Persistent Coverage Control with Variable Coverage Action in Multi-Robot Environment

Carlos Franco, Gonzalo López-Nicolás, Carlos Sagüés, Sergio Llorente

Abstract— **In this paper we tackle the problem of persistent coverage, where a team of agents covers an area whose amount of coverage degrades with time. The task is never accomplished completely due to the coverage decay and the agents have to revisit the domain persistently. In this framework, our contribution is a novel approach which consists not only in developing efficient motion to perform the coverage, but also in covering the domain with a variable coverage action. The agents can adapt its coverage power to the coverage error of the actuator domain, being able to reduce the energy consumption and the coverage error. We propose a new controller for the coverage power and we demonstrate by means of simulations that it is more efficient and flexible than developing the coverage with constant power.**

I. Introduction

The study of the persistent area coverage is of interest in a lot of applications such as: cleaning, painting, monitoring, lawn mowing, etc. In these problems, one or several agents have to develop an action while moving over an area. The energy consumption needed to cover the area is an important parameter and researchers have proposed different strategies and algorithms to address it.

The literature is nowadays focused on reducing the energy consumption of the coverage by reducing the path length. In general, the optimization problem is NP-hard and this is made by computing a suboptimal path which covers the whole domain [1], [2], [3]. In [1] authors compute paths for lawn mowing and milling problems by means of different algorithms. They give an approximation factor for each algorithm taking into account the particularities of each problem. In [4] a new strategy to increase the efficiency of back and forth motions while covering seabed is proposed. It takes into account the depth of the seabed, that modifies the range of the sensor, to adapt the interspacing of each lap avoiding overlaps. In [3], a strategy to divide area in cells and develop optimal path over each cell to achieve a complete and efficient coverage of the domain is presented. These papers deal with efficient path planning area coverage, but they do not take into account multi-robot environments. [5] and [2] address efficiency of multi-robot approach. In [2], authors propose to divide the area into cells and construct spanning trees as an efficient solution for the area coverage. The union of the spanning tree of each robot covers the whole domain. In [5] the efficiency of the coverage path is increased by adapting the speed of the agents to the frequency of variation of the features of the domain. There are also authors that propose to build "informative paths", that lies on computing a path taking into account how the variables of interest are spread instead of computing a path covering all the domain [6], [7].

On the other hand, there exist other authors which compute the path online [8], [9], [10]. In [8] a Kalman filter is used to minimize the estimation error as well as the motion energy of the agents. In [10] a decentralized gradient technique is used to compute a path within a horizon for a team of robots and then, the path is modified using the information of a decentralized data fusion between the neighbors to increase the efficiency of the coverage. In [9] a gradient technique is combined with a hierarchical grid decomposition of the domain to develop the coverage orderly and reduce the path length. These approaches are more flexible to changing environments since they compute the direction to move constantly and seem more appropriate to domains with decay. However, none of them addresses the persistent coverage.

In this paper we focus on developing persistent coverage by computing the path online. The contribution of this paper is the proposal of a coverage action with variable power, which allows to save energy and reduce error. As far as we know, this is the first time that this problem is addressed. There are applications that require a particular coverage level, and higher coverage leads to a waste of energy, as for example cleaning, or to bad results as painting. Here, we propose a controller with an action proportional to the weighted error of the coverage domain of each agent. For the motion, we rely on the design proposed in [11] based on a gradient for developing local coverage, and blob analysis to develop a global strategy, and we adapt it to a new coverage dynamics and error function. We assume that the coverage information of the domain is available for all the agents, but the coverage and motion actions are computed by each agent and then, the proposal is scalable.

The paper is organized as follows: Section II presents the problem formulation, introducing the coverage control and the motion control of the agents. Section III presents an algorithm to find objectives for the global strategy of coverage. Section IV shows simulation results of the control laws proposed and compares our proposal with constant coverage actuator. Finally, Section V provides the conclusions.

This work was supported by projects IPT-2011-1158-920000 of subprogram INNPACTO and by DPI2012-32100 from Ministerio de Economía y Competitividad, by DGA-FSE (grupo T04), and by grant B139/2010 by DGA.

C. Franco, G. López-Nicolás and C. Sagüés are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, 50018, Zaragoza, España (email: cfranco@unizar.es; gonlopez@unizar.es; csagues@unizar.es)

S. Llorente is with the Research and Development Department, Induction Technology, Product Division Cookers, BSH Home Appliances Group, 50016, Zaragoza, España (email: sergio.llorente@bshg.com).

II. Problem formulation

In this section we introduce the problem formulation abusing of the notation by including the dependencies of the variables just when they are defined. The aim is to reach a desired coverage level $\Lambda^*(x) > 0$ for all the points $x \in D_x$ over a bounded domain $D_x \subset \mathbb{R}^2$.

We assume that mobile agents are holonomic, $\dot{p}_i = u$, where $p_i(t)$ is the position of the i-th agent in a domain $D_p \subset \mathbb{R}^2$ and u is the motion. Then $p_i(t) = [p_{i_1}(t), p_{i_2}(t)]^T$ of each agent $\mathcal{A}_i$ of the team $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ of N agents. Note that D_p can be different from D_x.

A. Coverage control

Let us define $r(t)$ as the Euclidean distance between a point x and the position of the agent p_i, and $\alpha_i(r)$ as the coverage action that an agent develops over points at distance r. In this work, we restrict to actuators with limited range so that their covering function is positive $\alpha_i \geq 0$ in the interior of the coverage domains $\Omega_i(t)$, $i \in \{1, \ldots, N\}$ and zero elsewhere. The coverage action of the team of agents is defined as $\alpha(t, x) = \sum_{i \in \{1,\ldots,N\}} \alpha_i$. Furthermore we define $\Lambda(t, x) \in [0, \infty)$ as the coverage developed by a team of agents over a point x at time t.

The coverage information changes in each point x with the following differential equation:

$$\frac{\partial \Lambda}{\partial t} = A \cdot \Lambda + B \cdot \alpha \tag{1}$$

where $A, B \in \mathbb{R}$, $A < 0$ is the state gain and B the input gain. Note that $A < 0$ is required for stability. We assign $\Lambda(0, x) = 0, \quad \forall x \in D_x$, which means that at the beginning all points are assumed as uncovered.

At this point let us introduce $\Phi(x) \in (0, 1]$, $\forall x \in D_x$, as the priority to cover each point x. Φ is a map that weights the interest of the points in the domain to give more priority to particular zones of special interest. The quadratic coverage error of the domain is defined as:

$$e_{D_x}(t) = \int_{D_x} \Phi \cdot (\Lambda^* - \Lambda)^2 dx \tag{2}$$

Assuming we have one agent and it is always inside the coverage domain, we compute the evolution of the error:

$$\frac{\partial e_{D_x}}{\partial t} = -2 \int_{D_x} \Phi \cdot (\Lambda^* - \Lambda) \cdot (A \cdot \Lambda + B \cdot \alpha_i) dx \tag{3}$$

And reordering terms:

$$\begin{aligned}\frac{\partial e_{D_x}}{\partial t} = -2 \cdot \Bigg[&\int_{D_x} A \cdot \Phi \cdot \Lambda \cdot (\Lambda^* - \Lambda) dx + \\ &\int_{D_x} B \cdot \alpha_i \cdot \Phi \cdot (\Lambda^* - \Lambda) dx \Bigg].\end{aligned} \tag{4}$$

Where the first term drives Λ to 0, and in general increases the error, and the second term can be tuned by means of α_i to make it always positive or null achieving that the error decreases, or at least increases as less as possible.

Let us define $\sigma_i(r)$ as the normalized action of the agent such that represents the intensity of the actuators of the agents depending on the distance to its position and:

$$\int_{\Omega_i} \sigma_i dx = 1. \tag{5}$$

For example a lawn mower may be represented by a constant σ_i whereas a spray painter may be represented as a decreasing function with the distance r. Then we can write the action of an agent as a gain multiplied by the normalized action: $\alpha_i = K \cdot \sigma_i$. Here, we introduce the proposed controller, which is the main novelty of the paper:

$$K = C \cdot \left[\int_{\Omega_i} B \cdot \Phi \cdot \sigma_i \cdot (\Lambda^* - \Lambda) dx \right]^{2 \cdot q - 1}, \tag{6}$$

where $C > 0$ is the gain of the controller, $q \in \mathbb{N}$, and the powered term can be interpreted as the weighted error of the agent's domain. Thus, it is the formulation of a proportional controller taking into account the distribution of the robots' action in order to increase performance. Note that $K \in \mathbb{R}$ and $\sigma_i = 0$, $\forall x \notin \Omega_i$. In this way we obtain:

$$\begin{aligned}\frac{\partial e_{D_x}}{\partial t} &= -2 \cdot \Bigg[\int_{D_x} A \cdot \Phi \cdot \Lambda \cdot (\Lambda^* - \Lambda) dx + \\ &\qquad K \cdot \left(\int_{\Omega_i} B \cdot \sigma_i \cdot \Phi \cdot (\Lambda^* - \Lambda) dx \right) \Bigg] \\ &= -2 \cdot \Bigg[\int_{D_x} A \cdot \Phi \cdot \Lambda \cdot (\Lambda^* - \Lambda) dx + \\ &\qquad C \cdot \left(\int_{\Omega_i} B \cdot \sigma_i \cdot \Phi \cdot (\Lambda^* - \Lambda) dx \right)^{2 \cdot q} \Bigg].\end{aligned} \tag{7}$$

Therefore, the second term is never negative and the agents reduce the error according to the shape of the coverage action thanks to the proposed controller.

B. Motion control

In this section we obtain the motion control law for coverage purposes using ideas from [9], [11]. The objective of our motion control law is to keep decreasing the error. Thus, we want to minimize the variation of the error computed in (7) of each agent with respect to its own position. Then, we compute the gradient of the variation of the error with respect to the position of agent i:

$$\begin{aligned}u_i^{loc}(t) &= \frac{\partial}{\partial p_i} \left(\frac{\partial e_{D_x}}{\partial t} \right) = \\ &- 4 \cdot q \cdot C \cdot \left(\int_{\Omega_i} B \cdot \sigma_i \cdot \Phi \cdot (\Lambda^* - \Lambda) dx \right)^{2 \cdot q - 1} \cdot \\ &\int_{\Omega_i} B \cdot \Phi \cdot \frac{\partial \sigma_i}{\partial r} \cdot \frac{p_i - x}{\|p_i - x\|} \cdot (\Lambda^* - \Lambda) dx.\end{aligned} \tag{8}$$

From this gradient we can extract the direction of the motion, $\hat{u}_i^{loc}(t)$, to get the maximum benefit covering the neighborhood of the agent:

$$\hat{u}_i^{loc} = \frac{u_i^{loc}}{\|u_i^{loc}\|}, \tag{9}$$

if $\|u_i^{loc}\| \neq 0$ and null vector otherwise. However, gradient techniques are known to get stuck in local minima and, when the error in the coverage domain is low, the benefit of covering the neighborhood of the agent is small. Because the information decays in other parts of the domain, it may happen that the error over the domain e_{D_x} increases. Thus, we propose to combine the gradient strategy with a global law $u_i^{glo}(t)$ that depends on the coverage of the whole domain to bring the agents to places with higher error and improve the performance of the coverage.

To reach other parts in the domain with higher error we propose a strategy to select global objectives $p_i^{obj}(t) \in D_p$, which is developed in Section III, and a control law to reach the global objective. Let us define $d_i^{obj}(t)$ as the Euclidean distance from the position of the agent to the position of its objective, and $k_i^G(d_i^{obj}) \in [0,1]$ as the global gain. For example:

$$k_i^G = tanh\left(\frac{2 \cdot d_i^{obj}}{R}\right), \tag{10}$$

where R is the range of the coverage action. This function is near 1 until the distance to the objective is in the range of the coverage actuator and then it decreases to 0 when agent is in the objective. It is a good choice since it is maximum until the objective is being covered, but it is not the only one. Finally, the global control law:

$$u_i^{glo} = k_i^G \cdot \frac{p_i - p_i^{obj}}{\|p_i - p_i^{obj}\|}. \tag{11}$$

To combine both global and local control laws let us introduce the local error $\varsigma_{\Omega_i}(t) \in (-\infty, 1]$ as:

$$\varsigma_{\Omega_i} = \int_{\Omega_i} \Phi \cdot \sigma_i \cdot \frac{(\Lambda^* - \Lambda)}{\Lambda^*} dx \tag{12}$$

Compared with (2), this is a more qualitative error indicating that agent's neighborhood is satisfactorily covered when it is negative or 0 and viceversa. Let us also introduce a local weight $W_i^{loc}(t)$ and a global weight $W_i^{glo}(t)$:

$$W_i^{loc} = (\varsigma_{\Omega_i}^+)^\beta \tag{13}$$

$$W_i^{glo} = 1 - (\varsigma_{\Omega_i}^+)^\beta \tag{14}$$

where $\beta \in \mathbb{R}^+$ is a constant parameter to be adjusted depending on the desired behavior of the algorithm and the parameters of the problem, and $\varsigma_{\Omega_i}^+ = \max(0, \varsigma_{\Omega_i})$. Further details will be provided in the simulation section. The objective direction of the coverage $u_i^{cov}(t)$ is obtained with:

$$u_i^{cov} = W_i^{loc} \cdot \hat{u}_i^{loc} + W_i^{glo} \cdot u_i^{glo}. \tag{15}$$

Finally, we compute the motion control law $u_i(t)$ as follows:

$$u_i = k_i \cdot (1 - \varsigma_{\Omega_i}^+) \cdot u_i^{cov}, \tag{16}$$

where $k_i \in \mathbb{R}$ is the motion gain and represents the maximum velocity of each robot. The term $(1 - \varsigma_{\Omega_i}^+)$ slows down the robot to develop coverage in the neighborhood when the local error is high, and speed up the agent when the error is close to 0 to leave the area. Note also that the weights make the agents to obey local control law when the local error is high, moving slowly in the direction of the gradient of the error, and performing coverage carefully. And makes the agent to obey global control law when the local error is low, heading towards new uncovered areas rapidly to increase the performance of the coverage. This combination of local and global strategies was firstly presented in [9] where a proof of total coverage was given for environments without decay. In this case, reaching the total coverage of the area is not possible since agents are not able to cover all the domain simultaneously, but as it will be shown in the simulations a steady error is reached when the coverage decay is equal to the agents' coverage capability.

III. Selection of global objectives

In this section we propose a strategy to find areas with high error and to assign each agent its global objective p_i^{obj}. It is based on blob detection of the uncovered information previously introduced in [11]. We use this algorithm to find islands of uncovered information in the maps Λ, and then we compute their sizes and their centroids. With this information we propose a criterium to select the objective based on the uncertainty and the proximity of the blobs.

Let us define $\Psi(t) = \{\psi_1, \psi_2, ..., \psi_j, ..., \psi_M\}$ as the collection of $M(t)$ global objectives, $\psi_j(t) \in D_x$. Let us also define $\pi_j(t)$ as the collection of points of the domain composing each blob and whose global objective is $\psi_j(t)$, $\Pi(t) = \bigcup_{j=1}^{M} \pi_j$ as the collection of points of the domain assigned to objectives ψ_j. Finally, let us introduce $\mathfrak{T}$ as the tolerance or percentage of admissible error, and $\pi^\emptyset(t) = \{x \in D_x | (\Lambda^* - \Lambda) \leq \mathfrak{T} \cdot \Lambda^*\}$ as the collection of points that are covered. The method to select the global objectives is described in Algorithm 1.

The algorithm starts by checking if some of the M global objectives ψ_j have been covered. Those covered are erased from the list of global objectives Ψ and the points π_j assigned to the objective are released. It also checks if there are objectives that are closer than a distance R, which is the coverage radius of the agents. Those objectives are also erased and their points released to try to merge them and to get a bigger blob in the blob searching procedure. Afterwards, the domain to obtain the new blobs of the scene is computed by subtracting the covered points $\pi^\emptyset$ and the assigned points Π from the domain to cover D_x. With $blob(D_{blob})$ the candidate centroids ψ^f and the points π^f of the F regions of the space to be covered are obtained. Then, the candidate centroids ψ^f are checked to see if they belong to the points π^f of the blob. If the centroids ψ^f are inside the blob, they and the points of the blobs π^f are saved, whereas the blob domain is reduced with the points of the new found blob.

Once the checking is complete, it is possible that some centroids fall outside of their respective blobs. This is not a desired situation because due to the coverage range of the agent, it is possible that once the agent has arrived to the global objective, it cannot reach uncovered points causing a blockage. In this case, the image is eroded. This results

Algorithm 1 Blob-based algorithm for the selection of global objectives

```
Require: $D_x$, $\pi^{\emptyset}$, $\Psi$, $\pi_j$;
Ensure: $\Psi$, $\pi_j$;
 1: for $j = 1,..,M$ do
 2:     if $\psi_j \in \pi^{\emptyset}$ then
 3:         $\Pi = \Pi - \pi_j$; $\Psi = \Psi - \psi_j$;
 4:     end if
 5: end for
 6: for $j = 1,..,M$ do
 7:     $d_j^{min} = min(\|\psi_j - \psi_r\|), (r \neq j)$
 8: end for
 9: for $j = 1,..,M$ do
10:     if $d_j^{min} < R$ then
11:         $\Pi = \Pi - \pi_j$; $\Psi = \Psi - \psi_j$;
12:     end if
13: end for
14: $D_{blob} = D_x - \Pi - \pi^{\emptyset}$;
15: while $D_{blob} \neq \emptyset$ do
16:     $(\psi^1, ..., \psi^F, \pi^1, .., \pi^F) = blob(D_{blob})$;
17:     for f=1,..,F do
18:         if $\psi^f \cup \pi^f \neq \emptyset$ then
19:             $\Psi_{M+1} = \psi^f$; $\pi_{M+1} = \pi^f$;
20:             $D_{blob} = D_{blob} - \pi^f$;
21:         end if
22:     end for
23:     $D_{blob} = erode(D_{blob})$;
24: end while
25: Assign eroded points to the nearest blob;
```

in the elimination of the points in the domain that are in contact with covered or assigned points in such a way that the irregularities of the blob that cause the centroid to fall outside the blob are eliminated. Afterwards, blob analysis is repeated while the blob domain is not null. Finally, the eroded points are assigned to the nearest blob. It is possible but unlikely that, due to symmetries, no global objectives are found. In that case the nearest uncovered point is the only global objective until the symmetry breaks.

The choice of the objective p_i^{obj} for each agent i is done with a criterion that weights distance to the centroids, and coverage error inside the blob assigned to each centroid. We follow the same approach of [11] where the areas with the higher error are assigned to the closest agent iteratively.

This is an heuristic approach which follows intuition, driving agents to big and close uncovered areas, but others could be followed with the requirement that from p_i^{obj} agent j reach uncovered points. For example the one presented in [12] is valid and simply drives agents to the closest uncovered points. That solution is simpler, Although it is also less effective to develop the coverage.

IV. Simulation results

In this section we present simulation results of the control strategy proposed. The coverage domain D_x is a 100×100 units square area and $D_p = \mathbb{R}^2$. The state and the input gains are constant $A = -1/200$, $B = 1/400$, and the coverage priority is also a constant $\Phi = 1\ \forall x \in D_x$. The team is composed by 6 agents whose motion control parameters are: $\beta = 1/5$, $k_i = 3$ and whose coverage range is $R = 10$. In our problem K is saturated between 0 and 3000, and the tolerance $\mathfrak{T}$ is 20%. We choose $q = 1$ and $C = 12 \cdot 10^4$. We carry out several experiments changing Λ^* and σ_i to test the effectiveness of the proposal. Let us introduce the average absolute error:

$$\hat{e}_{D_x}(t) = \frac{\sqrt{e_{D_x}}}{\int_{D_x} \Phi dx} \tag{17}$$

and the integrated average error:

$$\bar{e}_{D_x}(t) = \frac{\int_0^t \hat{e}_{D_x} dt}{t} \tag{18}$$

We present a first simulation of the behavior of the system with $\Lambda^* = 50$ and:

$$\sigma_i(r) = \begin{cases} \frac{3}{\pi \cdot R^6}(r^2 - R^2)^2 & r \leq R \\ 0 & r > R \end{cases} \tag{19}$$

This is a quadratic function normalized with the relationship (5) used in other coverage approaches [13], [14].

The evolution of the coverage map is shown in Fig. 1. Whereas the evolution of the coverage power, average absolute coverage error, and a boxplot of the histogram of the coverage evolution is shown in Fig. 2. Fig. 3 shows the evolution of the motion control action. In 250 seconds the error reaches a steady value with an absolute average error around 11. The boxplot chart and the coverage maps show that most of the points are around the coverage objective but there are also points over and below the objective. The shape of the coverage action and the dynamical behavior of the coverage of the domain prevents that all the points reach and keep the objective with zero error simultaneously. The coverage power and the motion action seems to be noisy but is an accordion effect produced by showing 1000 time units. In the second simulation, we present in Fig. 4 the results of an experiment with the same parameters except $\Lambda^* = 100$. In this case the error reaches a similar percentage with an absolute steady value around 17. It takes around 300 units of time and the power consumption doubles the previous simulation.

In the third simulation we present a comparison between the constant coverage power and our variable coverage power with $\Lambda^* = 50$. We develop 100 simulations starting at random positions with the same variables than the first simulation, and also 100 with each one of the following values of the coverage power: $K = \{100, 300, 500, 700, 900\}$. In Fig. 5 we show the results. The evolution of the error with time shows that the variable coverage is the fastest in reaching the steady state and with a final value of 8. In this case, the only constant power that reaches a lower value is $K = 500$, which is a very similar value to the average variable power, in this case, the average power over time of our variable coverage is 499. Over $K = 500$, the error reaches a minimum and then it grows since the domain

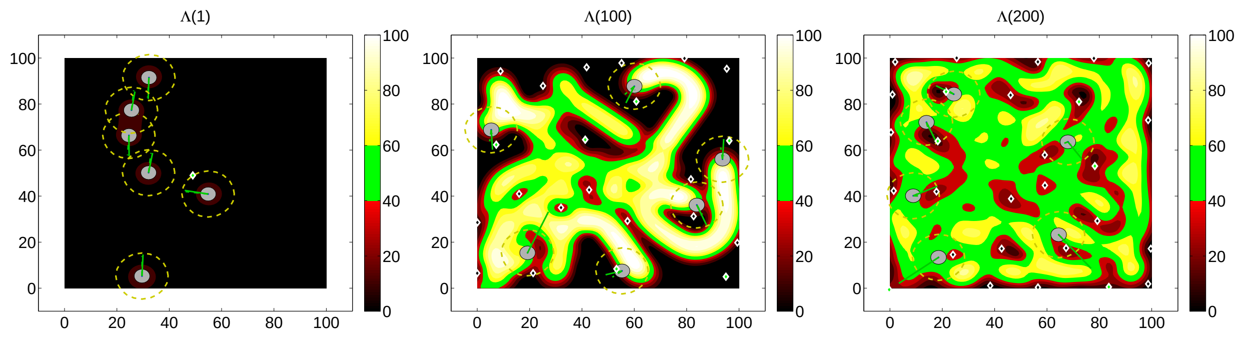

Fig. 1. Results of simulation 1. Evolution of the coverage map. Small circles represent the position of the agents and the coverage domain is represented by a dashed line. Green continuous straight line represents the total action. The small rhombi represent the global objectives. From 200 units of time and till the end, the average coverage level is stationary as Fig. 2 shows and the map looks similar.

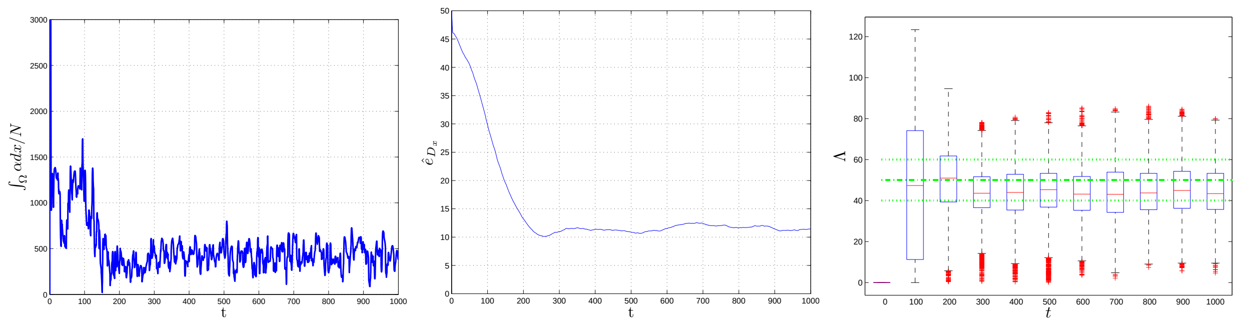

Fig. 2. Results of simulation 1. From left to right: evolution of the average action of the agents, evolution of the average absolute error $\hat{e}_{D_x}$, boxplot of the distribution of the coverage at several different times. Coverage objective $\Lambda^* = 50$, the steady error is reached around t=250 with a value around 11.

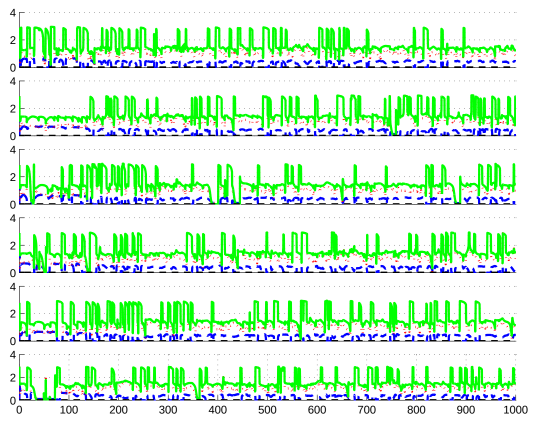

Fig. 3. Results of simulation 1. Motion action of the agents. x-axis represents time. Continuous line represents the total motion action. Dotted line the global component and dashed line the local component.

becomes over-covered. Integrating the error over time, the variable coverage has the lowest value. Finally, the average path length (PL) shows that low power results in shorter path lengths since the term $(1 - \varsigma_{\Omega_i})$ slows down agents when the domain is uncovered and viceversa, when the domain is covered or over-covered the path length grows and reaches around twice the variable coverage path length. The fourth simulation (Fig. 6) is carried out with the same parameters of the third one but $\Lambda^* = 100$. The conclusions are similar to the ones before. The variable coverage reaches a steady state faster and reaches a low error level. The only constant controller that reaches lower steady error is the one with $K = 1000$, very similar to the average coverage power of the variable controller that is 980. The integrated error is also lower than any of the constant power, and path lengths of lower coverage level are smaller, but it grows as the power grows. In these two simulations, it is shown how the same variable coverage controller is able to adapt the coverage power level to different coverage objectives.

V. Conclusion

In this paper we tackled the problem of persistent area coverage with a variable coverage power. We have developed a formulation and proposed a new control algorithm based on that formulation. Finally, we evaluate the performance of our control algorithm and the results are compared with the performance of the coverage with a constant power coverage actuator. The variable coverage is able to adapt its action to different references achieving lower error and less power consumption.

References

[1] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry: Theory and Applications*, vol. 17, no. 1-2, pp. 25–50, 2000.

[2] N. Agmon, N. Hazon, and G. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *IEEE International Conference on Robotics and Automation*, pp. 1698 –1703, May 2006.

[3] J. W. Kang, S. J. Kim, M. J. Chung, H. Myung, J. H. Park, and S. W. Bang, "Path planning for complete and efficient coverage operation of mobile robots," in *International Conference on Mechatronics and Automation*, pp. 2126 –2131, August 2007.

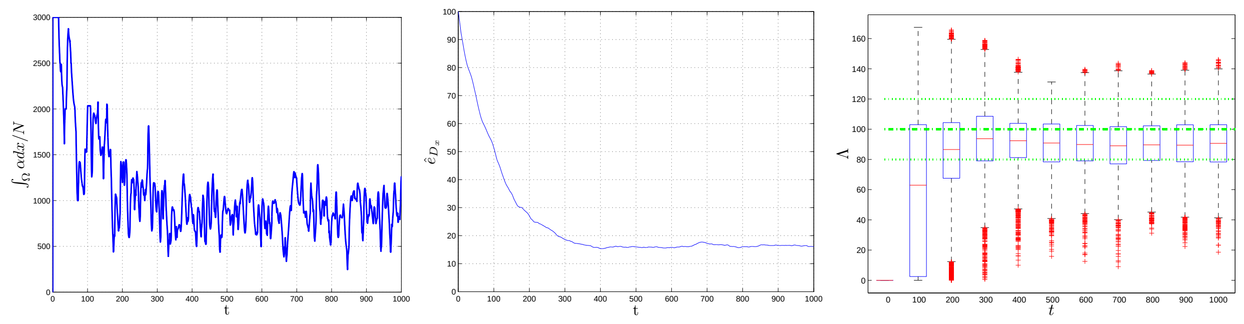

Fig. 4. Results of simulation 2. From left to right: evolution of the average action of the agents, evolution of the average absolute error $\hat{e}_{D_x}$, boxplot of the distribution of the coverage at several different times. Coverage objective $\Lambda^* = 100$, the steady error is reached around t=300 with a value around 17.

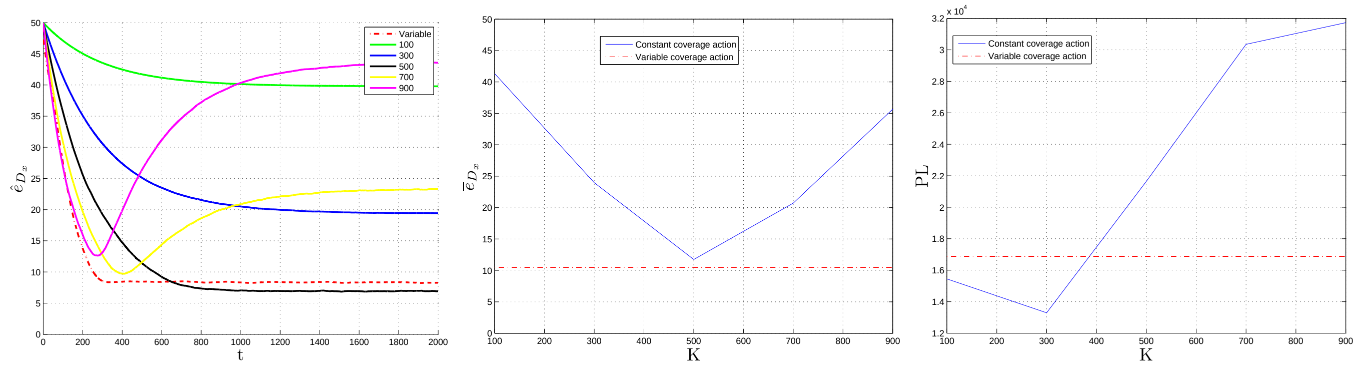

Fig. 5. Aggregated results of simulation 3. From left to right, average error over time, integrated average error $\overline{e}_{D_x}$ and average path length (PL). The coverage objective is $\Lambda^* = 50$ with an average power over time of the variable coverage power control law of 499.

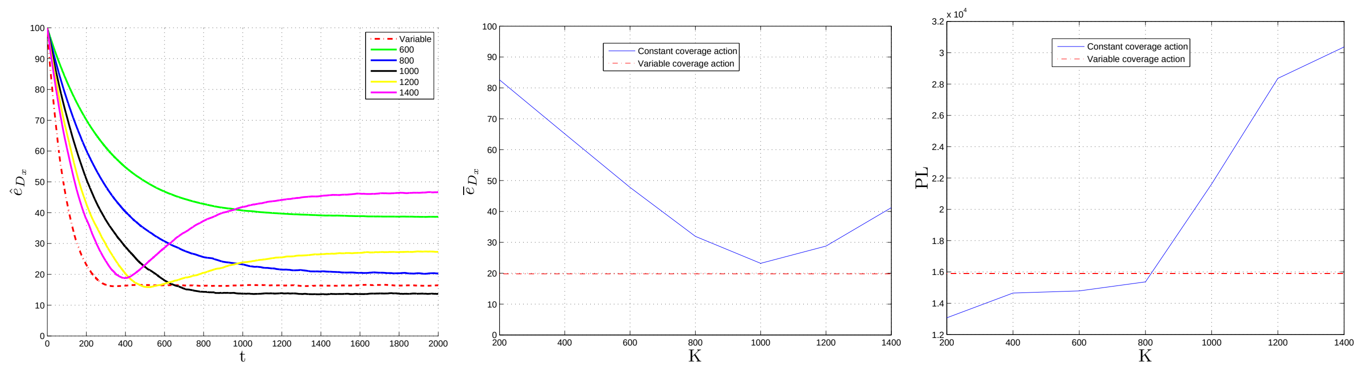

Fig. 6. Aggregated results of simulation 4. From left to right, average error over time, integrated average error $\overline{e}_{D_x}$ and average path length (PL). The coverage objective is $\Lambda^* = 100$ with an average power over time of the variable coverage power control law of 980.

[4] E. Galceran and M. Carreras, "Efficient seabed coverage path planning for ASVs and AUVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 88–93, Oct. 2012.

[5] S. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, pp. 410–426, Apr. 2012.

[6] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligent Research*, vol. 34, pp. 707–755, Apr. 2009.

[7] D. Soltero, M. Schwager, and D. Rus, "Generating informative paths for persistent sensing in unknown environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2172–2179, Oct. 2012.

[8] I. Hussein, "Kalman filtering with optimal sensor motion planning," in *American Control Conference, 2008*, pp. 3548–3553, June 2008.

[9] C. Franco, D. Paesa, G. Lopez-Nicolás, C. Sagüés, and S. Llorente, "Hierarchical strategy for dynamic coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5341–5346, Oct. 2012.

[10] S. K. Gan and S. Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 751–756, May 2011.

[11] C. Franco, G. López-Nicolás, D. Stipanović, and C. Sagüés, "Anisotropic vision-based coverage control for mobile robots," in *IROS Workshop on Visual Control of Mobile Robots (ViCoMoR 2012)*, vol. 1, pp. 31–36, Oct. 2012.

[12] I. I. Hussein and D. M. Stipanović, "Effective coverage control for mobile sensor networks with guaranteed collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[13] D. M. Stipanović, C. Valicka, C. Tomlin, and T. R. Bewley, "Safe and reliable coverage control," *Numerical algebra control and optimization*, vol. 3, no. 1, pp. 31–48, 2013.

[14] C. Song, G. Feng, Y. Fan, and Y. Wang, "Decentralized adaptive awareness coverage control for multi-agent networks," *Automatica*, vol. 47, no. 12, pp. 2749 – 2756, 2011.