A decentralized strategy for multi-robot sampling/patrolling: theory and experiments

Alessandro Marino, Gianluca Antonelli, A. Pedro Aguiar, António Pascoal, Stefano Chiaverini

***Abstract*—This paper presents a decentralized coordinated strategy for multi-robot patrolling missions. Patrolling is here interpreted within the framework of the sampling problem. To be applied in practice, several realistic constraints and the time/spatial variance of the information are explicitly taken into account. The proposed approach is well rooted in the concepts of Voronoi tessellations and Gaussian Processes. Each robot, based only on local information, computes the next point to visit according to a given performance criteria. Numerical simulations and experiments involving three autonomous marine surface robots in a harbour scenario at the Parque Expo site in Lisbon, Portugal, are presented.**

I. Introduction

Multiple robots are becoming pervasive, and there is tremendous potential for their use in a large number of applications. Classical application examples include: environmental exploration and sampling, sensor deployment, terrain coverage, air and ocean monitoring and patrolling [3], [15], [31]. Many of the envisioned applications show considerable overlapping, the key differences lying in the mathematical assumption underlying the problem formulations and also in the implementation details.

Patrolling can be defined as the act of walking or traveling around an area, eventually at regular intervals, in order to protect or supervise it. Patrolling, in a more general sense, may involve the task of visiting selected places in a given area in order to assess the state of the environment in relation to the presence of intruders or the occurrence of unforeseen events. If the entire terrain cannot be monitored at all times, each location in the target area is periodically monitored. In addition, in [16] a distinction is made between *coverage-oriented* patrol and *on-demand* patrol. The former involves visiting all target points as frequently as possible, while the latter involves monitoring target points to detect an intrusion by a mobile adversary or any detected event. In [23], an analysis of the main patrolling task issues and some multi-agent-based solutions are presented; a very detailed analysis was carried out by taking into consideration such aspects as the agents type, communication skills, coordination scheme, agents perception, decision-making, etc. In [8] and [18], graph-theoretical tools were used to find the optimal solution of a mathematical problem expressing a multi-robot surveillance mission. In [2], the authors analyzed non-deterministic paths for a group of homogeneous mobile robots patrolling a frontier, under the assumption of an hostile agent trying to enter the area, where the latter has full knowledge of the algorithm. In [26], a patrolling strategy for vehicles moving on a line surrounding the area to patrol was designed based on a Finite State Automata approach. Recently, in [28] a polynomial time algorithm for the minimum refresh time problem was developed. In [34], static limited range sensors were used to detect intruders moving in the environment.

The work in [14] describes how a fleet of underwater gliders is coordinated for sampling purposes. A Virtual Body and Artificial Potentials (VBAP) based strategy is adopted. In the proposed implementation, way-points are generated from the discretization of continuous trajectories that represents the VBAP output running on a centralized unit which assumes that the vehicles move at constant speed. In [22], an ocean sampling network of 10 gliders was deployed in the Monterey Bay, California. The approach uses scalable feedback control laws to coordinate the motion of robotic vehicles into sampling patterns, designed to maximize information in the data collected. In particular, the vehicles move around closed curves and could be arranged in sub-groups. In [32], the authors present algorithms that produce persistent monitoring missions for underwater vehicles by balancing path following accuracy and sampling resolution for a given region of interest. A cost function that takes into account two competing factors as the maximization of the information value along the path and the minimization of the deviation from the planned path due to ocean currents is defined. The solution is experimented on a single underwater glider. In [17], a method to find the trajectory that maximizes a well defined information metric and that takes into account constraints as fuel, energy and time consumption is proposed. The strategy makes use of branch and bound techniques to achieve efficient optimization and is applied to the single vehicle case. Simulation results are provided.

In a real world scenario, the problem of sampling/patrolling is characterized by several constraints detailed in Section II:

- Coordinated, multi-robot strategy: robots have to coordinate in order to reach a common objective;
- Decentralized algorithm;
- Robustness with respect to a wide range of events that include temporary communication or robot losses;

A. Marino is with the University of Salerno, Via Ponte don Melillo, 84084, Salerno (SA), Italy almarino@unisa.it

G. Antonelli and S. Chiaverini are with Università di Cassino e del Lazio Meridionale, Via G. Di Biasio 43, 03043, Cassino (FR), Italy, {antonelli,chiaverini}@unicas.it.

A.P. Aguiar is with the University of Porto (FEUP), Portugal, and Laboratory of Robotics and Systems in Engineering and Science (LARSyS), Lisbon, Portugal, pedro.aguiar@fe.up.pt.

A. Pascoal is with the Laboratory of Robotics and Systems in Engineering and Science (LARSyS), ISR/IST, Torre Norte 8, Av. Rovisco Pais, 1049-001 Lisbon, Portugal, antonio@isr.ist.utl.pt.

- Flexibility/scalability with respect to the number of robots;
- Possibility to tailor the strategy with respect to the communication capabilities;
- Real-time implementation.

To the best of our knowledge, the solutions described in the literature for sampling/patrolling do not fulfil all of the above realistic constraints fully. This places strong constraints on their applicability in real world scenarios. For these reasons, the solution proposed in the present paper builds on a solid mathematical framework but embodies in its structure selected heuristic mechanisms to make it applicable in practice. To assess its applicability, extensive validation tests were performed with real vehicles in a real environment.

The proposed solution builds strongly on two fruitful mathematical tools: Voronoi tessellations and Gaussian processes, which we describe briefly below.

A Voronoi tessellation or partition of a set with respect to a finite number of points (also called seeds) is a collection of regions (cells) with the following property: each point in a region related to a specific seed is closer to that seed than to any other [13]. An important property of Voronoi cells it that, given a seed, the corresponding cell can be computed based only on the position of the other seeds in its neighborhood.

This property has been recently used in the robotics community to distribute a region of interest among the robots in a team [7], [10]. In fact, each robot can compute its Voronoi cell provided that it is able to know its neighbors' positions by proper exteroceptive sensors and/or communication.

Patrolling shares several concepts with the problem of coverage or sampling, such as the need to maneuver a group of robots, equipped with proper sensors, so that the amount of information concerning a certain phenomenon is optimized in some sense. In the case, for example, of estimating a scalar field such as salinity and temperature in the ocean, it is the objective of the robots to measure the field at a reduced set of locations so that its estimate becomes also available at non-sampled points. An appealing mathematical tool used for this purpose is given by the theory of Gaussian processes [29], which also allow for the computation of the uncertainty involved in the estimation process [19].

In the algorithm proposed, Gaussian Processes allow us to address in an elegant manner both temporal and spatial variability objectives. The first is captured in the form of a forgetting factor, while the second reflects the need to patrol certain regions more often.

The two mathematical tools above have been also properly framed within a realistic communication infrastructure, in the sense that the algorithm can be tailored with respect to the available communication bit/rate among neighbor robots.

This paper builds upon and extends the work in [6], where the theoretical background was briefly described, and in [25] describing experimental details. With respect to these papers, the present one does an in-depth research of the approximation error underlying the decentralized solution, exploits the property of the Gaussian processes as a means to patrol non-uniform areas, provides an efficient computational strategy for the function to be maximized and enriches the experimental results with a decentralized intruder surrounding functionality. In [24], this algorithm has been experimental tested in the underwater case.

The paper is organized as follows. Section II describes briefly the key mathematical tools and background results exploited in this work. Section III introduces the patrolling strategy, including some implementation aspects. Finally, Section VI contains the results of numerical simulations and experiments.

II. Problem description and mathematical background

The problem at hand is characterized by challenging theoretical as well as implementation issues. The constraints listed in Section I, strongly motivated by the need to perform experimental validation of the algorithm derived, are here briefly discussed:

- As discussed at length in the literature, robotic missions such as the one addressed in this paper are more efficient by means of a coordinated, multi-robot strategy.
- Decentralization. One central computational unit represents a weak point for a multi-robot algorithm. This is particularly significant when a security application, such as patrolling, is considered;
- Robustness. When robots move around in the real world, they are necessarily confronted with a number of unexpected events that may seriously jeopardize the success of their missions. It is not realistic to design a multi-robot algorithm that is not *robust*, in a wide sense, with respect to the possibility that one or more robots simply stop functioning, or hold in place, or that the communication among them may experience temporary black outs;
- Scalability. Several definitions of scalability exist with respect to distributed computations, depending on the research community. Here, we are interested in the most conservative, i.e., the computational burden associated to each robot does not change with the number of robots;
- Communications. Different communication technologies come with different bandwidths and ranges that impact directly on the performance achievable with multiple vehicle patrolling algorithms. A reliable algorithm must be customizable with respect to the available communication bandwidth;
- Real-time. Each robot needs to take *decision* in real-time, thus preventing the use of off-line planning algorithms;
- In view of practical implementation, additional features such as, for example, obstacle avoidance policies, need to be considered.

These constraints are fundamental for experiments in a real scenario and, differently form other solutions, are naturally taken into account by the designed solution.

We now describe the patrolling problem addressed in this paper: consider a region $\mathcal{A} \in \mathbb{R}^l$, $l = 2, 3$ and a function $y = f(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \mathcal{A}$, with $f : \mathbb{R}^l \times [0, \infty) \rightarrow \mathbb{R}_{0+}$,

where f is application dependent. For example, in the case of patrolling/security applications y represents the level of safety of the environment at point $\boldsymbol{x}$ and at time t. For sampling problems, y can be, for example, the temperature or salinity in a given region.

In both the cases of patrolling and sampling, the function f exhibits a spatial correlation that is mainly affected by the nature of the underlying phenomenon under study and/or the sensor suite used. For example, a thermometer gives the temperature at a single point but it is likely that this value show correlation with other measurements taken in its neighborhood. It is also important to stress that the function may be time-varying, at a scale that once again is determined by the phenomenon under investigation. In practice, the function may exhibit a small time constant (e.g. in the case of atmospheric phenomena), or a longer one in the case of security applications.

Our main goal is to develop a strategy to estimate the function f by taking appropriate measurements using robots equipped with sensor suites. In what follows, we let N_r denote the number of robots and $x_{r,i}(t) \in \mathbb{R}^l$, $i = 1, 2, \ldots, Nr$ the position of robot i at time t. In addition, as required in Section II-A, each robot is assumed to be able to sense or receive the position of some neighbors, where the term neighbor indicates a robot $\boldsymbol{x}_{r,j}$ that is close to $\boldsymbol{x}_{r,i}$ with respect to a certain metric (for example the Euclidean distance). This estimation problem has been studied extensively in the literature; see for example [11], [14], [15], [22], and the references therein.

The patrolling task shares several aspects with sampling. In particular, at given instant the knowledge about the safety status of a location in the area depends on the team configuration and, therefore, on the robots' positions $\boldsymbol{x}_{r,i}, \forall i = 1, 2, \ldots, N_r$. Thus, given a robot with position $\boldsymbol{x}_{r,i}$, it is possible to state whether this position is safe or not based on the value of function f at this point; it can be argued that this information can be used to infer the status at other locations in the neighbourhood (*spatial* dependence). An example is represented by an intruder or a toxic substance spill at location $x_{r,i}$. Finally, it can be also argued that in a dynamic scenario, a location that has been marked as safe (because it was visited in the past) but has not been visited by any of the patrolling robots for a certain amount of time should no longer be considered as safe; instead, a high uncertainty should be associated to its status. Thus, high uncertainty must be associated not only to those cells that have remained unvisited but also to the cells that were visited but long back in time (*time* dependence). An example is represented by a moving intruder, a moving oil spill, or a changing temperature field.

The aim is, therefore, to estimate the map function $f(\boldsymbol{x}, t)$ by reducing the uncertainty in its knowledge; namely, by bringing the robots toward those locations characterized by a high degree of uncertainty. It is assumed that a robot will be able to measure the degree of safety, f, of a location by means of some sensor as, for example, a vision sensor. Moreover, it should be clear that the developed strategy is suitable both for patrolling and sampling as shown in [6]. In addition, the use of multi-robot systems requires a coordination mechanism among robots. To this aim, in the next section the well known Voronoi tessellation is briefly introduced as it will be useful as a way for both distributing the calculus of the function f and coordinate the motion of robots. A Gaussian process strategy is, instead, used to predict function f.

A. The Voronoi partition

Voronoi partitions (or diagrams) are subdivisions of a set $\mathcal{D}$ characterized by a metric with respect to a finite number of seed points belonging to that set.

Assuming that at the current time t the seed points are the robots' positions $\{\boldsymbol{x}_{r,1}, \boldsymbol{x}_{r,2}, \ldots, \boldsymbol{x}_{r,N_r}\}$, the corresponding N_r Voronoi cells, $Vor(\boldsymbol{x}_{r,i})$, $i = 1, 2, \ldots, N_r$ are given by

$$Vor(\boldsymbol{x}_{r,i}) = \{\boldsymbol{x} \in \mathcal{D} \quad | \quad \|\boldsymbol{x} - \boldsymbol{x}_{r,i}\| \leq \|\boldsymbol{x} - \boldsymbol{x}_{r,j}\|, \forall j\}.$$

The union of the Voronoi cells gives back the entire set and the intersection of two cells is always empty. The most important property of the Voroni tool for the use on decentralized robotics is that each robot can compute its own cell by applying a *local* algorithm, i.e., by simply knowing its position and the *neighbors*' positions, either by direct sensing or by communication. An example is reported in Figure 1 where the Voronoi tessellation of a three-dimensional set has been generated according to three randomly generated seed points. In this paper, neighbours must be intended in the Delaunay sense, i.e., other robots directly connected to a given teammate by an edge of the Delaunay triangulation [21]. Further details

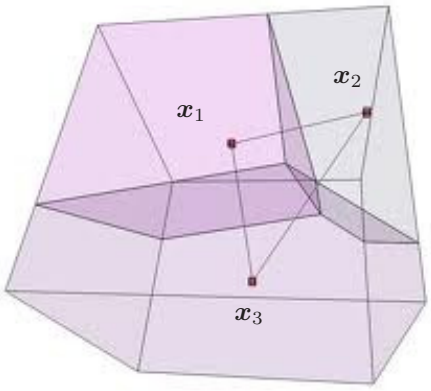

Fig. 1. Example of a Voronoi partition for 3 points in a 3D set.

on the Voronoi-based theory and its applications can be found in [13] or [27].

B. The Gaussian Processes

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [29]. One of the key features of Gaussian Processes is their potential to yield methods to predict the value of a function at any location, given a set of previously collected observations (either in space or in time), with an explicit representation of the uncertainty of that prediction. For this reason, they will

be used as a means to estimate the field f. What relates one observation to another in such cases is just the covariance function. In what follows we summarize the key facts about Gaussian Processes needed to understand the method that we propose. A comprehensive exposition of the theory can be found in [29].

We view at function $f(\boldsymbol{x}, t)$ as a zero-mean spatio-temporal Gaussian Process $f(\boldsymbol{x}) \sim \mathcal{GP}(0, \mathcal{K}(\boldsymbol{x}_1, t_1; \boldsymbol{x}_2, t_2)$ where $\mathcal{K}(\boldsymbol{x}_1, t_1; \boldsymbol{x}_2, t_2)$ is the covariance function. We will assume that the covariance function $\mathcal{K}$ is generically defined as

$$\mathcal{K}(\boldsymbol{x}_1, t_1; \boldsymbol{x}_2, t_2) = C(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|, |t_2 - t_1|) \quad (1)$$

with $C : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}^+$. Notice that both space and time are taken into account to handle also the non stationary case. For the sake of simplicity, in equation (1) we assume that the process is homogeneous, second order stationary and isotropic, which basically implies that the covariance only depends on the distance between two generic points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and on the absolute value of the time difference $t_2 - t_1$.

In the following, given the set S defined as $S = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_n, t_n)\}$ made of pair of locations $\boldsymbol{x}_i \in \mathcal{A}$ and instants of time t_i and the corresponding vector of observation $\boldsymbol{y} \in \mathbb{R}^n$, the symbol $\Sigma_S \in \mathbb{R}^{n \times n}$ represents the symmetric non-negative covariance matrix whose elements (i, j) is $\mathcal{K}(\boldsymbol{x}_i, t_i; \boldsymbol{x}_j, t_j)$.

Moreover, given a single element $(\boldsymbol{x}^*, t)$ and the set S, $\boldsymbol{\sigma}_{Sx}(\boldsymbol{x}^*, t) \in \mathbb{R}^n$ is a column vector whose i-th element is $\mathcal{K}(\boldsymbol{x}^\star, t; \boldsymbol{x}_j, t_j)$.

The objective is to predict $y^* = f(\boldsymbol{x}^*, t)$ at the generic location $\boldsymbol{x}^*$ and at the current time instant t based on the vector of observations $\boldsymbol{y}$. In the case of a multivariate normal distribution over a set S of random variables associated with n pairs of positions and time instants, the posterior distribution of y^* is characterized by a normal distribution $y^*|\boldsymbol{y} \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ with [29]:

$$\hat{\mu} = \boldsymbol{\sigma}_{Sx}(\boldsymbol{x}, t)^{\mathrm{T}} \boldsymbol{\Sigma}_S^{-1} \boldsymbol{y} \quad (2)$$

$$\hat{\Sigma} = \mathcal{K}(\boldsymbol{x}^*, t; \boldsymbol{x}^*, t) - \boldsymbol{\sigma}_{Sx}(\boldsymbol{x}^*, t)^{\mathrm{T}} \boldsymbol{\Sigma}_S^{-1} \boldsymbol{\sigma}_{Sx}(\boldsymbol{x}^*, t). \quad (3)$$

The best estimate of y^* is given by (2) and the uncertainty of the estimation is captured by its variance, described in (3). Thus, while the predicted value is useful for establishing the most likely appearance of the function f based on the available sensor data, it can also be misleading if considered in isolation. One of the key advantages of Gaussian Processes is, therefore, the possibility to compute the variance of each prediction.

Remark 2.1: In the problem considered, the i th robot measures the status of location $\boldsymbol{x}$ using dedicated sensors. We assume that the measurement made by robot i at position $\boldsymbol{x}$ and time t is given by

$$y = f(\boldsymbol{x}, t) + w_i,$$

where $w_i \sim \mathcal{N}(0, \sigma_i)$ is a white noise Gaussian Process with zero mean and standard deviation σ_i. For simplicity, we assume that $w_i = w \sim \mathcal{N}(0, \sigma)$, i.e., the robots are equipped with identical sensors. In this case, equation (2) becomes [29]

$$\hat{\mu} = \boldsymbol{\Sigma}_{Sx}{}^{\mathrm{T}} \left(\boldsymbol{\Sigma}_S + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \quad (4a)$$

$$\hat{\Sigma} = \mathcal{K}(\boldsymbol{x}, t; \boldsymbol{x}, t) - \boldsymbol{\sigma}_{Sx}(\boldsymbol{x}, t)^{\mathrm{T}} \left(\boldsymbol{\Sigma}_S + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\sigma}_{Sx}, \quad (4b)$$

where $\boldsymbol{I}$ is the identity matrix of proper dimensions.

In the above equations, the matrices $\boldsymbol{\Sigma}_S$ and $\boldsymbol{\sigma}_{Sx}$ are completely defined once the function C in equation (1) has been specified. According to [29] and [36], one possible choice for this function is the Square Exponential Covariance Function:

$$C(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|, |t_2 - t_1|) = \phi^2 e^{-\frac{\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|^2}{2\tau_s^2} - \frac{(t_2 - t_1)^2}{2\tau_t^2}}, \quad (5)$$

where ϕ is a weighting scalar parameter that will be selected to be unitary and the parameters τ_s and τ_t are positive scalars used to affect the space and time scales, respectively; the latter need to be properly designed [36]. In addition, it is worth noticing that the choice in equation (5) refers to isotropic domains and known constant τ_s and τ_t. The case of adaptive parameter estimation has also been addressed in the literature; see for example [30] and the references therein.

C. Application to the sampling/patrolling mission

From equation (3), the variance of the estimation at position $\boldsymbol{x}$ given the already acquired samples and the current time t takes the form

$$\hat{\Sigma}(\boldsymbol{x}) = C(0, 0) - \boldsymbol{\sigma}_{Sx}{}^{\mathrm{T}} \boldsymbol{\Sigma}_S^{-1} \boldsymbol{\sigma}_{Sx}. \quad (6)$$

It turns out that minimizing the uncertainty, which corresponds to minimizing the positive definite right-hand side of equation (6), is the same as maximizing (given the available degrees of freedom) the function

$$\xi_S(\boldsymbol{x}) = \boldsymbol{\sigma}_{Sx}{}^{\mathrm{T}} \boldsymbol{\Sigma}_S^{-1} \boldsymbol{\sigma}_{Sx}. \quad (7)$$

Note also that due to the time dependency of the covariance function in (5), a point that has been visited too far in the past (with respect to the time parameter τ_t) is candidate to be visited again. This feature is exploited by assigning proper time constants according to the applications.

It is interesting to reproduce graphically equation (7) for some case studies. In the simple case we consider, the region of interest is a planar square with unitary length. Three location have been visited, $S = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), (\boldsymbol{x}_3, t_3)\}$, with $\boldsymbol{x}_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{x}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$, and $\boldsymbol{x}_3 = \begin{bmatrix} 0.5 & 1 \end{bmatrix}^{\mathrm{T}}$ at time t_1, t_2 and t_3, respectively.

- First example (Figure 2 (top left)): $t_1 = t_2 = t_3 = 0$, current time $t = 0$, $\tau_s = 0.5$, $\tau_t = 8$.
- Second example (Figure 2 (top right)): $t_1 = t_2 = t_3 = 0$, current time $t = 0$, $\tau_s = 0.2$, $\tau_t = 8$.
- Third example (Figure 2 (bottom)): $t_1 = 0$, $t_2 = 7$, $t_3 = 10$, current time $t = 10$, $\tau_s = 0.5$, $\tau_t = 8$.

In the classical sampling task as in [20], [36], [37] the field to sample is related to a physical variable such as temperature, salinity, etc.; the *hyper-parameters*, ϕ, τ_s, τ_t, are usually identified or learned by training. On the other hand, in the case of patrolling a different point of view can be used: the hyper-parameters, in particular the constant τ_s and τ_t, are not

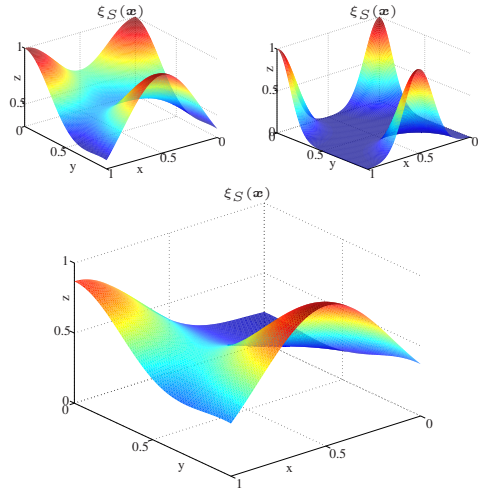

Fig. 2. Graph of the function (7). Top left: first example. Top right: second example. Bottom: third example.

related to any physical phenomenon. On the contrary, they are decided by the user and, then, known beforehand by the robots. Their value may be chosen so as to confer to the system specific behaviors, as shown in Section VI. As examples of the above concept, let us consider the following cases:

- a low value of the the parameter τ_s in equation (5) implies low *space* correlation between different locations. On the contrary, a larger value implies high correlation between even far locations. Such a feature may be used to take into account the range of robot' sensors. In addition, τ_s can be function of the location (i.e., $\tau_s = \tau_s(\boldsymbol{x})$); such a feature can be useful, e.g., for modeling different visibility conditions (eg., a robot equipped with a camera operating in a environment with non-uniform light conditions);
- a low value of the parameter τ_t in equation (5) implies low *time* correlation between cells. This means that the patrolling team needs to visit each position more frequently. On the other hand, a large value of τ_t implies that each robot does not need to get close to each cell too often to infer its status. Also in this case, the τ_t can be functions of the location (i.e., $\tau_t = \tau_t(\boldsymbol{x})$), in order to take into account the case of environments where some locations are more exposed to unexpected events than others (eg., in an harbor patrolling scenario it may be required to visit more frequently the harbour entry).
- by setting $\tau_t = \infty$ a static field is obtained as in the case of a coverage mission.

III. Proposed coordination strategy

Figure 3 illustrates the proposed control architecture for a single robot. At the top level, the planner is in charge of deciding the robot trajectory. This level will be described in Section III-A. At the lower level, the Null-Space-based-Behavioral (NSB) control approach allows following reference trajectory provided by the upper level, while properly handling unexpected events such as the presence of obstacles. The NSB has been widely used by some of the authors and its description will not be given here to avoid repetion; the interested reader will find in [4], [5], the details of this strategy and its properties.

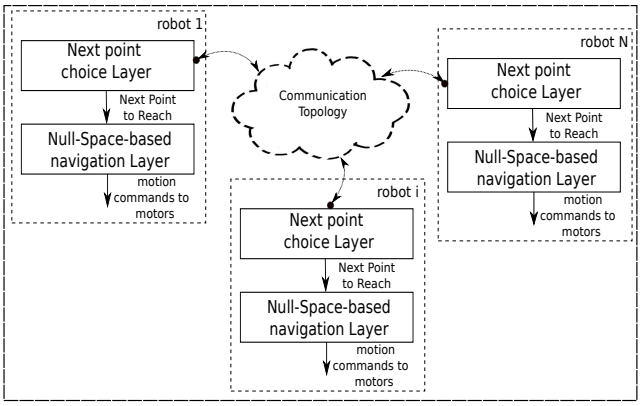

Fig. 3. Control architecture.

A. The top level: algorithm description

The top layer in Figure 3 represents the *core* of the proposed algorithm, i.e., the block in charge of computing the next point to be visited by each robot. As described in the previous Sections, our objective is the maximization of the function $\xi_S(\boldsymbol{x})$ expressed by equation (7). To this effect, a partition of the area $\mathcal{A}$ according to the Voronoi tessellations is computed, and each robot performs such a maximization in its own cell. The strategy designed for each robot is given in Algorithm 1.

Algorithm 1 Algorithmic description of the robot planner

loop
1) (if possible) exchange data with the neighbors
2) build its own cell: $Vor(\boldsymbol{x}_{r,i})$
3) select next via point $\overline{\boldsymbol{x}}_i$ in its own Voronoi cell
4) send $\overline{\boldsymbol{x}}_i$ to the NSB layer
end loop

In the remaining of this section, we will describe how to select the next point to visit. Maximization of $\xi_S(\boldsymbol{x})$ may be interpreted in several ways depending on the specific application; however, the ultimate goal is to make decisions on how to move the robots so as to increase the information available about the environment. For example, each robot may implement a pure reactive algorithm, based on the local gradient of $\xi_S(\boldsymbol{x})$.

In Figure 3, the top layers communicates among them to share useful information. In fact, whenever possible, each robot exchanges information with the neighbors to update its Voronoi cell and to maintain an estimate of the function $\xi_S(\boldsymbol{x})$ that depends on the state of the whole system [6]. Details about the information exchange and error in the estimation of $\xi_S(\boldsymbol{x})$ are given in Section IV.

In the methodology proposed, each robot chooses the next point to visit inside its own Voronoi cell $Vor(\boldsymbol{x}_{r,i})$. The most

uncertain points are represented by the lowest values of the function $\xi_S(\boldsymbol{x})$. Because $\boldsymbol{\Sigma}_S$ is a symmetric positive matrix, the lower bound of the quadratic form $\xi_S(\boldsymbol{x})$ is obviously 0. Notice that, from equations (3) and (5), its upper bound is ϕ^2. This means that, by setting $\phi = 1$, $\xi_S(\boldsymbol{x})$ is upper-bounded by 1. A proper threshold, θ, in the range $[0,\ 1]$ can be chosen and the set : $S_u(\theta) = \{\boldsymbol{x}_u \in Vor(\boldsymbol{x}_{r,i}) \mid \xi_S(\boldsymbol{x}_u) < \theta\}$ considered. The next point $\overline{\boldsymbol{x}}_i$ to visit belongs to the set $S_u(\theta)$. Among these points, $\overline{\boldsymbol{x}}_i$ is chosen according to some criteria (for example, the furthest point from the actual robot position, the point characterized by the lowest value of the function ξ_S, etc.). The choice of the constant θ plays an important role in the algorithm. Since the function $\xi_S(\boldsymbol{x})$ represents the degree of knowledge about the considered field at location $\boldsymbol{x}$ given the acquired samples, high values of θ lead to the inclusion, in the list of potential targets, points in the environment with a level of confidence sufficiently high. Instead, low values of θ cause to exclude from the list of the potential targets points of the domain with low values of $\xi_S(\boldsymbol{x})$ and, therefore, characterized by a low value of the confidence level.

Because the patrolling strategy solution must be decentralized and involve more than one vehicle, the optimal strategy that maximizes (7) may be extremely hard or even impossible to find in an analytically fashion. To overcome this hurdle and as stated in Section I, branch and bound techniques to find the optimal path according to some criteria were introduced in [17]. The solution is confined to a single vehicle case. In this work, we propose the following strategy. Let $\boldsymbol{x} = \boldsymbol{h}(s) = \boldsymbol{x}_{r,i} + (\boldsymbol{x}_u - \boldsymbol{x}_{r,i})s$ be a parametrization of the line segment joining the actual position $\boldsymbol{x}_{r,i}$ of the robot and a generic point $\boldsymbol{x}_u \in S_u$, with $s \in [0,\ 1]$; the next target in S_u is determined by

$$\overline{\boldsymbol{x}}_i = \min_{\boldsymbol{x}_u \in S_u} \frac{\int_0^1 \xi_S(\boldsymbol{h}(s))ds}{\|\boldsymbol{x}_{r,i} - \boldsymbol{x}_u\|}. \tag{8}$$

The heuristics behind the strategy (8) is that, among the unvisited points in the set S_u, the one characterized by the most unvisited path (normalized by the path length) is chosen. Another possible strategy would be to simply drive the robot toward the point of global minimum of $\xi_S(\boldsymbol{x})$ inside its Voronoi cell, that is, compute

$$\overline{\boldsymbol{x}}_i = \min_{\boldsymbol{x} \in Vor(\boldsymbol{x}_{r,i})} \xi_S(\boldsymbol{x}). \tag{9}$$

As an example, Figure 4 shows a comparison between the strategy in (8) and (9) in a $50\,\mathrm{m} \times 100\,\mathrm{m}$ rectangular environment with three vehicles. The space (τ_s) and time (τ_t) constants in equation (5) are $4.7\,\mathrm{m}$ and $400\,\mathrm{s}$, respectively. In particular, the performance index adopted is represented by the integral of $\xi_S(\boldsymbol{x})$ over the environment normalized by its area. It is worth noticing that, as $\xi_S(\boldsymbol{x}) \in [0,\ 1]$, the considered index belongs to the same interval. Because of the meaning of function $\xi_S(\boldsymbol{x})$,

- a performance index equal to 1 means that the field is completely known (no uncertainty about its value at all points);
- a performance index equal to 0 means that the field is completely unknown;
- a performance index equal to 1 can be asymptotically reached only in the case of static fields ($\tau_t = +\infty$).
- lower values of the time constant τ_t imply lower values of the index at steady-state.

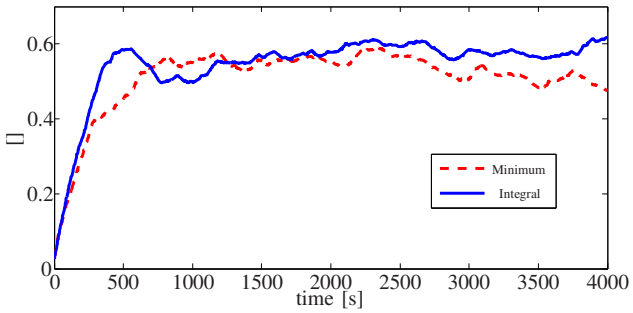

Fig. 4. Normalized integral of function $\xi_S(\boldsymbol{x})$ in a comparison between strategy in equation (8) (continuous blue line) and equation (9) (dashed red line).

In Figure 5, the same scenario is used with the strategy described by equation (8) for different values of the parameter θ introduced above. The figure shows that, in this scenario,

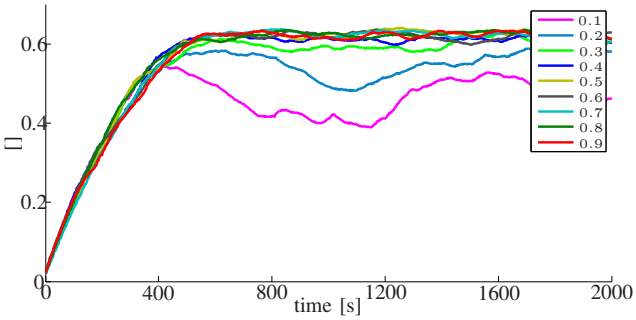

Fig. 5. Normalized integral of function $\xi_S(\boldsymbol{x})$ for different values of parameter θ with the strategy described by equation (8).

small values of θ lead to poor performance, as only points characterized by a very large uncertainty are considered. On the contrary, values higher than 0.5 allow to obtain better performance. In particular, it can be seen that, for this specific case, a good performance is obtained with $\theta = 0.5$.

IV. Implementation Issues and Error Analisys

Step 2 in Algorithm 1 requires that each vehicle computes its own Voronoi cell $Vor(\boldsymbol{x}_{r,i})$. The assumption made is that each robot has the capability to estimate the neighbor position either by direct sensing or communication. It is important to remark that this hypothesis cannot be verified in the case of robots that, because of their short range sensors, are not capable of communicating with their far neighbours. Representative work in this direction is described in [9].

A. Information exchange and approximation error

The main reason why non-parametric prediction by re-sorting to Gaussian processes is not popular for resource-constrained multi-agent systems is the fact that the optimal

prediction must make use of all cumulatively measured values. In this case, a robot needs to compute the inverse of the covariance matrix, the size of which grows as the robot collects more measurements, thus making the method unappealing for robots with limited memory and computing power. The maximization of (7) requires the knowledge of the set S of samples acquired by all the robots in different locations and at different time instants. Its computation requires, thus, either a centralized architecture or an all-to-all communication among the robots. The latter, in particular, would result in an unrealistic solution due to the limited channel bandwidth commonly available to the robots.

Reduction of information flow is, therefore, necessary. From a communications perspective, the sole assumption made in the proposed approach is that each robot is allowed to exchange information with its *neighbors*. However, because each robot chooses the next point to visit inside its own Voronoi cell, it only needs to estimate this function inside this cell. Then, each robot maintains a local discretized version of function (7) over the whole environment that is necessarily accurate only inside its partition.

To overcome the increase in complexity and to obtain a local good estimate of function $\xi_S(\boldsymbol{x})$, a number of approximation methods for Gaussian process regression have been proposed (eg., [33], [35]). Recently, an effective approximation method has been proposed in [38].

The key idea is to use the fact, illustrated in Figure 6, that in order to compute the function (7) at a given point $\boldsymbol{x}$, only samples from points that are not spatially too far away from it and have been acquired not too far in the past will contribute. To this effect, a suitable strategy is to cut *less informative* points according to a well defined criterion.

In accordance with the Gaussian formulation adopted, the samples that need to be exchanged are those whose distances from the considered Voronoi cell are below a certain threshold $\alpha_s\tau_s$ and that have been acquired not more than $\alpha_t\tau_t$ seconds in the past with respect to the current instant, with α_s and α_t positive scalars.

The parameters that affect the approximation error are the size of the search area, the resolution of the corresponding grid, the number of robots, the space and time constants, and the communication bit/rate. There are trade-off among the approximation error, the computational capabilities of the robots and the values of the parameters α_s and α_t: the larger the parameters are, the smaller the error. Usually, a tuning procedure is required to properly set the grid resolution and the α_s, α_t parameters in order to meet the quality of results and the computational burden constraints.

Figure 6 illustrates the underlying procedure graphically. In the figure, the current time instant is t_k and it is assumed (for graphical simplicity) that the robot is still. The samples acquired at each time instant t_{k-i} $(i = 1, 2..., m)$ are shown as crosses laying in a $x-y$ plane corresponding to the considered time instant. At the current time t_k and at a given point in space (the black point in the image), the function ξ_S depends on the already acquired samples belonging to a circle centered at that point with a radius depending on the space constant τ_s [38]. With regards to the samples acquired in the past, the effect of the time constant τ_t is the same as the reduction of the radius of the circles until it approaches zero for *very old* samples. As a result of this procedure, all the samples considered lay in a cone with base and height that depend on τ_s and τ_t, respectively.

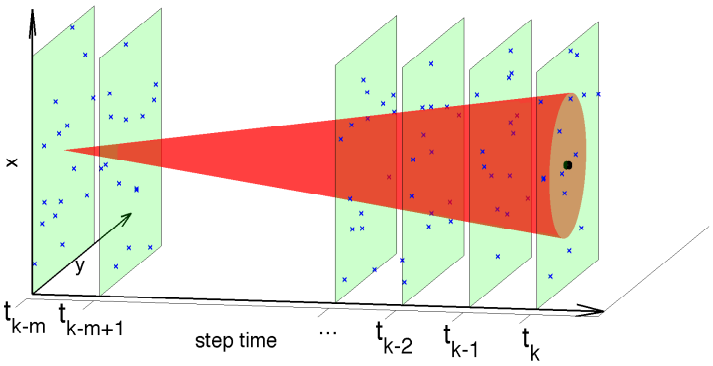

Fig. 6. Graphical representation of the principle underlying the approximation procedure of the function $\xi_S(\boldsymbol{x})$ based on temporal and spatial considerations. Blue crosses represent acquired samples and the black circle represents the location where function $\xi_S(\boldsymbol{x})$ must be computed. The points belonging to the red cone are those that will be preserved.

Two simulation scenarios were run for a sampling mission in order to illustrate the approximation adopted in our method. In both cases, the area explored is similar to the one used in the actual experiments in SectionVII, i.e., a $50\,\text{m} \times 100\,\text{m}$ rectangular map where the initial position of robots are randomly chosen. In the first campaign, several simulations (of about 100 steps) were run with a fixed number of robots ($N_r = 4$) and variable space (τ_s) and time (τ_t) constants. In particular, both τ_s and τ_t varies from 5 to 50 with α_s and α_t set to 4. Let ${}^i\hat{\xi}_{S_i}(\boldsymbol{x}, t)$ be the approximation of function $\xi(\boldsymbol{x})$ made by robot i $(i = 1, 2, \ldots, N_r)$ at location $\boldsymbol{x} \in \mathcal{A}$ and time t. The relative error is given by:

$$error = \max_{i,\boldsymbol{x},t} \frac{\left| {}^i\hat{\xi}_{S_i}(\boldsymbol{x}, t) - \xi_S(\boldsymbol{x}, t) \right|}{\xi_S(\boldsymbol{x}, t)} \tag{10}$$

with t belonging to the simulation interval. As seen in Figure 7 (top), the error does not exceed 4%.

In the second simulation scenario, the time and space constants were fixed to $\tau_t = 50\,\text{s}$, $\tau_s = 10\,\text{m}$ as well as α_s and α_t to 4, while the number of robots was made variable between 2 and 20. The error as in the previous scenario was calculated for each team size and is shown in Figure 7 (bottom).

B. Computational Burden

According to (7), the computation of the index in (8) requires the computation of the inverse of matrix $\boldsymbol{\Sigma}_S$. This could lead to a high computational burden due to the high

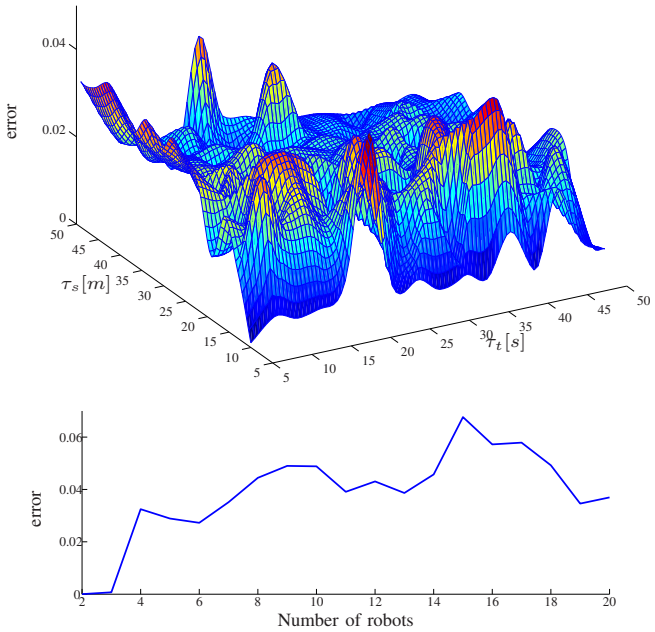

Fig. 7. Top. Surface plot of the maximum relative estimation error of function $\xi_S(\boldsymbol{x})$ as function of the time and space constants τ_t, τ_s with 6 robots. Bottom. Plot of the maximum relative estimation error of function $\xi_S(\boldsymbol{x})$ as function of the number of robots with fixed time and space constants $\tau_t = 50\,\text{s}$, $\tau_s = 10\,\text{m}$.

dimensions that this matrix can reach, especially, in static or slowly varying fields. For this reason, a simple strategy is adopted to iteratively compute $\boldsymbol{\Sigma}_S^{-1}$ as new locations are visited or new samples are received. Consider the case of a generic robot that, at time t_k has its own estimate of the function $\xi_S(\boldsymbol{x})$ based on the set $S = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_k, t_k)\}$ of acquired/received information. In addition, let us suppose that a new sample is acquired/received at time t_{k+1}, $(\boldsymbol{x}_{k+1}, t_{k+1})$ that leads to the set $S' = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_k, t_k), (\boldsymbol{x}_{k+1}, t_{k+1})\}$. With this notation, the value of the function $\xi'_S(\boldsymbol{x})$ at a generic location $\boldsymbol{x}$ at the instant $t \geq t_{k+1}$ is

$$\xi_{S'}(\boldsymbol{x}) \quad = \quad \boldsymbol{\sigma}_{S'}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Sigma}_{S'}^{-1} \boldsymbol{\sigma}_{S'}(\boldsymbol{x}). \tag{11}$$

The terms $\boldsymbol{\sigma}_{S'}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}_{S'}^{-1}$ are computed as

$$\begin{cases} \boldsymbol{\sigma}_{S'}(\boldsymbol{x}) & = & \boldsymbol{T}_{k+1} \begin{bmatrix} \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}) \\ C(\|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|, t - t_{k+1}) \end{bmatrix} \\ \boldsymbol{\Sigma}_{S'} & = & \begin{bmatrix} \boldsymbol{\Sigma}_{S_k} & \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1}) \\ \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1})^{\mathrm{T}} & 1 \end{bmatrix} \end{cases} \tag{12}$$

where $\boldsymbol{T}_{k+1} \in \mathbb{R}^{k+1 \times k+1}$ is a diagonal matrix with its diagonal terms defined by:

$$\boldsymbol{T}_{k+1}(i,i) = \begin{cases} \exp\left(-\frac{(t-t_k)(t+t_k-2t_i)}{2\tau_t^2}\right) & \text{if } i \leq k \\ 1 & \text{if } i = k+1. \end{cases} \tag{13}$$

The inverse of matrix $\boldsymbol{\Sigma}_{S'}$ in (12) can now be written as

$$\boldsymbol{\Sigma}_{S'}^{-1} = \begin{bmatrix} \boldsymbol{X} & \boldsymbol{y} \\ \boldsymbol{y}^{\mathrm{T}} & z \end{bmatrix}.$$

Using Schur complements and the inverse of block partitioned matrices, yields

$$\begin{cases} z = \dfrac{1}{1 - \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1})^{\mathrm{T}} \boldsymbol{\Sigma}_{S_k}^{-1} \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1})} = \dfrac{1}{1 - \xi_S(\boldsymbol{x}_{k+1})} \\ \boldsymbol{y} = -z \boldsymbol{\Sigma}_S^{-1} \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1}) \\ \boldsymbol{X} = \boldsymbol{\Sigma}_{S_k}^{-1} + z \boldsymbol{\Sigma}_{S_k}^{-1} \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1}) \boldsymbol{\sigma}_{S_x}(\boldsymbol{x}_{k+1})^{\mathrm{T}} \boldsymbol{\Sigma}_{S_k}^{-1}. \end{cases} \tag{14}$$

Equations (11)-(14) provide the mathematical equations useful to iteratively evaluate the objective function at a given point.

V. Literature comparison

In the literature, the problem of coverage or patrolling is often solved by means of an integral-based algorithm such as, for example, the one described in [7], known as *deployment*.

In addition to the scalar function to be maximized, for example the $\xi_S(\boldsymbol{x})$ function described in (7), in this case there is also the need to add a term that prevents all the robots from moving toward the same via point. Following [7], this is achieved by using a positive non-decreasing scalar performance function $g \in \mathbb{R}$ yielding a functional to be minimized of the form

$$\sigma(\boldsymbol{x}) = \int_{\mathcal{D}} \min_i g(\|\boldsymbol{x} - \boldsymbol{x}_{r,i}(t)\|) \xi_S(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{15}$$

The remarkable intuition behind [7] is that this integral index may be rewritten according to *local* contributions by resorting to the Voronoi partitions, as follows:

$$\sigma(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{n} \int_{Vor(\boldsymbol{x}_{r,i})} g(\|\boldsymbol{x} - \boldsymbol{x}_{r,i}(t)\|) \xi_S(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{16}$$

where $\boldsymbol{x} \in \mathbb{R}^l$ is a generic point in the set S, and set $Vor(\boldsymbol{x}_{r,i}(t))$ is the Voronoi partition associated with i th robot. One possible choice for function above is $g = \|\boldsymbol{x} - \boldsymbol{x}_{r,i}(t)\|^2$, leading to the *distortion problem* in [7].

It is to see that the associated Jacobian $\dfrac{\partial \sigma}{\partial \boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{x}) \in \mathbb{R}^{1 \times ln}$ is structurally decoupled and exhibits the structure

$$\boldsymbol{J} = \begin{bmatrix} \cdots & \int_{Vor(\boldsymbol{x}_{r,i})} \xi_S(\boldsymbol{x}) \, (\boldsymbol{x} - \boldsymbol{x}_{r,i})^{\mathrm{T}} \, \mathrm{d}\boldsymbol{x} & \cdots \end{bmatrix} \tag{17}$$

where the generic term reported is an $(1 \times l)$ matrix and the dependency of $\boldsymbol{x}_{r,i}$ on time has been omitted. Notice that the partial derivative of (15) is not trivial to compute, the details can be found in [7].

It is interesting to observe that the Jacobian can be rewritten as

$$\boldsymbol{J} = \begin{bmatrix} \cdots & m(\boldsymbol{x}_{r,i}) \left(\boldsymbol{c}(\boldsymbol{x}_{r,i}) - \boldsymbol{x}_{r,i}\right)^{\mathrm{T}} & \cdots \end{bmatrix}, \qquad (18)$$

where

$$m(\boldsymbol{x}_{r,i}) = \int_{Vor(\boldsymbol{x}_{r,i})} \xi_S(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, \quad \boldsymbol{c}(\boldsymbol{x}_{r,i}) = \frac{\int_{Vor(\boldsymbol{x}_{r,i})} \boldsymbol{x}\xi_S(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}{\int_{Vor(\boldsymbol{x}_{r,i})} \xi_S(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}$$

are the scalar mass and the l-dimensional centroid of the i-th Voronoi partition, respectively. A straightforward interpretation of equation (18) is that the stationary points of this function are the centroids of the Voronoi partitions. The problem then becomes how to find the Voronoi partitions of area $\mathcal{A}$ such that the position of each robot coincides with the respective partition's centroids. Such a configuration is called a *centroidal Voronoi configuration* and is, in general, not unique.

A general method to reach such a configuration is to resort to Lloyd's method [12], where, stated in simple terms, the robots move towards the centroids of their partition while updating the overall Voronoi tessellation.

The fact that the objective function in (15) is integral-based and contains the *repulsive* term g makes the robots reach its maximum with configurations that are not necessarily the most *rich* in information. The reason is twofold:

- integral-based approaches suffer in case of symmetric configuration of the function to be estimated;
- the approach can not be used in all the situations where the robots themselves impact on the definition of the function; i.e., whenever the robot are contributing to the informative process.

This can be better explained by resorting to a simple numerical example. Let us consider the case of a squared environment initially unexplored (i.e., $\xi_S(\boldsymbol{x}) = 0$, $\forall \boldsymbol{x}$); without loss of generality, a stationary field is assumed ($\tau_t = +\infty$ in (5)), in addition, the value of τ_s will not affect the following considerations. Three robots, starting from random positions, reach the configuration shown in Figure 8 (left); the path and the final Voronoi tessellation are also shown. The right plot shows that most of the field remains *uncertain*.

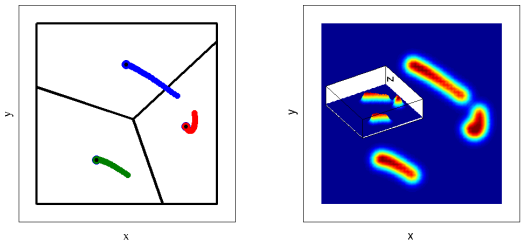

Fig. 8. First Simulation. Left: robot paths and final Voronoi partitions in the case of the centroidal Voronoi approach. Bigger bullets represent the partition centroids according to function ξ_S. Right: the ξ_S function; red color corresponds to larger values and blue color to smaller values.

In order to show that this drawback is not due to the stationarity of the chosen field, an additional numerical case study is shown with $\tau_t = 4$ and a *rich* initial shape for the function $\xi_S(\boldsymbol{x})$. Figure 9 shows the initial robot configuration (top left) and the $\xi_S(\boldsymbol{x})$ function (top right), as well as the steady state corresponding plots at the bottom.

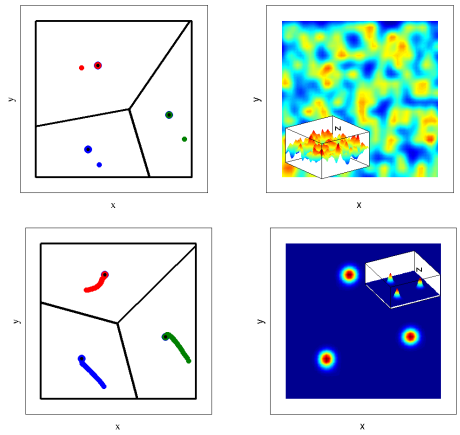

Fig. 9. Second Simulation. Top left: robot initial positions and initial Voronoi partitions with their centroid according to function ξ_S. Top right: the initial configuration of the ξ_S function. Bottom left: robot paths and final Voronoi partitions in the case of the centroidal Voronoi approach. Bottom right: the final configuration of the ξ_S function. Bigger bullets on the left plots represent the partition centroids according to function ξ_S while smaller bullets are the robots' positions. Red color on the right plots corresponds to larger values of function ξ_S and blue color to smaller values.

VI. Simulation Results

In this simulation case study of a patrolling mission, a rectangular $60\,\mathrm{m} \times 100\,\mathrm{m}$ environment is used as shown in Figure 10. The region with vertices $[0,\,0]$ m, $[0,\,100]$ m, $[50,\,100]$ m, $[50,\,0]$ m was divided in two zones: *Zone#2* with vertices $[25,\,0]$ m, $[25,\,50]$ m, $[50,\,50]$ m, $[50,\,0]$ m and *Zone#1* the remaining part of the area; because the former is closer to the entrance of the area (see Figure 10), it can be argued that it is a more vulnerable zone with respect to unexpected events such as attacks by one or more intruders and, thus, it should be more carefully patrolled. This concept is used in Section VI-B, while in Section VI-A the coverage case is briefly discussed and no distinction between the two zones is made. Moreover, both in the simulations and experiments, the parameter θ introduced in Section III-A was heuristically set to 0.5.

A. The constant field case

As a first example, in order to highlight the flexibility of the approach derived, the case of a static field is examined. For patrolling or sampling missions, this case is useful whenever the aim is to scan the field only once. This result can be obtained by simply setting $\tau_t = +\infty\,\mathrm{s}$.

The number of robots was set to $N_r = 6$, with the robots moving at a constant speed of $1\,\mathrm{m/s}$ for $2000\,\mathrm{s}$, and the space constant is $\tau_s = 4.7\,\mathrm{m}$. As the time constant is infinite, we expect each location to be chosen at most once as destination point and that, after a certain amount of time, all the locations will be sampled. This is exactly the situation shown by Figure 11 where, as in done in Section III-A, the

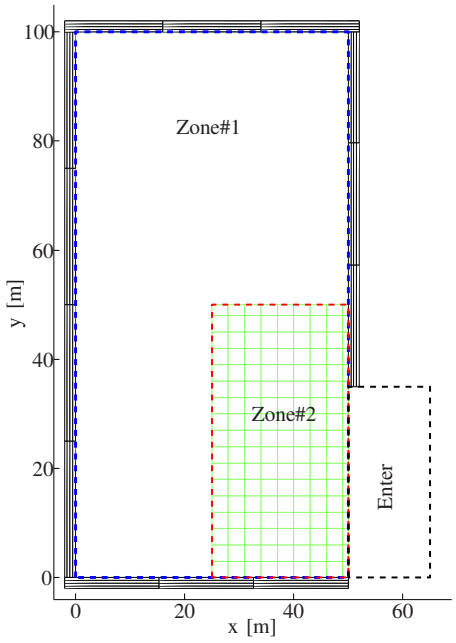

Fig. 10. First simulation case study. The two subregions, *Zone#1* and *Zone#2* exhibit different constraints for the patrolling task.

normalized integral of function $\xi_S(\boldsymbol{x})$ over the environment is shown.

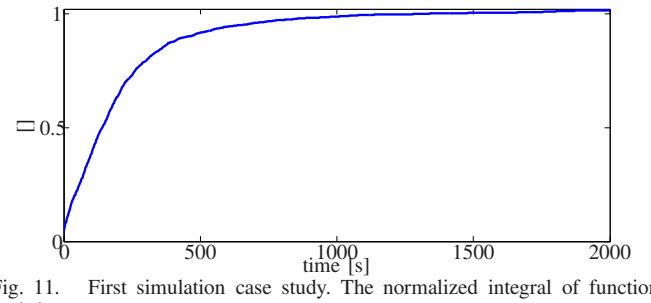

Fig. 11. First simulation case study. The normalized integral of function $\xi_S(\boldsymbol{x})$ over the environment as function of time.

B. The patrolling case

The same environment as in the previous case study is considered. Now, $N_r = 4$ and $\tau_s = 4.7$ m but, in contrast to the previous case, a finite value of the time constant, τ_t, is used. In particular, $\tau_t = 400$ s in *Zone#1*, and several simulations were run with τ_t varying from 50 s to 400 s in *Zone#2* with a step size of 50 s. Each simulation runs for 3000 s.

Figure 12 shows the integral of function $\xi_S(\boldsymbol{x})$ over the environment, normalized by the area of the overall surface. In all cases, after an initial transient phase, it reaches asymptotically an almost constant value of 0.6. The fact that the value 1 is not reached is due to the fact that the function ξ_S is time varying and decreasing in time at locations where the robots are not present. Thus, contrarily to what happens in the static case (i.e., $\tau_t = +\infty$), it is not possible to reach the maximum value of the normalized integral unless for large size teams and/or high values of the space constant τ_s. However, as the next point to visit is chosen in the set S_u (see Section III-A) this will not imply that the whole area will not be completely covered.

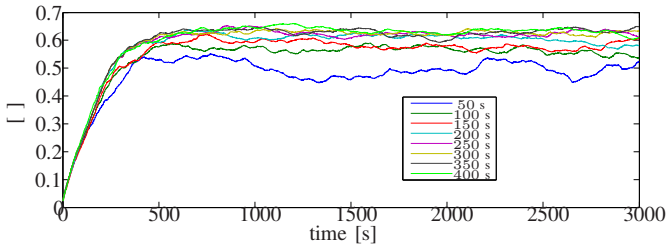

Fig. 12. Second simulation case study. The normalized integral of function $\xi_S(\boldsymbol{x})$ over the environment as function of time for different values of the parameter τ_t.

As can be seen, a uniform high value of τ_t leads to a larger value of the normalized integral of function $\xi_S(\boldsymbol{x})$. Lower values of τ_t in Zone#2 result in lower values of the integral with more noticeable oscillating behaviour. This can be explained by considering that the lower the time constant in Zone#2 is the larger is the rate of decrease of $\xi_S(\boldsymbol{x})$ in this zone.

With regards to Zone#2, Figure 13 shows the total time spent by the robots in it. The time is expressed as a percentage of the overall simulation duration, normalized by the team size N_r.

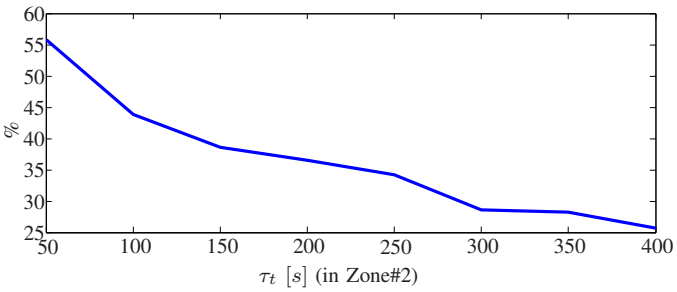

Fig. 13. Second simulation case study. Time spent by the tram in Zone#2 expressed as percentage of the overall simulation duration divided by the team size N_r. The time constant τ_t is constant and equal to 400 s in Zone#1, and varies from 50 s to 400 s in Zone#2.

Notice that, for $\tau_t = 400$ s (i.e., uniform time constant) the team spent 25% of the overall time in Zone#2 (Zone#2 is 1/4 of the overall environment). As τ_t decreases in Zone#2 from 400 s to 50 s, the time spent by the the team in this zone increases until it reaches over 50% of the time. What happens in this case is that one or more robots (not always the same) keep patrolling *Zone#2* with higher frequency.

C. Numerical Comparison

As done in [6], a numerical comparison between different approaches is reported here. Simulations were run for a dynamic field of 100 m x 100 m to be sampled, characterized by $\tau_t = 500$ s, $\tau_s = 5$ m, and three robots. The approach designed herein is compared with other three approaches:

- the lawnmower approach: the environment is divided in equal rectangular slices. Inside each slice, the robot

moves repeatedly up and down as in a lawn mowing maneuver;
- the random approach: the robots move randomly in the environment without any cooperation scheme;
- the deployment approach as described in Section III.

The performance index adopted for comparison is the integral of the function $\xi_S(\boldsymbol{x})$ over the domain, normalized to the maximum obtained value. The results are shown in Figure 14.

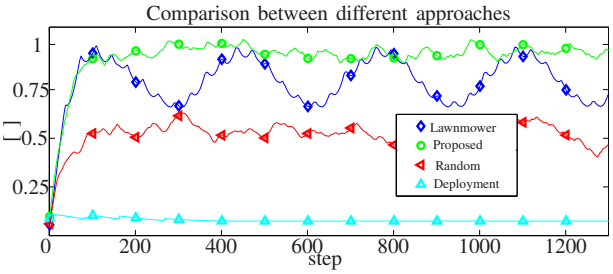

Fig. 14. Numerical comparison. Normalized Performance index for the lawnmowing motion (blue), proposed approach (green), random motion (red) and deployment (cyan).

The deployment approach in Section III causes the robots to reach a static configuration leading to a large portion of the environment characterized by low values of function (7). The random approach gives, obviously, better performance but worse than the lawnmower-like motion where the particular robot paths cause the index to increase, but with oscillations that are not present in the case of the proposed strategy based on the heuristics as in equation (8).

VII. Experimental results

Experiments were performed in July 2011 at the Parque Expo site, Lisbon, PT, using three Medusa autonomous surface robots designed and built by the marine robotics team of IST/ISR (Instituto Superior Técnico/Institute for Systems and Robotics) and shown in Figure 15 (red, black, and yellow robots). A video of the experiments can be found in [1].

Fig. 15. The three Medusa surface robots setup (the red, black and yellow robots).

Each robot, named Medusa, was entirely designed and assembled by the staff of the Dynamical Systems and Ocean Robotics Laboratory (DSOR-Lab) of the ISR/IST of Lisbon. They consist of two identical bodies, an upper body and a lower body with diameters and lengths of $0.15\,\mathrm{m}$ and $1\,\mathrm{m}$, respectively. The distance between the two bodies is $0.45\,\mathrm{m}$. The lower underwater body contains the actuator electronics together with a lithium polymer battery pack that allows, at the nominal speed of $1.0\,\mathrm{m/s}$, an autonomy of $6\,\mathrm{hours}$. The autonomy can be doubled by adding an extra battery pack. The upper part contains the processing unit together with navigation sensors (mainly, IMU and GPS). For control, navigation, and mission control systems implementation, each robot runs a standard Linux operative system on a $2\,\mathrm{GHz}$ Intel Core Duo-$1\,\mathrm{GB}$ RAM platform. The architecture adopted allows the use of MatLab both for test and real-time control purposes. The robots are equipped with a GPS localization system. Inter-robot surface and underwater communications are enabled via a Wi-Fi and acoustic modem networks, respectively. The maximum rated speed of the robots is $2.0\,\mathrm{m/s}$. At the moment no sensor is mounted on the robot to detect intruders in an harbour patrolling scenario or for coverage tasks: the uncertainty map is updated based only on the robots' positions.

A. First case study

Figure 16 shows the map of the site, with the robots moving in a $60\,\mathrm{m}\times70\,\mathrm{m}$ rectangular map. An obstacle consisting of a buoy is placed inside the patrolling region (the green dot in Figure 16). The position of the buoy is fixed and known in advance by the robots. The maximum robots speed was limited to $0.7\,\mathrm{m/s}$. In what follows, we summarize the results of an experiment that run for approximately 1 hour.

The robots exchanged information via WI-FI network. The τ_s and τ_t parameters were set to $3.7\,\mathrm{m}$ and $200\,\mathrm{s}$, respectively; the constant θ in Section III-A was set to 0.5.

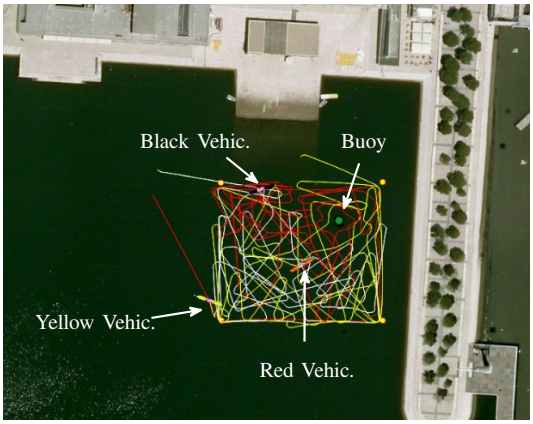

Fig. 16. First experiment case study. The map of the Parque Expo site in Lisbon with the paths described by the robots in a typical experiment. The robots (red, black and yellow) are restricted to move in a $60\,\mathrm{m}\times70\,\mathrm{m}$ rectangular environment.

In Figure 17, the sequence of steps performed by the robots is shown. In each frame, on the left are shown the Voronoi cells with the robots and the current targets (big bullets), while, on the right, the plot of function (7) is shown. The red color is representative of higher values of the function while the blue color of lower ones. Focusing the attention on the black robot, the following steps are shown:

1) the robot moves toward the current target as generated by the algorithm described in Section III,
2) the robot reaches the current target,

3) the robot chooses the next target inside its own Voronoi cell,
4) the robot moves toward the new target.

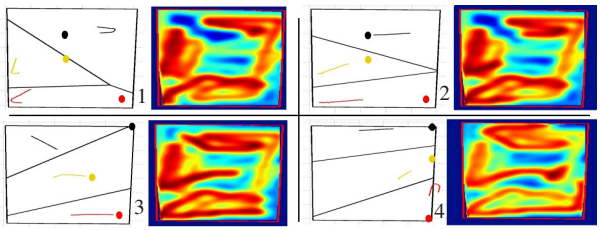

Fig. 17. First experiment case study. Frames of the experiments. In each frame, on the left are shown the Voronoi cells with the robots and the current targets (big bullets) while, on the right, the plot function $\xi_S(\boldsymbol{x})$ in equation (7) is shown (red color is representative of high values of the function, blue color of low values). Frame#1. The black robot is approaching the corresponding target. Frame#2. The black robot has approached the target. Frame#3. After the black robot has reached the target, the next one is generated inside its own cell. Frame#4. The black robot moves toward the new target.

Finally, in Figure 18 the integral of the function in $\xi_S(\boldsymbol{x})$ over the environment normalized by its area is plotted with respect to time. Notice how the function almost reaches a steady value as confirmed by the simulations in Section VI.

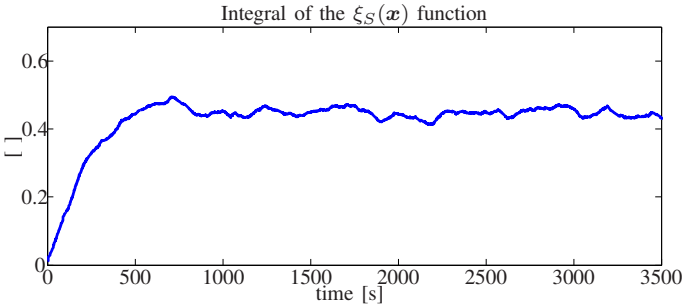

Fig. 18. First experiment case study. The normalized integral of the function in equation (7) over the environment.

B. Second case study

In the case of patrolling, it is worthwhile to enrich the architecture with an additional functionality useful when a menace, such as an intruder, is observed. In this case, in addition to patrolling, the robots must surrounds an eventual intruder that enters the area. The intruder (the robot in the yellow circle in Figure 19) is represented by a small surface robot equipped with GPS and wifi and remotely guided. The strategy can be summarized in the following steps:

- as long as the intruder is out of the sensor visibility range of any of the patrolling teammates, they keep patrolling according to the strategy described in the sections above;
- once any of the patrolling robots has detected the intruder, a message is sent to its neighbors and the corresponding location marked as unsafe. The information propagates throughout the team;
- the robots drive at maximum speed in order to rapidly surround the robot;
- the robots surround the intruder and track its motion in formation;

Fig. 19. Second experiment case study. The intruder enclosed in the yellow circle surrounded by the patrolling robots.

For the sake of clarity, as patrolling robots are not yet equipped with any detection sensor, each robot knows at each time instant the intruder's GPS position. However, any action is undertaken only when the distance between the intruder and any of the patrolling teammates is below a given threshold ($10\,\mathrm{m}$ in the presented experiment). In order to short the detection time, an environment smaller than the one described in the experiment above has been used ($30\,\mathrm{m}\times 50\,\mathrm{m}$).

Figure 20 shows the team configuration at the time instant the yellow robot detects the intruder. As described above, at this stage a message is sent to its neighbors.

Once the intruder has been reached, the patrolling teammates have to position according to a regular polygon keeping a distance of $8\,\mathrm{m}$ from the intruder.

In Figure 21, the distances over time of the robots from the intruder are shown. High error in the time interval $[500,\ 700]\,\mathrm{s}$ are due to instantaneous wrong values provided by the intruder's GPS (that is much cheaper with respect to the ones of the patrolling robots).

Finally, in Figure 22 the overall paths are shown. The patrolling robots (the red, yellow and black dots) keep the formation while tracking the intruder (the cyan dots).

VIII. Conclusions

A distributed approach for multi-robot patrolling was proposed. The strategy adopted exploits a probabilistic framework to define the robot's motion strategy by resorting to Gaussian Process analysis and design tools. The objective is to choose the next point to reach in order to reduce the overall uncertainty about the knowledge of a given field. The Voronoi tessellation is used as a means to distribute the objective among the robots. Based on these techniques, each robot chooses the next point to reach inside its Voronoi cell in order to reduce the uncertainty at unexplored locations. Experiments were run using three autonomous marine robots available at the ISR/IST of Lisbon. Future work will address the problems that arise due to limited communications and anysotropic sensing.

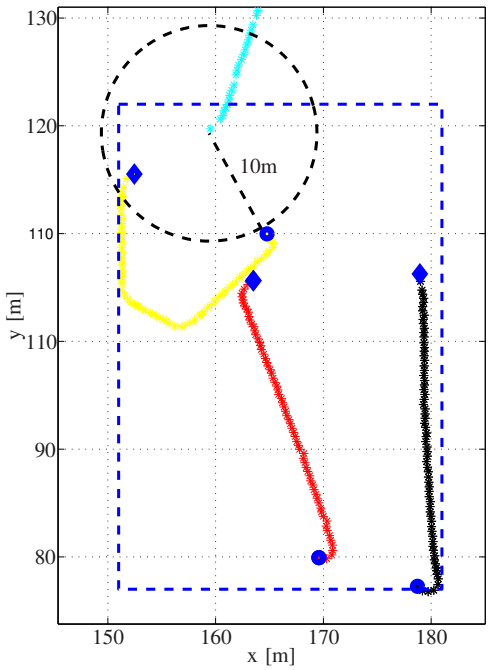

Fig. 20. Second experiment case study. The intruder detection instant. The dashed blue line is the boundary of the area to patrol. Red, yellow, black, dots represent the path of the patrolling robots, while the cyan dots the intruder's path. Blue diamonds are the starting positions while blue circles the final ones.

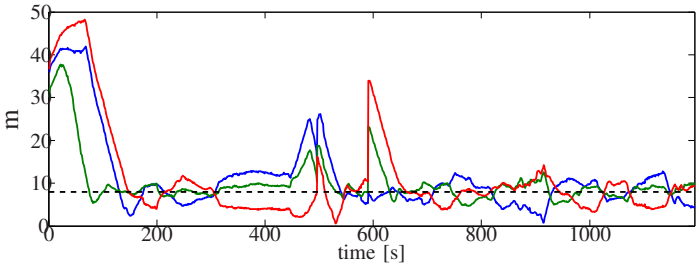

Fig. 21. Second experiment case study. The distance of the patrolling robots from the intruder. The dashed horizontal line represents the desired value.

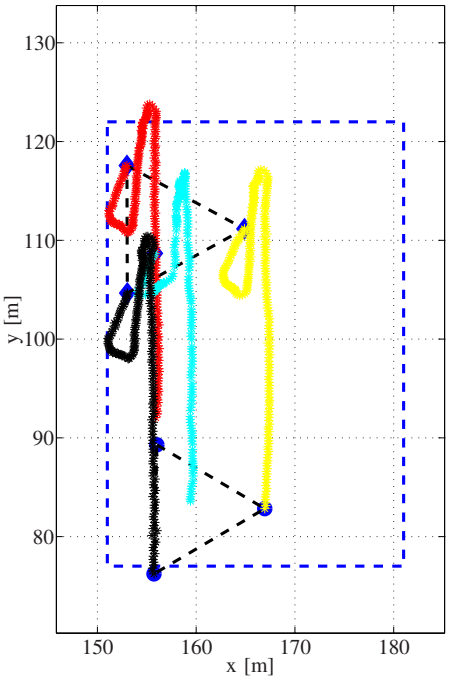

Fig. 22. Second experiment case study: the robots paths. Red, yellow, black, dots represent the paths of the patrolling robots, while the cyan dots the intruder's path. Blue diamonds are the starting positions while blue circles are the final ones.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n. 231378 (STREP project Co^{3}AUVs) and under the Italian Government under grant FIRB n. RBFR08QWUV (NECTAR) and PRIN 2008 n. 2008EPKCHX (MEMONET).

Our special thanks go to the technical staff of the Dynamical Systems and Ocean Robotics Laboratory (DSOR-Lab) of the ISR/IST of Lisbon for their invaluable support in making the MEDUSA robots available for tests.

REFERENCES

[1] *http://automatica.diiie.unisa.it/video/VideoLisbonExpo.avi*.

[2] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proceedings 2008 IEEE International Conference on Robotics and Automation*, pages 2339–2345, Pasadena, CA, May 2008.

[3] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, , and V. Corruble. Recent advances on multi-agent patrolling. *Proceedings of the Brazilian Symposium on Artificial Intelligence*, 2004.

[4] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5):985–994, October 2009.

[5] G. Antonelli, F. Arrichiello, and S. Chiaverini. The Null-Space-based Behavioral control for autonomous robotic systems. *Journal of Intelligent Service Robotics*, 1(1):27–39, Jan. 2008.

[6] G. Antonelli, S. Chiaverini, and A. Marino. A coordination strategy for multi-robot sampling of dynamic fields. In *Proceedings 2012 IEEE International Conference on Robotics and Automation*, pages 1113–1118, St Paul, MN, May 2012.

[7] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009.

[8] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. *Procedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302–308, 2004.

[9] J. Cortes, S. Martinez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.

[10] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. on Rob. and Aut.*, 20(2):243–255, 2004.

[11] E. Creed, J. Kerfoot, C. Mudgal, S. Glenn, O. Schofield, C. Jones, D. Webb, T. Campbell, M. Twardowski, G. Kirkpatrick, et al. Automated control of a fleet of Slocum gliders within an operational coastal observatory. In *Proceedings of MTS/IEEE Conference Oceans*, volume 2, pages 726–730, 2003.

[12] Q. Du, M. Emelianenko, and L. Ju. Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations. *SIAM Journal on Numerical Analysis*, 44:102–119, January 2006.

[13] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[14] E. Fiorelli, N.E. Leonard, P. Bhatta, D.A. Paley, R. Bachmayer, and D.M. Fratantoni. Multi-AUV control and adaptive sampling in Monterey Bay. *IEEE Journal of Oceanic Engineering*, 31(4):935–948, 2006.

[15] A. García-Olaya, F. Py, J. Das, and K. Rajan. An online utility-based approach for sampling dynamic ocean fields. *IEEE Journal of Oceanic Engineering*, (2):185–203, 2012.

[16] C. Gui and P. Mohapatra. Virtual patrol: a new power conservation design for surveillance using sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[17] G. Hollinger and G. Sukhatme. Stochastic motion planning for robotic information gathering. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[18] A. Kolling and S. Carpin. Multi-robot surveillance: an improved algorithm for the graph-clear problem. In *Proceedings 2008 IEEE International Conference on Robotics and Automation*, pages 2360–2365, Pasadena, CA, May 2008.

[19] A. Krause, C. Gestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *IPSN 06*, Nashville, Tennessee, USA, April, 19-21 2006.

[20] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of Gaussian Processes: An exploration-exploitation approach. In *International Conference on Machine Learning*, pages 449–456, 2007.

[21] D. Lee and B. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Parallel Programming*, 9(3):219–242, June, 1980.

[22] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang. Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay. *Journal of Field Robotics*, 27:718–740, 2010.

[23] A. Machado, G. Ramalho, J.D. Zucker, and A. Drogoul. Multi-agent patrolling: an empirical analysis of alternative architectures. *Lecture Notes in Computer Science*, 2581:155–170, 2003.

[24] A. Marino and G. Antonelli. Experimental results of coordinated coverage by autonomous underwater vehicles. In *Proceedings 2013 IEEE International Conference on Robotics and Automation*, pages 4126–4131, Karlsruhe, D, May 2013.

[25] A. Marino, G. Antonelli, A.P. Aguiar, and A. Pascoal. Multi-robot harbor patrolling: a probabilistic approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, PT, October 2012.

[26] A. Marino, L. Parker, G. Antonelli, and F. Caccavale. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In *Proceedings 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[27] A. Okabe, B. Boots, and K. Sugihara. *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 1992.

[28] Fabio Pasqualetti, Antonio Franchi, and Francesco Bullo. On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Transaction on Robotics*, 28:592–606, 06/2012 2012.

[29] C. E. Rasmussen and C. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[30] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *In Porecedeengs of 20th International Joint Conference on Artifical Intelligence*, San Francisco, CA, USA, 2007.

[31] R. Smith, J. Das, H. Heidarsson, A. Pereira, F. Arrichiello, I. Cetinic, L. Darjany, M. Garneau, M. Howard, C. Oberg, M. Ragan, E. Seubert, E. Smith, B. Stauffer, A. Schnetzerand G. Toro-Farmer, D. Caron, B. Jones, and G. Sukhatme. The USC center for integrated networked aquatic platforms (CINAPS): Observing and monitoring the southern California bight. *IEEE Robotics and Automation Magazine, Special Issue on Marine Robotic Systems*, 17(1):20–30, Mar 2010.

[32] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme. Persistent ocean monitoring with underwater gliders: Adapting spatiotemporal sampling resolution. *Journal of Field Robotics*, 28(5):714 – 741, Septemeber 2011.

[33] A. J. Smola and P. Bartlett. Sparse greedy Gaussian Process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

[34] Y. Wang, X. Wang, B. Xie, D. Wang, and D. P. Agrawal. Intrusion detection in homogeneous and heterogeneous wireless sensor networks. *IEEE Trans. Mob. Comput.*, 7(6):698–711, 2008.

[35] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[36] Y. Xu and J. Choi. Mobile sensor networks for learning anisotropic gaussian processes. In *American Control Conference*, St. Louis, MO, USA, June, 10-12, 2009.

[37] Y. Xu and J. Choi. Adaptive sampling for learning Gaussian Processes using mobile sensor networks. *Sensors*, 11(3):3051–3066, 2011.

[38] Y. Xu, J. Choi, and S. Oh. Mobile sensor network navigation using Gaussian Processes with truncated observations. *IEEE Transactions on Robotics*, 27(6):1118–1131, 2011.