Automatic Sensor Placement For Complex Three-dimensional Inspection and Exploration

François-Michel De Rainville, Christian Gagné, and Denis Laurendeau

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

Abstract

This paper describes an end-to-end system for positioning sensors to observe desired parts of arbitrary initially unknown three-dimensional environments. The system finds the appropriate number of sensors and their configuration using a derivative-free cooperative optimization algorithm relying on a visibility estimation cost function. The system is tested of scenarios with different types of sensors and observation tasks. The first scenario implies six degrees of freedom floating robots to inspect a broken part of the international space station, while the second scenario involves three degrees of freedom rovers gathering information on a stuck vehicle.

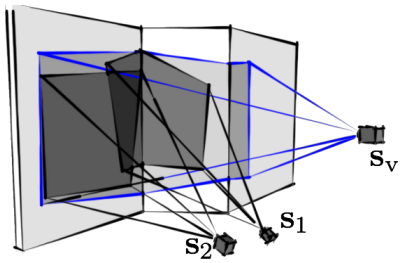


Figure 1. : Sensor placement problem to observe a non-planar wall. The region of interest, in light grey, is induced by the virtual sensor $\mathbf{s}_v$ field of view (blue pyramid). The fields of view of real sensors $\mathbf{s}_1$ and $\mathbf{s}_2$ are shown in dark grey.

1 Introduction

Advances in mobile computing and robotic miniaturization recently enabled researchers to design robotic systems made of multiple relatively simple independent robots to solve tasks a single sophisticated robot cannot [5]. The individuals in a group of cooperative robots often have the advantage of being physically simpler, which lead to more affordable, scalable, and robust systems. These characteristics make collectives of robots well suited for inspection and surveillance of hazardous or unattainable environments, common for operations such as in space module inspection and spatial exploration.

We propose an automated deployment system for mobile sensors to capture complex three-dimensional environments. We envision telepresence applications where a user investigates a scene that is inaccessible to humans. We define the *virtual sensor* $\mathbf{s}_v$ as the object manipulated by the operator to control the desired view of the environment. As shown in figure 1, the virtual sensor $\mathbf{s}_v$ defines the area for which real sensors $\mathbf{s}_i$ (the group of mobile robots) need to gather information to reconstruct the virtual view. In general, the intrinsic parameters of the virtual sensor, such as its resolution and field of view, will be greater than the ones of the real sensors. Thus, numerous well-positioned sensors have to combine their sensing capacity to observe the user-selected region with the requested accuracy.

This problem is related to the well-known art gallery problem [13], where one wants to guard an art gallery using a minimal number of guards. Our problem diverges from the original art gallery problem on a few points. First, we want to work in arbitrary and initially unknown three-dimensional environments. Second, we consider guards with limited sensing capabilities with respect to the field of view and resolution. Third, we require that the resulting resolution matches a predetermined accuracy in every point of the observed environment. These differences make our problem applicable to a wider range of real world applications where robots have to automatically deploy themselves to gather specific information in new or unknown environments.

Different approaches have been explored to solve the art gallery problem. A first class of algorithms, called Next Best View algorithms [4, 14], is most commonly used for inspecting manufactured objects but which were applied recently to indoor scene acquisition. The goal of the algorithm is to acquire automatically and with the fewest views possible all the surfaces of a three-dimensional object enclosed inside a defined volume. Af-

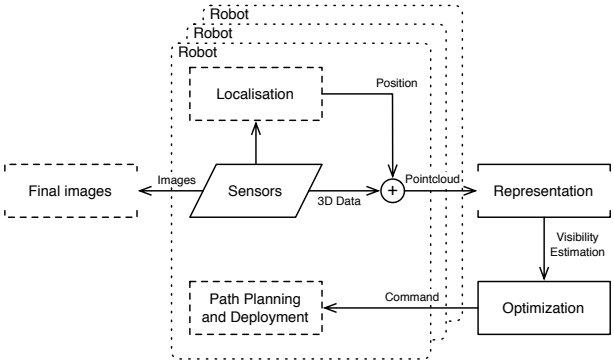


Figure 2. : Proposed system. Algorithms in the "robot" box run locally on each robot, while the other algorithms are centralized on an external controller.

ter each acquisition, the algorithm computes the next best position where each sensor should be placed in order to maximize the coverage of the unknown volume. While Connoly [4] and Pito [14] proposed a deterministic solution to this problem, Nüchter et al. [12], and Tarbox and Gottschlich [18] used a stochastic algorithm to increase the efficiency and decrease the number of views. The common ground between their work and our approach is the simulation of the view a sensor would have from different poses in the environment. Based on these simulations, they selected the best one and move the sensor to this position. However, their approach rely on planning sensor placement one step at a time, which divides the observed space in small non-connected parts and generates a greater number of views than required.

Other multisensor solutions use gradient methods to optimize an observability function [17]. The observability function combines the visibility of each sensor under a single function and performs the optimization globally, making every sensor aware of what the other sensors are observing. This strategy allows for a better control of a group of sensors as a whole. Unfortunately, the visibility function has to be differentiable, a condition that is rarely met in general three-dimensional space due to discontinuities and occlusions. In this paper we propose an end-to-end system capable of optimizing sensor placement in arbitrary and unknown three-dimensional environments.

2 Rationale

This section briefly describes the end-to-end system used to deploy a group of mobile sensors in a full three-dimensional environment. As shown in figure 2, the system contains two major phases, optimization and deployment, while the sensors never stop their sensing of the environment.

The **optimization** phase is responsible to find the appropriate number of sensors and their configuration given the geometry of the environment and the desired virtual view. It uses a visibility estimation procedure as cost function to optimize this configuration. The optimization is run on a centralized unit for all robots. After a given time interval, the optimization algorithm is paused and the best configuration is sent to the robots for the deployment phase. The cost function used for the optimization is described in section 3, while the optimization algorithm is detailed in section 4.

The **deployment** of the sensors is made at constant intervals between optimization phases. The robots plan their displacements locally, using a standard path planning algorithm and localization system. Once all sensors reach their position, the optimization algorithm resumes from its previous state. The deployment phase is not a subject of innovation in the current work and will not be described here.

In the proposed system, the robots continuously **sense** their environment. The collected information is stored in a centralized **representation** accessible to the optimization algorithm. An appropriate three-dimensional representation, containing all the information required by the cost function, is described in section 3.2.

3 Visibility Cost Function

As defined previously, the problem faced is to place an undetermined number of sensors to observe entirely and with a defined accuracy a region of the space delimited by the field of view of a virtual sensor. The present section will outline the mathematical details of the cost function used in the optimization as well as the information retrieval procedures from the selected representation.

3.1 Cost Function

Let $\mathcal{E} \subset \mathbb{R}^3$ be the bounded environment to be observed with a given precision. $\mathcal{E}$ is delimited by the virtual sensor $\mathbf{s}_v$ field of view. The precision of a camera sensor $\mathbf{s}_i$ at any point $\mathbf{e} \in \mathcal{E}$ is defined by its sensing density D, that is the number of pixel per area unit covering this point.

$$D(\mathbf{s}_i, \mathbf{e}) = \frac{\text{number of pixels}}{\text{area}} \tag{1}$$

Sensor $\mathbf{s}_i$ is defined to be in $\mathcal{P} = \mathbb{R}^\delta$, a δ degrees of freedom space. The sensing density $D(\mathcal{S}, \mathbf{e})$ of a group of sensors $\mathcal{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ is defined as the union of the densities among $\mathcal{S}$.

The problem is to match the virtual sensor sensing density $D(\mathbf{s}_v, \mathbf{e})$ with the real sensors sensing density $D(\mathcal{S}, \mathbf{e})$ for the whole environment $\mathcal{E}$. This is accom-

plished by minimizing their difference

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\in\mathcal{P}^n, n} \int_{\mathcal{E}} \max\left(\frac{D(\mathbf{s}_v, \mathbf{e}_j) - D(\mathcal{S}, \mathbf{e}_j)}{D(\mathbf{s}_v, \mathbf{e}_j)}, 0\right) \mathrm{d}\mathbf{e}, \quad (2)$$

where the max operation between the normalized difference and 0 is to saturate the quality measure when sensing the target space with a density higher than requested. Note that equation 2 also minimizes the number of sensors in $\mathcal{S}$.

For general three-dimensional environments, the exact geometry of $\mathcal{E}$ is unknown. Consequently, the integral of equation 2 cannot be solved analytically. However, with a good representation of the environment geometry, the sensing density at any point in the environment can be approximated reasonably well. In fact, the sensing density of traditional line-of-sight sensors depends on their intrinsic parameters, the distance to the point, and the orientation of the surface at the observed point. The intrinsic parameters are constant, while the distance and the normal can be obtained from the geometric representation of the environment. The numerical integration of the integrand in equation 2 can be obtained by estimating the sensing density at N points $\mathbf{e}_j$ in the virtual sensor field of view, such that it can be approximated by

$$\hat{\mathcal{S}}^* = \arg\min_{\mathcal{S}\in\mathcal{P}^n, n} \frac{1}{N}\sum_{j=1}^{N} \max\left(\frac{D(\mathbf{s}_v, \mathbf{e}_j) - D(\mathcal{S}, \mathbf{e}_j)}{D(\mathbf{s}_v, \mathbf{e}_j)}, 0\right). \quad (3)$$

3.2 Information Retrieval

The chosen representation for the three-dimensional data is an occupancy grid [6] encoded as an octree structure [19]. This representation contains the necessary information required to compute equation 3, i.e. the distance and the orientation of the closest surface to the sensor.

Octrees are well known for their fast ray-tracing operation [1], which can return the first occupied voxel hit by the ray. The distance is simply the distance between the two voxel centres, or more precisely between the sensor position and the centre of the occupied voxel hit. The normal at this point can be found by computing one iteration of the marching cube algorithm [11] centred on the hit voxel and its 26 direct neighbours. From all the normals, we choose the one that has minimal angle with the virtual sensor $\mathbf{s}_v$ optical axis if it is a line of sight sensor and the normalized sum of all normals otherwise.

4 Sensor Placement Optimization

The summation in equation 3 qualifies a sensor placement for a given environment and virtual sensor. The derivative of this cost function cannot be calculated due to the specificity of arbitrary three-dimensional environments. Thus, we must rely on less restrictive global optimization algorithms. Derivative-free optimization algorithms, such as evolution strategies [2, 9], manage optimization problems where the only information that can be retrieved from a process $f([x_1 \cdots x_n])$ is its function quality $[y_1 \cdots y_m]$, where $f : \mathbb{R}^n \mapsto \mathbb{R}^m$. However, these algorithms do not handle naturally functions with a variable number of inputs n, e.g. a varying number of sensors. This section proposes a two-level derivative-free optimization algorithm designed to minimize the number of sensors at the first level and to optimize their placement at the second level.

4.1 Cooperative Optimization

Cooperative optimization has been proposed by Potter and De Jong [15] as a method to coadapt subsolutions to solve a problem by decomposing it automatically into subproblems. The framework uses the explicit notion of modularity to assemble multiple subcomponents (i.e. the sensors), to solve a global problem (i.e. multisensor placement).

```
1  initialize 𝔓 ← {𝔖_i, i = 1,...,n};
2  choose ℜ ← {select_random(𝔖_i), i = 1,...,n};
3  while ¬stop do
4      foreach species 𝔖_i ∈ 𝔓 do
5          𝔖_i ← generate_solutions(𝔖_i);
6          evaluate(𝔖_i, ℜ\r_i);
7          𝔖_i ← update(𝔖_i);
8      ℜ ← {select_best(𝔖_i), i = 1,...,n};
9      if improvement < T_i then
10         𝒥 = {j | contrib(𝔖_j) < T_c, j = 1,...,n};
11         𝔓 ← 𝔓 \ ⋃_{j∈𝒥} 𝔖_j;
12         ℜ ← ℜ \ ⋃_{j∈𝒥} ℜ_j;
13         add a new species 𝔓 ← 𝔓 ∪ {𝔖′};
14         ℜ ← ℜ ∪ {select_random(𝔖′)};
```

Algorithm 1: Cooperative co-evolution.

The complete cooperative optimization algorithm proposed by Potter and De Jong [15] is presented in algorithm 1. First, a population $\mathfrak{P}$ of n species $\mathfrak{S}_i$ is initialized randomly, each species optimizes one sensor. One representative $\mathbf{r}_i$ of each species is copied in a representative set $\mathfrak{R}$. Then, 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 lines 5 to 7, a generate-update optimization scheme [3] is used to position the sensors, although any type of algorithm could be used. More details on the selected procedure is given in the next section.

Once an iteration of the species level optimization is completed for each species, their current best solution forms the new representative set $\mathfrak{R}$. Then, the improvement of the evolution is verified at line 9. If the global solution quality, given by the representatives set evaluated

with equation 3, has not improved by a given threshold T_i over the last T_l 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 the population along with their representative from $\Re$. The contribution of any species $\mathfrak{S}_i \in \mathfrak{P}$ is given by the quality ratio of $\Re$ with and without $\mathfrak{S}_i$. Finally, a new species $\mathfrak{S}'$ 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 produces a pressure on the system to find useful placement for the sensors enabling the search for both the number of sensors and their optimal position.

4.2 Covariance Matrix Adaptation Evolution Strategy

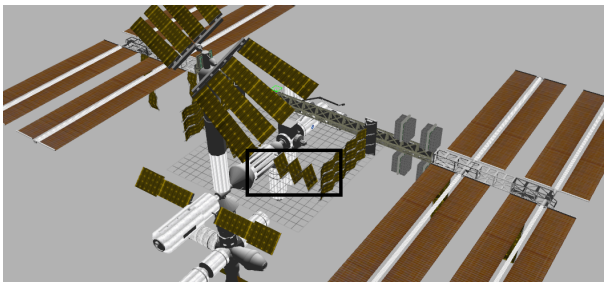
The algorithm chosen to optimize each sensor position (lines 5 to 7 of algorithm 1) is the $(1 + \lambda)$ variant of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [9]. CMA-ES is a derivative-free optimization technique based on the self-adaptation of a multivariate normal search distribution. For details on CMA-ES, the interested reader is referred to [9].

Few modifications have been made to the original algorithm to fit the sensor placement problem and cooperative optimization scheme. First, every time a species is added, the step size of all CMA-ES strategies is doubled to favour coadaptation to new species and increase the chances of escaping local minimas. Second, CMA-ES is known for its log-linear convergence on optimums due to the adaptation of the step size and covariance matrix. However, in situations where coadaptation is necessary, contraction of the search space below a certain level is undesirable. In cooperative optimization algorithm, we prevented the search space from contracting if its smaller axis falls below 10 cm.

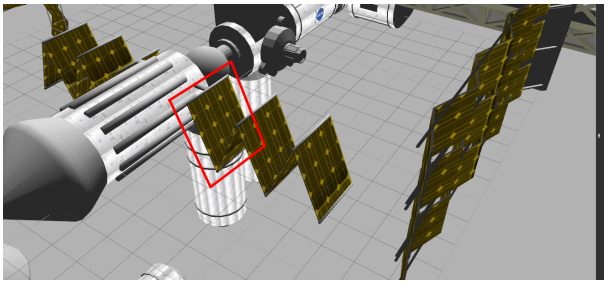
5 Experiments

We conducted two different simulation experiments with different environments and type of sensors. The first set of experiments took place in space around the International Space Station (ISS), while the second set of experiments took place on the ground. All experiments were performed using Gazebo a multirobot simulator [10] combined with ROS the robot operating system [16].

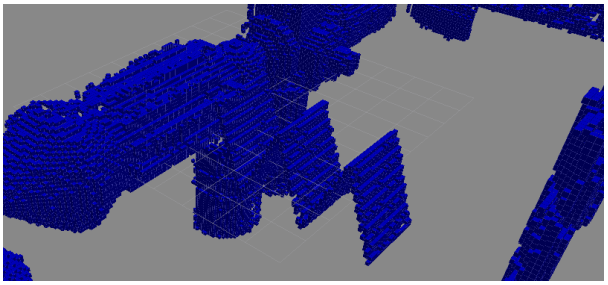
The experiments were run with the occupancy grid of the environment loaded initially. For each set of experiments, the occupancy grid was built offline with a robot similar to the one used in the experiments. The robot was controlled manually and was commanded movement patterns analogous to those the optimization algorithm and path planning would produce. This allowed to offload the central computer that had also to run the complete multi-robot simulator and to accelerate the experiments.



(a) Complete ISS three-dimensional model used for the simulations.



(b) View of the photovoltaic array to be observed by the sensors.



(c) Three-dimensional octree built by scanning the ISS 3D model within the simulator. Each blue cube is an occupied voxel in the occupancy octree.

Figure 3. : International Space Station inspection environment

5.1 International Space Station Inspection

The first simulation experiments took place around the ISS, where a user wanted to inspect one of the photovoltaic arrays. The user placed a virtual camera so that the desired part of the array was entirely seen by the virtual sensor, as shown in figure 3(b).

The virtual camera intrinsic parameters were set to $f = 1597.71\,\text{px}$, $u = v = 1\,\text{px}$ and the horizontal and vertical field of view were respectively 130° and 100°. The sensors participating in the optimization process had six degrees of freedom: their position $[x, y, z]$, orientation $[\phi, \theta]$ (pan and tilt) and a zoom factor ζ. Their intrinsic parameters were the same as the virtual camera for f, u, and v, while their field of view at a zoom factor of 1 was 62°

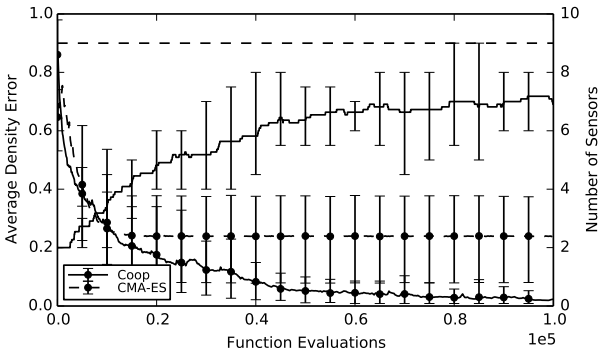


Figure 4. : Average results for the two optimization algorithms for the experiment around the ISS. The circle marked lines report the average density error with the 5%-95% confidence interval range presented by the error bars, while the unmarked lines show the average number of sensors.

horizontally and 48.2° vertically. The three-dimensional sensor used on these robots is a rotating laser range finder similar to a Hokuyo URG-04LX.

Experiment were run with the presented cooperative algorithm (Coop) and a global optimizer for comparison. The parameters for the cooperative algorithm were:

- an improvement length $T_l = 5n$ iterations, where n is the current number of sensors;
- an improvement threshold $T_i = 10\%$;
- a contribution threshold $T_c = 0.25$.

The Coop algorithm had a pool of 10 sensors available and started the optimization with 2. The global optimizer was the well-known CMA-ES [9] described earlier. The default parameters for CMA-ES were selected, except for the number of offspring which was set to 200. Since CMA-ES cannot tune by itself the number of sensors, n was set to 9, the highest number of sensors found in any run of the cooperative algorithm. Both algorithms were stopped after 100 000 evaluations or when the error was lower than 0.01. The optimization algorithm implementations that were used come from DEAP, an evolutionary algorithm framework [8].

Figure 4 shows the average results of 11 repetitions of the ISS experiment. The figure exposes how the cooperative algorithm outperforms a classical global optimization algorithm even if it has to find the number of sensors in addition to their placement. The error bars for the two algorithm almost stop overlapping at half of the allowed optimization time. The average final errors are 0.02 and 0.24 with standard deviation of 0.020 and 0.125 respectively for Coop and CMA-ES. The average final number of sensors for Coop is 7.18 with a standard deviation of 1.11.

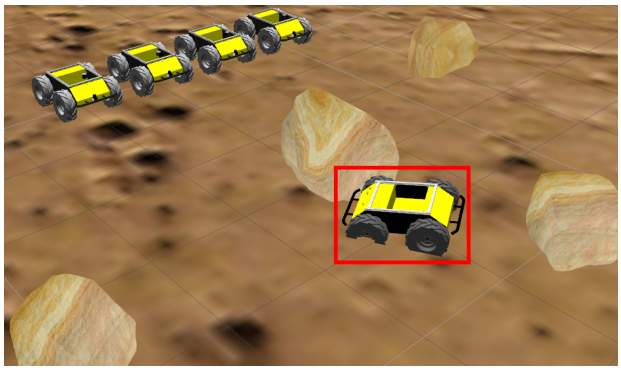


Figure 6. : Ground environment with stuck vehicle highlighted. The four other robots are at their initial position.

Figure 5 presents the median configuration for both optimization algorithms. We observe, on the one hand, that the cooperative algorithm is able to observe almost entirely the desired surface with the desired density. We also note that all seven sensors contribute to the final solution, as they are all oriented directly towards the surface. On the other hand, the poor performance of the global optimization algorithm is explained by the fact that not all nine sensors contribute to the final solution. In fact, it seems that the global optimizer is unable to find which sensors do not contribute to observe the target. Thus, we can state that the internal mechanism quantifying the contribution of each sensors helps the cooperative algorithm to find good solutions even in complex observation scenarios.

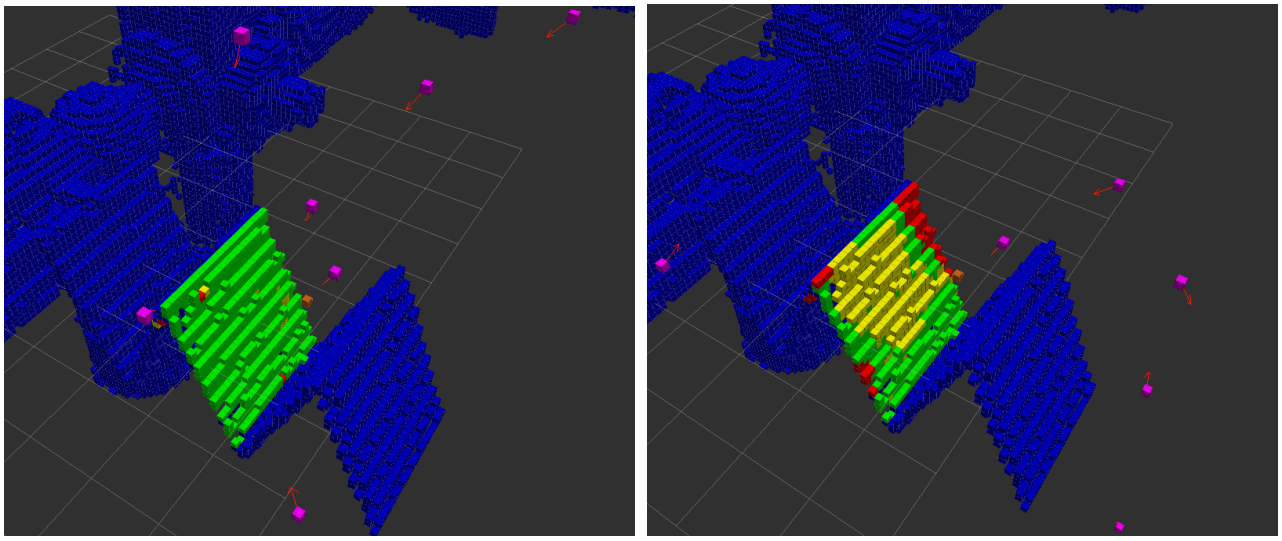
Finally, defining a constrained region around the target would probably help the global optimization algorithm to find better solutions. However, defining such constraints is often hard and time consuming even for simple environments. Thus, the advantage of the cooperative algorithm to be able to work without these constraints is a major advantage.

5.2 Ground Inspection of a Stuck vehicle

The second set of experiments involved sensors mounted on wheeled robots that were requested to observe a stuck vehicle as shown in figure 6. In these experiments, the user had to set a bounding box around the entire vehicle so that the sensors had to observe it from all angles.

The target was delimited by a box centred on the stuck vehicle, the desired sensing density D was set to $5 \times 10^5\,\mathrm{px/m^2}$ for every surface inside the bounding box. The real sensors are mounted on Clearpath Robotics Husky[1] like vehicles. The camera sensors had the same resolution and field of view as in the ISS experiment, but without a zooming factor. Thus the sensors had three degrees of freedom: position $[x, y]$ and orientation θ (pan).

[1] http://www.clearpathrobotics.com/husky/



(a) Cooperative optimization algorithm (average error of 0.0147)

(b) CMA-ES optimization algorithm (average error of 0.1773)

Figure 5. : Median visibility results for the ISS inspection environment. The green cubes are the voxels in the virtual sensor field of view observed by the real sensors $\mathcal{S}$ with at least the required density d, the yellow cubes are the voxels observed but with a density lower than required, and the red cubes are the voxels not seen by any sensor. The violet cubes (with arrows) are the position and orientation of the sensors, while the dark orange cube is the position of the virtual sensor and the blue cubes are the rest of the environment.

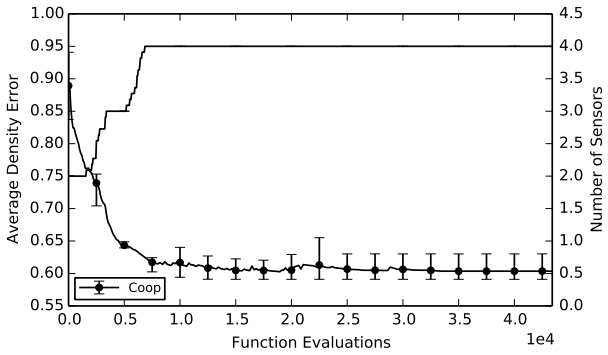


Figure 7. : Average results for the cooperative optimization algorithms in the ground experiment. The circle marked line reports the average density error with the maximum and minimum range presented by the error bars, while the unmarked line shows the average number of sensors.

The Huskies also carried a rotating Hokuyo URG-04LX as three-dimensional sensor.

Figure 7 presents the average error over 11 independent runs of the ground experiment. We observe that the average final error is 0.60 with standard deviation of 0.015. This high final error is mainly caused by the fixed height of the sensors. As a matter of fact, their height does not allow them to see the top and the interior of the stuck

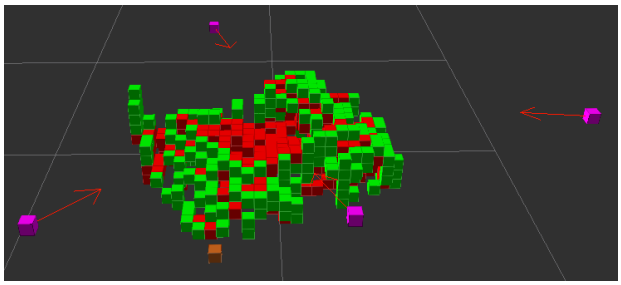


Figure 8. : Median visibility results for the ground experiments. The colour code is the same as in figure 5.

vehicle, preventing them to sense a greater number of surfaces in the desired bounding box. Figure 8 presents the visibility data of the median results. We observe that most of the side voxels are perfectly covered (green voxels), while the interior of the vehicle is responsible for most of the error. The final images taken from the four sensors of the median result are shown in figure 9, while the corresponding placement is shown in figure 10. This shows that the cooperative algorithm is able to cope with scenario where the available sensors cannot cover the entire target.

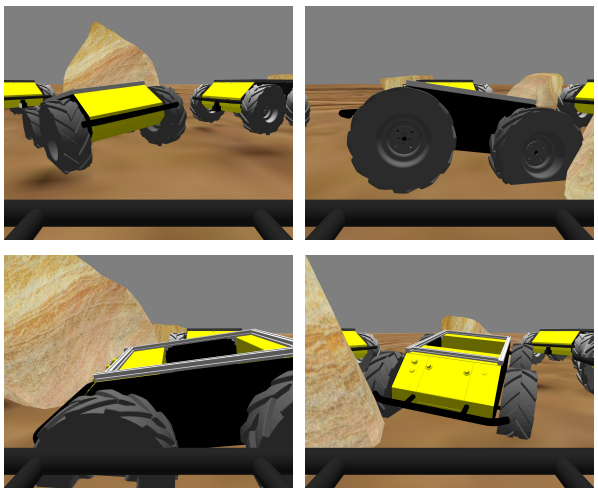


Figure 9. : Views acquired by the four Huskies of the median result in the stuck vehicle experiments. The Huskies are ordered left-right and top-down.

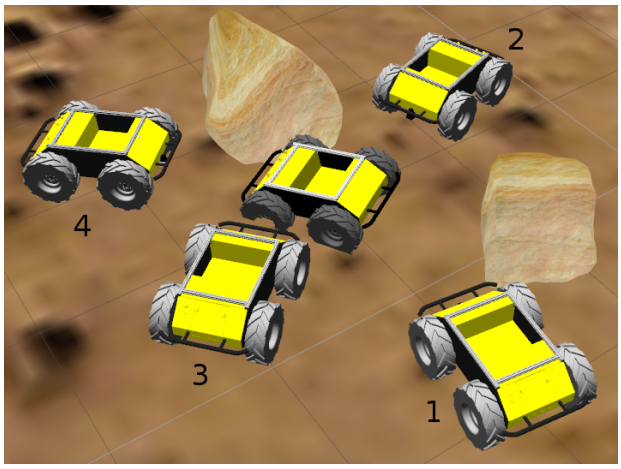


Figure 10. : Huskies final position in the median result of the stuck vehicle experiments. Numbers correspond to the views shown in figure 9.

6 Conclusion

This paper presented a framework to position mobile robotic sensors to observe desired parts of arbitrary three-dimensional environments. The framework has been proved useful in scenarios borrowed from the space domain and different type of sensors, where it outperforms classical global optimization algorithm and finds an appropriate number of sensors to observe the target. Future work will focus on the following:

- computing the pixel density in overlapping regions using super-resolution algorithms [7],
- using multiview virtual sensor to enable stereo on the target,
- and integrating the displacement cost in the cost function to reduce displacement and energy loss.

Acknowledgement

This work has been supported by grants from the FRQ-NT (Québec) and NSERC (Canada).

References

[1] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Proc. of Eurographics*, 1987.

[2] H.-G. Beyer and H.-G. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, (1):3–52, 2002.

[3] Y. Collette, N. Hansen, G. Pujol, D. Salazar Aponte, and R. Le Riche. On object-oriented programming of optimizers – Examples in Scilab. In P. Breitkopf and R. F. Coelho, editors, *Multidisciplinary Design Optimization in Computational Mechanics*, chapter 14, pages 527–565. Wiley, 2010.

[4] C. Connoly. The determination of next best views. In *Proc. of the International Conference on Robotics and Automation*, pages 432–435, 1985.

[5] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

[6] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[7] S. Farsiu, R. L. Robinson, M. Elad, and Milanfar. Fast and robust multi-frame super-resolution. *IEEE Transaction on Image Processing*, 13(10): 1327–1344, 2004.

[8] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research, Machine Learning Open Source Software*, 2012.

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[10] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *Proc. of the International Conference on Intelligent Robots and Systems*, 2004.

[11] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[12] A. Nüchter, H. Surmann, and J. Hertzberg. Planning robot motion for 3D digitalization of indoor environments. In *Proc. of the International Conference on Advanced Robotics*, pages 222–227, 2003.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[14] R. Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(10): 1016–1030, 1999.

[15] M. A. Potter and K. A. De Jong. Cooperative coevolution : An architecture for evolving co-adapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2001.

[16] 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.

[17] M. Schwager, D. Rus, and J. J. Slotine. Unifying geometric, probabilistic, and potential field approachs to multi-robot deployment. *International Journal of Robotics Research*, 30(3):371–383, 2011.

[18] G. H. Tarbox and Gottschlich. Planning for complete sensor coverage in inspection. *Computer Vision and Image Understanding*, 61(1):84–111, 1995.

[19] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.