2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Cooperative coverage for surveillance of 3D structures

Antonio Adaldo, Sina Sharif Mansouri, Christoforos Kanellakis, Dimos V. Dimarogonas, Karl H. Johansson and George Nikolakopoulos

Abstract— **In this article, we propose a planning algorithm for coverage of complex structures with a network of robotic sensing agents, with multi-robot surveillance missions as our main motivating application. The sensors are deployed to monitor the external surface of a 3D structure. The algorithm controls the motion of each sensor so that a measure of the collective coverage attained by the network is nondecreasing, while the sensors converge to an equilibrium configuration. A modified version of the algorithm is also provided to introduce collision avoidance properties. The effectiveness of the algorithm is demonstrated in a simulation and validated experimentally by executing the planned paths on an aerial robot.**

I. Introduction

Coverage problems for mobile sensor networks have attracted a notable volume of research in the past few decades, because they constitute a flexible model for numerous applications, such as deployment, inspection and surveillance. Coverage problems can be divided in two large categories. Static coverage problems [1]–[3] are about finding a good placement of the sensors, while dynamic coverage problems [4], [5] are about deploying the sensors to survey an environment continuously, until it has been searched sufficiently well.

In traditional research works on static coverage problems [1], the sensors have circular sensing patterns, meaning that their sensing power is maximal at their own position, and decays with the distance from the sensor. This model is only appropriate to describe omnidirectional sensors, such as temperature sensors or circular laser scans. This type of problem is usually addressed with Voronoi tessellations and the Lloyd algorithm [6], with each sensor being assigned a subregion corresponding to its Voronoi cell.

In more recent works [5], [7], the sensors have anisotropic sensing patters, which allows to model a larger variety of sensing devices, such as monocular cameras or cone-shaped laser scans. However, these models are still deficient to describe certain coverage problems, such as the inspection of a surface or the surveillance of a 3D structure, because they do not capture the morphology of the structure.

Moreover, in the related literature, there have been many works that have addressed the Coverage Path Planning (CPP) problem in 2D spaces and fewer approaches that have addressed coverage of 3D spaces [8]. In [9] a complete survey was presented on CPP methods in 2D and 3D. Towards the 3D CPP, Atkar et al. [10] presented an offline 3D CPP method for spray-painting of automotive parts. Their method used a CAD model and the resulting CPP should satisfy certain requirements for paint decomposition. In [11], the authors presented a CPP with real time re-planning for inspection of 3D underwater structures, where the planning assumed an a priori knowledge of a bathymetric map and they adapted their methodology for the case of an autonomous underwater vehicle. In [12], the authors introduced a new algorithm for producing paths that cover complex 3D environments. In this case, the algorithm was based on off-line sampling with the application of autonomous ship hull inspection, while the presented algorithm was able to generate paths for structures with unprecedented complexity.

In the area of aerial inspection, [13] presented a time-optimal UAV trajectory planning for 3D urban structure coverage. In this approach, initially the structures to be covered (buildings) were simplified into hemispheres and cylinders and in a later stage the trajectories were planned to cover these simple surfaces. In [14], the authors studied the problem of 3D CPP via viewpoint re-sampling and tour optimization for aerial robots. More specifically, they presented a method that supports the integration of multiple sensors with different fields of view and considered the motion constraints on aerial robots. Moreover, in the area of multi-robot coverage for aerial robotics in [15], a coverage algorithm with multiple UAVs for remote sensing in agriculture has been proposed, where the target area was partitioned into k non-overlapping sub-tasks and in order to avoid collision, different altitudes have been assigned to each UAV and security zones were defined, where the vehicles are not allowed to enter.

In this paper, we consider a problem of static coverage of a 3D structure with a network of mobile sensors. The target application is the surveillance of a building or infrastructure with a network of aerial robots endowed with cameras. Our contribution can be summarized as follows. On one hand, the sensors have anisotropic sensing patterns that depend not only on the alignment between the observed point and the line of sight of the sensor (as in [5] and [16]), but also on the alignment between the normal to the surface and

A. Adaldo, D. V. Dimarogonas and K. H. Johansson are with the Department of Automatic Control, School of Electrical Engineering, KTH Royal Institute of Technology, Osquladas väg 10, 10044, Stockholm. Emails: {adaldo,dimos,kallej}@kth.se

S. Sharif Mansouri, C. Kanellakis and G. Nikolakopoulos are with the Robotics Group at the Control Engineering Division of the Department of Computer, Electrical and Space Engineering, Luleå University of Technology, Luleå SE-97187, Sweden. Emails: {sinsha,chrkan,geonik}@ltu.se

This work has received funding from the European Unions Horizon 2020 Research and Innovation Programme under the Grant Agreement No.644128, AEROWORKS.

the line of sight of the sensors. To deal with this type of sensing pattern, we leverage on our previous work [16] to define a generalized notion of Voronoi tessellation, and we introduce a coverage function that measures how well the structure is surveilled by the sensors. We also abstract the structure to surveil into a finite set of *landmarks*, so that the sensors are not required to perform complex computations for partitioning a continuous environment. We also propose an extension of the algorithm to avoid collisions among the agents as well as with other objects nearby. Finally, another contribution of this paper is the experimental evaluation of the proposed scheme. The advantages of our approach with respect to existing algorithms for cooperative surveillance include the following: environment abstraction, path planning and obstacle avoidance in are collected into an unified framework; by using a generalized concept of Voronoi tessellation, we obtain formal convergence results analogous to those obtained for classical coverage problems of planar environments; reconfiguration in the event of an agent joining or leaving the network in mid mission is straightforward. The indoor trials demonstrate the applicability of this method towards surveillance tasks using autonomous real-life aerial platforms.

II. Preliminaries

The null vector in $\mathbb{R}^n$ is denoted by 0_n. A vector in $\mathbb{R}^n$ is also intended as the corresponding column vector in $\mathbb{R}^{n\times 1}$. The cross product between two vectors $u, v \in \mathbb{R}^n$ is denoted by $u\times v$. The skew operator is denoted by $S(\cdot)$ (i.e., $S(u)v = u \times v$). The identity matrix in $\mathbb{R}^{n\times n}$ is denoted by I_n. The transpose of a matrix $M \in \mathbb{R}^{n\times m}$ is denoted by $M^\intercal$. The Euclidean norm of a vector $u \in \mathbb{R}^n$ is denoted by $\|u\|$. The set $\mathcal{S}^2 = \{u \in \mathbb{R}^3 : \ \|u\| = 1\}$ is called the *unit sphere*. A vector $u \in \mathcal{S}^2$ represents a direction in $\mathbb{R}^3$. The kinematics of a vector $u \in \mathcal{S}^2$ can be described as

$$\dot{u} = S(\omega)u, \tag{1}$$

where ω is called the *angular velocity* of the vector. Since $\dot{u} = \omega \times u \perp u$, a vector $u \in \mathcal{S}^2$ which evolves according to (1) remains forever in $\mathcal{S}^2$.

III. Problem Statement

A. Setup

A *landmark* represents a point or a small area of interest within the surface of an object to inspect. A landmark $\ell = (p_\ell, u_\ell)$ is defined by a position $p_\ell \in \mathbb{R}^3$ and an unity-norm vector u_ℓ which represents the outward normal to the body evaluated at the position of the landmark. A set of landmarks offers an abstract and approximate representation of the external surface of a body. The higher is the number of landmarks, the higher is the approximation accuracy. A landmark-set representation of the surface of a 3D structure can be obtained, for example, from a point cloud or from a 3D model, by using opportune sampling algorithms (see for example [17], [18]), as we explain in Section VI.

A *mobile sensor* is a formal abstraction for a generic mobile and directional sensing device, (such as an aerial robot endowed with a camera, as we describe in our motivating example). A mobile sensor $\sigma = (p_\sigma, u_\sigma, f_\sigma)$ is defined by a position $p_\sigma \in \mathbb{R}^3$, a direction $u_\sigma \in \mathcal{S}^2$, and an upper-bounded function $f_\sigma : \mathbb{R}^3 \times \mathcal{S}^2 \times \mathbb{R}^3 \times \mathcal{S}^2 \to \mathbb{R}$. The function f_σ defines the sensing pattern of the mobile sensor σ; namely, $f_\sigma(p_\sigma, u_\sigma, p_\ell, u_\ell)$ is a measure of the visibility of a landmark ℓ attained by the mobile sensor σ. This function is also called the *footprint* of the mobile sensor σ.

Definition 1: The visibility $\mathrm{vis}(\sigma, \ell)$ of a landmark ℓ attained by a sensor σ is defined as

$$\mathrm{vis}(\sigma, \ell) = f_\sigma(p_\sigma, u_\sigma, p_\ell, u_\ell). \tag{2}$$

Definition 2: The coverage $\mathrm{cov}(\sigma, \mathcal{L})$ of a finite set $\mathcal{L}$ of landmarks attained by a mobile sensor σ is defined as

$$\mathrm{cov}(\sigma, \mathcal{L}) = \sum_{\ell\in\mathcal{L}} \mathrm{vis}(\sigma, \ell). \tag{3}$$

Definition 3: Let $\mathcal{L} = \{\ell_1, \dots, \ell_M\}$ be a finite ordered set of landmarks, and $\Sigma = \{\sigma_1, \dots, \sigma_N\}$ be a finite ordered set of mobile sensors. Let $\mathcal{P} : \{1, \dots, M\} \to \{1, \dots, N\}$ be a map that assigns each landmark to one of the sensors, in the sense that each landmark ℓ_j is assigned to the sensor $\sigma_{\mathcal{P}(j)}$. Finally, let $\mathcal{L}_i = \{\ell_j \in \mathcal{L} : \mathcal{P}(j) = i\}$ be the set of the landmarks assigned to sensor σ_i. We define the *coverage score function* associated to the set $\mathcal{L}$ as

$$\mathrm{Cov}_{\mathcal{L}}(\Sigma, \mathcal{P}) = \sum_{i=1}^{N} \mathrm{cov}(\sigma_i, \mathcal{L}_i). \tag{4}$$

Definition 4: Let $\mathcal{L}$, Σ, $\mathcal{P}$, and $\mathcal{L}_i$ with $i \in \{1, \dots, N\}$ be defined as in Definition 3. The tuple $(\mathcal{L}, \Sigma, \mathcal{P})$ is said to be a *Voronoi tessellation* if the following two conditions are satisfied:

- for each $\ell_j \in \mathcal{L}$, $\mathrm{vis}(\sigma_{\mathcal{P}(j)}, \ell_j) \geq \mathrm{vis}(\sigma_i, \ell_j)$ for all $\sigma_i \in \Sigma$;
- for each $\sigma_i \in \Sigma$, it holds that $\partial\,\mathrm{cov}(\sigma_i, \mathcal{L}_i)/\partial p_i = 0_3$ and $S(u_i)\partial\,\mathrm{cov}(\sigma_i, \mathcal{L}_i)/\partial u_i = 0_3$.

Voronoi tessellations as by Definition 4 constitute a generalization of the classical Voronoi tessellations considered, for example, in [6]. In fact, Definition 4 reduces to classical Voronoi tessellations if, for each $\sigma \in \Sigma$, we set $f_\sigma(p_\sigma, u_\sigma, p_\ell, u_\ell) = \|p_\sigma - p_\ell\|^2$. By Definition 4, if $(\mathcal{L}, \Sigma, \mathcal{P})$ is a Voronoi tessellation, then $(\Sigma, \mathcal{P})$ constitutes a local optimum of the coverage score function $\mathrm{Cov}_{\mathcal{L}}(\cdot, \cdot)$.

B. Control objective

Given a set $\mathcal{L}$ of landmarks and a network Σ of mobile sensors, our objective is to control the motion of the sensors and the assignment $\mathcal{P}$ of the landmarks so as to drive the tuple $(\mathcal{L}, \Sigma, \mathcal{P})$ to a Voronoi tessellation, while increasing the value of the coverage score function.

IV. Control Algorithm

In this section, we describe a control algorithm that achieves the objective described in Section III-B. For this algorithm, we leverage on the results presented in [16], and we tailor the algorithm to our motivating application of structure surveillance. We consider an ordered set $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$ of $M \in \mathbb{N}$ landmarks, and an ordered set $\Sigma = \{\sigma_1, \ldots, \sigma_N\}$ of $N \in \mathbb{N}$ mobile sensors. For each mobile sensor σ_i, we denote $\sigma_i(t) = (p_i(t), u_i(t), f_i)$, but we also write σ_i (without time dependency) to speak of the mobile sensor as a physical device. Similarly, we denote $\Sigma(t) = (\sigma_1(t), \ldots, \sigma_N(t))$, but we also write Σ to speak of the sensor network as the ensemble of the physical sensors. The kinematics of each sensor σ_i is defined by

$$\dot{p}_i(t) = v_i(t), \tag{5}$$

$$\dot{u}_i(t) = S(\omega_i(t)) u_i(t), \tag{6}$$

where $v_i(t)$ and $\omega_i(t)$ are the control inputs, and constitute, respectively, the linear velocity and the angular velocity of the sensor.

Each sensor σ_i is assigned a subset $\mathcal{L}_i(t)$ of the landmarks. Note that, even if the set $\mathcal{L}$ is fixed, the set $\mathcal{L}_i(t)$ of the landmarks assigned to a particular sensor changes over time. The landmark assignment is equivalently defined by the time-varying map $\mathcal{P}_t : \{1, \ldots, M\} \to \{1, \ldots, N\}$ such that $\mathcal{P}_t(j) = i \iff \ell_j \in \mathcal{L}_i(t)$.

The proposed control law for each mobile sensor is simply a gradient climb of the sensor coverage:

$$v_i(t) = k_v \frac{\partial \operatorname{cov}(\sigma_i(t), \mathcal{L}_i(t))}{\partial p_i}, \tag{7}$$

$$\omega_i(t) = -k_\omega S(u_i(t)) \frac{\partial \operatorname{cov}(\sigma_i(t), \mathcal{L}_i(t))}{\partial u_i}, \tag{8}$$

where k_v and k_ω are positive gains, and $\partial \operatorname{cov}(\sigma_i(t), \mathcal{L}_i(t))/\partial p_i$ denotes the gradient (taken as a column vector) of $\operatorname{cov}(\sigma_i(t), \mathcal{L}_i(t))$ with respect to p_i (similarly for $\partial \operatorname{cov}(\sigma_i(t), \mathcal{L}_i(t))/\partial u_i$). Note that each mobile sensor can compute (7) and (8) based only on the knowledge of $p_i(t)$, $u_i(t)$ and $\mathcal{L}_i(t)$, and does not need to communicate with the other sensors.

To complete the definition of the control algorithm, we only need to specify how the landmark assignment changes over time (i.e., how the sets $\mathcal{L}_i(t)$ are updated). For this part of the algorithm, we let the sensors send some data to each other on discrete time instants. Namely, for each sensor σ_i we define a time sequence $\{t_{i,k}\}_k$, with $k \in \mathbb{N}$, such that, at each time $t_{i,k}$, σ_i opens a communication with another agent. For the purposes of the algorithm, the way that these sequences are generated is irrelevant. For example, they can be pre-assigned, periodic, event-triggered, or triggered by an user. The only requirement is given by the following assumption.

Assumption 1: There exists a finite upper bound on the intervals $t_{i,k_{j'}} - t_{i,k_j}$ between two consecutive communications with the same sensor σ_j.

Naturally, a lower bound on these intervals must also exist for implementation purposes.

A communication instance initiated by σ_i with another sensor σ_j consists in σ_i passing some of its landmarks to σ_j. A landmark $\ell \in \mathcal{L}_i(t_{i,k})$ is passed if and only if σ_j has a better vision of ℓ than σ_i has (i.e., if $\operatorname{vis}(\sigma_j(t_{i,k}), \ell) > \operatorname{vis}(\sigma_i(t_{i,k}), \ell)$). The actions executed during a communication instance are formalized in Algorithm 1.

Algorithm 1 Actions executed by a sensor σ_i at communication time $t_{i,k}$.

```
open communication with $\sigma_j$
$\Delta := \emptyset$
for $\ell \in \mathcal{L}_i(t_{i,k})$ do
    if $\operatorname{vis}(\sigma_j(t_{i,k}), \ell) > \operatorname{vis}(\sigma_i(t_{i,k}), \ell)$ then
        $\Delta := \Delta \cup \{\ell\}$
    end if
end for
$\mathcal{L}_j(t_{i,k}^+) := \mathcal{L}_j(t_{i,k}) \cup \Delta$
$\mathcal{L}_i(t_{i,k}^+) := \mathcal{L}_i(t_{i,k}) \setminus \Delta$
```

Our control algorithm is now obtained simply as Algorithm 2.

Algorithm 2 Control algorithm for each sensor $\sigma_i \in \Sigma$

```
if $t \notin \{t_{i,k}\}$ then move according to (7) and (8)
else execute Algorithm 1
end if
```

A few remarks on the control algorithm are due.

Remark 1: Sensors communication is only pairwise and intermittent, which makes the algorithm suitable for embedded processors with wireless communication channels. The sensors need to be within each other's communication radius, at least as often as to satisfy Assumption 1. This requirement is mild if we consider that the sensors are surveying the same physical object.

Remark 2: The algorithm allows the sensors to have heterogeneous sensing patterns. However, in this case, the sensors need to know each other's footprint.

The properties of the algorithm are formalized as the following Theorem 1.

Theorem 1: Consider a set $\mathcal{L}$ of landmarks, a network Σ of mobile sensors, and an initial assignment $\mathcal{P}_0 : \mathcal{L} \to \Sigma$ of the landmarks. Under Algorithm 2 with Assumption 1, the tuple $(\mathcal{L}, \Sigma(t), \mathcal{P}_t)$ converges to a Voronoi tessellation, while the coverage score function $\operatorname{Cov}_{\mathcal{L}}(\Sigma(t), \mathcal{P}_t)$ is monotonically nondecreasing.

A proof of Theorem 1 can be obtained by modeling the sensor network as a hybrid system, and then using an invariance principle for hybrid systems (Theorem 8.2 in [19]). The main argument of the theorem is that the coverage score function is upper-bounded and strictly increasing along the system dynamics. A complete proof is omitted here because of space restrictions, but the interested reader is referred to our previous work [16] for the proof of a similar result.

V. Collision Avoidance

Collision avoidance is easily incorporated in the control algorithm by modifying Definition 1 to take into account the possible proximity of the sensor to other bodies, such as the other sensors in the network. Namely, consider the following collision avoidance function:

$$\phi(p,b) = \begin{cases} 0, & \|p-b\| > \rho, \\ 1/\rho^2 - 1/\|p-b\|^2, & \|p-b\| \le \rho, \end{cases} \tag{9}$$

where $\rho > 0$ is a safety radius. Note that $\phi(p,b) \le 0$, $\phi(p,b) < 0$ if $\|p-b\| > \rho$, and that $\phi(p,b) \to -\infty$ when $p-b \to 0$. The results in this section can be generalized to other collision avoidance functions with these characteristics, but for simplicity we are going to refer to (9).

The definition of the visibility of a landmark is modified as follows.

Definition 5: The *visibility* of a landmark ℓ attained by a mobile sensor σ while avoiding collisions with bodies $b \in \mathcal{B}$ is defined as

$$\operatorname{vis}(\sigma,\ell,\mathcal{B}) = f_\sigma(p_\sigma,u_\sigma,p_\ell,u_\ell) + \sum_{b\in\mathcal{B}} \phi(p_\sigma,b). \tag{10}$$

Note that incorporating a collision avoidance term in the visibility function allows also to model effects such as excessive proximity to another body hindering the performances of the sensor. For example, the body may obstruct the field of view of the sensor or influence the motion of the sensor in an undersired way.

Definitions 2, 3, and 4 are easily modified in the same way. First, we define

$$\operatorname{cov}(\sigma,\mathcal{L},\mathcal{B}) = \sum_{\ell\in\mathcal{L}} \operatorname{vis}(\sigma,\ell,\mathcal{B}). \tag{11}$$

Then, we let $\mathcal{B}_i(t) = \{p_j(t) : \sigma_j \in \Sigma \setminus \{\sigma_i\}\} \cup \tilde{\mathcal{B}}$, where $\tilde{\mathcal{B}}$ is a set of fixed points. This choice is motivated by the fact that the sensors should not collide with each other, but neither with other static bodies in the vicinity. For example, one can have $\tilde{\mathcal{B}} = \mathcal{L}$, which captures the requirement to avoid excessive proximity with the structure under surveillance. Consequently to (11), we define

$$\operatorname{Cov}_{\mathcal{L}}(\Sigma,\mathcal{P}) = \sum_{i=1}^{N} \operatorname{cov}(\sigma_i,\mathcal{L}_i,\mathcal{B}_i). \tag{12}$$

The tuple $(\Sigma,\mathcal{L},\mathcal{P})$ is called a Voronoi tessellation if

- for each $\ell \in \mathcal{L}$, $\operatorname{vis}(\sigma_{\mathcal{P}(j)},\ell,\mathcal{B}_{\mathcal{P}(j)}) \ge \operatorname{vis}(\sigma_i,\ell,\mathcal{B}_i)$ for all $\sigma_i \in \Sigma$;
- for each $\sigma_i \in \Sigma$, there exists a neighborhood $\mathcal{N} \subset \mathbb{R}^3 \times \mathcal{S}^2$ of $(p_{\sigma_i},u_{\sigma_i})$ such that $\operatorname{cov}(\sigma_i,\mathcal{L}_i,\mathcal{B}_i) \ge \operatorname{cov}((p,u,f_{\sigma_i}),\mathcal{L}_i,\mathcal{B}_i)$ for all $(p,u) \in \mathcal{N}$.

The control law (7)-(8) for the motion of each sensor is modified by substituting $\operatorname{cov}(\sigma_i(t),\mathcal{L}_i(t))$ for $\operatorname{cov}(\sigma_i(t),\mathcal{L}_i(t),\mathcal{B}_i(t))$.

When we use the modified definition of Voronoi tessellation and the modified version of the control law introduced in this section, Theorem 1 is still valid, while each sensor σ_i avoids collisions with the bodies in $\mathcal{B}_i$. In fact, when two agents are sufficiently close, the contribution of the collision avoidance term $\phi(\cdot,\cdot)$ to the control input prevails upon any other contribution, and the distance between the two agents cannot be reduced further.

VI. Landmark Extraction

The 3D surface of the object to inspect is represented in a point cloud form containing the landmark positions (i.e., p_ℓ for each $\ell \in \mathcal{L}$). This representation approximates the surface based on the density of the contained points. To fully define a landmark ℓ, we need to associate an unity norm vector u_ℓ to each of these points p_ℓ, where u_ℓ corresponds to the normal to the surface in p_ℓ. Depending on the origin of the point cloud, the distribution of the points can be non-uniform for different parts of the structure, leading to inaccurate normal vector estimation. Therefore, the point cloud is down-sampled up to the desired Level of Detail (LoD) to maintain the original shape of the structure, using the *voxelized grid approach* [20]. A voxel grid filter down-samples the point cloud computing a substitute point for all points inside the voxel with their centroid, thus providing a uniform spatial distribution for the points.

For the estimation of the unity-norm vectors u_ℓ, the *Kd-tree algorithm* [21] is used for finding the nearest neighbor. Considering the query point p_l and k neighboring points the aim is to find an orthogonal basis that describes this set of points, using Principal Component Analysis (PCA). For each neighborhood of points a plane is defined from it's centroid $\overline{p}$ and it's normal u_ℓ. Thus, the plane fitting can be formulated as the Least-Squares (LS) problem with cost-function the orthogonal distance defined as $(p_\ell - \overline{p}) \cdot u_\ell$ between the plane and query point p_l. This formulation leads to the Covariance matrix $\mathcal{C}$ (Equation 13) created from the nearest neighbors of the query point. For each point p_ℓ, the covariance matrix $\mathcal{C}$ is formed as follows:

$$\mathcal{C} = \frac{1}{k}\sum_{\ell=1}^{k} (p_\ell - \overline{p}) \cdot (p_\ell - \overline{p})^\intercal. \tag{13}$$

Considering the covariance matrix eigenvalue decomposition, the estimate of the unit normal of the surface will be parallel to the eigenvector that corresponds to the smallest eigenvalue Namely, given that

$$\mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j}, \tag{14}$$

with $j \in \{1,2,3\}$, and where λ_j are eigenvalues of the covariance matrix, and $\vec{\mathrm{v}_j}$ are the eigenvectors that form an orthogonal frame, the unit norm is derived from the eigenvector $\vec{\mathrm{v}_0}$ that corresponds to the smallest eigenvalue, so we set $u_\ell = \vec{\mathrm{v}_0}$. In this paper, the Point Cloud Library (PCL) [17] is used to down-sample the point cloud and calculate the normals to the surface of the object.

VII. Numerical Simulation

In this section, we present a numerical simulation of the proposed algorithm. The simulation is built upon the middleware ROS [22], and the controller of each sensor is implemented as a single ROS node. In particular, for each simulated sensor σ_i, there is a controller node that executes Algorithm 2 and computes $v_i(t)$ and $\omega_i(t)$, and a simulator node that integrates the kinematics (6). This setup presents at least two advantages: first, the simulation reproduces

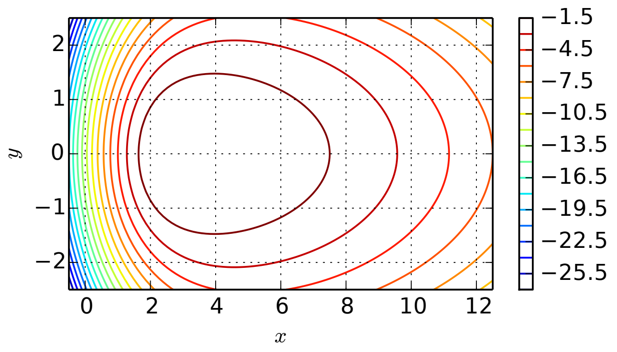

Fig. 1: Contour plot of footprint (15) with $p_\sigma = 0_3$, $u_\sigma = [1, 0, 0]^\intercal$, $p_\ell = [x, y, 0]^\intercal$, $u_\ell = [1, 0, 0]^\intercal$, $D = 2.5$, $\alpha = 0.515$, $\beta = 0.485$ and $\gamma = 1.0$.

faithfully the distributed nature of the control algorithm; second, the same ROS nodes used in the simulation can be used to execute the control algorithm on physical robots. The simulation features $N = 4$ simulated robotic sensors and a landmark set extracted from a point cloud of a cylinder with the procedure described in Section VI. The sensors' footprint is designed to model the sensing patter of a monocular camera:

$$\begin{aligned} &f(p_\sigma, u_\sigma, p_\ell, u_\ell) = \\ &-\|p_\sigma + Du_\sigma - p_\ell\|^2 \cdot \left(\alpha + \beta \frac{u_\sigma^\intercal(p_\sigma + Du_\sigma - p_\ell)}{\|p_\sigma + Du_\sigma - p_\ell\|}\right) \\ &-\gamma\|p_\sigma + Du_\ell - p_\ell\|^2 \cdot \left(\alpha + \beta \frac{u_\ell^\intercal(p_\sigma + Du_\ell - p_\ell)}{\|p_\sigma + Du_\ell - p_\ell\|}\right), \end{aligned} \tag{15}$$

with a continuous extension for $p_\sigma + Du_\sigma - p_\ell = 0_3$ and for $p_\sigma + Du_\ell - p_\ell = 0_3$. Here $D > 0$ is an optimal distance for the visibility of a landmark while $\alpha, \beta, \gamma > 0$ regulate the shape of the footprint. Note that by design choice this footprint is nonpositive, which makes all the coverage values nonpositive. A coverage value closer to zero means that a better coverage is attained. Figure 1 shows a contour plot of footprint (15). The coverage score function is computed including a collision avoidance function (i.e., according to (12)), with $\mathcal{B}_i(t) = \{p_j(t) : \ \sigma_j \in \Sigma \setminus \{\sigma_i\}\}$.

A communication instance is triggered by a sensor when the norm of its velocity goes below a certain threshold, with a minimum interval of 1.0 seconds between two consecutive instances initiated by the same sensor. The initial positions of the sensors are taken as $[-1.5, -1.5, 0]^\intercal$, $[-1.5, 1.5, 0]^\intercal$, $[-1.5, -1.5, 1]^\intercal$ and $[-1.5, 1.5, 1]^\intercal$. The simulation is run for 100 seconds. The results of the simulation are illustrated in Figures 2 and 3. Figure 2 comprises six snapshots of the configuration assumed by the sensor network during the simulation. In Figure 2, each sensor σ_i is represented as a colored arrow, with the tail of the arrow being the position p_{σ_i} of the sensor, and the direction of the arrow being u_{σ_i}. Each landmark ℓ is represented as a colored point p_ℓ (the unit vector u_ℓ is not shown to avoid cluttering the pictures). The color of a landmark corresponds to the sensor that is responsible to cover that landmark at that particular time.

Figure 3 illustrates the coverage attained by the sensors throughout the simulation, as well as the total coverage attained by the network. All the quantitites are negative since the footprint used is negative semidefinite. From picture 3, we can see that the coverage attained by each sensor presents large discontinuities across the communication instants, since landmarks are transferred at those instants. On the other hand, we can see that the total coverage attained by the network is nondecreasing, and converges to a value corresponding to the equilibrium configuration that we can see in the last snapshot of Figure 2. The small spikes in the plot must be attributed to the short delays in the message transmission in ROS.

VIII. Experimental Results

A. Experimental Setup

The proposed method has been evaluated with the utilization of the Ascending Technologies NEO hexacopter, depicted in Figure 4. The platform has a diameter of $0.59\,\mathrm{m}$ and a height of $0.24\,\mathrm{m}$. The length of each propeller is $0.28\,\mathrm{m}$, as depicted in Figure 4. This platform is capable of providing a flight time of $26\,\mathrm{min}$, and can reach a maximum airspeed of $15\,\mathrm{m/s}$ and a maximum climb rate of $8\,\mathrm{m/s}$, with payload capacity up to $2\,\mathrm{kg}$. The platform has an onboard Intel NUC computer with a Core i7-5557U and 8 GB of RAM. The NUC runs Ubuntu Server 14.04 with ROS. Additionally, multiple external sensory systems (e.g. cameras, laser scanners, etc.) can be operated on the platform. Regarding the onboard sensory system, a monocular camera developed by Skybotix AG (weight of $0.088\,\mathrm{kg}$, depicted in Figure 4), is attached on the protective case on the front side. The camera was operated in $20\,\mathrm{fps}$ with a resolution of 640x480 pixels.

The proposed method, established in Sections IV and V, has been entirely implemented in Python. The inputs for the method are the same as described in Section VII, plus the position controller sampling time. However, to better suit the dimensions of the laboratory, we set $D = 1.3$ and $\gamma = 0.5$. The generated paths are sent to the NEO platform through the ROS framework.

The platform contains three main components to provide autonomous flight, which are a Mo-Cap system for pose extraction, a Multi-Sensor-Fusion Extended Kalman Filter (MSF-EKF) [23] and a linear Model Predictive Control (MPC) position controller [24]–[26]. The MSF-EKF component fuses the obtained pose information and the NEO IMU measurements. This consists of an error state Kalman filter, based on inertial odometry, performing sensor fusion as a generic software package, while it has the unique feature of being able to handle delayed and multi-rate measurements, while staying withing computational bounds. The linear MPC position controller [26] generates attitude and thrust references for the NEO low-level attitude controller. The overall schematic of the experimental setup is presented in Figure 5.

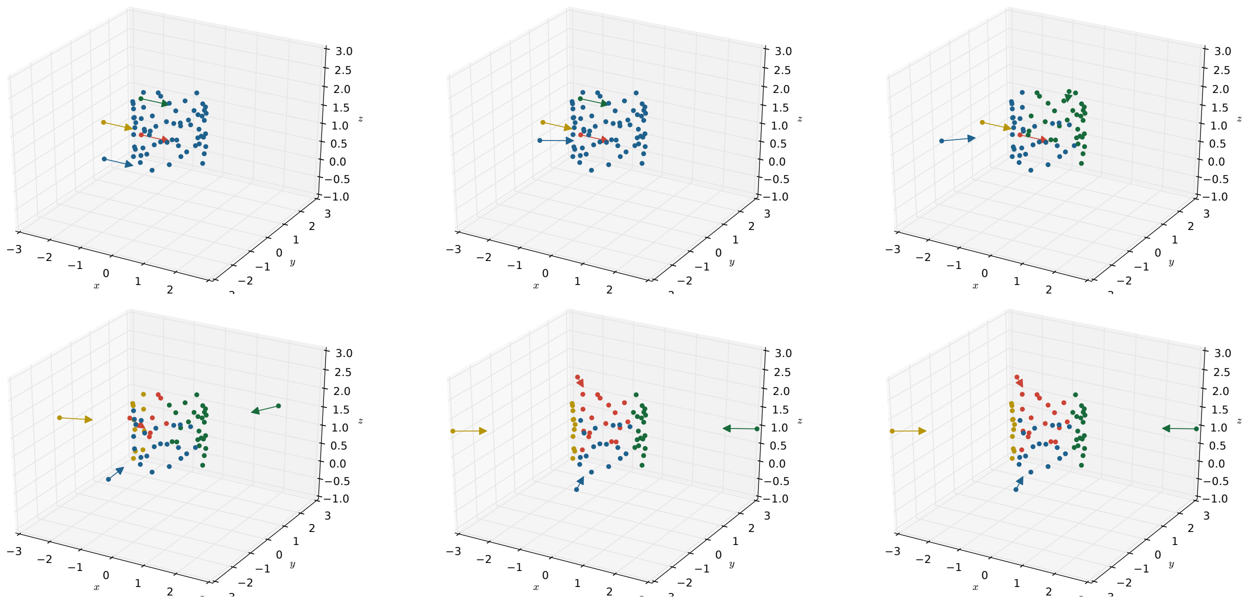

Fig. 2: Snapshots of the configuration assumed by the sensor network. Each sensor σ_i is represented as a colored arrow, with the tail of the arrow being the position p_{σ_i} of the sensor, and the direction of the arrow being u_{σ_i}. Each landmark ℓ is represented as a colored point p_ℓ. In lexicographical order, with $t = 0$ being the start of the simulation, the snaphots are taken at: $t = 0$, $t = 5$, $t = 25$, $t = 50$, $t = 85$, $t = 100$.

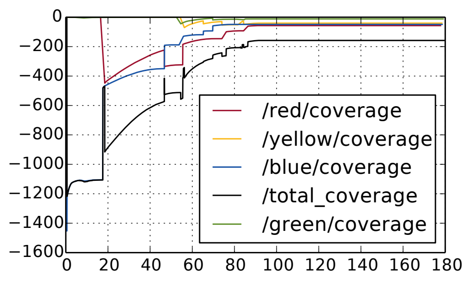

Fig. 3: Coverage attained by each sensor and total coverage attained by the network throughout the simulation. The short spikes in the total coverage signal are due to asynchronous message transmission among ROS nodes.

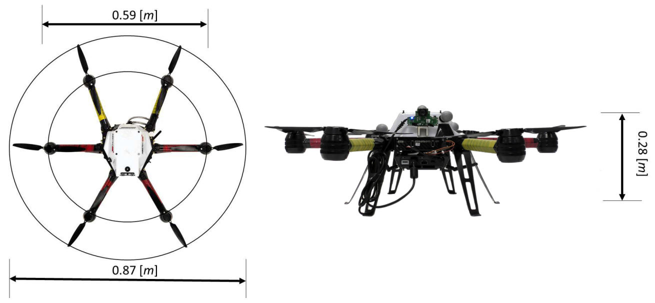

Fig. 4: AscTec NEO platform with attached monocular camera.

B. Experimental Evaluation

For demonstrating the applicability of the presented method, an indoor experiment has been performed in the FROST lab at Luleå University. Figure 6 depicts the artificial structure to inspect assembled for the experimental trial. The cylindrical structure in the pictures is provided for visual purposes, while for the landmark generation we use a cylinder object is with radius of $0.165\,\mathrm{m}$ and height $1\,\mathrm{m}$. For the safety of the flight, and to avoid ground effect and collisions with the ceiling, the landmarks are extracted only between $0.5\,\mathrm{m}$ and $1.1\,\mathrm{m}$ height.

In this experiment, the Vicon Motion-capture (Mo-cap) system has been utilized for precise object localization, and this information is utilized by the NEO for the task execution. The waypoints generated by the algorithm are converted into position-velocity-yaw trajectories as the input for the linear MPC controller [26] for the navigation. This is done by taking into account the sampling time T_s and the desired velocity along the path $\vec{V}_d$ which are $200\,Hz$ and $0.5\,m/s$ respectively. Figure 6 depicts the position of two agents in the left, while in the middle and right the image streams from the two agents are shown. Additionally, the reference path, the trajectories followed by the agents and the point cloud of the object are shown in Figure 7. The starting point of the agents are $[-1.7, -1.5, 0]^\intercal$ and $[-1.7, -1.5, 0]^\intercal$, while the final position is $[0, 1.3, 0.68]^\intercal$ and $[0.5, -1.7, 0.8]^\intercal$. The error between the reference trajectories and the obtained trajectories is mainly caused by ground effect and error in attitude controller. Furthermore, Figure 7 also includes the point cloud that used for obtaining the landmark set. Finally, Figure 8 shows the coverage attained by each agent along

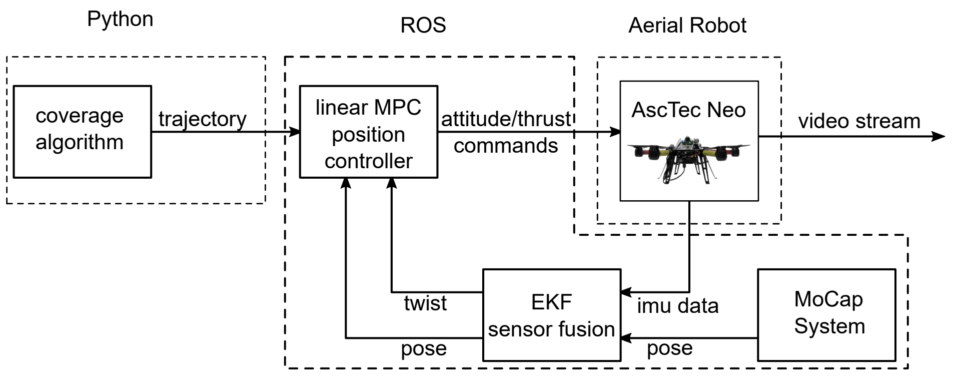

Fig. 5: Software and hardware components used for experimental setup.

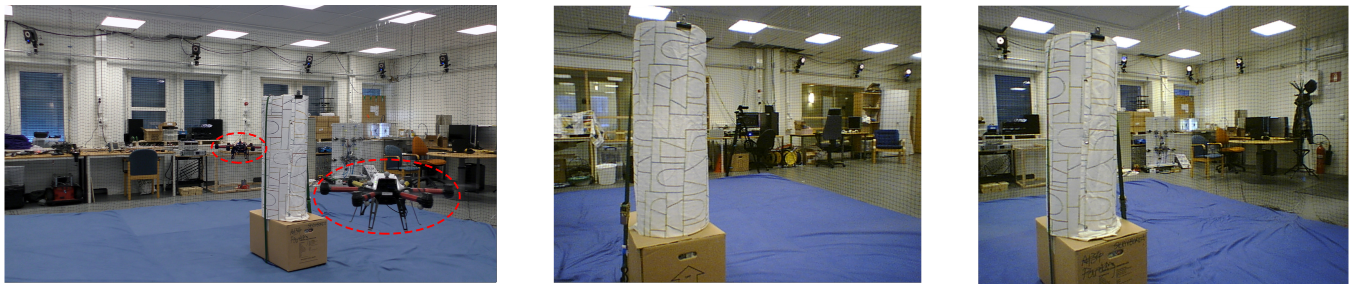

Fig. 6: On the left is the position of two agents in the experiment, on the middle is the visual feedback of the first agent, and on the right is the visual feedback of the second agent.

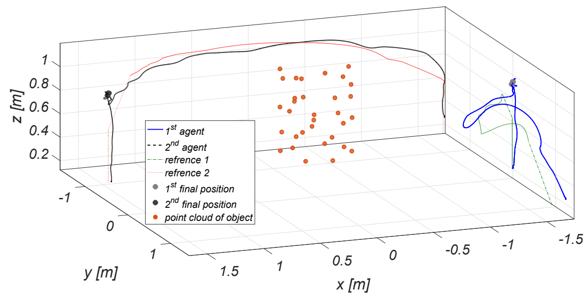

Fig. 7: Trajectories followed in the indoor experiment.

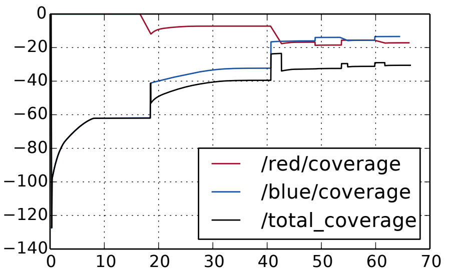

Fig. 8: Coverage attained by each agent along its path, and total coverage attained by the network. The short spikes in the total coverage signal are due to asynchronous message transmission among ROS nodes.

its path, and the total coverage attained by the network.

IX. Conclusions

Motivated by applications in the field of robotic surveillance, in this paper we have investigated a coverage problem of 3D structures with robotic sensor networks. We have introduced an abstraction of the structure to surveil, and we have generalized the concept of Voronoi tessellation to fit our problem, defining a coverage score function as a measure of the quality of the surveillance attained by the sensor network. Then, we have proposed a control algorithm that drives the sensors to a Voronoi tessellation while the coverage score function is nondecreasing. We have also extended the algorithm to deal with collision avoidance problems. The algorithm has been validated by a simulation and an experiment with an aerial robot.

Future developments of this works include the extension of the proposed framework to dynamic coverage, and in particular to inspection problems (e.g., detecting faults on a 3D structure). We are also investigating how to improve the coverage score beyond local optima.

References

[1] J. Cortés, S. Martínez, T. Karata, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete Partitioning and Coverage Control for Gossiping Robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[3] Y. Stergiopoulos, M. Thanou, and A. Tzes, "Distributed Collaborative Coverage-Control Schemes for Non-Convex Domains," *IEEE Transactions on Automatic Control*, vol. 60, no. 9, pp. 2422–2427, 2015.

[4] I. I. Hussein and D. M. Stipanovic, "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.

[5] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, “Distributed dynamic coverage and avoidance control under anisotropic sensing,” *IEEE Transactions on Control of Network Systems*, 2016, to appear.

[6] Q. Du, V. Faber, and M. Gunzburger, “Centroidal Voronoi Tessellations: Applications and Algorithms,” *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[7] Y. Stergiopoulos and A. Tzes, “Spatially distributed area coverage optimisation in mobile robotic networks with arbitrary convex anisotropic patterns,” *Automatica*, vol. 49, no. 1, pp. 232–237, 2013.

[8] E. Galceran and M. Carreras, “Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor,” in *IEEE International Conference on Robotics and Automation*, 2013.

[9] ——, “A survey on coverage path planning for robotics,” *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[10] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar, “Exact cellular decomposition of closed orientable surfaces embedded in R^3,” in *IEEE International Conference on Robotics and Automation*, vol. 1, 2001, pp. 699–704.

[11] E. Galceran, R. Campos, N. Palomeras, M. Carreras, and P. Ridao, “Coverage path planning with realtime replanning for inspection of 3d underwater structures,” in *IEEE International Conference on Robotics and Automation*, 2014.

[12] B. Englot and F. S. Hover, “Sampling-based coverage path planning for inspection of complex structures,” 2012.

[13] P. Cheng, J. Keller, and V. Kumar, “Time-optimal uav trajectory planning for 3d urban structure coverage,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[14] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, “Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots,” *Autonomous Robots*, pp. 1–20, 2015.

[15] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, “Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots,” *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[16] A. Adaldo, D. V. Dimarogonas, and K. H. Johansson, “Hybrid coverage and inspection control for anisotropic mobile sensor teams,” in *IFAC World Congress*, 2017, to appear.

[17] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation*, 2011.

[18] D. OuYang and H.-Y. Feng, “On the normal vector estimation for point cloud data from smooth surfaces,” *Computer-Aided Design*, vol. 37, no. 10, pp. 1071–1079, 2005.

[19] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability and Robustness*. Princeton Univerisity Press, 2012.

[20] E. Eisemann and X. Décoret, “Fast scene voxelization and applications,” in *ACM symposium on Interactive 3D graphics and games*, 2006.

[21] K. Zhou, Q. Hou, R. Wang, and B. Guo, “Real-time kd-tree construction on graphics hardware,” *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, p. 126, 2008.

[22] Robot Operating System (ROS). [Online]. Available: http://www.ros.org/

[23] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “A robust and modular multi-sensor fusion approach applied to mav navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[24] K. Alexis, G. Nikolakopoulos, and A. Tzes, “Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances,” *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[25] ——, “Model predictive quadrotor control: attitude, altitude and position experimental studies,” *IET Control Theory & Applications*, vol. 6, no. 12, pp. 1812–1827, 2012.

[26] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, “Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system,” in *Robot Operating System (ROS) The Complete Reference*, A. Koubaa, Ed. Springer, (to appear).