimal paths for CSCP, CLAN, CSCP-Alts, CLAN-Alts, Exploration only, and Exploration only with CLAN (Left to Right).

the discrete workspace. Herein, we describe a method based on adapting to different clustering techniques to solve these computational difficulties.

Cluster analysis is an unsupervised learning procedure which is used in many disciplines for data grouping and class discovery without the need for data labels. We utilize clustering to provide multiple groupings of workspace vertices that the sensor network can be assigned to observe. This cluster analysis for sensor configuration (CLAN) strategy is outlined in Fig. 4. CLAN utilizes an adaptive switching between *partition*-based and *density*-based clustering, which qualitatively correspond to *exploratory* and *exploitative* sensor configuration strategies, respectively. We make the determination based on the mean variance of the region of interest $\bar{P}_\ell[i,i]^{(\boldsymbol{\tau})}$ and that of the entire environment $\bar{P}_\ell[i,i]$. If $\bar{P}_\ell[i,i]^{(\boldsymbol{\tau})} \geqslant \bar{P}_\ell[i,i]$ then we perform exploratory clustering, else exploitative clustering. When the vertices along the path are not as variable as the environment, the inequality condition indicates we are near an optimal solution and therefore should observe only high variance groupings and reject the low variance vertices.

Exploratory Clustering: Exploratory clustering is concerned with spatial coverage rather than grouping spatially dense high variance regions. Such clustering can be accomplished using partition based algorithms such as K-Means [22]. We utilize weighted K-Means++, which uses the variances of each vertex as weights and an initialization strategy as described in [23]. We utilize $P_\ell[i,i]^{(\boldsymbol{\tau})}$ as the weighting, set the number of clusters $N_c = N_s$, and utilize the vertices corresponding to $\boldsymbol{\tau}$ as the data points.

Exploitative Clustering: In contrast, exploitative clustering finds clusters of high density, or tight groupings of high variance vertices. We utilize Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [24], which finds dense groupings and rejects outliers. HDBSCAN determines the relative density of data points using a notion of core density. However, we override this by computing a transformed distance matrix $\boldsymbol{\mathcal{T}} := \boldsymbol{\omega}\boldsymbol{\omega}^\intercal \odot \boldsymbol{D}$, where we compute the element-wise product of $\boldsymbol{\omega} := (P_\ell[i,i]^{(\boldsymbol{\tau})})^{-\frac{1}{2}}$ and the region of interest distance matrix $\boldsymbol{D}$. Therefore, high variance vertices have a spatially attractive force and low variance vertices have a spatially repellent force.

Adaptive Cluster Analysis for Sensor Configuration

if $\bar{P}_\ell[i,i]^{(\boldsymbol{\tau})} \geqslant \bar{P}_\ell[i,i]$ **then**
  Obtain $\boldsymbol{s}_k$ from Weighted K-Means++ with $N_c = N_s$, data points corresponding to $\boldsymbol{\tau}$, and data weights $P_\ell[i,i]^{(\boldsymbol{\tau})}$
else
  Find region of interest transform weights $\boldsymbol{\omega} = (P_\ell[i,i]^{(\boldsymbol{\tau})})^{-\frac{1}{2}}$
  Find the transformed distance matrix $\boldsymbol{\mathcal{T}} = \boldsymbol{\omega}\boldsymbol{\omega}^\intercal \odot \boldsymbol{D}$
  Obtain $\boldsymbol{\varsigma}, \boldsymbol{\psi}$ from HDBSCAN($\boldsymbol{\mathcal{T}}$)
  $N_c = \min\{N_s, |\boldsymbol{\varsigma}|\}$ clusters $\boldsymbol{\tau}_1^{(i)}, \ldots, \boldsymbol{\tau}_{|\boldsymbol{\varsigma}|}^{(i)}$ from sort($\boldsymbol{\varsigma}$)
  Obtain $\boldsymbol{s}_k$ as per equation (6)
end if
for $k \in N_c$ **do**
  Find each sensor position per equation (6)
  Find each cluster radius $\varrho_k = \max\{\varrho^{\min}, \min\{\varrho^{\max}, d_{\max_k}\}\}$
  if $(\boldsymbol{\nu}_k \wedge \boldsymbol{\tau}^{(i)}) = \mathbf{0}$ **then**
    Translate $\boldsymbol{s}_k$ to nearest neighbor
  end if
end for

Fig. 4. Pseudocode for sensor configuration with cluster analysis.

We then perform HDBSCAN with a minimum cluster size of 2 and extract clusters pertaining to leafs on the condensed cluster tree. HDBSCAN then returns the stability of each cluster $\boldsymbol{\varsigma}$ along with the class probability for each vertex $\boldsymbol{\psi}$.

If we obtain at least one stable cluster from HDBSCAN, we proceed to obtain the associated points corresponding to each cluster $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{N_c}$ where the number of clusters is defined as $N_c = \min\{N_s, |\boldsymbol{\varsigma}|\}$ and we take the largest clusters from the sorted $\boldsymbol{\varsigma}$. We provide $\boldsymbol{\tau}$ and $N_c = \min\{N_s, |\boldsymbol{\tau}|\}$ as the input parameters, and assign $\boldsymbol{\psi} = \mathbf{1}$. If all points are classified as noise, we treat it as a single cluster.

After obtaining the clusters by exploitative clustering, we can find the sensor position as the weighted average of points in each cluster. For $k \in N_c$ we find $\boldsymbol{s}_k$ as:

$$\boldsymbol{s}_k := \textstyle\sum_{j=1}^{|\boldsymbol{\tau}_k|} \boldsymbol{\psi}_{k_j} \boldsymbol{p}_{\boldsymbol{\tau}_{k_j}} / \sum_{j=1}^{|\boldsymbol{\tau}_k|} \boldsymbol{\psi}_{k_j} \tag{6}$$

HDBSCAN does not need to configure all N_s sensors if $N_c < N_s$, thereby reducing the number of sensor configurations required each iteration.

For both exploratory and exploitative clustering, the radius is $\varrho_k = \max\{\varrho^{\min}, \min\{\varrho^{\max}, d_{\max_k}\}\}$, where $d_{\max_k}$ is the maximum distance found between any point in the cluster and $\boldsymbol{s}_k$. We shift the k^{th} sensor position to the closest data point if $(\boldsymbol{\nu}_k \wedge \boldsymbol{\tau}) = \mathbf{0}$.

An additional benefit of clustering is the ability to use it as a dimensionality reduction technique for the direct optimization of the TDIG metric. By fixing the cluster centers, we may optionally *polish*, the solution by optimizing only the radius parameter for each sensor. Fig. 5 shows the initial and 10^{th} iterations for 'polished' variations of CLAN for each region of interest (path, alternates, entire environment).

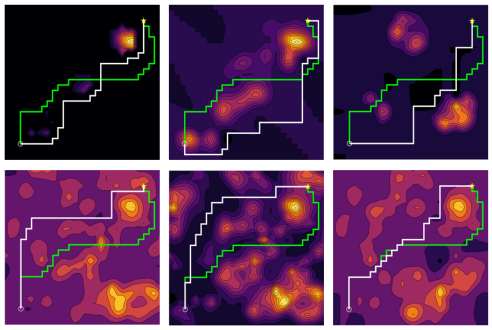

Fig. 5. Initial (Top Row) and 10^{th} Iteration (Bottom Row) field estimates and estimated optimal paths for polished variants of CLAN, CLAN-Alts, and Exploration only with CLAN (Left to Right).

E. Algorithm Termination

After the statistical model of the environment has been created as detailed in III-B, Dijkstra's algorithm is used to compute the new candidate path plan $\boldsymbol{\pi}_\ell^*$. The algorithm terminates if this path satisfies the termination threshold as $\mathrm{Var}_\ell(\boldsymbol{\pi}_\ell^*) < \varepsilon$. We note that small values of ε indicate a high desired path cost confidence and vice versa.

IV. Results and Discussion

In this section, we assess the performance of the aforementioned method on both the example field we presented and a numerical study with randomly generated environments.

A. Performance Analysis on Example Field

Each method was run for 25 iterations and the path variance was recorded as shown in Fig. 6. A few key observations can be made from this single example. The CSCP, CLAN, and CLAN with polishing all perform the best over the 25 iterations. CSCP took an average of 181 seconds, but CLAN method only took 67 seconds and with polishing 121 seconds. This shows that the CLAN approach can have comparable convergence to low path variance in significantly less time. The task-driven methods converge to nearly 0.01 whereas exploration is nearly 10 times larger. This shows the superiority of task-driven sensor configuration. Finally, the CLAN approach with and without polishing struggles to converge quickly when performing exploration.

B. Performance Analysis on Randomly Generated Fields

We performed a numerical study in which we ran 100 trials on environment areas varying from $9km^2, 25km^2, 49km^2$, workspace resolutions of 121, 441, 961, 1681, and 2601 grid points, and number of sensors varying from 1, 3, 5, 7, and 9 sensors. The sensors were constrained between

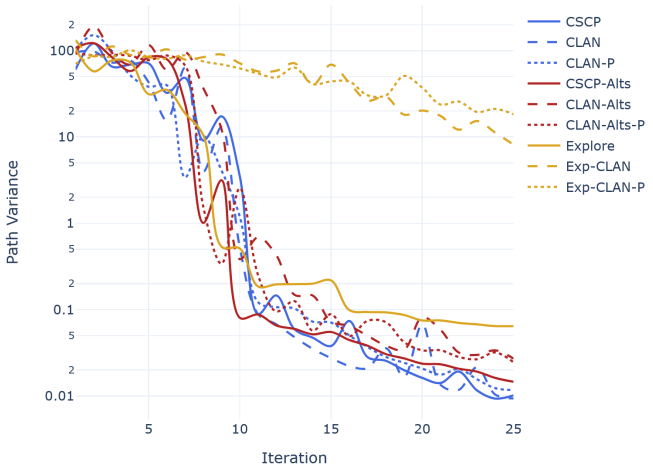

Fig. 6. Iterations vs. Path Variance for the studied methods

	Iterations	Observations	Field %	Time [sec]
CSCP	**8.56**	47.84	65.22	164.97
CLAN	11.17	53.81	60.50	**71.45**
CLAN-P	15.80	**46.18**	**56.50**	140.41
CSCP-Alts	**9.45**	**51.11**	70.86	201.36
CLAN-Alts	16.99	73.56	**62.56**	**85.01**
CLAN-Alts-P	12.40	53.93	67.31	168.17
Exp	**14.16**	**73.44**	**98.45**	501.72
Exp-CLAN	36.67	195.39	98.73	**461.81**
Exp-CLAN-P	32.86	173.41	99.01	799.13

TABLE II - Average Results of the Numerical Study

heights of $0.1km$ and $1km$, with FoV radius was equal to half the height. We used a sensor noise modeled as $\sigma_k^2 = \frac{1}{2}\log(1+\exp^{\pi\varrho_k^2}) - 0.1505$, which is monotonically increasing for $\varrho_k \geqslant 0$. Table II shows the average results of the numerical study excluding $N_s = 1$, and Fig. 7 shows N_s versus the average iterations and time for each method.

1) Task-Driven vs. Exploration Results: The results indicate optimizing sensor configuration which explores the entire environment proves to be more expensive for all metrics. This shows that coupling sensor configuration and path-planning is both physically and computationally beneficial by orders of magnitude in certain cases.

2) Direct Optimization vs. Adaptive CLAN: Assessing the performance between direct optimization with CSCP and using the adaptive CLAN approach, we notice that the CLAN strategy saves a significant amount of computation time. By polishing the result, we see that CLAN-P has the fewest number of required sensor observations and required field coverage of any method. The drawback that can be visualized in Fig. 7 and Table II is that the CLAN approach does not perform as well when doing exploration rather than task-driven sensor configuration.

V. Conclusions

In this paper we described a coupled sensor configuration and path-planning algorithm which updates its environment estimate using Gaussian Process Regression after iteratively obtaining measurements from a sensor network. We present

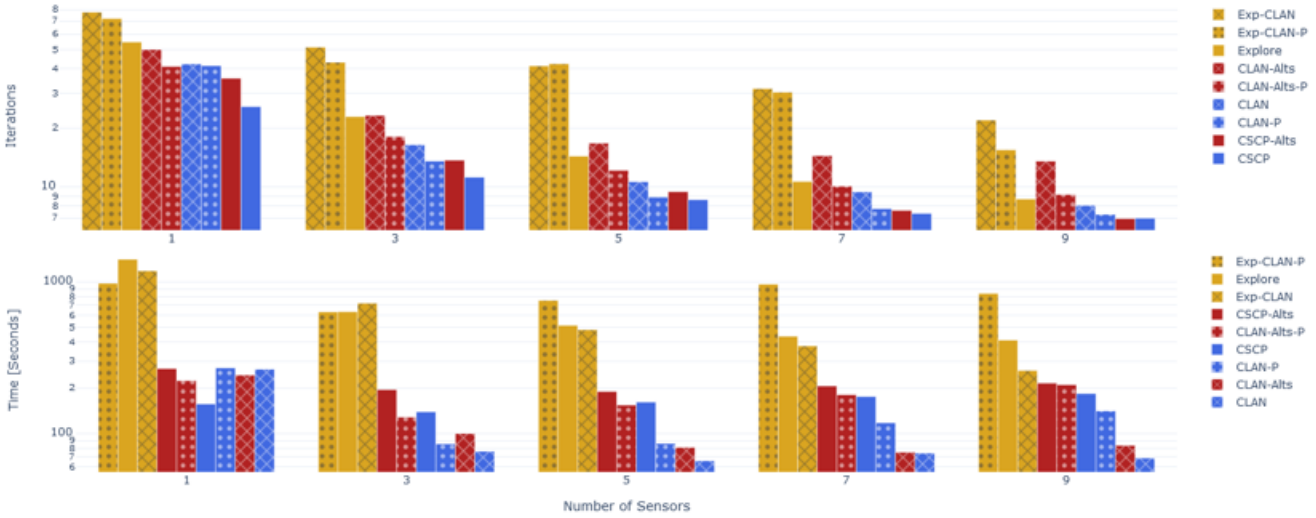

Fig. 7. Number of Sensors vs. Average Iterations and Average Time for the studied methods

a fast approximation via an adaptive cluster analysis scheme which leverages exploratory and exploitative clustering depending on the environment variance and path cost variance. We show that the task-driven approaches outperform environment exploration and that the adaptive cluster analysis can achieve performance close to direct optimization but in significantly less time. Finally, we show that the adaptive cluster analysis can be used as a dimensionality reduction scheme by finding sensor FoV centers and then optimizing just the sensor radial coverage.

Acknowledgement: This research is funded in part by AFOSR grant #FA9550-17-1-0028.

REFERENCES

[1] S. M. LaValle, "Motion Planning," *IEEE Robotics Automation Magazine*, vol. 18, no. 1, pp. 79–89, Mar. 2011.

[2] A. S. H. H. V. Injarapu and S. K. Gawre, "A survey of autonomous mobile robot path planning approaches," in *2017 International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE)*, Oct. 2017, pp. 624–628.

[3] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, Dec. 2016.

[4] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, Jan. 2020.

[5] D. J. MacKay, "Introduction to Gaussian processes," *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.

[6] C. E. Rasmussen and K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2005.

[7] S. Fine, "Efficient SVM Training Using Low-Rank Kernel Representations," p. 22.

[8] V. Tresp, "A Bayesian Committee Machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, Nov. 2000.

[9] C. K. I. Williams and M. Seeger, "Using the Nyström Method to Speed Up Kernel Machines," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 682–688.

[10] D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin, "Structure discovery in nonparametric regression through compositional kernel search," in *International Conference on Machine Learning*, 2013, pp. 1166–1174.

[11] A. B. Abdessalem, N. Dervilis, D. J. Wagg, and K. Worden, "Automatic Kernel Selection for Gaussian Processes Regression with Approximate Bayesian Computation and Sequential Monte Carlo," *Frontiers in Built Environment*, vol. 3, 2017.

[12] D. Ramsden, "OPTIMIZATION APPROACHES TO SENSOR PLACEMENT PROBLEMS," p. 80, 2009.

[13] D. Cochran and A. O. Hero, "Information-driven sensor planning: Navigating a statistical manifold," in *2013 IEEE Global Conference on Signal and Information Processing*, Dec. 2013, pp. 1049–1052.

[14] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, *Robust Sensor Placements at Informative and Communication-Efficient Locations*, 2010.

[15] S. Li, H. Zhang, S. Liu, and Z. Zhang, "Optimal sensor placement using FRFs-based clustering method," *Journal of sound and vibration*, vol. 385, pp. 69–80, 2016.

[16] D. Yoganathan, S. Kondepudi, B. Kalluri, and S. Manthapuri, "Optimal sensor placement strategy for office buildings using clustering algorithms," *Energy and buildings*, vol. 158, pp. 1206–1225, 2018.

[17] R. Sarrate, J. Blesa, and F. Nejjari, "Clustering techniques applied to sensor placement for leak detection and location in water distribution networks," in *22nd Mediterranean Conference on Control and Automation*. Palermo, Italy: IEEE, Jun. 2014, pp. 109–114.

[18] B. S. Cooper and R. V. Cowlagi, "Interactive planning and sensing in unknown static environments with task-driven sensor placement," *Automatica*, vol. 105, pp. 391–398, Jul. 2019.

[19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "Section 24.3: Dijkstra's algorithm," *Introduction to Algorithms*, pp. 595–601, Jan. 2001.

[20] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press, 2012.

[21] H. Durrant-Whyte and T. C. Henderson, "Multisensor data fusion," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Cham: Springer International Publishing, 2016, pp. 867–896.

[22] J. Wang and X. Su, "An improved K-Means clustering algorithm," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, May 2011, pp. 44–46.

[23] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," p. 11.

[24] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.