Multisensor Placement in 3D Environments via Visibility Estimation and Derivative-Free Optimization

François-Michel De Rainville[1], Jean-Philippe Mercier[2], Christian Gagné[1],
Philippe Giguère[2], and Denis Laurendeau[1]

***Abstract*— This paper proposes a complete system for robotic sensor placement in initially unknown arbitrary three-dimensional environments. The system uses a novel approach for computing the quality of acquisition of a mobile sensor group in such environments. The quality of acquisition is based on a geometric model of a camera which allows accurate sensor models and simple occlusion computation. The proposed system combines this new metric with a global derivative-free optimization algorithm to find simultaneously the number of sensors and their configuration to sense accordingly the environment. The presented framework compares favourably with current techniques working in two-dimensional environments. Furthermore, simulation and experimental results demonstrate the ability of the system to cope with full three-dimensional environments, a domain still unexplored by previous methods.**

I. INTRODUCTION

Recent advances in mobile computing and miniaturized robotics allow groups of small and relatively simple robots to accomplish tasks that a single sophisticated robot could not achieve alone. The robustness, flexibility, and scalability features of multirobot systems [1] make them well-suited for inspection and surveillance applications. For instance, a group of robots could be deployed to explore hazardous or remote environments, such as defective nuclear power plants (e.g., Fukushima) or outer space sites (e.g., Mars).

This work focuses on automatic placement of a group of mobile sensors in initially unknown arbitrary three-dimensional environments. More specifically, we are motivated by telepresence applications where an external user investigates a scene that is inaccessible to humans. By moving a *virtual sensor* $\mathbf{s}_\mathrm{v}$ in the environment, this user can define the scene he wants to observe. The virtual sensor is a tool manipulated by the user just as a standard camera would be, but it is not limited by physical constraints. Therefore, it can be placed anywhere in the environment and have unrealistic parameters such as a very high resolution and very large field of view. The problem we tackle in this work is to automatically deploy a fleet of mobile robots (with potentially low resolution sensors) to acquire all the information contained in the region of interest designated by the virtual sensor $\mathbf{s}_\mathrm{v}$ with a minimal number of sensors. Our method can be applied to any generic cost function capturing different aspects of the environment. However, in this paper, we are specifically interested in the visual inspection of an environment. Thus, the cost function will be tied to the satisfaction of a desired pixel density, which is defined as the number of pixel observing a given surface.

Related problems have been approached from different angles in the sensor placement literature. In Sec. II, we review recent work in the *next best view* and sensor network research area that inspired this work. To solve our formulation of the sensor placement problem, we first introduce, in Sec. III, the pixel density measure to efficiently quantify a camera's ability to observe a general three-dimensional space. Then, we present, in Sec. IV, a combination strategy to merge the acquisition performance of many sensors into a single quality measure for all the sensors. Afterwards, we describe in Sec. V a novel divide-and-conquer optimization approach for sensor placement that allows the simultaneous identification of the right number of sensors and their position. Finally, in Sec. VI and VII, we present simulations and real world experiments that compare the proposed framework performance against established methods.

II. RELATED WORK

The problem addressed in this work is closely related to the well-known art gallery problem, which consists of watching over an entire art gallery with a minimal number of guards. In this paper, not only do we want to observe an entire region of interest defined by virtual sensor $\mathbf{s_v}$, but we also require this target to be sampled at a given pixel density. Thus, concepts from sensor placement and visibility estimation are combined in a global optimization framework to solve our problem.

On the one hand, Next Best View (NBV) strategies address the coverage problem by searching greedily for the next sensor position that maximizes the acquisition of the unobserved volume based on past observations. The search is made by simulating views of the sensor in an environment model built online. Deterministic solutions to this problem are presented in [2,3] using an enclosing sphere around the object to be observed and cutting the unoccupied volume as it is sensed. This enclosing sphere hypothesis has been used jointly with a stochastic optimization for view planning [4]. It was proposed later to use NBV planners to cover an indoor environment with a mobile robot, dropping altogether the enclosing volume hypothesis [5,6]. These planners deal

This work is supported by grants from the FRQ-NT and NSERC. The authors would like to thank Annette Schwerdtfeger for proofreading this manuscript.

[1]Laboratoire de vision et systèmes numériques, Département de génie électrique et de génie informatique, Université Laval, Québec, Québec, Canada G1V 0A6 francois-michel.de-rainville.1@ulaval.ca, {christian.gagne, denis.laurendeau}@gel.ulaval.ca

[2]Département d'informatique de génie logiciel, Université Laval, Québec, Québec, Canada G1V 0A6 jean-philippe.mercier.2@ulaval.ca, philippe.giguere@ift.ulaval.ca

poorly with multisensor scenarios since they tend to fragment space and require a greater number of views than the optimal solution.

On the other hand, the sensor network literature solves the sensor placement problem in multisensor scenarios by relying on the gradient of an observability function shared by all sensors. Climbing the gradient for each sensor results in maximizing the visibility of the whole network. A first type of gradient method allowed the deployment of a mobile sensor network following a potential field induced by the environment bounds and the other sensors [7]. A second type of gradient has been derived from the Voronoi decomposition of the environment where each Voronoi cell, centred on a sensor, is moved following a utility function [8]. These gradient methods have been combined to allow convergence to either types of gradient through a parameter [9]. This unification allowed positioning of quadrotor helicopters to observe a planar environment entirely [10]. However, these approaches are limited by the fact that, in arbitrary three-dimensional space, occlusions create artifacts causing the derivation of the cost function to be dependent on the geometry of the environment.

While NBV strategies provide a means to compute the quality of a sensor's position given its model and the environment, gradient methods propose ways to combine multiple sensors and obtain solutions with fewer viewpoints. We claim that combining these concepts helps to solve multisensor placement in general three-dimensional spaces. In fact, we show in the next sections that embedding the combination function of gradient methods in the NBV quality estimation framework enables the optimization of viewpoints.

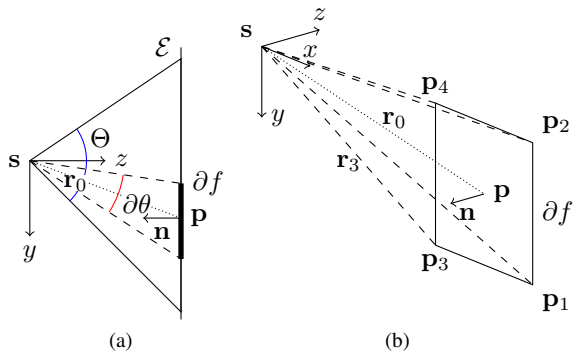
III. CAMERA PIXEL DENSITY

The visual inspection of an environment requires the observation of its components with a desired precision. For camera sensors, this can be measured by the pixel density: the more pixels per unit area, the more precise the observation is. In the remainder of the section, we develop a complete approach to evaluate this.

We define the pixel density of a camera sensor at any point in the environment using a pinhole camera model imaging on a discrete sensor matrix. Let $\mathcal{E} \subset \mathbb{R}^3$ be the environment the robots evolve in and $\mathbf{s} \in \mathcal{P}$ be the sensor position and orientation, with $\mathcal{P} = \mathbb{R}^\delta$ representing a sensor's δ degrees of freedom. A sensor footprint $\mathcal{F}$ comprises every point $\mathbf{p} \in \mathcal{E}$ inside its field of view which is visible from its position.

A two dimensional cut of a directional sensor viewing frustum is shown in Fig. 1(a). To compute pixel density $d(\mathbf{s}, \mathbf{p})$ of sensor $\mathbf{s}$ at $\mathbf{p}$, we compute the area of a footprint $\partial f \subseteq \mathcal{F}$ containing a constant number of pixels and subtended by angles $\partial\theta$ and $\partial\phi$, which are fractions of the sensor's field of view Θ by Φ. We assume that ∂f is small enough to be planar at $\mathbf{p}$.

The plane supporting ∂f is given by $(\mathbf{q}-\mathbf{p})\cdot\mathbf{n} = 0$, where $\mathbf{q} \neq \mathbf{p}$ is any 3D point on that plane and $\mathbf{n}$ is its normal. This plane is the base of a pyramid of vision associated with $\mathbf{p}$, as shown in Fig. 1(b). The pyramid edges $\mathbf{r}_1$ to $\mathbf{r}_4$ are



(a) (b)

Fig. 1. (a) Two dimensional cut of sensor $\mathbf{s}$ viewing pyramid and footprint ∂f covering point $\mathbf{p} \in \mathcal{E}$ (Φ and $\partial\phi$ are hidden by the orthogonal projection); (b) Components required to compute the area of ∂f at $\mathbf{p}$ (for readability, only $\mathbf{r}_3$ is annotated).

given by rotating $\mathbf{r}_0$, a vector between sensor $\mathbf{s}$ and point $\mathbf{p}$, by extrinsic rotation matrices of angle $\pm\frac{\partial\phi}{2}$ and $\pm\frac{\partial\theta}{2}$.

$\mathbf{r}_1$ to $\mathbf{r}_4$ intersect the plane at points $\mathbf{p}_1$ to $\mathbf{p}_4$ respectively, giving the four corners of the convex polygon ∂f. The corners are $\mathbf{p}_k = \delta_k \cdot \mathbf{r}_k$ with $k = 1, \ldots, 4$ and $\delta_k = \frac{\mathbf{p}\cdot\mathbf{n}}{\mathbf{r}_k\cdot\mathbf{n}}$. The area of ∂f is thus given by

$$a(\mathbf{s}, \mathbf{p}) = \frac{\|\overline{\mathbf{p}_1\mathbf{p}_4}\|\|\overline{\mathbf{p}_2\mathbf{p}_3}\|}{2} \sin\varphi, \tag{1}$$

where $\|\cdot\|$ is the norm of a vector and φ the angle between vectors $\overline{\mathbf{p}_1\mathbf{p}_4}$ and $\overline{\mathbf{p}_2\mathbf{p}_3}$. Finally, the pixel density $d(\mathbf{s}, \mathbf{p})$ of sensor $\mathbf{s}$ with focal length f and pixel size $u \times v$ at point $\mathbf{p}$ is given by

$$d(\mathbf{s}, \mathbf{p}) = \begin{cases} \frac{\rho}{a(\mathbf{s},\mathbf{p})} & \text{if } v(\mathbf{s}, \mathbf{p}) \text{ is true} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $v(\mathbf{s}, \mathbf{p})$ is a Boolean visibility function that is true when $\mathbf{p}$ can be seen by sensor $\mathbf{s}$ and false otherwise, and

$$\rho = \frac{4f^2 \tan\left(\frac{\partial\theta}{2}\right) \tan\left(\frac{\partial\phi}{2}\right)}{uv}. \tag{3}$$

Therefore, (2) has four elements: the sensor intrinsic parameters, the visibility between $\mathbf{s}$ and $\mathbf{p}$, the distance between these two points, and the environment surface normal. The first variable is constant for a sensor, and the other three can readily be extracted from a geometric representation of the environment such as an occupancy octree [11].

IV. VISIBILITY COST FUNCTION

As mentioned earlier, evaluating the views one at a time and selecting the next best point of view to place the next sensor causes a fragmentation problem that leads to a resulting sensor network much larger than the optimal size. This section describes the original cost function that was developed to solve the sensor placement problem outlined in the introduction.

We recall that the problem is to place a sensor network $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots\}$ so that a region of interest $\mathcal{I} \subseteq \mathcal{E}$, delimited by the view of a virtual sensor $\mathbf{s}_\text{v}$, is observed with the

requested pixel density. We evaluate the placement quality of a given network S as:

$$f(\mathbf{s}_\mathrm{v}, S) = \int_{\mathcal{I}} f_\mathrm{e}(\mathbf{s}_\mathrm{v}, S, \mathbf{i})\, \mathrm{d}\mathbf{i}, \tag{4}$$

where $f_\mathrm{e}(\mathbf{s}_\mathrm{v}, S, \mathbf{i})$ reports the error between the pixel densities of $\mathbf{s}_\mathrm{v}$ and sensor network S at point $\mathbf{i} \in \mathcal{I}$. This translates into the following minimization problem:

$$\min_{S \in \mathbb{S}} f(\mathbf{s}_\mathrm{v}, S), \tag{5}$$

$$\text{s.t.} \quad \frac{f(\mathbf{s}_\mathrm{v}, S \backslash \mathbf{s}_i) - f(\mathbf{s}_\mathrm{v}, S)}{1 - f(\mathbf{s}_\mathrm{v}, S)} \geq T_\mathrm{c}, \ \forall \mathbf{s}_i \in S, \tag{6}$$

where $\mathbb{S}$ is the possible sensor network domain. Equation (6) enforces the constraint that all sensors making the network S should make a relative reduction of the error by at least a value of T_c. Stated otherwise, we do not want to use more sensors than necessary, such that the contribution of each unit to the performance of the network should be above a given value.

We propose to use the bounded error function

$$f_\mathrm{e}(\mathbf{s}_\mathrm{v}, S, \mathbf{i}) = \max\left(0, \frac{d(\mathbf{s}_\mathrm{v}, \mathbf{i}) - f_\mathrm{c}(S, \mathbf{i})}{d(\mathbf{s}_\mathrm{v}, \mathbf{i})}\right), \tag{7}$$

where the maximum operation between 0 and the normalized difference is capped by the highest density requested and f_c is a function combining the sensors pixel density. This characteristic helps to focus on regions where desired pixel density is not reached, rather than improve a part of the sensor field that is already observed with the desired resolution.

The last component of the cost function is a combination function $f_\mathrm{c}(S, \mathbf{i})$ that computes the global pixel density for a group of sensors where more than one sensor observe the same point. [9] proposed the use of a p-norm to merge the pixel densities. Given that we are basically interested in covering the region of interest, we chose $p = \infty$ so that only the sensor observing the point with the greater density is retained:

$$f_\mathrm{c}(S, \mathbf{i}) = \max_{\mathbf{s}_i \in S} d(\mathbf{s}_i, \mathbf{i}). \tag{8}$$

That way, no assumption is made on the underlying merging algorithm, which should perform at least as well as the single best sensor observing this part of the environment.

Unfortunately, arbitrary three-dimensional environments contain occlusions that make it difficult to compute the geometry of $\mathcal{I}$ to find an analytical solution to (4). Nonetheless, we can rely on Monte Carlo numerical integration to find a solution to our problem, transforming the integral in (4) into

$$\hat{f}(\mathbf{s}_\mathrm{v}, S) = \frac{1}{m} \sum_{j=1}^{m} f_\mathrm{e}(\mathbf{s}_\mathrm{v}, S, \mathbf{i}_j), \tag{9}$$

with points $\mathbf{i}_j \in \mathcal{I}$ being random samples taken such that their corresponding positions are uniformly distributed over the camera image sensor. The impossibility to define the geometry of $\mathcal{I}$ in arbitrary three-dimensional environments also prevents the derivation of the cost function $\hat{f}$, thus forbidding the use of optimization algorithms requiring the computation of derivatives (e.g., Jacobian matrix). The next section presents an derivative-free optimization algorithm to solve our sensor placement problem.

V. Optimizing Sensor Placement

The modelling presented in the previous section provides a realistic and precise way to evaluate sensing performance over the region of interest in an unknown environment. However, as stated before, gradient-based optimization cannot be used as is. We propose to integrate the sensing model in a divide-and-conquer derivative-free optimization algorithm tailored to sensor placement. This is achieved by merging two existing algorithms, namely CMA-ES, which is used to optimize the configuration of each sensor individually, and cooperative optimization, which allow a joint optimization of all sensors, where sensors can be added and removed on the fly. The amalgam of these two optimization approaches is novel to the best of our knowledge, while being particularly fit for sensor placement optimization in the proposed context.

A. Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [12] is a derivative-free global optimization technique that uses self-adaptation of a multivariate normal search distribution to maximize the probability of generating fitter candidate solutions. It is one of the most powerful black-box optimization algorithms available in the literature [13]. In its $(1 + \lambda)$ form used here, the CMA-ES algorithm creates, at every iteration, λ independent candidate solutions

$$\mathbf{s}_j = \mathbf{s}_\text{best} + \upsilon, \ \forall j \in \{1, \ldots, \lambda\}, \tag{10}$$

where $\mathbf{s}_\text{best}$ is the best solution obtained at the last iteration, and $\upsilon \sim \mathcal{N}\left(0, \sigma^2 \mathbf{C}\right)$ a random variable following a multivariate normal distribution. From these λ variations, statistics are evaluated at each iteration to adjust the covariance matrix $\mathbf{C}$ and the step size σ according to the feedback received from the cost function. For more details on CMA-ES the reader is referred to [12].

In our sensor placement problem, a CMA-ES solution $\mathbf{s}$ stands for the complete set of parameters of a single sensor. In the next section, we explain how these strategies cooperate and enable the placement of several sensors.

B. Cooperative Optimization

Cooperative optimization decomposes an optimization problem into subproblems automatically and combines the subsolutions to form an optimal global solution [14]. In our case, the sensor configurations $\mathbf{s}$ are the subsolutions, which are optimized separately into what is called a *species*, noted $\mathfrak{S}$. Cooperation between the species to form a sensor network occurs only at evaluation time, where each candidate solution $\mathbf{s}_j^i \in \mathfrak{S}^i$ is combined with a representative $\mathbf{h}^k$ from each of the other species to assess its quality.

In our context, the quality assessment of a candidate solution $\mathbf{s}_j^i$ is conducted using (9), in which $S = \{\mathbf{h}^1, \ldots, \mathbf{h}^{i-1}, \mathbf{s}_j^i, \mathbf{h}^{i+1}, \ldots, \mathbf{h}^n\}$. The *representative set*

1 initialize $\mathfrak{P} \leftarrow \{\mathfrak{S}^i \ : \ i = 1, \ldots, n\}$;
2 choose $\mathfrak{H} \leftarrow \{\text{select\_random}(\mathfrak{S}^i) \ : \ i = 1, \ldots, n\}$;
3 **while** $\neg$*stop* **do**
4 **foreach** *species* $\mathfrak{S}^i \in \mathfrak{P}$ **do**
5 $\mathfrak{S}^i \leftarrow \text{variate}\left(\mathbf{h}^i, \mathcal{N}(0, (\sigma^i)^2\mathbf{C}^i)\right)$;
6 evaluate$(\mathfrak{S}^i, \mathfrak{H}\backslash\mathbf{h}^i)$;
7 $\sigma' \leftarrow \sigma^i$; $\mathbf{C}' \leftarrow \mathbf{C}^i$;
8 update the step size σ';
9 **if** $\exists j \ : \ f\left(\mathbf{s}_j^i\right) \leq f\left(\mathbf{h}^i\right)$ **then**
10 $\mathbf{h}^i \leftarrow \arg\min_{\mathbf{s}_j^i \in \mathfrak{S}^i} f(\mathbf{s}_j^i)$;
11 update covariance matrix $\mathbf{C}'$;
12 **if** $\text{diag}((\sigma')^2\mathbf{C}') > \gamma$ **then**
13 $\sigma^i \leftarrow \sigma'$; $\mathbf{C}^i \leftarrow \mathbf{C}'$;
14 $\mathfrak{H} \leftarrow \{\text{select\_best}(\mathfrak{S}^i) \ : \ i = 1, \ldots, n\}$;
15 **if** *improvement* $< T_{\text{i}}$ **then**
16 remove species in $\mathfrak{P}$ with *contribution* $< T_{\text{c}}$;
17 $\mathfrak{H} \leftarrow \mathfrak{H}\backslash\mathfrak{H}^-$;
18 **for** $i = 1, \ldots, n$ **do**
19 $\sigma^i \leftarrow 2\sigma^i$;
20 add a new species $\mathfrak{P} \leftarrow \mathfrak{P} \cup \{\mathfrak{S}'\}$;
21 $\mathfrak{H} \leftarrow \mathfrak{H} \cup \{\text{select\_random}(\mathfrak{S}')\}$;

Alg. 1. Cooperative coevolution with $(1 + \lambda)$-CMA-ES.

$\mathfrak{H} = \{\mathbf{h}^1, \ldots, \mathbf{h}^n\}$ is composed of the best candidate of each species. At all time, $\mathfrak{H}$ represents the candidate configuration of the sensor network.

The proposed cooperative framework with the incorporated CMA-ES steps is presented in Alg. 1. To begin with, a population $\mathfrak{P}$ is initialized randomly with n species $\mathfrak{S}_i$. One representative $\mathbf{h}^i$ of each species is copied in $\mathfrak{H}$. The optimization loop begins at line 3 where, in turn, each species is given a chance to optimize the position of the sensor it represents. At line 5, candidate solutions are generated by varying the species representative $\mathbf{h}^i$ using the usual CMA-ES procedure. Then, these solutions are evaluated in cooperation with $\mathfrak{H}$ excluding $\mathbf{h}^i$ at line 6 and the species' CMA-ES parameters are updated at lines 7 through 13. On lines 9 to 11, the representative $\mathbf{h}^i$ of species i is replaced by the best variation $\mathbf{s}_j^i \in \mathfrak{S}^i$ if it has a better fitness.

CMA-ES is known for its log-linear convergence to optima due to the shrinking of its sampling distribution $\mathcal{N}$. This contraction is desirable up to a certain level in cooperative optimization, as too small a sampling distribution prevents coadaptation between species. Thus, we introduce (lines 12 and 13) a mechanism to prevent contraction of the sampling distribution when any element on the diagonal of $(\sigma')^2\mathbf{C}'$ falls below a given threshold γ.

Once an iteration of single sensor optimization is completed for each species, their current best solution is copied into $\mathfrak{H}$. Then, improvement is verified at line 15. If the global solution quality, evaluated over $\mathfrak{H}$ with (9), has not improved by a given threshold T_{i} in the last I generations, the system is considered as *stagnating*. In this case, unproductive species (i.e., those contributing less than a threshold T_{c}) are removed from $\mathfrak{P}$ along with their representative $\mathfrak{H}^-$ from $\mathfrak{H}$. The relative contribution of any species $\mathfrak{S}^i$ is given by

$$c(\mathfrak{S}^i) = \frac{\hat{f}(\mathbf{s}_{\text{v}}, \mathfrak{H}\backslash\mathbf{h}^i) - \hat{f}(\mathbf{s}_{\text{v}}, \mathfrak{H})}{1 - \hat{f}(\mathbf{s}_{\text{v}}, \mathfrak{H})}. \tag{11}$$

On line 19, any remaining species undergoes a diversification step to help coadaptation of converged species to the new configuration. Finally, a new species, initialized randomly, is added to the population and the cooperative optimization continues until the termination criterion is reached. The process of removing and adding species when stagnation occurs is pressuring the system to find useful solutions both on number of sensors and optimal position aspects.

Another interesting feature with the proposed cooperative optimization is that species do not need to be homogeneous. In fact, the species are optimized in isolation and they interact only through (9), where the sensors density are independent. This allows $\mathfrak{P}$ to be composed of heterogeneous sensors with different intrinsic parameters and degrees of freedom.

VI. SIMULATION RESULTS

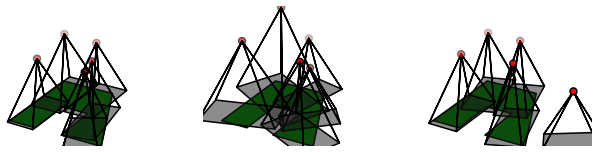
We conducted two different sensor placement simulations. The first one involves four degrees of freedom cameras hovering over a two-dimensional nonconvex closed polygon. This setup is similar to the one presented in [10] for their rectangular field of view cameras. The second simulation experiment extends our approach results to three-dimensional environments using cameras mounted on a heterogeneous group of aerial and ground robots in two different scenarios: a disaster-stricken building and a museum. These two different scenarios demonstrate the ability of the system to find the ideal number of sensors and place them in different conditions.

A. Two-dimensional Target

The first experiment allows direct comparison with the decentralized gradient controller (DGC) [10]. In this setup, a group of cameras mounted on aerial robots is requested to observe entirely the interior of a nonconvex polygon. The environment is shown in Fig. 2.

The camera parameters were set to a focal length $f = 10\,\text{mm}$, pixel size $u = v = 10\,\mu\text{m}$, and fields of view $\Theta = 40°$ by $\Phi = 70°$. The optimization was conducted for sensor positions $\mathbf{s} = [x, y, z, \theta]$ and for a fixed number of sensors $n = 5$, in an environment known *a priori*. Thus, for this test, lines 15 to 21 of Alg. 1 do not execute because the number of sensors is fixed. The sensor's initial position x, y was chosen randomly inside the polygon to be observed at an altitude z between 30 and 40 cm and an orientation θ between 0 and 360°.

The desired pixel density $d(\mathbf{s}_{\text{v}}, \mathbf{i}), \ \forall \mathbf{i} \in \mathcal{I}$ for the cooptimization controller (CC) was chosen as $0.3\,\text{Mpx/m}^2$. The initial standard deviation σ^i and covariance matrix $\mathbf{C}^i$ of the CMA-ES were respectively set to 0.3 and the identity matrix so that CMA-ES can explore the complete environment.



(a) CC, $1.3e7$ $\frac{\text{mm}^2}{\text{px}}$ (b) DGC, $3.3e9$ $\frac{\text{mm}^2}{\text{px}}$ (c) Glob, $4.9e9$ $\frac{\text{mm}^2}{\text{px}}$

Fig. 2. Median configuration found for the two-dimensional five-sensor scenario, with the area per pixel metric [10].

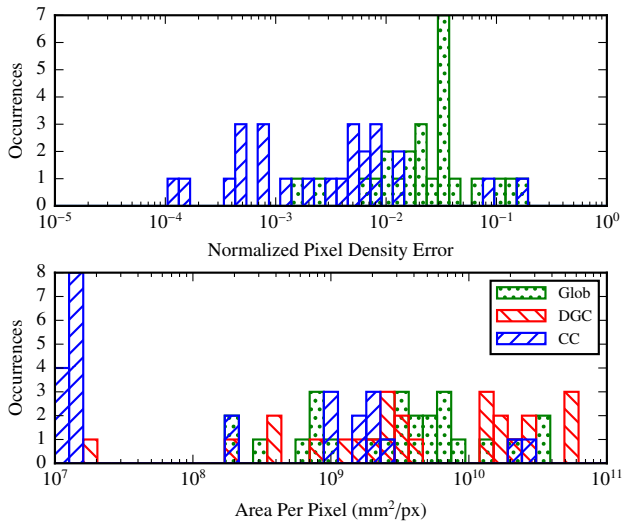


Fig. 3. (Top) Distribution of the 25 configurations found by CC according to the normalized density error. (Bottom) Same configurations for CC and those found for DGC according to the area per pixel metric presented in [10], lower is better.

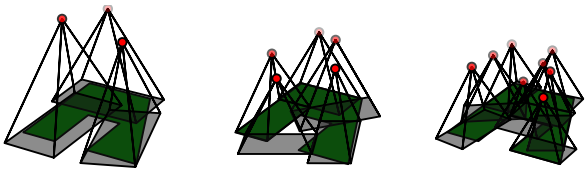
Parameter γ that prevents contraction was set to 0.05. The parameters for the DGC were $a = \frac{uv}{f^2} = 10^{-6}$, the initial prior $w = 2^{16}$, and the gains $k = 10^{-6}[1\ 1\ 0.1\ 10^{-9}]$, which are the values used in [10] for a similar environment. A global standard CMA-ES algorithm (Glob) has also been used to compare the performance of CC against an algorithm working on all sensors at once. All algorithms were stopped when the best solution did not improve over 50 iterations. DEAP' [15] CMA-ES implementation was used.

Fig. 3 presents the distribution of the results achieved by the three algorithms on the two dimensional environment experiment. The top plot shows the results for CC and Glob with the Normalized Pixel Density Error (NPDE) metric of (9). It shows that CC reaches consistently better configurations by working on each sensor separately compared to Glob. Moreover, the bottom plot of Fig. 3 presents the 25 configurations for CC and Glob but with quality computed using the area per pixel metric presented in [10]. It shows that CC generally outperforms DGC even if the optimization are conducted on a different metric. Fig. 2 presents the median final configuration achieved by each algorithm. We see that the advantage of CC comes from its ability to minimize the overlap between the sensors view globally and lead to their alignment with the borders of the environment. Glob also aligns the sensors with the borders of the environment,

TABLE I

AVERAGE NORMALIZED PIXEL DENSITY ERROR AND NUMBER OF SENSORS WHEN VARYING REQUIRED DENSITY.

Required Density (d)	Algo.	Avg. (Std. dev.) NPDE	Avg. (Std. dev.) Num. Sensors
$0.15\,\text{Mpx/m}^2$	CC	0.0046 (0.0141)	3.60 (0.49)
	DGC	0.1316 (0.0852)	4 (0)
	Glob	0.0037 (0.0086)	4 (0)
$0.3\,\text{Mpx/m}^2$	CC	0.0178 (0.0323)	5.28 (0.66)
	DGC	0.1566 (0.0798)	6 (0)
	Glob	0.1432 (0.0106)	6 (0)
$0.5\,\text{Mpx/m}^2$	CC	0.0234 (0.0274)	7.56 (1.02)
	DGC	0.2086 (0.0637)	8 (0)
	Glob	0.0499 (0.0270)	8 (0)



(a) $d = 0.15\,\text{Mpx/m}^2$, NPDE $= 0.0000131$ (b) $d = 0.3\,\text{Mpx/m}^2$, NPDE $= 0.0059$ (c) $d = 0.5\,\text{Mpx/m}^2$, NPDE $= 0.0107$

Fig. 4. Median configuration found by CC for the two-dimensional problem when varying the required density.

but does not identify which sensors are useless (minimal contribution), thus sometimes leaving a sensor with absolutly no contribution. The DGC seems to have reached a local optimum where the sensors observe a considerable amount of space outside of the environment.

In the next simulations, we investigated the capacity of Alg. 1 to find the required number of sensors n to observe the entire region of interest with the correct pixel density. The same polygon was used as in the previous experiments, but we varied the desired pixel density d to assess how many sensors are allocated to the task by the method. The optimization was free to add or remove sensors when the cost function is not lower by 2.5% than it was $5n$ iterations earlier, where n is the current number of sensors (i.e., line 15 of Alg. 1). When stagnation occurs, sensors contributing less than $\frac{1}{4n}$ to the combined solution are removed (i.e., line 16 of Alg. 1). These parameters are chosen so that the optimization runs long enough before adding a new sensor and each sensor contributes relatively equally to the solution.

Tab. I presents the average final error and number of sensors for 25 runs of the algorithms on a two-dimensional environment with three different required densities. The number of sensors for DGC and Glob has been chosen by rounding up the number of sensors found by CC. The CC algorithm equals or outperforms the two other algorithms by using less sensors on average. Fig. 4 shows the median configuration for each of the three sets of experiments. We clearly observe that CC is able to find the required number of sensors and their positions to sense the polygon with the desired density.

Fig. 5. Three-dimensional damaged building environment.

B. Three-dimensional Targets

This section presents two scenarios demonstrating the capacity of our framework to handle three-dimensional environments with occlusions. The setup for the full three-dimensional simulations involved the complete physical simulation of the robots and their sensors. The Gazebo robot simulator [16] combined with the ROS robot operating system [17] were used for this purpose. In the simulations, each robot was equipped with a laser range finder for localization, mapping, and trajectory planning, and a Microsoft Kinect to model the world in three-dimensions and act as camera sensor. The parameters for the Kinect camera were $f = 531\,\text{px}, u = v = 1\,\text{px}, \Theta = 62°$, and $\Phi = 48°$. These sensors were selected as they are commonly used on real robots, including ours. The robots were globally localized, and their trajectories were planned by a dynamic window approach algorithm [18].

The *a priori* unknown 3D environment was handled by alternating optimization and displacements. First, a robot was sent to the position where the virtual sensor $\mathbf{s}_\text{v}$ was placed to acquire an initial 3D model of the environment. Then, the optimization was run on this model. After 25 iterations of the Alg. 1 main loop, robots were sent to the positions given by the representative set $\mathfrak{H}$. At all times during the simulations, all robots refined the global 3D model with the data they captured using their 3D sensor. When the robots completed their displacement, the optimization-displacement sequence continued until no more improvement of the NPDE was made over 100 iterations or the NPDE decreased below 1%.

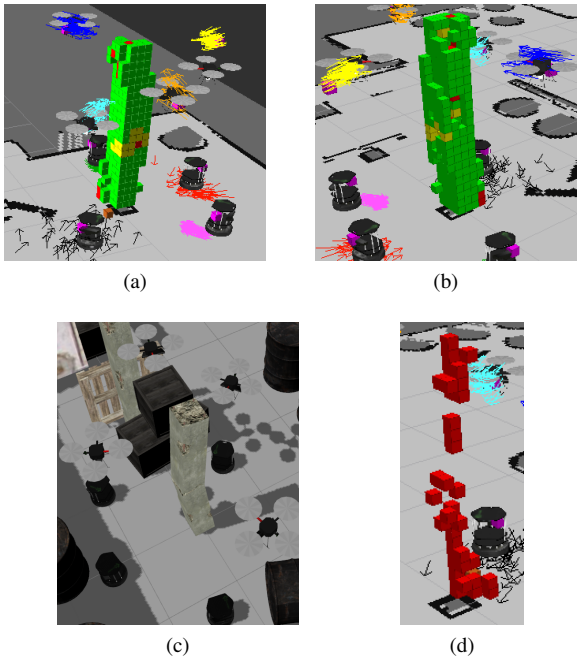
The representation chosen to model environment was an occupancy grid encoded as an octree [11]. This representation allows the retrieval of the necessary information required to compute (2), i.e. the visibility, the distance, and the orientation of the closest surface to the sensor in any given direction. The occupancy grid was updated with the down-sampled data obtained from the simulated noisy Microsoft Kinect. The smallest resolution of the octree was set to 0.1 m, which corresponds approximately to the noise level of the Microsoft Kinect at 5 m range.

The first three-dimensional environment is depicted in Fig. 5. A group of ground and aerial robots was requested to acquire information about a damaged column threatening to collapse in a building. The region of interest was specified in Fig. 5 by a bounding box around the column (highlighted), with the density $d(\mathbf{s}_\text{v}, \mathbf{p})$ set to $0.5\,\text{Mpx/m}^2$ for every surface inside the bounding box. For comparison purposes, a global CMA-ES algorithm configuring the maximum number of sensors was run, along with the cooptimization controller. Two types of experiments were run. The first type, called *Unknown*, started with a completely unknown environment which was modelled online during the experiments. The second type, called *Pre-loaded*, began with a known environment which was loaded from one of the previous runs.

TABLE II

AVERAGE NORMALIZED PIXEL DENSITY ERROR FOR THE DAMAGED BUILDING EXPERIMENT WITH AN INITIALLY UNKNOWN AND AN INITIALLY PRE-LOADED ENVIRONMENT.

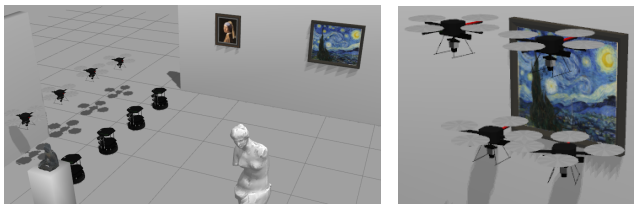
Algorithm	Unknown Avg. (Std. dev.) NPDE	Pre-Loaded Avg. (Std. dev.) NPDE
CC	0.175 (0.0423)	0.206 (0.0246)
Glob	—	0.564 (0.0835)



(a) (b) (c) (d)

Fig. 6. Final solution of the damaged building experiments: (a) and (b) the optimisation data where the green, yellow, and red cubes are the voxels in the virtual sensor field of view observed by the real sensors S with, respectively, at least the required density d, a lower density than required, and no visibility from any sensor; (c) 3D simulator view of the final configuration; and (d) occupied voxels inside the column (occluded by the perfectly covered voxels of (a)).

Tab. II presents the results of five repetitions with the initially unknown environment and 25 repetitions with the preloaded environment in the damaged building. The difference between the initially unknown and pre-loaded environment is not significant as per a Wilcoxon rank-sum test. We also verify that CC outperforms the Glob algorithm. Fig. 6(a) and Fig. 6(b) present the controller data at the end of one simulation. We observe that most of the voxels in the region of interest are marked as perfectly covered, but that the errors in Tab. II are higher than 0. Fig. 6(d) reveals the source of this



(a) Initial configuration (b) Final solution

Fig. 7. Three-dimensional museum environment.

TABLE III

AVERAGE NORMALIZED PIXEL DENSITY ERROR FOR THE MUSEUM EXPERIMENT WITH AN INITIALLY UNKNOWN AND AN INITIALLY LOADED ENVIRONMENT.

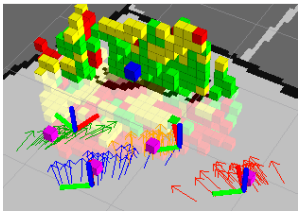
Algorithm	Unknown Avg. (Std. dev.) NPDE	Pre-Loaded Avg. (Std. dev.) NPDE
CC	0.0102 (0.0046)	0.0077 (0.0022)
Glob	—	0.169 (0.0897)

error. Noise in the acquisition sensors, namely the Kinects, creates occupied voxels occluded by another layer of occupied voxels in the occupancy grid. These occluded voxels create a constant error in the evaluation function since they cannot be observed by any sensor. This phenomenon prevents the cost function from decreasing any further. However, the sensor placement is still close to optimal as shown by the large number of exterior voxels marked as perfectly covered.

The second set of simulations, with the same eight robots, took place in the museum environment depicted in Fig. 7(a). A user wanted to observe a given painting with a higher resolution than what the sensors on the robots could actually achieve. Thus, several robots had to combine their sensing capabilities to achieve this task. The region of interest was communicated to the robots through the simulation of a virtual camera sensor $\mathbf{s}_\mathrm{v}$ with parameters $f = 1000\,\mathrm{px}$, $u = v = 1\,\mathrm{px}$, $\Theta = 35°$, and $\Phi = 30°$ and positioned in front of the painting. Thus, the required pixel density $d(\mathbf{s}_\mathrm{v}, \mathbf{p})$ was not uniform in the entire region of interest, but is rather given by (2) with $\mathbf{s}_i = \mathbf{s}_\mathrm{v}$.

Tab. III presents the results of five repetitions with the initially unknown environment and 25 repetitions with the preloaded environment in the museum. Again the difference between the initially unknown and preloaded environment experiments is not significant according to a Wilcoxon rank-sum test. The average final NPDE shows that CC finds a solution very close to the optimal one in this scenario while Glob could not. A final configuration is presented in Fig. 7(b). An interesting fact is that even if CC was allotted four ground robots and four aerial robots, it used only the four aerial robots in the final configuration in 28 of the 30 experiments. In fact, it was impossible for the wheeled robots to achieve the correct pixel density due to the limited height of their sensors. This shows that the presented cooptimization controller was able to select the most appropriate type of sensors from an heterogeneous set to execute a given sensing mission successfully.

(a) Final configuration



(b) Optimisation data (using the same legend as in Fig. 6)

Fig. 8. Room used for the real world experiments.

VII. REAL WORLD EXPERIMENTS

This section presents real world experiments conducted indoors, as presented in Fig. 8(a). For these experiments, a 2D map of the environment was build beforehand with a simultaneous localization and mapping technique [19] since the robots operated on a floor. During the experiments, the robots were localized using adaptive Monte Carlo localization [20], and their trajectories were planned by a dynamic window approach algorithm [18]. Each robot was broadcasting its position to prevent any collision during displacements. Four wheeled robots, similar to those used in simulations, were synchronized through a ROS Multimaster setup. The robots were Create mobile platforms equipped with a Microsoft Kinect, a Hokuyo URG-04LX-UG1 lidar, and a gyroscope. The robots were linked through the building Wi-Fi network and operated by an Asus Eee-PC. The optimization and 3D model ran on an Intel Core i7 920 desktop with 8GB RAM.

The mobile sensors had to generate a view of the red car door, as if taken just above the central cardboard box. The parameters for the virtual camera sensor $\mathbf{s}_\mathrm{v}$ were $f = 6666\,\mathrm{px}$, $u = v = 1\,\mathrm{px}$ with a field of view of $\Theta = 60°$ and $\Phi = 45°$. This setup covered the complete interior part of the door. The real sensor parameters were $f = 531\,\mathrm{px}$, $u = v = 1\,\mathrm{px}$ with a field of view of $\Theta = 62°$ and $\Phi = 48°$. The robots had to position themselves close to the target to obtain the required pixel density.

Five independent runs were conducted with very similar results. The average time of an experiment was 10 minutes, where 90% of the time was spent on displacing the robots and the other 10% was used by the optimization process. The optimization process was run for 50 iterations and the best positions found were sent to the robots. Once the robots arrived at their position, the optimization was continued. This alternating process was run until no further displacements were needed.

Fig. 8(b) shows the controller data at the end of one repre-

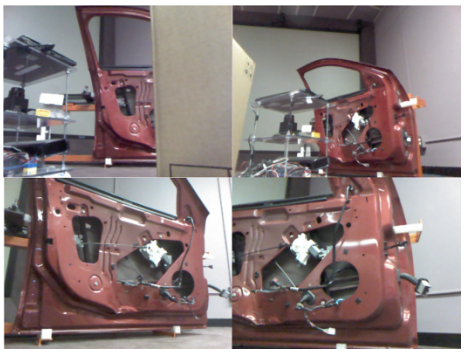


Fig. 9. Images captured with the robot's camera at the end of an experiment.

sentative run out of the five. We observe that most of the door is covered with the correct pixel density. Moreover, Fig. 9 shows the images captured from the robots' camera at the end of the experiment. We notice that the entire area of the door is covered by the combined photographs. Furthermore, the robot taking the top right image is placed such that the robot in front of it does not occlude its vision of the top right part of the door. We also notice how the robots managed to move as close as possible to the target so that their pixel density is maximized while still achieving good visual coverage. To the best of our knowledge, the presented controller is the first to achieve sensor placement considering this level of accuracy in initially unknown, nonconvex, three-dimensional environments. A video showing the complete process of the sensor placement experiment is available at http://vision.gel.ulaval.ca/~fmdrainville/msensor_opt.mp4

VIII. CONCLUSION

In this paper, we proposed a framework for mobile robotic sensor placement in arbitrary three-dimensional environments. As a first original contribution, we proposed a model to estimate pixel density obtained by a sensor network on 3D environments with an arbitrary geometry. The second main contribution of the paper is a divide-and-conquer derivative-free global optimization method for determining the placement of sensors in the environment, resulting from an original combination of cooperative coevolution with CMA-ES. By joining these together, we are able to optimize in real time the placement of a variable number of (possibly heterogeneous) sensors to maximize pixel density in the area of interest.

The proposed framework has been compared favourably to a state-of-the-art method limited to the control of robotic cameras observing a two-dimensional target. Furthermore, simulations have shown that satisfactory placement can also be obtained in different three-dimensional scenarios for multiple sensing tasks. The simulation also demonstrated that the presented controller can cope efficiently with the placement of heterogeneous sensors. Finally, we have implemented the framework on real robots and have shown the capabilities of the whole system experimentally.

This work can be extended in several ways. First, the pixel density in overlapping sensed regions could be computed by merging the information from different sensors instead of completely removing the information of the less precise sensors. Second, a degradation of the cost function performance when sensed regions overlap could be added to help the controller to minimize the number of sensors. Finally, the optimization process could include a measure of displacement length to be minimized during the optimization to reduce the amount of energy required to move the robots in the environment.

REFERENCES

[1] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, 2005, vol. 3342, pp. 10–20.

[2] C. Connolly, "The determination of next best views," in *Proc. of the Int. Conf. on Robotics and Automation*, 1985, pp. 432–435.

[3] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 21, no. 10, pp. 1016–1030, 1999.

[4] G. H. Tarbox and Gottschlich, "Planning for complete sensor coverage in inspection," *Comput. Vis. and Image Underst.*, vol. 61, no. 1, pp. 84–111, 1995.

[5] A. Nüchter, H. Surmann, and J. Hertzberg, "Planning robot motion for 3D digitalization of indoor environments," in *Proc. of the Int. Conf. on Advanced Robotics*, 2003, pp. 222–227.

[6] C. Dornhege, A. Kleiner, and A. Kolling, "Coverage search in 3D," in *Proc. of the Int. Symp. on Safety, Security, and Rescue Robotics*, 2013, pp. 1–8.

[7] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. of the Int. Symp. on Distributed Autonomous Robotics Systems*, 2002, pp. 299–308.

[8] J. Cortés, S. Martínes, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004.

[9] M. Schwager, D. Rus, and J. J. Slotine, "Unifying geometric, probabilistic, and potential field approachs to multi-robot deployment," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 371–383, 2011.

[10] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proc. IEEE*, vol. 99, no. 9, pp. 1541–1561, 2011.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[12] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, 2006, pp. 75–102.

[13] A. Auger and N. Hansen, "Benchmarking the $(1 + 1)$-CMA-ES on the BBOB-2009 testbed." in *Proc. of the Genetic and Evolutionary Computation Conf. Companion*, 2009, pp. 2459–2466.

[14] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving co-adapted subcomponents," *Evolut. Comput.*, vol. 8, no. 1, pp. 1–29, 2001.

[15] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *JMLR, MLOSS*, 2012.

[16] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. of the Int. Conf. on Intelligent Robots and Systems*, 2004.

[17] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[18] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, pp. 34–46, 2007.

[20] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Int. J. Robot. Res.*, vol. 22, no. 12, pp. 985–1003, 2003.