Coverage path planning based on waypoint optimization, with evolutionary algorithms.

DAVID STRUBEL

N° 1

THÈSE présentée par

DAVID STRUBEL

pour obtenir le

Grade de Docteur de
l'Université de Bourgogne

Spécialité : **Instrumentation et informatique de l'image**

Coverage path planning based on waypoint optimization, with evolutionary algorithms.

Unité de Recherche :
Imagerie et Vision Artificielle
Vibot ERL CNRS 6000

Soutenue publiquement le 09 may 2019 devant le Jury composé de :

PASCAL VASSEUR	Rapporteur	Professor at LITIS, UNIROUEN, UNIHAVRE, INSA, 76000 Rouen, France
XEVI CUFÍ	Rapporteur	Professor at VICOROB, University of Girona, 17071 Girona, Spain
SAMIA AINOUZ	Examinateur	Assistant Professor at LITIS, UNIROUEN, UNIHAVRE, INSA, 76000 Rouen, France
FRANCK MARZANI	Examinateur	Professor at ImViA ,University of Burgundy, 21000 Dijon, France
DAVID FOFI	Directeur de thèse	Professor at ImViA ,University of Burgundy, 71200 Le Creusot, France
OLIVIER MOREL	Co-encadrant de thèse	Assistant Professor at ImViA ,University of Burgundy, 71200 Le Creusot, France

ACKNOWLEDGEMENT

It was a long and somewhat laborious process which could not have been possible without the help of many people whom I have to thank. Of course, among the first people to thank are my supervisors, without whom all this would not have been possible.

David Fofi who always has good advices to solve the astronomical amount of problems that occurred during this thesis. Thanks for always being available, smiley and a precious support.
Olivier Morel whom I must thank for his investment and professionalism.
Naufal Bin Saad who was a valuable help in understanding the habits and customs of the UTP and Malaysia.

I would also like to thank the rapporteurs Professor Pascal Vasseur and Doctor Xeví Cufi assistant professor a as well as the members of the jury Doctor Samia Ainouz and Professor Franck Marzani.

I need to thanks a lot Alexandra Elbakyan for the creation of Sci-hub. Sci-hub offer at any researcher the ability to do their works and to share the humanity advance.

I also has to thanks many friend which help a lot for many thing:

Guillaume and Mojdeh for their friendship and good advices.
Aamir Shahzad my guide in the maze of UTP.
Annick Le Berre and Pierre Hary for their precious friendship and obstinacy teaching me a more appropriate English.
Cansen for his motivation and obstinacy in always wanting to do better.
Nirbhay and Pham for their kindness and support.
Tilman for introducing me to computer vision.

Some other people to thanks:
Mark N ; Nathan C; Nathan P; Abir ; Goh ; Rishu ; Marwan ; Tom son; Maguie ; Mamounia & Papili; Nathalie C; Desiré; O Aubreton; Bertuca family; Mrs Cabot;Fabrice M ;Marc B; Toufia; Ilyes; Zawawie ;

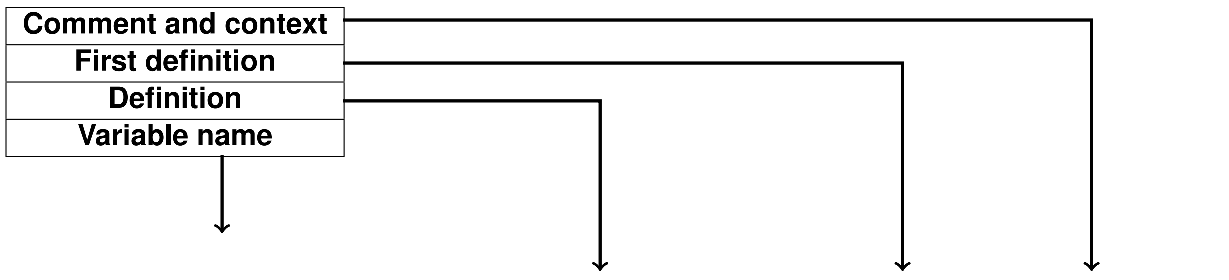
Pour conclure et en français; je tiens à remercier ma famille (Regis, Marie Christine, Charlotte et la petite zoé) pour leur soutiens constant dans cette épopée que fut cette thèse.

Contents

0.1/ ACRONYM LIST

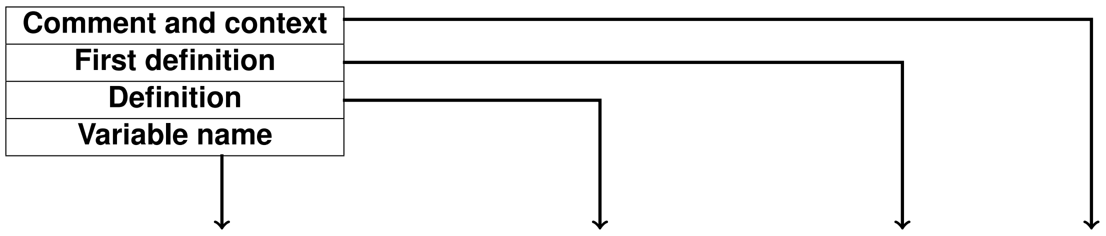
UAV - Unmounted Aerial Vehicle
PTZ - Pan Tilt Zoom
AGP - Art Gallery Problem
NP-Hard - Non Polynomial Hard
NP-Complete - Non Polynomial Complete
WSN - Wireless Sensor Network
WSAN - Wireless Sensor and Acutor Network
PSO - Particles Swarm Optimization
VSN - Visual Sensor Network
BIP - Binary Integer Programming
EA - Evolutionary Algorithm
SA - Simulated Annealing
CMA-ES - Covariance Matrix Adaptation Evolution Strategy
BFGS - Broyden-Fletcher-Goldfarb-Shanno (Quasi-Newton method)
MAGA - Multi-Agent Genetic Algorithm
HEA - Hybrid Evolutionary Algorithms
PCB - Printed Circuit Board
PI-PSO - Probability Inspired binary PSO
CR-PSO - Craziness based PSO
CPP - Coverage Path Planing
WRP - Watchman Route Problem
TSP - Travelling Salesman Problem
PNWCC - Novel Previous-Next Waypoints Coverage Constraint
DNA - DeoxyriboNucleic Acid
MOEA - Multi Objective Evolutionary Algorithms
MOP - Multi Objective Problem
SGA - Simple Genetic Algorithm
MOEA/D - Multi Objective Evolutionary Algorithms decomposed
NSGA - Non-dominated Sorting GA
NSGA-II - Non-dominated Sorting GA II
BMPGA - Bi-objective Multi Populations GA
QGA - Quantum-inspired GA
DoF - Degree of Freedom
RS - Random Selections
DoE - Design of Experiments
GT - Ground Truth
NW - Number of Waypoints
GAPSO - Genetic Algorithm Particles Swarm Optimization
FoV - Field of View

0.2/ EQUATION VARIABLES



k and k-coverage	Where k represent the number of cameras useful to cover some region of the area.	Sub-section :2.1.1	All
P	Polygon used to represent an art galery (room).	Sub-section :2.1.2.1	Only the Section 2.1.2
n	Number of vertices of P	Section :2.1.2.1	Only the Section 2.1.2
v_i	The i^{th} vertices of the polygon P	Section :2.1.2.1	Only the Section 2.1.2
x_i	The i^{th} guard (coordinate position in P)	Section :2.1.2.1	Only the Section 2.1.2
y	A point $y \in P$	Section :2.1.2.1	Only the Section 2.1.2
X	A set containig the minimum number of guards X useful to fully cover the polygons P	Section :2.1.2.1	Only the Section 2.1.2
g	The number of guards x needed to cover P such that we have a set $X = \{x_1 \ldots x_i \ldots x_g\}$	Section :2.1.2.1	Only the Section 2.1.2
x	Sensor position	Section :2.1.3	Only the Section 2.1.3
r	Sensor power radus	Section :2.1.3	Only the Section 2.1.3
$\vec{x}$	Input vector and possible solution	Section :sec:GeneralEAforn	Only the Section 3.2.2
$f(\vec{x})$	Cost function to evaluate the $\vec{x}$ answer	Section :3.2.2	Only the Section 3.2.2
$\vec{X}$	Global optimum solution	Section :3.2.2	Only the Section 3.2.2
E	Set of m constraints composed by e_j for the j^{th} constraint function	Section :3.2.2	Only the Section 3.2.2
$F(\vec{x})$	Cost fonction to include the $\vec{x}$ answer, with evaluate also the constraints cost	Section :4.1	All
G	An occupation grid of the area	Section :4.1	4.1
g_i	The i^{t}h point of the grid G should be covered by a camera	Section :4.1	All

Table 1: Equation notation I.

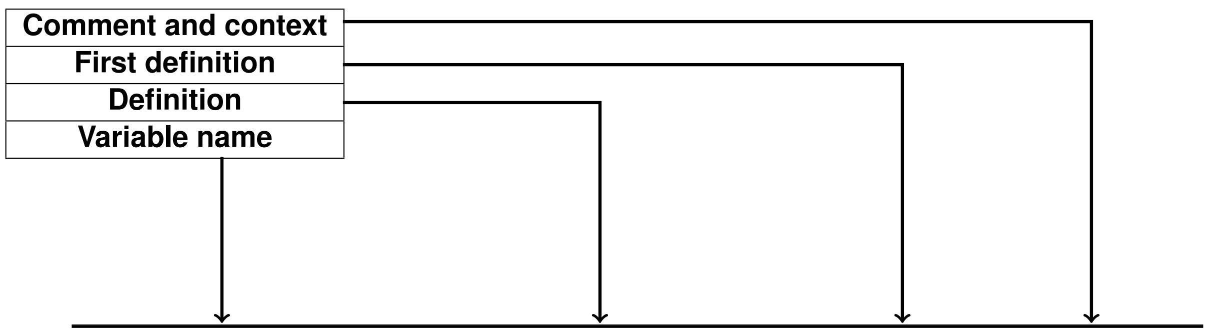


m	The number of points in the grid G	Section :4.1	All
B	The list of the points g_i from the grid G which are covered	Section :4.1	All
p_i	The weight of the point g_i on the grid G	Section :6.1.1	Only the Section 6.1.1
P	The list of p_i which contain the weighting of the area	Section :4.1	Only the Section 4.1
G'	The list of point g_i (grid similar then G) with the non-interesting zones removed	Section :4.1	The following section
U	The zones without interest to be covered noted the set U. $U \in G$	Section :4.1	Only the Section 4.1
(x, y, z)	Camera position	Section :4.2.1	All
(α, β, γ)	Three degrees of freedom for the camera orientation: the rotation in pan, tilt, and roll angles	Section :4.2.1	All
f	The focal lens	Section :4.2.1	Following sections
v	The input camera with the necessary camera properties	Section :4.2.1	All
Obj_l	The l^{th} object in the scene	Section :4.2.1	Only the Section 4.2.1
$f(...)$	The function to compute camera projection	Section :4.2.1	Following sections
$Wr \times Hr$	The size of the rectangle projection of the camera in width and height	Section :4.2.1	Following sections
$S_w \times S_h$	The size of the camera sensor in width and height	Section :4.2.1	Following sections
V	A solution. V contain all individual positions and orientations of the set of cameras for a predefined focal length, sensor size and related map depending on the problem	Sub-section :4.2.2	Only the Section 4.2.2
Vs	An acceptable solution which respect the set of constraint	Sub-section :4.2.2	Only the Section 4.2.2

Table 2: Equation notation II.

Variable name	Definition	First definition	Comment and context
W	Size of the boundaries of the area to cover in width	Sub-section :4.3.1	All
H	Size of the boundaries of the area to cover in height	Sub-section :4.3.1	All
ϵ	The *soft* constraint	Sub-section :4.3.2	Only the Sub-section 4.3.2
ε'	The *hard* constraint	Sub-section :4.3.2	Only the Sub-section 4.3.2
$C(Vs)$	The cost function to evaluate the area coverage which includes the constraints	Section :4.3.3	Following sections
n	The number of cameras in the network	Section :4.2.2	Following sections
N	The number of cameras in the network	Section :4.4	Only the Section 4.4 and 6.3.2
Sp	The search space size for one camera	Section :4.4	Only the Section 4.4 and 6.3.2
V_k	Velocity of a particle of the kth iteration	Section :5.1.1	Only the Section 5.1.1
P_i	Best solution of a particle	Section :5.1.1	Only the Section 5.1.1
P_g	The best solution from the swarm	Section :5.1.1	Only the Section 5.1.1
ω	The inertia of the swarm	Section :5.1.1	Only the Section 5.1.1
$b1$ and $b2$	Random value between 0 and ϕ_g for b_1 or ϕ_p for b_2	Section :5.1.1	Only the Section 5.1.1
X_k	The k^{th} particle which includes this own position at the current time	Section :5.1.1	Only the Section 5.1.1

Table 3: Equation notation III.



A_{room}	Area of the room (length $\times$ width)	Section :6.1.1	Only the Section 6.1.1
A_{Wall}	Obstacles area (length $\times$ width)	Section :6.1.1	Only the Section 6.1.1
A_{Cam}	Area covered by the camera in the maximum size of z	Section :6.1.1	Only the Section 6.1.1
NWayPoint	Number of waypoints	Section :6.1.1	Only the Section 6.1.1
Threshold Rate	Objective threshold rate	Section :6.1.1	Only the Section 6.1.1
S	A solution of waypoints set	Section :6.1.1	Only the Section 6.1.1
$evalCost$	Cost function	Section :6.1.1	Only the Section 6.1.1
α_i	An angle of the curve in the trajectory as in Figure 6.2	Section :6.1.2	Following sections
$Size(\alpha)$	The number of curves in all the trajectory	Section :6.1.2	Following sections
$Distance$	The distance of the path	Section :6.3.1	Following sections

Table 4: Equation notation IV.

1

Introduction

The project of self camera organization has been initiated by a partnership between the CISIR and the previously called Le2i and since January 2019 ImVia. The project aims to develop a smart system composed by camera mounted on UAVs. The system has to be available to self-organize and auto-adapt to a dynamic environment. Concretely the system should act as a base to perform cooperation work between UAVs, to detect security issues in a complex and dynamic environment. This initial project is ambitious and contains numerous technical locks. To be realistic, this project must be resized and the research have to be focused.

The project start with a basic idea to develop a smart system composed by a camera mounted on a UAV. The first step, is to manage the different positions of a UAV, inside the area to control. The goal here is to find the best viewpoints for the surveillance. To monitor correctly the area, the viewpoints selected must cover the quasi totality of the environment.
It is important to find the relevant waypoints and create a shorter path passing by all of them in order to efficiently control and monitor a given zone. Our concern here is to find the strategic positions for the waypoints in order to control the area despite the environmental constraints. The constraints can be various as discussed in details in the following sections (the shape of the area, the UAV limitations, the image quality requirements, are some example of constraints). Estimating efficiently the path plan with strategic waypoints poses is an important technical constraint to create a complete and autonomous system of smart cameras mounted on UAVs. So far, no optimal solution has been found. Nevertheless, some solutions have been applied (see Section 2) depending on different constraints, but all of those proposed in the literature are imperfect and can be upgraded. In order to propose a novel approach for estimating the best path planning for UAV under constraints, the proposed solutions are taking this complex problem and split it into sub-parts.

The first sub-part is the estimation of the position of each waypoints and the second part is the computation of a path going through all the waypoints previously deduced. Despite the apparent simplicity of the proposed solution, numerous problems appears such as the difficulty to find the optimal position for all the waypoints.
Estimating the best waypoint positions is finally similar to cameras positioning. The major part of this topic is focused on finding the best cameras positionning (or waypoints) which is an open challenge. The path planning passing by the waypoints is addressed with inclusion of the estimated number of waypoints. The results presented in this thesis were published for peer-review in international conferences such as URAI 2016 and IV 2017.

1.1/ Camera positioning

The first step of the problem is to solve the camera positioning, i.e. how to estimate an optimal viewpoint selection to ensure an acceptable coverage. It is known that an efficient camera positioning is a bottleneck in many applications, as for example in the video surveillance field [1, 2, 3, 4, 5], where an efficient camera positioning is essential to monitor correctly an area. The following section will deal with the questions:

- What is a good position and orientation for a camera (i.e. a good camera pose)?
- What are the purposes of the application requiring camera positioning?

These questions have already been investigated and some solutions have been proposed in the literature.

1.1.1/ Efficient pose in a camera network: challenges and objectives

The first point to address is the definition of camera poses (or set of cameras). In computer vision, the pose of a camera is composed of its position in space and orientation (or viewing direction), i.e. 3 translations and 3 rotations in a world coordinate frame. The second point to be addressed is how to define a ”good” or ”optimal” camera pose(s)?
To do so, it is essential to identify the purposes, tasks and priorities of the application:

- What is the final goal?
- What are the important features for the application (e.g. to track an object, to have a high-resolution mapping, etc.)?
- What are the shooting conditions and physical constraints?

All of these aspects will have an incidence on the definition and formalization of a ”good camera pose”. For instance, in [6], the purpose is to detect tags placed on people torso which forces the camera to be positioned at a certain height with a viewing direction almost parallel to the ground; on the contrarily, in [7], a camera is mounted on a UAV to monitor a vast outdoor area, which forces the viewing direction almost perpendicular to the ground.

These two articles share the same objective, i.e. to get the best possible coverage of an area. But because of the constraints and secondary objectives (e.g. number of cameras, resolution, luminosity, tracking, etc.), it leads to different formulations and approaches.

The next sections of this thesis focuses on positioning the camera to maximize the viewing areas. The viewing area or coverage rate of the area is directly related to the estimated position of each camera and their orientation. To get the best coverage, it is essential to find the best pose for each camera, depending on the constraints and possible secondary goals.

The camera positioning for maximizing the coverage rate is an open challenge studied this last decade and still open. In the following chapters we will introduce the problem

and some of the most interesting solutions studied in the literature. The research of the last decades shows the complexity of the problem in its plurality and the solutions that have been proposed.
Thanks to the literature, the problem complexity and the multiplicity of constraints is an open problem. In fact it will reveal the importance of the constraints depending on the solution proposed and the importance of well defined constraints for the problem formulation and the solutions adopted.
Based on state of the art in the chapter 2, a precise problem formulation is proposed adapted to the UAVs constraints in the chapter 4 and the algorithms proposed will be explained in the chapter 3.
The last chapters of this thesis is dedicated to our proposed solution and the experimental results obtained by simulations of complex area; first for waypoints positioning in chapter 5, followed by the coverage path planning in the chapter 6.

2

STATE OF ART

Contents

The surveillance and control domain is a wide field of research which contains a lot of aspects such as: object tracking, object recognition, 3D mapping and area coverage among others. Our work focused specifically on the latter, i.e. finding a procedure which allows to capture a minimal number of images of a given area maximizing the coverage of the area. This can be achieved using a set of visual sensors. Many parameters have thus to be taken into account: size and shape of the area itself, field of view of the cameras, number of cameras (or views) to name a few. The following chapter will survey the different methods and techniques proposed in the literature to solve this problem. Keeping in mind that our final goal is to find out a technique that works both indoor and outdoor, flexible enough to handle two scenarios: (1) a set of cameras observing simultaneously the area; (2) a single camera moving along a pre-computed path to cover the area.

2.1/ Camera positioning

The first step for solving the area coverage problem is the camera positioning, i.e. how to estimate an optimal viewpoint selection to ensure an acceptable coverage. It is known that an efficient camera positioning is a bottleneck in many applications, as for example in the video surveillance field [1, 2, 3, 4, 5], where an efficient camera positioning is essential to monitor correctly an area. The following section will deal with the question of:

- What is a good position and orientation for a camera (i.e. a good camera pose)?
- What are the purposes of the application requiring camera positioning?

These questions have already been investigated and some solutions have been proposed in the literature.

- Object coverage:
  In Hoppe et al. [8] a good coverage is defined by the ability to have full 3D reconstruction of an object (in their 3 dimensions) with no occlusion. In this work, some prior knowledge on the object is exploited such as a rough surface description (mesh). The camera follows a trajectory ”around” the object and its viewing direction is oriented towards the center of the mesh (see Figure 2.1). Since it is a matter of covering the 3D surface of an object, in a next-best-view strategy, this application remains too far from the one we wish to implement. It is therefore barely applicable to our problem.

- Path to cover:
  The point here is to observe the entire trajectory commonly taken by users (car, pedestrians, . . .). When the area to cover is a known place, the main trajectory taken by the users can be estimated or extracted [10]. If the area to cover is a road, for instance, then the trajectory of the driver is known [11]. In this condition, the aim is to cover the common trajectory of the user as presented in [11, 10, 12, 9] (see the Figure 2.2). The path coverage is interesting due to the restricted area to cover: not an entire area, but only a path within a given area, that can be seen as a priority sub-area.

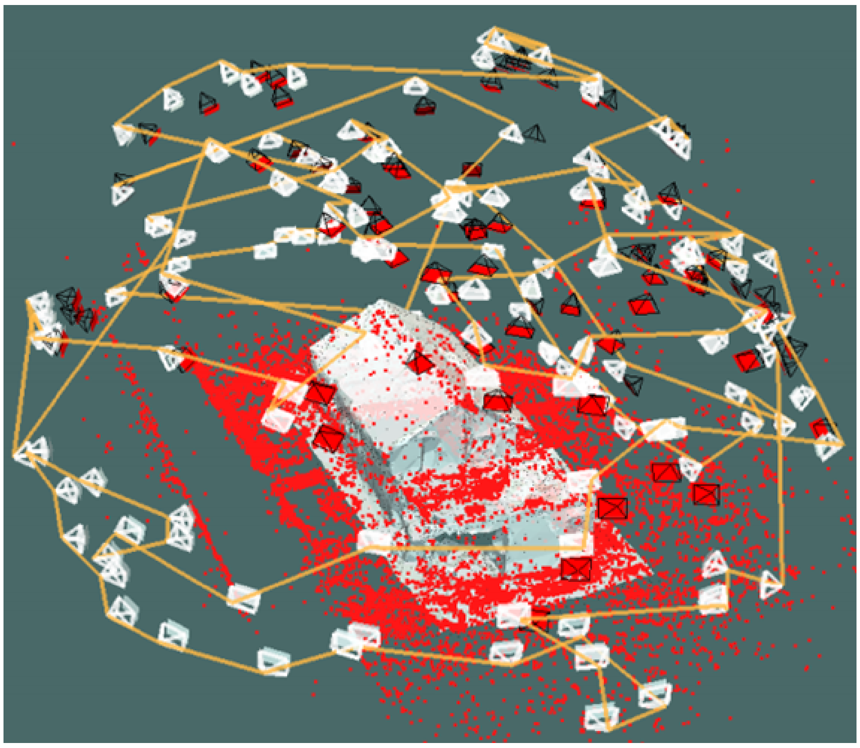


Figure 2.1: Full coverage of an object in the 3D space. The coverage is made by selecting a set of adapted waypoints. The coverage must be good enough to enable reconstruction of the 3D shape of the object without any occlusion. Result here are obtained by the method of Hoppe et al. [8].

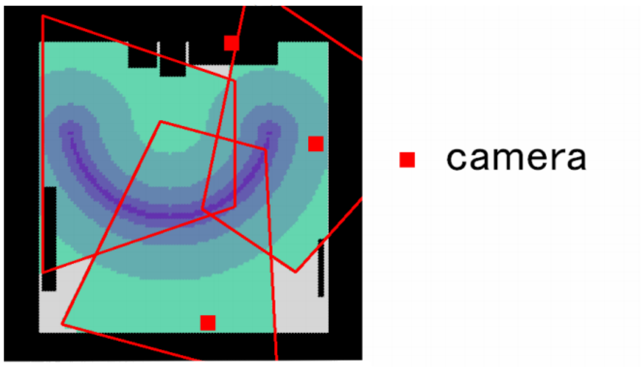


Figure 2.2: Illustration of ”path to cover” of Nikolaidis et al. [9]. The aim is to focus on covering a road (walking path) in a small room by using only 3 cameras.

- Coverage priority:
  A natural way of defining the coverage in a context of insufficient number of cameras, is to define as priorities. In [5, 13, 14], some pre-defined regions are set as ”priority” and called respectively ”region of interest”, ”crucial sub-area” (see Figure 2.3) and ”importance space weighting”. In the proposed solutions by [5, 13, 14], the camera poses are in priority affected to this specific and restricted

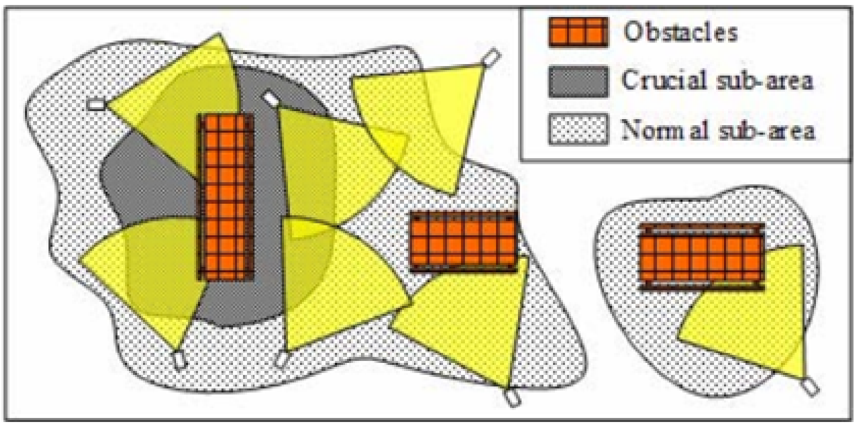


Figure 2.3: Map of an area to cover with crucial sub-area (region of interest), the normal sub-area and obstacle. This map is an example of area coverage introduced in Jiang et al. [13].

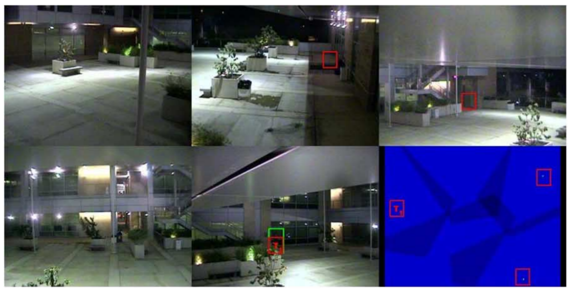
region which has the effect to neglect the other parts of the area.
If the environment is composed of some regions of interest, there should be also ”normal” sub-areas. These ”normal sub-areas” should be covered, but with lower priority. Furthermore, some sub-areas can be defined as ”no interest”, which mean ”not to be covered”. In [13, 14] for example, the obstacles are defined as ”no interest” regions with also the consequence to be occluding area. The idea is to keep a maximum of freedom in the camera network positioning and allow the system to handle local priority and constraints.

- Inside or outside area:
  Another important feature to define the coverage is related to indoor/outdoor scenes. The area to cover can be typically a room with walls (indoor). Each wall must be a considered as an obstacle occluding the camera field of view, which results in having to manage the visibility of the environment according to these obstacles and to the position of the cameras. For outdoor scenes, it is often necessary to take into account the size of the environment according to the reduced field of view of the camera. This has the effect of increasing the number of required cameras (or views) and leveraging the combinatory.

2.1.1/ Additional constraints

The common points in all the examples discussed is the aim of maximizing the coverage rate. The positioning of the cameras, and its effects on the coverage rate, is thus constrained by the application itself, the context and the type of observed scenes.
Of course, additional constraints can have also a significant impact on the camera pose. Some of the more common constraints found in the literature are listed below:

- The numbers of cameras:
  In many cases, the number of cameras used for the coverage should be minimised such as in [4, 14, 6]. Limiting the number of cameras is primordial to decrease


(a) Initial coverage for 5 cameras dedicated to detect the input of target.

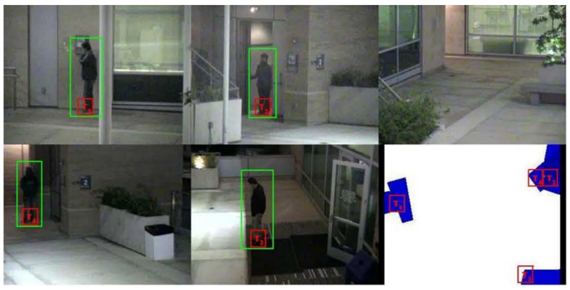

(b) The covered area when the objective have to track several targets.

Figure 2.4: Illustration of an covered area for tracking target. Experiment from Ding et al. [3]

the computation time and the bandwidth. It also reduces the cost of the setup [15]. Reducing the number of cameras and optimizing their poses to get an optimal coverage are closely related tasks, not necessarily competing. Too few cameras can shrink the coverage rate by leaving black-holes in some areas of the scene. Too many cameras can result in too much overlaps and unnecessary redundancies.

- Object tracking:
  Constraints can arise by the objective of detecting and localising a given target [3, 2, 16, 17, 18, 6]. In such a case, camera poses must be estimated in order to track one or more targets, and possibly, be dynamically adapted. These applications very often requires an adaptation of the camera orientation (viewing direction) more than its position. This is the reason why previous work used PTZ cameras (Pan, Tilt and Zoom) as in [3, 19, 2] (see Figure [3]). Keeping a full area covered and at the same time tracking efficiently one or more targets can be contradictory. The solution is then the result of a trade-off between coverage and tracking, as in [3] and [19]. In Liu et al. [19], target tracking in a wide area is decomposed in two steps: detection and localization. Each of these steps is performed independently on each camera. Area coverage is essential to detect the targets, less for localization as the priority is, in this step, to track a target previously detected within the covered area. In the entire camera network used, one camera may be in detection mode while another is in location mode. Obviously, by adding target tracking as a constraint, camera poses and coverage are usually less efficient because of the sub set of cameras assigned to the tracking.

- Luminosity and environmental setup:
  Intrinsic image quality is also a constraint that can guide area coverage. The quality of an image can result in sufficient brightness or an almost uniformly distributed histogram. In other words, the captured images must be such that they guarantee a usable signal. For example, Reddy et al. [20] addressed first the coverage problem of a complex area and in second time, target localization. In order to decide which target must be tracked, the quality of the image is taken into account to avoid dark areas where the target is hardly detectable. In this case, the tracking and coverage trade-off discussed in the previous paragraph is ruled by the image quality.

- Energetic cost:
  Authors suggests to estimate camera positioning or path planning by minimizing a cost function that represents the energy consumption, such as in [19, 21]. For instance, in Lui et al. [19], the objective is to cover most of an area to detect whether a target is entered in the scene or not. Secondly, the target is tracked by smart and autonomous cameras of the network. The camera set is randomly distributed in the area and the coverage problem represents, in this case, the selection of the best cameras in order to detect the target. The selection of the cameras is estimated by both maximizing the area coverage and minimizing the energy consumption. The consumption can be consequently reduced by restricting the number of cameras set in detection mode. Indeed, the cameras are more or less power-consuming depending on the activated mode (which can be ”detection”, ”tracking” or ”sleep”). If we consider now path planning, the energy cost can be represented by the distance between two cameras or views and energy minimization is equivalent to finding the shortest path [22, 23].

- Multi coverage:
  Among the numerous possible constraints, the multi-coverage is interesting (as for example in [24, 4, 25, 26, 27]). It can be seen as a coverage problem where one or a few specific sub-areas must be covered by a minimum of k cameras at the same time, that is why it is also called k-coverage. Multi-coverage does not necessarily mean priority: a sub-area which have to be covered by several cameras is not necessarily a sub-area which must be covered in priority (more details in Section 4.1.1.5). However, mobilizing multiple cameras in a given sub-area means that fewer cameras can be used to cover the rest of the overall area. This effect can be compensated by adding cameras, if allowed. Contrarily, full-coverage of the area and k-coverage of some sub-areas will conflict and lead to a trade-off.

- Resolution:
  In order to keep or increase the quality of the captured images, a minimum resolution threshold or value can be fixed and used as a constraint [10, 20, 14, 25, 28]. The focal length, the size of the pixel grid or the camera-target distance can serve as a measure for the resolution. However, in many applications, it is easier to adapt the distance than changing the focal length (which can be fixed and affect the calibration) or, of course, changing the size of the pixel grid. In most of the previously cited works, the distance from the target to the camera along the optical axis is therefore used as a measure for the resolution.

> Full coverage will tend to move the cameras at the farther distance (or higher elevation) in order to maximize the field of view. Resolution constraint will impose the cameras to be positioned at a certain range of distance or elevation. Full coverage and resolution constraint will lead to a trade-off between guaranteeing most of the area to be covered and a sufficient image resolution. The trade-off is particularly beneficial when the number of cameras is more important than the optimal need, in this case, the distance will be reduced and the resolution subsequently increased.
> In [20] the problem has been formalized by using a Gaussian function in order to define the proper distance between the cameras and the target to keep an acceptable resolution for the application.
> Here, the depth of view is used to define the range of distances. The focus point and the aperture of the camera will define the optimal distance and range in which the target is optimally focused [29]. Constraining the positioning with the depth of view can be seen somehow as a constraint on the resolution itself.

The constraints are numerous and varied, we just introduced a few of them which seems interesting to us and related to our work. Among them, some are closely related and can be interconnected, they can even be combined as in [20], where the targets coverage, luminosity, resolution, are all associated to find the best camera positions maximizing the area coverage and the tracking target with good visibility condition.
One interesting point to study is the impact of these constraints on the full coverage itself as we have seen that they introduce a trade-off between their own particular goal and the main goal of area coverage but also between themselves. No need to say thus, that the constraints have to be chosen carefully and accordingly weighted as in any multi-objective problems.

2.1.2/ Art gallery problem

The Art Gallery Problem (AGP) is a theoretical and historical problem closely related to the camera positioning. The AGP is commonly cited in the literature as a source of the problem for camera positioning (for full coverage). It is also commonly used to estimate the complexity of the task (notably due to the room shape) and as starting point to find an appropriate solution. The problem of cameras positioning can be formulated as the AGP paradigm (as example [30, 31, 24]). For these reasons, the AGP has to be well understood before going further.

In the following sections a definition of the AGP is given with a brief history. After a more general introduction to AGP, we will describe some interesting solutions found in the literature and discuss their limitations.

2.1.2.1/ Definition of the paradigm

The art gallery problem is a geometrical problem introduced by Victor Klee in 1973. The problem was the estimation of the number and the position of useful guards to cover an art gallery. The particularity of an art gallery is the complexity of the room shape, with many walls to place the paintings. The shape complexity of the room make the estimation of guards number even more difficult.
In order to formulate properly the problem, the room is assimilated as a polygon P, com-

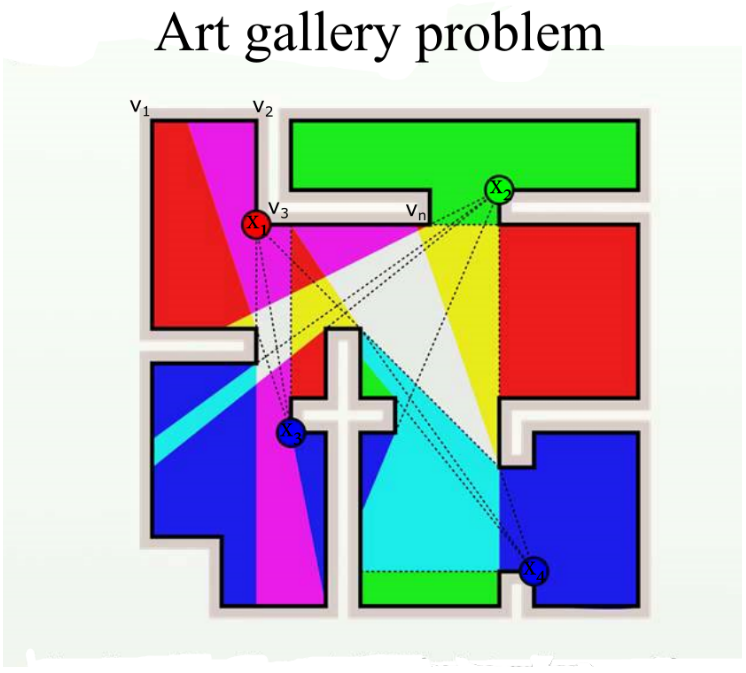


Figure 2.5: Illustration of the AGP. The gallery is covered by 4 guards ($x_1; ...; x_4$) for a polygon composed by n vertice ($v_1; ...; v_n$).

posed of n vertices ($v_1; v_2; \ldots v_n$). The vertices are linked by n edges ($v_1v_2; \ldots; v_{n-1}v_n$) to make the shape of the Polygon P (or room).

A guard x is inside the room $x \in P$. A guard x can cover or see any point $y \in P$ if the segment xy is not intersected by one of the boundaries (walls) of the polygon P, in order to have $xy \subseteq P$. The polygon P is considered as fully covered when for any position of the point y in the polygon, at least one guard can see it .
A guard x can have a $360°$ field of view to cover all around him, with no depth of field limitation (except the wall obstacles). Clearly that means the guard can see and monitor the entire room from one side to the other side if there is no obstacles around to occlude its vision. For example, if the shape of the room is a triangle, quadrilateral or another simple polygon, at any position taken by one guard, this guard can monitor all the area despite the size of the art gallery (see Figure 2.5).

When the polygon is more complex, it is necessary to estimate the minimum number g of guards x and the position of the guards in polygon P. The set of minimum number of guards are listed in X. Where X contains the useful x to fully cover the polygons P, with g the minimums number of guard in order to have a set of points $X = \{x_1 \ldots x_i \ldots .x_g\}$. So that every point y in P are cover by at least on guard x of the set of X.
The AGP in addition to estimating the numbers of guards, is also interested in finding the optimal position of this restricted number of guards. These two questions can be solved at the same time by using one of the solutions proposed.

2.1.2.2/ Solutions

The advances on the AGP since its formulation in 1973 are numerous. The following paragraphs present some of the main proposals and contributions..
The first, and one of the most important, is the proof given by Chvàtal in 1975 [30]. The proposed polygon must be covered by a minimum of guards, the proof of Chvàtal links the minimum number of guards g to the number of vertices n. A polygon composed by n vertices need in the worst case a minimum number of guards equal at $n/3$. The Chvàtal proof is based on the triangulation of the polygon. The triangulation is made based on the vertices of the polygon.
The proof given by Chvàtal is also confirmed by the work of Fisk few years later (1978). The work of Fisk is also based on triangulation and colouring node. It is probably the easiest way to understand and also give a solution to estimate the pose of each guard. It is recommended to begin by the Fisk proof before the Chvàtal one, despite the chronology order as it is recommended in [32]. The book of O'Rourke et al. [32] is an early work about the AGP with the formulation, proofs and advancement of the field clearly explained.
After the important work of Chvàtal which enable the estimation of the minimum number of guards in the worst case consequently fix a limit to the AGP. Thus, the research has been oriented toward the optimal guard positioning. The goal is to find the best algorithms to solve the AGP for all kind of polygon while reducing the complexity in time (reducing the $O(...)$).
In perspective, the work of Toussaint and Avis (in 1981) is the reference and propose a solution working in $O(nlogn)$. This work has been followed and upgraded until the solution of Couto, Resend and Souza 2011 [33]. The solution finally proposed reduce the complexity to $O(n^3)$ in the worst case.

2.1.2.3/ Limit of AGP and camera coverage relation

The AGP can be considered as a reduction of the best camera poses estimation to maximize the coverage of a complex area. Based on the algorithms developed to solve the AGP and the strong relation between AGP and the cameras positioning (for maximum coverage problem), loigically, some proposed algorithms have been developped. They try to extend the AGP to the problem of camera positioning as example in [34, 20, 28].
The algorithms developed for AGP cannot be applied directly on the problem of cameras positioning for maximum coverage. The main reasons are the cameras limitations such as the field of view and the depth of field (see [15, 35]). The cameras limitations makes unreadable the algorithm proposed to solve the AGP; because the AGP is considering the guard with no limitations for the depth of field and field of view. Due to these differences, the geometric model of AGP may not be applicable for perspective cameras.

Also another reason that could make the AGP solution not applicable for the camera positioning is the diversity of cameras in the same system (as perspective cameras with different focal length or associate to non-perspective cameras such as omnidirectional camera). The AGP may have many guards, they are all interchangeable. The interchangeability is due to the guards ability (or skill) to monitor the area. Contrarily, it is possible for a camera network to have different kind of cameras with different lenses. Finally the perfect assumption for the AGP formulation create important weakness when it is time to replace the guards by cameras. These weaknesses make the algorithms form

AGP not adapted for our problem, as it is shown in [9, 14].

However, some part of the AGP formulation and especially some proofs are still of importance, as the proof of Chvàtal [30] or the NP-hard complexity proof. In fact, the AGP formulation as a NP-hard problems is available in the book of O'Rourke section 9.2[32]. NP-hard means that the problem cannot be solved in a deterministic manner with a reasonable time. In order to demonstrate the AGP NP-hard, the first part to be considered is the reduction of the problem to another well-known problem for this complexity. The relation is made by reducing the AGP with a polygon composed by holes to another standard problem: in the demonstration, the 3SAT is used to be exact. The 3SAT is a restriction of the SAT problem with atleast 3 literals in each clause (example of 1 clause with 3 literals $(x \vee \neg y \wedge z)$) , the SAT has to satisfy a boolean expression written with only AND $\wedge$, OR $\vee$, NOT $\neg$) by assigning the appropriate value to the variable of the boolean expression (True or False).
Once the AGP is reduced to 3SAT (see [36]) and because 3SAT is an NP-complete problem, the AGP is also considered as NP-complete or NP-hard but under conditions: the room have to be composed of holes. In addition, another work of Lee and Lin 1986 proved the complexity of AGP *without hole* also by reducing the AGP to another well known problem (for more explication see the book of O'Rourke section 9.3 [32]). Despite the limit of AGP, numerous articles are based on it to formulate the problem as in [28, 31]. For example in [28] a similar approach to AGP is used in order to estimate the occluded regions. As explained earlier the cameras positioning can be reduced as an AGP [31], notably by removing most of the constraints due to the camera properties (as depth of field and field of views).

In the literature numerous articles uses the AGP as reference to explain the complexity of the problem of camera positioning, for example [37, 30, 24, 4]. The problem of camera positioning for maximum coverage is at least NP-hard or NP-complete as stated above. The complexity of the problem will have an impact on the solution tested to solve and optimize it.
Another impact of AGP in the problem of camera positioning is the shape of the rooms. In [35, 14, 20, 28], the shape of the room to cover is similar to the definition of an art gallery (in AGP). The art gallery room is a complex polygon composed by many vertices which may occlude the view (as explain in Section 2.1.2.1 and as in the Figure 2.5). This phenomenon can be imputed to the link made between the number of vertices and the effective number of guards to cover it. In addition, the occlusion formulation made for the AGP is commonly used. The occlusion in AGP is defined by a segment $xy \not\subset P$ where $x \in P$ is the guard position $y \in P$ is a point in the room P. Moreover the usage of a complex room inspired by AGP is therefore a good choice in order to verify the effectiveness of the algorithms developed for the cameras positioning in a complex environment.

The AGP can be interpreted as the historical source of the cameras positioning and give a beginning of a solution about the problem, its formulation and its complexity. Despite that, the AGP is not the only source to refer about the cameras positioning for maximizing the coverage. Some clue and algorithms can be found in other related fields.

2.1.3/ Wireless sensor network

The Wireless Sensor Network (WSN) can be, as the AGP, considered like an inspiration for the problem of cameras positioning to maximize the coverage. The WSN is an active field of research related in many aspects to camera positioning. These sections are focused on the WSN and its relation with the cameras positioning.

What is the Wireless Sensor Network (WSN)?

The WSN is a distributed network of sensors or in some cases actuators. It can also be called WSAN (Wireless Sensors and Actuators Network). Each sensor of the network acts as a relay for the information to the rest of the network. The sensors are at the same time the nodes and the relay for the network. The node has the purpose to transmit the information to the other node. The information can be centralized or not :

- When the system is centralized, the information has to be transmitted node to node up to the centralized agent. The computation and the decision about the network is taken by the centralized agent before the transmission back to the node.
- Otherwise, the node is the sensor, collects information and decides to communicate with the other nodes depending on the situation (example when the target is detected in their field). The node manage itself or with its neighborhood the information and computation before reacting in consequences.

The information collected by the sensors are vast depending on the final application and the capacities of the sensors ability. The WSN is used in different fields for various applications such as Telecom with antenna positioning [38], military surveillance field [19, 39], airport surveillance [40], video surveillance and tracking [19], environmental monitoring [21] etc. Consequently, the sensors is able to collect numerous type of informations depending the need: temperature, movements, images, song and also some actuator can be used as radio frequency for example.
The applications of WSN are wide, especially since the WSN can be exploited in many applicative fields. The WSN tries to optimize a network of sensors in different aspects, for example [17] focus on an adapted architecture efficient enough for data transfer (here the data are images) ; otherwise in [18] the WSN are dedicated to adapt the network around static nodes and energetic resources in order to keep the network connected.
In our case, the most interesting aspect of the WSN, is the coverage of an area with constraints. The other disciplines of the WSN such as the network optimization will not be addressed in the following document. Only the problem of coverage is studied: the other discipline are not considered as the first or main objective but can be some secondary objectives after the problem of coverage which has to be taken in account for the optimization.

2.1.3.1/ Sensor at 360

The WSN refers commonly to sensors or actuators with no restriction in the view angle, considered to own a $360°$ field of view. The sensor field of view can be represented as a circle in a 2D plan like in [41, 26, 42] (as illustrated in the Figure 2.6) and in some case a spherical shape for the 3D environment examples in [27, 38].
Each sensor have a position x in the area and a power range. From the sensor power, the radius r of the circle is deduced from x the center (see Figure 2.6). This circle gives the area covered by a sensor in the simplest case. The simplest case correspond to a flat area without any obstacle as shown in [41, 26] (see the Figure 2.7). In addition, more

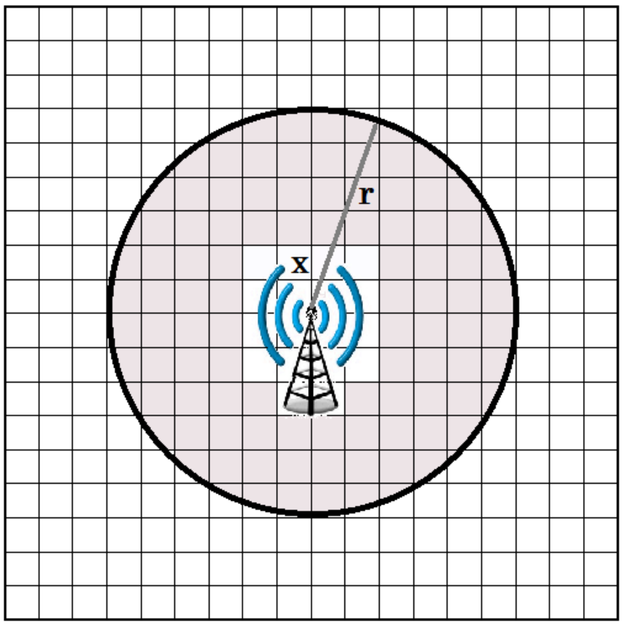


Figure 2.6: An omnidirectional sensor centred on x, with a radius r for the range.

complex solutions can be used. A more complex solution but also more realistic is the one proposed by Wang et al. [38] where the obstacles reflief is taken into account in the process. In Zhang et al. [26], more complex model have been developed where, despite of a flat ground without obstacle, each sensor is represented by a perception radius and a communication radius. The communication radius a bit bigger than the perception radius. These two radius with the same center corresponds, first to area covered by one antenna (sensors for the perception) and the second radius to the distance of emission/reception of the data (actuators for the transition). To have an efficient coverage of the area, the antennas must be placed in order to have connections with other antenna but without too much overlap of the perceptive field.

The solution proposed in order to optimize the positioning of the WSN for a circular sensors can be numerous. Mostly two different ways are applied for the sensor which have circular angle of view or spherical.

- The first solution use a heuristic based on geometry construction as in Medhi et al. [27]. This approach gives a good coverage solution but is usually greedy and can be quickly limited due to an important number of sensors required. Moreover if some external constraints are added. For example in Zhang et al. [26] the greedy solution was tested and optimized by using a ”partition and shifting” strategy in order to upgrade the result. The limit of this solution is this greediness of the process. In addition, it is not applicable to the problem of cameras positioning due to the reduced field of view of a camera.

- The second solution intends to find an efficient and quick solution to optimize initial random position, for each sensor of the network. This solution includes many differ-

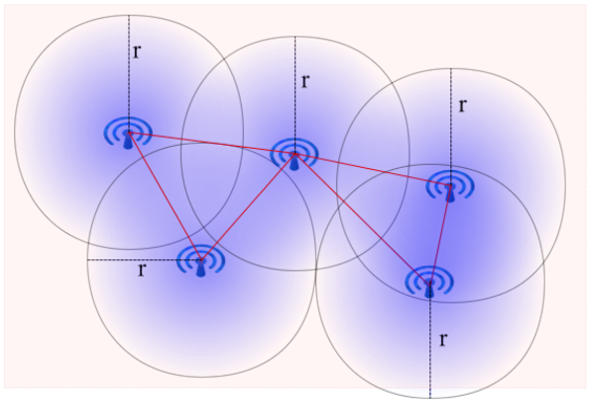


Figure 2.7: Illustration fo a simple wireless sensor network with omnidirectional sensor centred on x, with a radius r.

ent families of algorithms focused on optimization. Among the family of algorithms, the evolutionary algorithms (disused in detailed in Chapter 3) is commonly used as in [41, 38], and [42].
These solutions propose to optimize the position in order to maximize the coverage depending on constraints. The method of optimization have to be adapted to the problem. In Chakrabarty et al. [42] the integer linear programming is chosen in order to maximize the coverage with two types of sensor. One standard with a smaller area coverage but with a smaller cost ($100m$ radius for 150$) and the other sensors can cover a wider region ($200m$ radius for 200$) . The objective is to cover the region while reducing the financial cost using integer linear programming adapted to the problem of coverage optimization.
In [38] and [41] the solutions proposed are based on two different evolutionary algorithms in order to optimize the sensors positioning. In Kulkarni et al. [41] the camera positioning with a multi-coverage is solved by using an evolutionary algorithm called Particles Swarm Optimization (PSO). The objective is to optimize the position of the sensor in order to have an efficient coverage of the area and also enough redundancy to keep the network workable if one or few sensors fail. In Wang et al. [38], an evolutionary algorithms is also used to optimize the position of antennas. The objectives in this paper is to give the best coverage of an area with taking in count the relief of the area. The relief make the coverage estimation of each sensor even more complex and costly in terms of time computation. The genetic algorithm is used in order to find quickly a position for each antenna of the network.

Among the solutions proposed, the second, based on optimization of a set of sensors position depending on the constraints is the most intersting and the more flexible to the addition of new additional constraints (and secondary objectives).
The following sections is dedicated to validate if these solutions are applicable to the problem of positioning a camera set despite the camera constraints.

2.1.3.2/ Visual sensors

Positioning visual sensor can be, by numerous aspect, closely related to AGP and the WSN. These previous methods must be even more constrained to enable the usage of

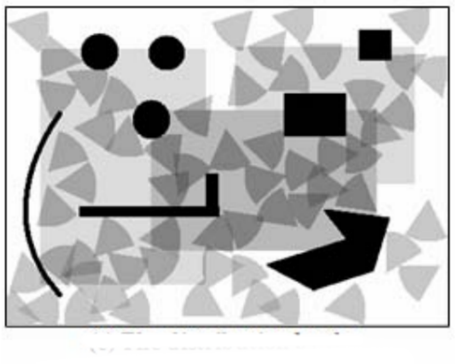


Figure 2.8: Illustration of the area coverage with visual sensors. The area is covered at 47%. Experiment form Jiang et al. [13]

a perspective camera due to the limited field of view. Visual sensors embed different types of cameras and modalities, even though the most commonly used and studied is the perspective camera such as in [24, 26, 29, 21, 13] (see Figure 2.8). These constraints makes the camera positioning relatively more complex, as it is necessary from AGP point of view (see Section 2.1.2) to pass from guards to cameras. Contrarily, WSN already takes into account the limited depth of field which makes it more suited to perspective cameras.

Heuristic-based solutions applied to omnidirectional sensors cannot be deployed for perspective cameras (which would require the definition of new heuristics and not only an adaptation), however optimization-based solutions can be adapted to it by only adjusting the constraints. The following sections will discuss such kind of techniques and solutions.

2.2/ Solutions not based on evolutionary methods

The algorithms used to estimate the pose of a camera set in order to maximize the coverage are numerous. Two main classes can be defined: the first one, which is called ”constructive”, is a step-by-step positioning of the cameras, one after the other; the second one comprises the optimization-based solutions. Most of these approaches fall under either the AGP paradigm or the WSN paradigm, or even a combination of both.

2.2.1/ The constructive solution

The camera positioning can be done by a progressive construction, i.e. a deterministic method is applied to locate iteratively one camera after another or to adjust their positions based on an initial set-up.
For instance, in Liu et al. [19], a constructive solution is applied in order to select the smart cameras of a network. Each smart camera is a node of the network transmitting information and images. The smart cameras are fully autonomous in terms of energy and decision-making (no central master). The nodes can be set to three different modes:

- Sleepy mode. It is used in order to save the energy consumption. The camera is turned off which means no computation tasks are performed but the network is listened at regular intervals to wait for the wake-up call.

- Detection mode. In this mode, the camera is turned on, but with a low frame rate. Just a few computations are done to detect if a target enters the field of view. Some information may be transmitted by the network. This mode consumes more energy than the previous one, however the smart cameras can still stay in this mode for a long time.

- Tracking mode. This mode is the more active, thus the more energy consumer. The camera is turned on with a high frame rate and numerous computations are done to track and localize the targets. Also more informations have to be transmitted by and to the network. Localizing and tracking a target is a collaborative and distributed task between several smart cameras of the network.

The objectives in [19] are multiple, depending on the state of the cameras. The most interesting for our application is to keep under control the area as long as possible, for target detection. Numerous smart cameras are randomly distributed in the area (as an air-drop in a battlefield) and the aim would be to select the minimal number of cameras which allows to maximize the coverage by setting them in detection mode. The solution proposed by Liu et al. in [19] rely in the usage of a constructive algorithm. The network of cameras self-organize itself after a ”discussion”. Also each smart camera is able to estimate precisely its own localization: this is a key information to select the cameras to set up in detection mode. The solution proposed is directly inspired by the distributed network communication protocol.
All the sensors are in sleepy mode at first. The cameras wakes up regularly and send a call at the neighborhood: if they receive no reply, this means no other camera is awake around and the camera switches to the detection mode. If a few replies are received (the threshold must be set up), this means that the required density of the camera around is not reached yet. Thus the camera switches or keeps on detection mode. Otherwise the camera switches back on sleepy mode until the next wake-up (in this case, the density threshold is reached). Each smart camera follows this protocol but the sleeping time is inherent to each camera to avoid they all wake-up simultaneously. After a certain time, the network is well enough organized to cover the area in detection mode. The area is considered covered when the density of camera is good enough.
The method introduced by Liu et al. in [19] is efficient. The solution proposed has the advantage to work in wide areas and to be dynamic. For example, if a camera does not have any more power, the network will self-reconfigure. But it suffers also from some drawbacks. First, it requires a high number of cameras with some of them being ”useless” because of the sleepy mode. Also this method is really dependent on the network communication and the capability to localize accurately each camera. Finally, extracting a sub-set of cameras among a randomly distributed network (in which the cameras are static) does not give a sufficiently accurate localization to allow a appropriately satisfying coverage (see Figure 2.9).

Another cameras pose estimation by construction is proposed by Höster et al. [14]. The solutions proposed is based on a greedy search heuristic. The objective is to find the positions and orientations for a camera set with a fixed pan, in the environment inspired

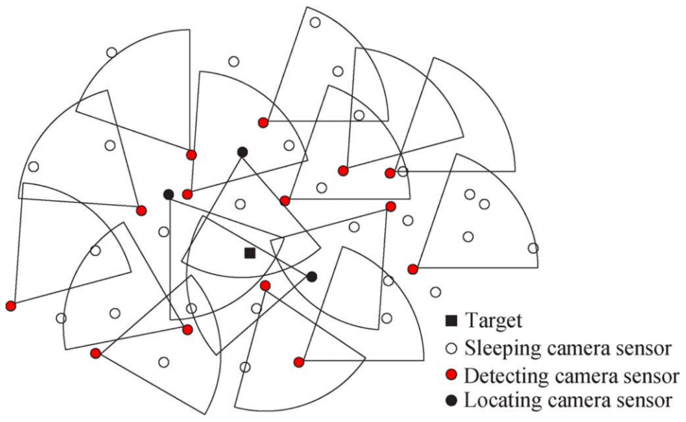


Figure 2.9: Illustration of the area coverage with smart cameras and a target tracking objective. The smart cameras can be in 3 modes (sleepy, detecting , locating). Experiment made in Liu et al. [19]

by the AGP.
In [14], a first greedy search solution has been presented before another algorithm which extend the greedy solution. The algorithms developed in [14] is called Dual Sampling.
The dual sampling is an incremental method. First step, is the initialization of the position for all the cameras. A random initialization for the position and the orientation must be appropriate.
Second step, is the selection of one point of the area that is not covered yet. The area is discretized by several points, with each point that must be covered by at least one camera. Around the selected point, several positions and orientations are tested for the cameras nearby. The possible position are obtained by sampling the area around the point to cover. Finally the best cameras position and orientation is kept. The best cameras positions and orientations depends on the number of other points globally covered in the area. The second step is repeated and the set of uncovered control points are reduced at each iteration. This procedure is applied until the stopping criterion is reached. Which means, enough points of the area are covered.
The constructive solution have some inconvenient, notably in terms of efficiency. Indeed this solution is limited by the number of cameras and size of the area due to the exponential difficulty.

In Nikolaidis et al. [9], the cameras placement is studied to cover a basic mobile robot trajectory. The trajectory of the mobile robot is modelled as the regions of interest, with a gradually decreasing interested from the trajectory center.
The solution applied in [9] lie in performing a local optimization one camera after another with the “steepest decent method” (related to the gradient decent). If this local optimization gives a better solution, the network of camera is modified, otherwise, the cameras stays at the same place. This operation is repeated until the convergent arrangement is obtained or if no more upgrade can be found. The result presented in the experiment done by Nikolaidis in [9] are intresting despite the simplicity of the area and the very small amount of cameras used (maximum four). The main limitation is due to the number of steps required to optimize independently each camera of the network. Also a multitude of local optimization is not obviously the same or better than a global optimization.

Ma et al. [40] proposed a solution for the problem of finding the minimum camera barrier coverage. The objective of the camera barrier is to cover only the boundary of an area, to be able to detect target intrusion within the perimeter. The perimeter is relatively wide and can be considered at some point as an area to cover composed by a big hole in its center.
The region to cover is divided into numerous sub-regions which are inter-connected. Each sub-region must be “full-view covered”. The full-view covering is defined as follows: regardless of the direction in which the target is moving, there must be a camera to detect his/her face - in the case, for example, of a video surveillance of a building.
The proposed solution is based on constructive solution with adapted heuristic. The heuristic used is presented in details in [40]. The global idea is to divide the perimeter in different sub-regions and apply the method proposed to have a full-view coverage at each sub-region. Dividing the region into sub-regions allows to speed up the processing time of the algorithm.
This solution is well suited for visual coverage of the perimeter but much less for coverage of an entire surface, which is the case we would like to address. The first limitation is the number of cameras used to cover the area. However, this method require a large number of cameras, this definition implies many overlaps to have the full view coverage.

The previous solutions and algorithms presented in [19, 40, 9, 14] are based on constructive methods for the camera placement and their local optimization. The positioning of the camera network is carried out camera per camera successively and iteratively. This method has some consequence, notably, the fast increasing number of iterations required to have a sufficiently accurate solution. The time complexity is even more problematic while increasing the size of the area and even more while increasing the number of cameras. Also these solutions are extremely dependent on the formulation and the constraints and cannot be easily adapted to other closely related problems (new constraints or modified objectives). The poor adaptability of the method is mostly due to the use of heuristics designed for very specific problems.

2.2.2/ Linear programming optimization and limits

Different methods of linear optimization were applied and tested in the literature. Linear optimization, when based on a judiciously adapted formulation, can be very effective on small convex areas which remains very restrictive This is the reason why linear optimization is often used for comparison with more flexible and efficient methods more as a research contribution (as in [4, 15]).

The following section will show the interest and the limitation of the linear programming based on examples from the literature.

2.2.2.1/ Linear programming

The linear programming is applicable for the linear and convex problems in order to minimize a linear and convex cost function.
The paper of Erdem et al. [28] is based on AGP and WSN by proposing a fusion of the two paradigms. Some modifications have been proposed to impose a field of view limitation which is not initially included in the AGP. On interesting aspect is how some camera properties have been modelized to fit with the problem of AGP.

In [28], PTZ cameras are set up with the task of zooming, 50mm or 35mm. The area to cover and the cameras parameters are discretized. The solution proposed being usable using an omnidirectional camera or considering a PTZ as an omnidirectional camera while performing an efficient angular sweeping. The omnidirectional cameras are simulated by PTZ camera with a non-continue zoom as in the experimentation proposed in [28]. Where the PTZ can have two focal lengths at 50mm and 35mm. Finally the solution proposed is to discretize the area to cover and also the different possible parameters for a camera (as: localization, orientation, focal...). In order to have a combinatorial formulation of the problem. Thanks to this formulation close to the Binary Integer Programming (BIP) and the application of a well-known method "Branch and Bound" in order to optimize the cameras placement.
This solution proposes a good coverage with the minimum of cameras in a reasonable time. The main limit of the solution is due to the use of omnidirectional or simile omnidirectional cameras.

Zhao et al. [6] intended to find the optimal position for a camera set in order to maximize an indoor area coverage (similar to an art gallery room). The coverage of an indoor area is not the main objective, which is tag detection in the scene. The solution proposed for the coverage is to adapt the number of points of interest which must be covered by the camera depending on the coverage rate.
The area is discretized as a grid. The grid is composed of smartly selected points. Each point of the grid represents a potential location of the target. Cameras are located in a few fixed positions on the walls of the room. The adapted grid and the restricted camera positioning are used to limit the size of the search space (as a number of possible solution). Thanks to this limited search space a linear optimization with the BIP can be used. BIP is a popular method that has been widely used such as in [6, 28]. In [6], the solution proposed is using BIP formulation, the smart sampling of the grid and branch and bound from LP_solve libraries to optimize the camera poses.

2.2.2.2/ Limitation of linear method

The methods of linear optimization were applied and tested to solve the problem of camera positioning for maximum coverage. In some cases, due to a well-adapted formulation or a restricted area and camera number, this solution is efficient enough. In some other cases, the methods were studied, but finally rejected because of their relatively bad performances when the number of cameras is high, or the room size and so on, as example [43, 4, 15]). Indeed, linear optimization is quickly challenged, or even failed, when the complexity of the problem increases (the solution can then converge to a local minimum).

Wang et al. [44] proposes a solution with an atypical problem formulation. The solution proposed in [44] will modulate the point sampling with respect to the room shape complexity (more points on the boundaries or obstacles and less on the center or "flat" areas). The idea is to have an area discretized with precision by using the minimum number of points.

The main advantage of this solution is to propose an area representation with enough sharpness and a minimum number of points. Less points to describe the area to cover mean also a gain in time efficiency during the camera poses estimation (solving the cost function is faster). Despite this interesting solution, the results presented in the experiments does not appear to be conclusive.

The main problem of the linear optimization appears when the problem becomes too complex. The complexity can come from the formulation and also from the additional constraints. But, in many cases, the increased complexity comes from the increased size of the search space. In practice, when the objective is to place a larger number of cameras or when the number of positions and orientations is too large, linear optimization may not work.

2.2.3/ Game theory

Among all the possible solutions, an atypical method is to apply the game theory [45]. The game theory is used to optimize the viewing direction of the cameras as in [2, 3, 45, 46]. These articles are based on game theory to find an equilibrium (also named Nash equilibrium) between two contradictory objectives. The objectives are, on one hand, to maximize the camera resolution and, on the other hand, to perform a multi-target tracking.

Soto et al. [2] proposed a network containing a dozen of PTZ cameras. That means the position of the cameras are fixed and the solution proposed found the best orientations with the appropriate focal length to track most of the targets.
To do so, the camera are smart enough to communicate with the close neighbors and adapt the pan, tilt and zoom depending on the objectives. The objectives have been defined by an utility function (or local cost function). The goal is to track most of the targets as possible with the better resolution. The cameras have reached the goal when they obtained a desired image resolution for all the visible targets. The multi-target issue can seem far from the coverage maximization we are interested in. But the number of targets may be higher than the number of cameras. The higher number of targets push the camera tracking to be an interesting solution to maximize the coverage of an area. In this case, the quality of the coverage will also depend on the number of targets and the importance of the resolution constraint.
In [3] and [46], different experiments have been proposed with a number of targets which increases progressively.
Furthermore, the decentralized solution (as proposed in [46] and [3]) is more adapted to prevent the security issues such as wrong transmission and interceptions by hostile opponents. Obviously the security and the integrity of the system are primordial key points in the surveillance application. To have the decentralized system the cameras must have some autonomy.
In the experiment proposed in [2] and [3, 46], each camera is smart enough to have its own tracking and control module. Also the cameras are able to communicate with each other, until a consensus is reached. The consensus is found when a Nash equilibrium is found between the two contradictory objectives. In this case it is a win win situation for both objectives.
So the objectives are not independently optimized, moreover the solution proposed have to optimize the objectives simultaneously in order to reach a consensus. The consensus is then also reached when it is no more possible to upgrade one of the objectives without downgrading the other one.
In the experiments of [3, 46], the consensus is found thanks to the cameras communications. The cameras communications have to maximize numbers of target covered with the higher resolution. The experiment is based on numerous targets moving freely in the area. Also in [3], one of the target must be covered in priority with a high resolution. This has an impact on the other camera positions. The results of the experiment as shown in

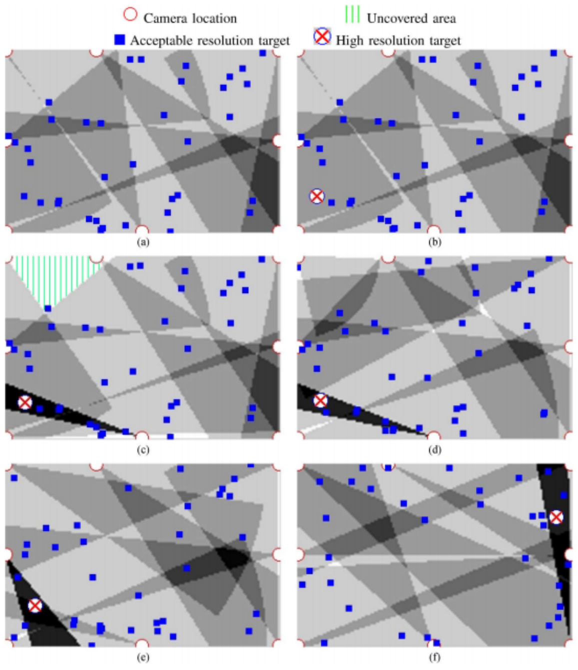


Figure 2.10: Result of coverd area after the game theory optimization. The objective of the game therory is to maximize the tracking and the resolution. The results shown are from the experiment made in [3].The images are representing the iterative sequences for the targets tracking and maximized resolution.

Figure 2.10 from [3] shows a really efficient global coverage with almost all the targets covered at every time-frame despite the movements of the targets.
The advantage of these solutions is the suitable result and the dynamic reconfiguration of the system for a reasonable size of the area and a decentralized computations. The decentralized computation of the system allows to process directly the information on each camera.
Otherwise this method has some limitations, as shown in the experiment, the area is relatively restricted and numerous cameras with a fixed position are needed. A consequence is that a sufficient and significant overlap is required. In this case the number of sensors is not perfectly optimized to cover the area. Moreover to use properly this method for maximum coverage, it requires a large number of simulated targets in the region to observe (there should be more targets than cameras). Finally this solution is more adapted for a self-reorganization for a set of PTZ cameras.

2.2.4/ Sum-up

In order to summarize the previous section, the Table 2.1 is proposed. It contains the main related papers. The first column is for the references, the authors name and the year of publication. The second column is dedicated to the best solution used. In fact some articles addresses few methods and only the best one is named here. The next columns are dedicated to specify the optimized parameters. The ✔ in the X and Y columns indicates if the solutions proposed optimize the cameras position along the X and Y axis. The position of the camera in X and Y can be randomly posed on the areas, noted as "random

dispersion” in these column. Finally in the X and Y columns, the ”0” indicates the case where the positions are not optimized or randomly placed.
The columns Pan, Tilt and Roll shows with a ✔ the solutions which enable the optimization of the rotation of the cameras. The 8^th column is used to precise if the focal length is optimized or fixed. When the focal length is optimized but with a discrete range of values, the annotation ”(discrete)” appears. The 9^th column shows the area representation. This representation is manly effective on the grid design. The 10^th column shows the maximum number of cameras placed and optimized during the experimentations presented in the articles. The last columns are dedicated to specify the possible secondary objectives which have to be optimized.

Table 2.1: Sum-up ref

Ref	**Best solution**	X	Y	Pan	Tilt	Roll	Focal lenght	Coverage room	Number of cameras	Secondary objectives and constraints		
[40] Ma 2012	heuristic	✓	✓	✓	0	0	fix	2D	≈ 10	barrier coverage	conectio depen-dance	
[19] Liu 2010	heuristic	random dispersion		0	0	0	fix	2D	≈ 600	resolution	tracking	
[10] Bodor 2005	non-linear	✓	✓	✓	0	0	fix	2D		tracking	resolutio	
[28] Erdem 2006	branch and bound	✓	✓	✓	0	0	✓ (discrete	omni di-rectional (by pan)	≈ 10	cost reduction	resolutio	DoF
[2] Soto 2009	game theory	0	0	✓	✓	0	✓	2D	≈ 10	resolution	tracking	multi target
[3] Ding 2012	game theory	0	0	✓	✓	0	✓	2D	≈ 10	resolution	tracking	multi target
[46] Song 2008	game theory	0	0	✓	0	0	✓	2D	≈ 14	resolution	tracking	multi target
[9] Nikolaidis 2009	heuristic	✓	✓	✓	0	0	fix	2D	2 to 4	Region of interst	trajector coverage	
[6] Zhao 2008	branch and bound	✓	✓	✓	0	0	fix	2D	≈ 10 (8-11)	tag detection	tag visbility	
[35] Yabuta 2008	linear programming relaxation	✓	✓	✓	0	0	fix	2D discrete square	≈ 20	region of interest		
[14] Hörster 2006	branch and bound	✓	✓	✓	0	0	✓ (discrete	2D	≈ 10	cost reduction		
[44] Wang 2017	multistage grid subdivi-sion	✓	✓	✓	✓	0	✓	2D	9 to 15	region of interest	big area	

2.3/ Solution based on evolutionary methods

The solutions discussed here, formulate the problem of camera positioning for a maximum coverage (as in the section 2.2) from various points of view. Until now the formulations and solutions commonly used have not been approached yet. In numerous works the cameras positioning for maximum coverage is assimilated to a problem of optimization. Due to the complexity of the problem (as introduced in the AGP section 2.1.2), the linear optimization or heuristic-based may not be appropriate. Other solutions could be the application of some stochastic methods with an appropriate meta-heuristic. Among the various possibilities the Evolutionary Algorithm (EA) often used. The following sections, are focused on the different algorithms based on the EA. The EA family are commonly used in the literature to optimize a set of cameras for maximum coverage depending on different secondary objectives.

2.3.1/ Among the EA algorithms

In this subsection different algorithms to solve the problem of camera positioning are discussed. The algorithms presented attempts to optimize the coverage depending on different specific constraints with different formulation adapted to the constraints.

The work of Zhao et al. [4] presents few solutions to optimize the position of the camera set . Among the solutions proposed, a greedy algorithm with a local optimization, an integer programming with a solver, a sampling algorithms and the Simulated Annealing (SA) from the EA family have been compared. Also the SA is used to customize the sampling mechanism in order to propose a new customized solution. In Zhao et al. [4], the problem is formalized as a BIP (binary integer programming). The goal is to find among a restricted number of possible positions and orientations the best pose for a set of cameras. The best pose for the camera set should cover the area, despite of the obstacles and the regions of interest.
Among the solutions proposed, the greedy algorithm has a good approximate solution, as long as the number of cameras is relatively low. The problem of the greedy algorithm and the integer programming solver is to have an important risk to get stuck in a local optimum (local minima). This risk increases proportionally with the size of the environment. Otherwise, the sampling technique with the SA is more adapted to the problem with time constraints and offer a satisfactory solution.
The solution proposed have some limitations: the major drawback is due to the experiment proposed. In fact, the experiment is made in a very small room with only few poses possible for each camera. Consequently, the solution proposed can appear better than a real situation and could be quite worse in a bigger area (in terms of computation time and pose optimization). The more interesting aspect in this work [4] is the usage of the SA from the EA family which allows to have an efficient coverage much faster than the integer programming solver for a similar result.

In the recent work of Akbarzadeh et al. [43], the cameras positioning for outdoor area has been designed as an optimization problem with few algorithms tested. Among the algorithms investigated, the SA and GA (Genetic Algorithm) both from the EA Family are the more promising. The work of Akbarzadeh et al. [43] is interesting in many points.
The problem formulation is one of the interesting point. The goal is to cover an outdoor area with multi-coverage ($k-$coverage discussed later in the section 4.1.1.5), the relief of

the terrain and obstacles may generate potential occlusions. Due to several reliefs of the considered maps, the cameras are always posed at the same distance from the ground (size of a tripod). Consequently the altitude of the cameras is automatically deduced from the relief. Moreover, depending on the position and orientation of the cameras, the relief can be source of occlusions for the cameras and has to be taken in account. All these elements combined make the problem formulation rather complete and realistic.
Among the solutions studied in [43], the non-linear has been tested with a quasi-Newton optimization method (called BFGS see in [43] section C), the SA and the CMA-ES (Covariance Matrix Adaptation Evolution Strategy) from the EA family with some mechanisms similar to a Genetic algorithm (see in [43] section D) have been tested too. The result of the comparison gives a clear advantage to the algorithms from the EA family in term of coverage rate with a fix number of cameras but also in term of time computation. The SA gives a close coverage rate (sometime slightly better) than the CMA-ES and also the SA has a better computation time. Otherwise the CMA-ES is more efficient in average with a reasonable computation time close to SA. The advantage of the CMA-ES appears more important in the bigger area. The quasi-newton and other algorithms tested are far away in terms of time and coverage, compared to the EA solutions which are more appropriate in this case. This work shows [43] the efficiency of the EA algorithm in a vast and realistic environments.

Chrysostomou and Gasteratos [15] presented an optimization system for maximum coverage with several constraints for two closely related problems. First of all, the goal is to cover an indoor area inspired by the AGP with a minimum of cameras. The minimum is fixed depending on a required coverage threshold rate. Secondly, the goal is to maximize the coverage for a fix number of cameras. These two problems are relatively close and just few elements have to be adapted to pass from one to another.
The coverage of the area is related to the camera visibility and few constraints have been proposed to control it, such as the visibility, viewing angle, field of view, resolution, viewing distance, and occlusions. The solution proposed to optimize the positions, orientations and some of the camera parameters (as the focal length), is to apply an algorithm from the EA family called Bee Colony. The Bee Colony is inspired by the Ant colony algorithm and associated to the bee exploration in the nature. The Bee Colony is in this paper [15] the more efficient of the two problems formulation (minimum of cameras and maximum of coverage with a fixed number of cameras) after a brief comparison with a GA and a branch and bound algorithms.
The experiment proposed is relatively limited. Only one room has been tested with only one test per algorithm, which is not relevant for stochastic algorithm (due to the importance of randomness). Despite that, the results proposed are encouraging.

The works presented in this section has the common point to apply different algorithms from the EA family to optimize the problem by maximizing the coverage for camera positioning. The algorithms from the EA family appear well adapted to the problem. To go further in the EA some specific branch such as the GA have to be studied.

2.3.2/ Solution used Genetic Algorithm or close related

Among the solutions applied from the EA to optimize the coverage area, the GA or algorithms closely related have been studied, as in the following section with some examples. In the solutions proposed such as [15, 20, 43], the GA has been applied as a comparator.

The GA and solution strongly inspired by the GA are not only used as a comparator, but it can appear more efficient as in [47, 39, 13, 25].

In Van et al. [47] the problem of camera positioning for video surveillance is studied. A building with several floors is examined and should be covered by a set of camera. Subsequently, the environment has to be considered in the three dimensions (3D) with the ceiling occlusion. The solution proposed is to fix the altitude of the cameras for each level of the building and considering each level as one independent room to optimize. The optimization is based on a basic GA with a customized crossover and mutation in order to fit the problems. For the crossover, a swap is done between two cameras from different sets. For the mutation, a Gaussian distribution is used to mutate some parameters of the cameras in a set. The GA is used with an elitists selection. All these parameters of the GA are essential to have an appropriate optimization adapted to the problems and to understand the performance of it (for more details about the GA and these parameters see the next Section 3.3).
Finally the area is covered with 32 cameras (for 3 levels) with relatively good coverage, few black hole and an acceptable level of overlapping. In this case, the GA is well appropriate and works efficiently.

In Jiang et al. [13] the solution for the coverage problem in a wide area with regions of interest and obstacles have been optimized using a standard GA. In [13], the GA has been tested for a wide area with numerous camera orientations. The goal is to find the best orientations to maximize the coverage for a camera set randomly placed in the space. The experiment has been performed in a vast area $400X300m^2$ with 60 cameras. All the cameras are embedding similar characteristics i.e. the same depth and field of view. The number of cameras is not enough to control the full area and choices must be made among the different regions of interest. The GA proposed in this article offers the best result with around 47% of coverage, with a minimum of overlapping.
The solution proposed with the GA is interesting and work well to optimize the orientation of the cameras in a relatively big and complex area. The main element missing in the solution proposed is an experiment with a number of cameras supposedly enough to fully cover the area.

In Wang et al. [25] a variant of the classical GA has been used. The Multi-Agent Genetic Algorithm (MAGA is from [48])) has been developed by the fusion of two algorithms, the GA and the multi agents systems. The main advantage of this algorithm is that it is supposed efficient on the optimization for a huge number of dimensions (Zhong et al. estimate the efficiency range around 20 to 10 000 dimensions). In this case, the number of dimensions means the number of parameters to optimize for a given problem.
MAGA is used in [25] to optimize the area coverage. The area to cover is a 3D space area and most of the volumetric space has to be covered. Unlike the other articles, the cameras have to be positioned in the 3 dimensions and the orientation as well. The constraints of volumetric space associated with the optimization of the position, orientation and cameras property, increase greatly the size of the search space (also the number of dimension to optimize). Despite this interesting solution, the experiment proposed are not enough consistent to properly assess it, despite that, the MAGA appears promising in term of potential increasing search space and solution quality.

The solution proposed by Topcuoglu et al. [39] is a Hybrid Evolutionary Algorithms (HEA). The HEA is based on the GA with different modifications, notably on the operators. For instance, the crossover is redesigned in order to have two different parts of

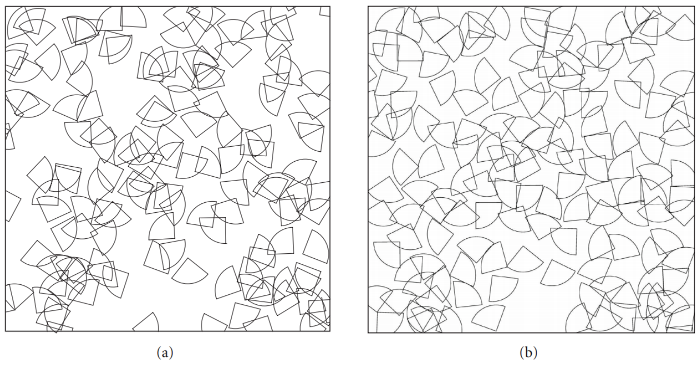


Figure 2.11: Illustration of the area coverage by a set of cameras randomly scattered (a). The cameras orientation has been optimized to maximize the area coverage (b). Experiment made by Xu et al. [5]

it. One of the parts is a local crossover and the other is a more *classical* crossover. The local crossover (called Proximity Based Crosover in Topcuoglu et al. [39]) is a crossover modified to have just a small amount of genome modified. This extends to a crossover with slight modifications and consequently a proximity between the solution and its offspring. The HEA proposed in this article is relatively close to another EA algorithms called the Mimetic algorithms. The solution applied in [39] is presented with different experimentations dedicated to maximize the visibility and minimize the cost of the sensors network. The experiments compare a simple random selection to the HEA. The result of this experiment shows a real efficiency in terms of minimizing the number of useful sensors and maximizing the total utility, where the utility is related to the other objectives.

From the point of view of these articles, the GA appears appropriate for the optimization of numerous cameras in a vast area as shown in [13]. The GA optimization can be a good starting point to develop a new method by tuning and customizing the numerous parameters as in [39, 25] or [47]. Also, despite the example presented, the GA is relatively under-exploited according to its customizability and efficiency.

2.3.3/ Solution used Particle swarm optimization or closely related

To maximize the coverage under various constraints the EA family is commonly used with different algorithms. One of those is the Particle Swarm Optimization (PSO) as refered in [5, 49, 20, 50, 29, 51, 41]. In this following section, the usage of PSO and similar algorithms is discussed for the problem of coverage optimization.

In Xu et al. [5], the PSO is used to optimize the orientation of several cameras (around 150). The cameras have been randomly positioned in the area. All the cameras have the same properties and cannot change position, but may adjust orientation to any directions (PTZ cameras). The area is designed in 2D and the viewing direction is described with only one rotation in pan (around the z axis). Finally the PSO is used to optimize the viewing directions in order to have the best coverage. The PSO optimize the cameras to reach a coverage around 65% after 1000 iteration and 20 particles for each iteration. The benefit between the initial random dispersion and the optimized viewing direction (with PSO) is around 12 points of percentage. The initial coverage (before the optimization) in the experiment was around 53% after the random dispersion (see the result in Figure 2.11. In this case, the PSO provides a relatively quick optimization and more suitable solution.

In Fu x et al. [51], the same problem as in [5] is discussed: the optimization of the orientation for an important camera set (100 cameras) using PSO. The difference is the number of parameters to optimize. In [51], the viewing direction is not only optimized on pan but also on the tilt axis. The PSO is compared with other algorithms such as the SA. Finally the PSO outperforms the other solutions experimented in this article despite an important number of parameters to optimize (100 cameras with 2 parameters per camera).

In Zhou et al. [49], the objective is to cover an indoor area inspired by the AGP to detect targets. The room is represented by a set of possible target positions. The coverage of the target is optimized for a fixed number of cameras. The experiments made in [49] show the ability of PSO in term of efficiency and speed compares to a hierarchical method. The proposed hierarchical approach is a greedy constructive heuristic (see more in [49] section III). The experiment is made in a basic square room described by a grid of 81 possible targets location which all have the same importance and have to be covered.
The result of the experiment in [49] showed the slight advantage of the hierarchical method. In fact, the hierarchical method gives a better solution, but requires more computation time. Otherwise the PSO provides an efficient solution close to the hierarchical method. The solution proposed by the PSO is also time efficient, between 2 to 6 times faster than the hierarchical method. The PSO combines result efficiently and fast computation time, consequently PSO appears more adapted to a realistic solution. Conversely the experiment proposed can be considered as limited due to the poor number of possible target locations in a simple room and the low level of choice for the cameras poses. An experiment in a bigger environment will probably show an even more important gap between the PSO and the Hierarchical method.

Zhou et al. [49] and Reddy et al. [20] proposed a similar solution for a similar problem formulation. The goal is to maximize the coverage depending on some target priority which must maximize the resolution using PSO. Moreover the solution proposed in [20] has been also taken into account the visibility parameters as depth of field and light intensity. All these constraints affect greatly the coverage results. Finally the experiments shows the efficiency of PSO for a small environment with few camera (around 7 cameras) for a multi objectives problem. The advantage in [20] is the addition of more objectives and constraints compared to [49]. Moreover the area has a better discretization which allows the confirmation of the efficiency of PSO for the coverage with multi-objectives in a small room.

Similarly in Fu y [29], the PSO is used to optimize the problem of maximum coverage. The camera poses must cover the totality of a small square room. The orientation of the camera has to be optimized in pan and tilt. In addition, all the cameras have an identical focal length. It is a relatively basic objective with few common constraints. The interesting part of this article [29] is the usage of PI-PSO knwon as Probability Inspired binary PSO. To use this algorithm, the problem has to be adapted. This adaptation gives an original formulation based on a combinatorial problems. Despite the atypical formulation, the solution proposed is greedy in term of cameras in order to cover a small space without obstacles.

In the recent work of Maji et al.[50], the finality is different but the problem can be considered similar to the problem of cameras positioning. The objective is to position several transistors in a rectangular Printed Circuit Board (PCB). The transistors must have a rectangular shape with different sizes and ratios. Despite the apparent simplicity due to the rectangular shape for the chips, some potential additional constraints, as the relation dependence between chips, or the strict non-overlapping of the transistors, makes the problem of positioning transistor becomes more complex. The solution is to optimize the positions and orientations of the chips on the board using EA. The PSO is chosen and more particularly the Craziness based PSO (CR-PSO). The interest of the CR-PSO is the variety of possible solutions introduced during the optimization compared to a classical PSO. The variety introduced during the optimization process by the CR-PSO helps to pass over the several local minimas. The more interesting aspect is to use a PSO in a similar problem as camera positioning which was commonly handled with a linear convex optimization as in [52] and based on the work on the floor planning defined Boyd and Vandenberghe in their book Convex optimization [53](8.8 page 438).

The solutions proposed based on PSO are promising. The PSO or closely related methods have numerous advantages as shown in the previous sections. The main advantages lie in the efficiency and the fast optimization, as well as the fast implementation. In fact, the PSO is relatively easy to implement and just few parameters need to be set-up to obtain an efficient result. The simplicity of the implementation is due to the numerous frameworks developed in all languages and the low numbers of parameters to set-up. Among the few parameters to set-up mainly the number of particles and the global inertia have an important impact (see more about PSO later in the Section 5.1.1). On the other side, the PSO is not really efficient to optimize a lot of parameters at same time (compared to other EA). The PSO can quickly be limited due to the increasing size and complexity of the area. Due to this limit the PSO has been customized as in [50] and [51]. The popularity of the PSO is mostly due to the association of the quick implementation, fast computation and efficient enough optimization for various problem as coverage optimization.

2.3.4/ Sum-up

In order to summarize the previous section, the Table 2.3.4 is proposed; which contains the relevant articles. The first column is dedicated to the references, the author names and years and the second is dedicated to the best algorithms. In these articles most of the time, several solutions have been tested. The second column shows the best algorithm among the algorithms tested. The third column notifies if the tested algorithms are from the EA family. The 4^{th}, 5^{th} and 6^{th} columns precises if the algorithm proposed optimize respectively on x,y and z coordinates the position of the cameras. In some cases the position on x and y are not optimized but randomly distributed. The 7^{th}, 8^{th} and 9^{th} columns notifies if the algorithm proposed optimize respectively on pan, tilt and roll rotations of the cameras (if no optimization, the orientation is fixed). The 10^{th} column is dedicated to mention if the focal length is optimized or fixed. The column for coverage representation specifies whether the area to cover is designed in 2D or 3D (11^{th} column). In some cases, the 2D representation of the area is enriched (noted with a ”+” or ”+relief”). The 12^{th} column shows if the experiments proposed are indoor or outdoor (in, out). The

13th column is dedicated to the maximum number of cameras tested in the experiments. The last columns are dedicated to list the main secondary objectives and constraints.

Table 2.2: Sum-up of the solution based on evolutionary method for maximize the coverage.

	Best algorithms	Other EA tested	X	Y	Z	Pan	Tilt	Roll	Focal length	Coverage representation	Indoor outdoor	Number of cameras	Objectives and contraints	
[39] Topcuoglu 2009	HEA GA	✔	✔	x	o	✔	o	o	fix	2D	out	10 to 200	max visibility	min cost
[20]Reddy 2012	PSO	o	✔	✔	o	✔	o	o	fix	2D	in	≈ 10	tracking	light intensity
[49]Zhou 2011	PSO	o	✔	✔	o	✔	o	o	fix	2D	in	1 to 20	tracking	resolution
[15]Chrysostomou 2012	Bee Colony	✔	✔	✔	o	✔	✔	o	(2)	2D	in	≈ 10	cost reduction	resolution
[47]Van 2009	GA	o	✔	✔	o	✔	o	o	fix	2D+	in	≈ 32	min overlap	multi level
[5]Xu 2011	PSO	o	random dispersion			✔	o	o	✔	2D	out	50 to 600	region of interest	
[43]Akbarzadeh 2013	CMA-ES	✔	✔	✔	relief	✔	✔	o	fix	2D +relief	out	10 to 110	relief	region of interest
[50]Maji 2015	CR-PSO	✔	✔	✔	o	0	o	✔	✔	2Drectangle		≈ 15	Rectangular proportion	dependency
[4]Zhao 2013	SA*	✔	✔	✔	o	✔	o	o	fix	2D	in	6 to 30	tracking	
[25]Wang 2009	MA-GA*	o	✔	✔	✔	✔	✔	o		3D	in	<30	Visible FoV	cuboid obstacle
[13]Jiang 2010	GA	o	random dispersion			✔	o	o	fix	2D	in	≈ 60	region of interest	
[29]Fu.y 2014	PSO	o	✔	✔	0	✔ (4)	✔	o	fix	2D	in	15 to 25	resolution	focus
[51]Fu.x 2010	PSO	o	✔	✔	✔	✔	✔	o	fix	2D	out	<150		

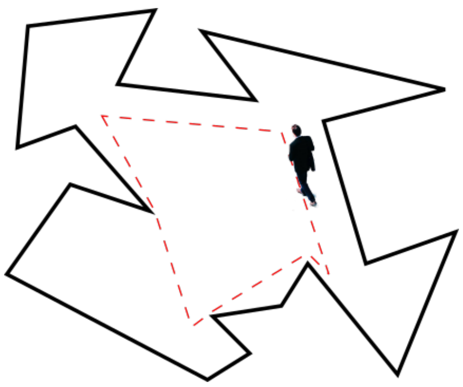
2.4/ Coverage path planning

The coverage path planning has for objective to design the more efficient path to cover an area. The path has to cover all or at least most of the area using the shortest path.
The solutions highlighted in the literature until now was about placing numerous cameras or robotics cameras (as PTZ cameras and smart camera) for vast area composed by multiple obstacles. The solutions proposed until now remains reliable only to monitor a vast area continuously with several cameras. The disadvantage of positioning a camera set appears quickly with the cost in computation time. The expensive cost is due to the several cameras and the communication network required to centralize the collected images. All the images must be collected in a real time.
In fact, in some applications, the area needs to be controlled periodically. The periodic control of the area does not require an installation for the set of fixed cameras. Until now the installation was composed by a set of immovable cameras unlike the solution presented in section see Section 2.3 and 2.2). The periodical control requirement can be illustrated with some example : the cartographies of the area is need just one time and can be completed with one fly over the area roughly bounded as in [54, 55], forest fire detection which requires a periodic fly over specific and vast region as in [56], the hoovering robots which needs to cover all the room with possibility some on-line computation [23, 57, 58, 59] and the agriculture. About the agriculture application, a UAV needs to fly over the field few times in a year in order to control by photography the hydration, the maturity and anything else of the field as in [55, 60, 61, 62, 63, 64]. For all these applications, the area must be covered but does not need a coverage of all the area instantly and continuously. The solution commonly proposed is the usage of only one sensor mounted on a mobile robot. The Mobile robot need to be adapted to the task and the environment, as flying [65], driving [12, 66], swimming [54].
The mobile robot with the sensor moves in the space to cover all the area. In this case, the objective is to determine the best path for the mobile robot to cover the area. The best path is dependent on the constraint of each problem. However, the common point in theses papers is to obtain the shorter path. The following section is focused on finding the best Coverage Path Planing (CPP) for a sensor mounted on a mobile robot. In most cases, the sensor is a perspective camera. To estimate the best CPP, different algorithms and methodologies have been applied in the literature to solve various problems.

The following sections are focused on the different algorithms and methodologies applied to optimize the CPP problem. First, the watchman route problem is introduced to highlight the origin of the CPP problem and the relation with the cameras positioning for optimal coverage. In a second time, the more popular solutions are discussed.

2.4.1/ AGP to watchman route problem

Before exploring the solutions proposed for the CPP, it is essential to understand the role of the Watchman Route Problem (WRP). The WRP is closely related to the CPP and the WRP has impacted the research about the CPP. The following sections are focused on the definition of the WRP, the proposed solutions in order to solve it, and finally, a brief discussion is drawn about the WRP .



(a)

Figure 2.12: Watchman route problem answer illustration.

2.4.1.1/ Definition of the watchmen route problem

The Watchman Route Problem is introduced for the first time by Chin and Ntafos in 1987 [67]. The problem of the WRP can be summarized in one sentence :

”How to calculate a shortest route contained inside a polygon such that any points inside this polygon is visible from at least one point of the route?”.

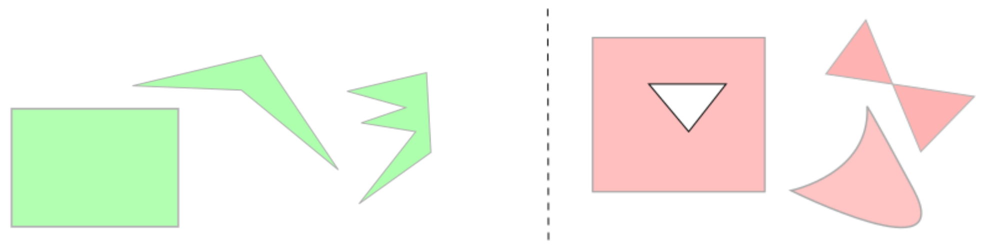
The guard has to cover an area represented by a polygon (see the illustration of the problems in Figure 2.12). The guard is considered as *perfect* with no restriction in the field of view (the guard can see at $360°$) and no restriction in the depth field (the guard can see from on extremity of the room to another. The guard can see the opposite wall excepted if an obstacle occludes the view). The guard abilities are directly inspired by the AGP (see 2.1.2) The shape of the polygon is primordial and affect greatly the possible solutions and the complexity to solve the WRP. The WRP problem is by many aspects closely related to the AGP. The AGP (see Section 2.1.2) is commonly considered as the root of the WRP. In fact, the WRP is not only focused on standing position, but on finding an optimal path. The path has to be optimized to cover all the points which compose the polygon and the path has to be as shorter as possible.

The next section will introduce the possible methods and algorithms applicable to solve or atleast optimize a solution for a WRP.

2.4.1.2/ Solutions

The WRP problem can be solved under some conditions. The solutions to solve it are applicable only if the polygon is simple. A polygon is considered as simple, when the boundary of it are composed of continue straight lines that do not intersect between them. To close this definition, it is important to precise the simple polygon does not have any hole (see Figure 2.13).

To solve the WRP, few algorithms were developed. The algorithms proposed in the literature enables successively better algorithm to solve the problem. The first interesting algorithm for the WRP with a simple polygon is the early work of Tan et al [68]. Tan et al.[68] proposed an algorithm for a fast computation time. The solution proposed works with a simple polygon composed by n vertices and its complexity belongs to polynomial



(a) Example of simple polygons. (b) Example of complex polygons.

Figure 2.13: Few examples to illustrate the simple polygon shape.

time $O(n^5)$.
Others contribution about WRP were then proposed. Dror et al. [69] proposed a better time complexity. In fact, in [69], the considered solution with a simple polygon is working in $O(n^3 logn)$ complexity. The deterministic solution applied is also usable in closely related problems of the WRT as the zoo-keeper problem and safari problem which makes this contribution versatile and generic. Two other algorithms [68, 69], are usable only in the case of simple polygons to deliver the optimal solution which is the shortest path for a full room coverage.

To have a more general solutions for WRP, the proposition lie in having an efficient algorithm working also with complex polygons. A possible solution could be to look for an efficient approximation by optimizing the position of the guards and their respective paths. An efficient approximation means a path acceptably short. In fact the path computed can not be certified as the "best path" but only as the shorter found.

Several efficient approximations have been proposed such as in [70] and [31].

In Packer [31], the algorithm proposed is based on splitting the problem into two sub-problems. The first sub-problem consider the search of a set of points which can be good enough to cover all the area despite a restricted visibility range. These points are called waypoints. Once the set of waypoints to cover all the polygons are found, the second sub-problem consider the creation of a path passing by all these points. This second sub-problem is similar to a classic Travelling Salesman Problem (TSP). The TSP tries to answer the questions asked by a travelling salesman "**What is the shortest path passing by each city only one time and return to the starting city?**". In the TSP, the cities are the nodes on the interconnected map and the roads are the connections between them. The TSP is a well known NP-hard and NP-complete problem when applied in some conditions, as described in [71].
Finally the solution used for the TSP can be applied for the second sub-problem of WRT. Subsequently, TSP can be applied for the second sub-problem of WRT solution proposed by Packer [31].

In Faigl [70] a similar method to the one proposed by Packer [31] has been proposed to approximate the shorter path. The problem is also split into two sub-problems. The first is to optimize the position of the waypoints and the second is to schedule the position in order to create a path planning (directly inspired by the TSP). Furthermore, the solution proposed by Faigl in [70] is applicable to a watchman with a restricted visibility range i.e. equivalent to a $360°$ field of view with a *restricted depth of field*. This restriction affects greatly the waypoints positioning. Due to this constraint, more waypoints needs to be placed to cover an area. Once the set of waypoints for the polygon coverage is found, the second sub-problem is to create a path passing by all these waypoints. The algorithms proposed to perform such task are also inspired by the solution given for the optimization of the TSP.

2.4.1.3/ Limit and consequences

The methods proposed to solve the WRP are really interesting and gives a good solution in some specific conditions. These conditions make his methods special cases. In fact the optimal solution, i.e. the shortest path which covers all the area (polygon), are usable only if the polygons observe some rules. The polygon must be simple and take into account the guard ability (no viewing constraints).
When the polygon is more complex, the optimal solution cannot be reached. The methods applied to solve the WRP for complex area are promising. Especially the problem splitting into sub-problems. Despite this interesting aspect, the solutions proposed are most of the time limited due to the area representation. This representation associated to the geometric methodologies to find the waypoints gives a crucial importance to the number of vertices inducing also the number of cameras.
A large number of vertices imply consequently that the first sub-problem may be difficult and time-consuming to solve. Consequently this method is not the more appropriate for vast and complex outdoor areas which would require numerous vertices to describe. The biggest limit of the WRP is the ability of the watchmen. Indeed, in the original problem, the watchmen are considered having a perfect visibility. The solution proposed solving the WRP are not usable with the accumulation of new visibility constraints. In Faigl [70], the WRP started to be extended by adding a constraint on the visibility. In this condition the problem is slightly modified to shift from self-organizing map to coverage path planning. But despite this addition of a restricted depth of field, the proposed solution based on geometric heuristic do not allows the accumulation of more constraints.

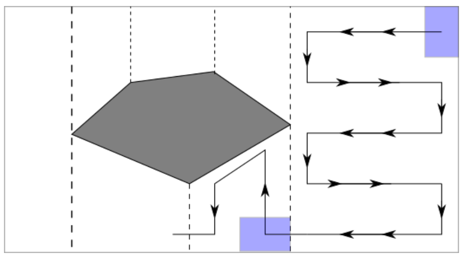
The AGP then WRP extensively impacted the vision, the design of cameras coverage problem and coverage path planning field. Among the solutions discussed in the previous section, numerous articles are based and/or refers to the AGP and WRP. Our work is consequently inspired by those formulations and will propose to extend it.

2.4.2/ CPP solutions

To optimize the CPP problems many solutions were proposed. The different algorithms and methodologies proposed are discussed in the following sections. The methodologies proposed can be organized in several branches. The most important branch to optimize the CPP corresponds to the usage of sweep associate to a cellular decomposition.

2.4.2.1/ Cellular decomposition and sweep

To solve the CPP problem, the most common solution is considered to be the cellular decomposition. The cellular decomposition is by some aspects inspired by the methodologies presented in the WRP. To recall one of the interesting solution for the WRP, the methodology starts by splitting the problem in two sub-problems and optimize those independently. The first sub-problem solve the search of the best waypoints and the second



(a)

Figure 2.14: Illustration of simple cell decomposition with sweep.

sub-problem corresponds to finding the shortest path passing by all the waypoints (like the TSP). The Cellular decomposition also split the problem of CPP in two sub-problems. The first sub-problem is focus on the decomposition of the complex area in several cells. Each cell has to be a simple polygon (basically a rectangle or latter any quadrilateral polygon). Inside each cell a sweeping has be applied in order to cover all the area (see Figure 2.14). The second sub-problem is to find the path which can connect each simple polygon. The problem has to take in account, the sweep start and end to find the global optimized shorter path. Where the global optimized shorter path take in account the sweep trajectory of each simple polygon and the path between the simple polygons.

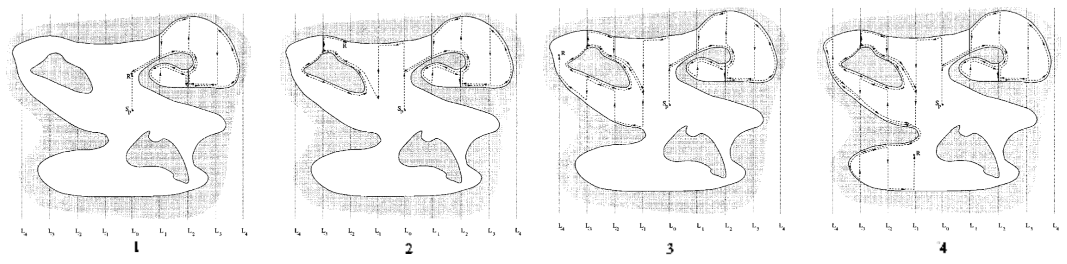
Finally the cellular decomposition is made by decomposing the complex area, choosing the appropriate sweep and finding the shortest path to connect the cells.

Decomposition

The decomposition in cells has for objective to split a complex area in several sub-areas. Each sub-area has to be simple in term of shape in order to apply a sweep. Since the 90s, numerous algorithms have been developed for the cellular decomposition. Among the algorithms for cellular decomposition, 3 types of decomposition were deduced as shown in the survey of Choset [72] and Galceran and al [54].

- Approximate: An approximate decomposition is based on discretization of the area. The free space of the area is represented by a set of cases (grid). Each case of the grid has to be covered by a mobile robot. A case is considered completely covered if the robot is on the associated position. Which means that the frequency of the grid is defined by the covered area of the mobile robot. The approximate decomposition by cases is convenient to describe the area but is greatly limited due to the low sampling frequency of the gird and the limited amount of possible trajectories.

- Semi approximate: The semi approximate decomposition is partly based on the discretization of the space. The idea is to create a set of large cells. The width of the cells is fixed and the height is relative to the area boundaries (see in [72]). The semi approximate cell decomposition allows to have cells with two parallel sides with a fixed size (right and left) and the two other sides adapted to the boundaries (up and down. See the illustration of a semi approximation in Figure 2.15). The width of the cells are chosen accordingly to half of the focal length



(a)

Figure 2.15: Illustration of a semi approximate decomposition outcome from the survey of Choset [72]. The 4 figures illustrates the trajectory of the robot over the time.

of the camera mounted on the mobile robot. The area is covered by following the boundaries of the cells with sweep to cover cells after cells. The advantage of the semi approximate decomposition is the ease to describe a vast area and the low computation time required. Additionally, this decomposition can be made on-line by the robot and do not requires a high level of knowledge of the area. The disadvantage is the non-optimization of path due to the cells decomposition. The simplicity of this decomposition can generate case where the robot have to cover many times the same place. The semi approximate cellular decomposition do not allow to have a well optimized path planning.

- Exact cellular decomposition: The exact cellular decomposition describe the area by creating juxtaposed geometrical regions. The size of the regions (or cells) are not depending on the robot ability contrarily to the previous decomposition. The large size of the cells allows the mobile robot to do back and forth motion to cover all the cells (also called sweep as in Figure 2.16). The cells have to be ordered to conserve an efficient and short path going through all the cells. The global path distance have to be reduced by an appropriation of the path passing through cells. The exact cellular decomposition became popular and numerous algorithms have been proposed. The proposed algorithms can be a faster decomposition or a deduction of more appropriate shape depending on the basic rectangular cells. Among the numerous exact cellular decomposition propositions: the trapezoidal decomposition for its simplicity and historic importance, the Boustrophedon decomposition and Morse-based Cellular Decomposition can be cited for their importance in the construction of other exact cellular decomposition. These algorithms and other are clearly summarized in the survey of Carreras and Galceran [54]. The exact cellular decomposition has been still studied since the survey of Carreras and Galceran [54] and upgraded, as in [73], proposing a decomposition for concave or multiple polygons.

Sweeps

The back and forth or sweep is an essential element of the CPP by cellular decomposition. The sweep has to cover the entirety of the cells. The cells are splitting the area in relatively simple polygons (as shown in the paragraph 2.4.2.1 and the Figure 2.14). The sweep needs to be adapted to the shape of the cells and also must start and finish in a ap-

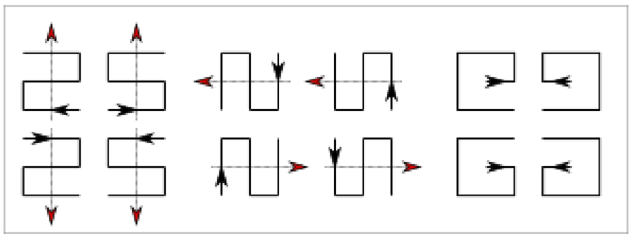

(a)

Figure 2.16: Illustration of the different sweeps and spirals.

propriate position taking into consideration the path to the next cells. For that the starting point and the ending point of the sweep are crucial for the global path planning. In Torres et al. [73], the sweep is function of a direction and can go clockwise or counter-clockwise to have a start and end in the appropriate position for the transition. This enables to have 8 different applicable sweeps at each cells depending on the most appropriate start and finish; according to the computed path. The sweeps are adapted depending on :

- The turn sides (clockwise or counter-clockwise)
- The directions (horizontal or vertical)
- The finishing positions (start and stop from the same side or opposite side)

The sweep can be also alternatively switched for a spiral as proposed in Jimenez et al. [74]. The proposed sweep and spiral are adaptable depending on the context. The sweep can have two directions and for the spiral two turn sides (clockwise or counter-clockwise). The different sweeps and spirals are illustrated in the Figure 2.16.

To have an adapted sweep, the footprint has to be defined depending on the camera ability and the objectives. In [73], the size of the sweep is dependent on the area covered by a camera and a considered sufficient amount of overlaps for the 3D reconstruction (see also[22]). In Li et al. [7], the footprint of the camera with a small pan is considered. Due to the pan, the camera projection is not a rectangle but a trapezoid. Consequently the sweep size is adapted by taking the larger side of the trapezoid for the sweep dimension. Also for the grid decomposition, the sweep size is directly related to the coverage ability of the mobile robot (as example in [58, 75]).

One extra element necessary for the sweep can be in some cases the external element such as the wind. In [58, 76], the external condition are taken into account in the cost function and it influence the sweep.

To summarizes the sweeps requires to be adapted to the area and the relation between cells. To choose the appropriate sweeps the following element have to be considered:

- The size of sweep have to be defined depending on the ability of the camera (footprint size).
- The direction (horizontal or vertical).
- The starting and finishing position (start and stop from the same side or opposite side)

To obtain the more appropriate sweep, it is primordial to know before the cells scheduling. The cells scheduling are a crucial part of the path planning.

Path planning

The aim of the path planning is to find the short path going through all the cells. Finding the best path planning passing by all the waypoints or in this case all the cells, is a complex problem which can be formulated as a TSP. The path planning became even more complex in the case of exact cellular decomposition or for any decomposition which requires a sweeping inside each cells. Indeed, in this case, the goal is to have the shortest path planning with taking into account each sweep and transition between cells. In the previous paragraph 2.4.2.1, the different sweeps and spirals have been detailed. The start and the end of the sweep are the crucial elements to consider during the path planning computation. Obviously, to find the best scheduling between each cells, the algorithm induces by the TSP are commonly used.

When the area is decomposed in cells, the goal is to schedule the right-of-way from one cell to another in order to create an efficient path. The scheduling of the cells can be optimized by using the paradigm of TSP with the same algorithms to have an acceptable path planning. The TSP is a well known problem and is deeply studied from long time ago. First is essential to remember the TSP is an NP-complete and NP-hard problem as prooved in Karp in [71].

To optimize the TSP, numerous algorithms have been tested and some of them have been specially applied to schedule the cells to have the shortest path. In An et al. [77], two algorithms were tested before developping a third. The algorithms used are based on branch and bound. The algorithm developed in [77] is called Novel Previous-Next Waypoints Coverage Constraint (PNWCC). The algorithms presented in [77] proposes at same time a schedule of each cells as well as a smooth trajectory without sharp edges usable for non holonomic driving robots.

In the survey of Carreras et Galceran [54], the solutions proposed to solve the TSP are asked to compute an exhaustive walk trough the adjacency graph. These solutions are workable only for computingg small adjacency graph. The GA is popular to optimize the TSP and is commonly used to evaluate the influence of parameters as in [78] [79]. About the CPP problem, the GA is also used to optimize the TSP part as in Jimenez et al. [74], where the GA is applied to find an optimized schedule retrospectively of an exact cellular decomposition of a complex polygon.
The GA is announced to be appropriated to obtain an optimized solution for the TSP. In some situations, a TSP is not realistic and some external constraints can be added such as the wind effect, turbulence, or holonomy constraint as in [80, 81, 54]). The addition of external constraints may leads to more complex scheduling of the cells.

2.4.2.2/ Other solutions

Among the algorithms developed for the CPP, some interesting methods have to be studied. Some algorithms has been discussed in Choset [72] as approximate solution (see Section 2.4.2.1).

The approximate cellular decomposition was briefly approached in the previous Section

2.4.2.1. The approximate cellular decomposition was considered not interesting due to the low sampling frequency of the grid. Conceptually, the works of [57, 59, 58] are going further by using a higher sampling frequency of the area to cover. The solution based on regular grid discretization may be compared to approximated cellular decomposition with a high sampling frequency as in [23]. Due to the relatively fine grid decomposition which engender an increased size of the cases number, the algorithms to optimize the path passing by all the cases of the grid has to be adapted . Consequently, new navigation strategy has been developed.

In Luo et al. [57], the goal is to have a complete coverage by visiting all the cases of a grid. To visit all the cases of the grid, a neural-neighborhood analysis and neural dynamic programming approaches are adapted to have an efficient CPP for the robots. In Simon et Luo [59], they proposes an similar method with a neural network solution for dynamic and non planar area.

The work of Lee et al. [58] proposes another solution based on smooth spiral path. The idea is to propose to follow the boundaries of the room in order to fully cover and use a smoothed spiral to cover the area. The solution proposed is compared to other methods. The main advantage of these method is the on-line ability and the smoothed trajectory (see also [75]).

Before to conclude about some of the algorithms usable to optimize the coverage path planning the more elementary has to be discussed. In fact, among the algorithms proposed until here the full random was not discussed. In Liu et al. [82], a really basic method is developed to cover an area with a mobile robot (hover robot) for on-line path planning in a dynamic environment. The algorithm proposed is based on random directions. The basic idea is to move forward until the detection of an obstacle (ultrasonic or bumper). When an obstacle is detected, the robot turns randomly. This solution is really basic and is based on the idea saying that if the room is not too big or not specially complex, with enough time, the mobile robot will cover all. Obviously this solution is not optimized and not suitable in many cases.

3

Genetic algorithms

Contents

3.1/ Darwin and the natural selection

The theory of evolution was introduced by Darwin and it inspired the computer science for developing optimization algorithms. To understand the algorithm it is important to go back to the origin. The following section is focused on the fundamental theories of the natural selection and its history.

3.1.1/ Darwin theory

Darwin has studied the differences between individuals from the same species and tried to establish a classification of the different sub-species. It appeared some individuals from the same species and from different countries had some slight differences. These variations were studied and explained by the Darwin theories in The Origin of Species, published in 1859.

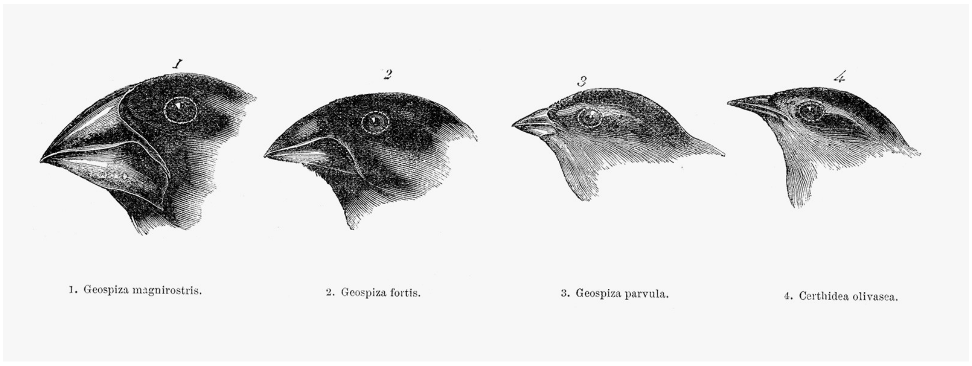


Figure 3.1: Finches from Galapagos archipelago extract od The Origin of species by C.Darwin .

The origin of species details are grouped and classifier in the theory of evolution. This theory uses the concept of adaptation introduced presciently by J.B. de Lamarck, and deeply studied by Darwin. The adaptation explains the relation between the environment of the individual and the differences generated by natural selection. This observation was first made on birds, called geospiza, or finches, (chaffinch) from Galapagos archipelago. Darwin noticed the difference of their beaks (see the Fig 3.1). The shape of the beak was correlated of the specificity of each island. Finches with the biggest beak correspond to the island with the biggest seeds.
This observation was formulated and explained by Darwin by the adaptation of the birds in their environment.
The adaptation is partially due to the natural selection. Indeed the selection is done by the reproduction of the strongest individuals.
The reproduction concerned 2 individuals (one male, one female). Each individual is in competition with the other individuals of the same species. In its condition only the stronger and the more adapted individuals have a chance to have a progeny (an off-spring). Therefore generation, after generation, the more adapted individual itself reproduced and mute, while the species adapt to their environment. In this case, the strongest finches is the one with an appropriate beak in order to eat more seeds.

3.1.2/ Biologic evolution

The Darwin theory of evolution was contested during long time until the confirmation by the progress of biologic sciences, especially with the genetic progress. The progress in this field was used to study the mechanism of the natural selection and evolution. One of the important progress and confirmation of the theory is by using the analyze of DNA code. DNA for DeoxyriboNucleic Acid contains all the useful information: growth, life and reproduction of life.
The discovery of DNA code permits to confirm the genetic proximity between some species. Moreover the DNA permitted the evaluation of their evolution and code modification in the same species from different location and several generation gap. Thanks to the genetic and DNA the evolutionary process has been explained deeply and clearly.

In the biology, every living element is composed by cellular. Inside each cellular the DNA code is stored. The DNA composes the chromosomes. The chromosomes have a central role in the definition of one individual and in the reproduction process (see the Fig 3.2).

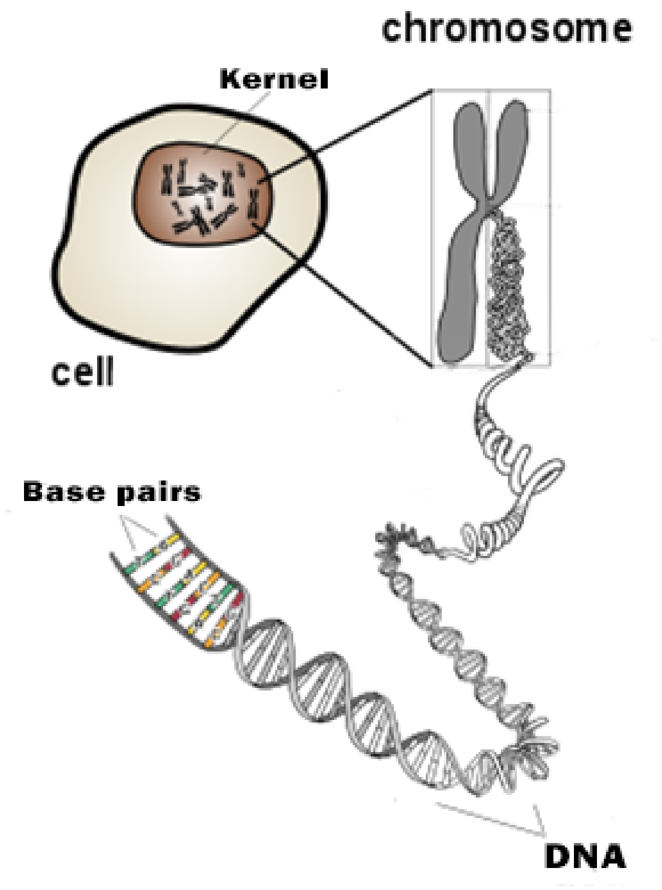


Figure 3.2: Biologic representation, cell to chromosome until DNA.

As introduced earlier, the evolution is possible by a natural selection. The selection is done by survival and reproduction ability of the best individuals. Better mean more adapted to the specific environment, like for the geospiza the size of the beaks depending on the size of the seed from his island. In the island with the big seed the geopiza with the small beaks was not the most adapted and had more difficulties to find food. This weakness makes the geopiza with the bigger beaks in better position to reproduce and the DNA of the individuals with biggest beak is transmitted to the next generation.
The understanding of the reproduction mechanism in term of succession and transfer of the natural ability is primordial. It is explained in part by crossover in the cellular state.

Sexual reproduction. The living element, for example the geospiza, use sexual reproduction. The sexual reproductions assumes to merge part of the chromosome from the two individuals to create their descendants. The selection is essential in order to keep the more adapted individuals of the species.
Among the reproduction mechanism, the crossover and the mutation are the ones with the biggest impact.

Crossover. The crossover is the action of merging the chromosomes of two individuals in order to have a new child.
It is an essential factor to preserve the individual ability of the geospiza. But the natural selection and the crossover cannot be considered as the only useful element to evolve.

Gene The gene is a subset of the DNA code. Commonly, the DNA is cut in many thousands genes where each gene can represent a specified function or ability.

Mutation. Contrary to the crossover which allows only the ability preservation from one generation to another, the mutation introduces some “anomalies”. The anomalies can become in some cases a biologic advantage. The mutation affects only rare chromosomes of the individuals. The chromosomes affected are not fully mutated but only one or two genes are affected.
The mutation changes the little piece of DNA code to introduce variety in the genetic code by the small “anomalies”.
Most of the time the small mutations are not consequent for the individual but generation after generation the mutation can be preserved and spread in the population.

The giraffe can be taken as an example:
In the arid environment, the giraffe with the longest neck has more chance to survive due to its empowered ability to find food. The giraffe with a neck a bit longer than the other, can become more attractive for the natural selection (in this case more food mean stronger and then more attractive). The natural selection pushes the best individual to reproduce together and by the crossover mechanism conserves the small advantage given. The initial mutations giving at few giraffes a longer neck and by the process of natural selection associate to the crossover allows this advantage from a small mutation to become the norm. Mutation after mutation and generation after generation the giraffe observe the average length of their neck increased. Finally the actual giraffe is the result of a long and complex evolutionary process.
The mutation can also be the source of degenerate animals but in this case the natural selection by the reproduction (and crossover) will not allow the preservation of the individual and thereby the mutated chromosome will disappear.

3.2/ The evolutionary algorithms

The Evolutionary Algorithm (EA) is a big family of algorithms and include many meta-heuristic used in the field of optimization and artificial intelligence.
The evolutionary algorithms are inspired by the biologic mechanism to design meta-heuristics. The origin of the inspiration can be various such as the genetic, insect work, animal behavior, ... (see Table 3.1).
The biologic inspiration is not the only element used to define the evolutionary algorithms. All the algorithms in this family are dedicate to optimize iteratively a population of solutions. The EA family is not part of deterministic process and use a randomized function in order to evolve.
To summarize, the EA has most of its attribute :

Inspiration or group	**Algorithm**
Based on memorization.	Neuroevolution
	Learning classifier system
Animal inspired and swarm algorithms	Particle swarm optimization (PSO)
	Bee colony
	Ant colony
	Mimetic algorithm
	Shuffle frog
Swarm algorithms	Addaptatif dimensional search
	Gaussian adaptation
	Genetic Algorithm
	simulated annealing
Combinatorial	Harmony search
Genetic	Genetic programing
	Evolutionary programing
	Evolutionary strategies
	Evolutionary programing
	DarwinTunes
	Genetic Algorithm

Table 3.1: List of some basic EA.

- Bio inspired.
- Using randomness (not fully deterministic).
- Based on population.
- Evolving a set of solutions to optimize a given problem.

These definitions are not the strict characteristics for all the EA. They are rather the most common element of the major part of the vast EA family.

3.2.1/ Historic

The EA are relatively recent and do not have a fixed origin. It is the result of more than a decade of research and improvements. The premise of the EA can be considered to be the work of Robbins et al. [83] in 1951. More commonly, the beginning of the EA are on the late 50s with the works of Bremermann [84], Friedberg [85], Box [86]. They proposed different algorithms based on the evolving solutions to optimize solution for any given problem.
During almost the three next decades, the research had slowly progressed and remained rather unknown. Mostly due to the low computation power at this time and also to some methodological short comings of those early approaches.
Despite of this difficulty, the fundamental works of Holland [87] and Fogel have been essential to the progress and to popularize the EA.
In 90s, due to the fast increasing computation power, the EA became more popular and numerous new algorithms from EA family have been designed as listed in the Table 3.1.

About the applications of the EA, the engineering field (examples in [88]) and the multiplication of the conferences in EA allowed the democratization of this family of algorithms. The EA took profit of three main and independent methodologies; evolutionary programming, evolution strategies and genetic algorithms (GA).

- **The evolutionary programming**, especially the work of Fogel is based on the finite state machine. The goal is to predict events based on the inputs. It was one of the premises of machine learning and classification.

- **The evolution strategies**, especially the work of Rechenberg ([89]), proposed a strategies based on deterministic selection and random mutation. The goal was to solve difficult experimental problems with discrete or continuous search space. Rechenberg applied particular the evolutionary strategies for aerodynamic profile design.

- **The genetic algorithm**, is probably the most polyvalent and tunable method belonging to the EA family. The GA offers an adaptive processes to optimize a solution. The GA will be precisely detailed in the Section 3.3.

The GA have more particularly attracted the interest of the research with the work of Holland and Goldberg. The popularity of the GA is most probably due to the increasing power of computation (in 80s, 90s) associated to fundamental progress and the numerous possible applications in the optimization field.

Since the late 90s the research of EA have been focused on the Multi Objective Evolutionary Algorithms (MOEA) [90, 91, 92]. The MOEA is the logic extension of the EA to give solution to problem more and more realistic which requires to manage different objectives and constraints potentially contradictory. The MOEA is discussed in the Section 3.4

3.2.2/ General formulation

Most of the EAs have to optimize one or several problems using an iterative process to evolve towards an assumed optimal solution. The EA can be formulated as the optimization of a given problem, in order to obtain the best possible solution. The possible solution must be formalized as a input vector $\vec{x}$. Where each element of the vector ($\vec{x}$) is one input or a dimension (x_i) to optimize the given problem.

$$\vec{x} = \{x_1, ...x_n\} \in \Omega \tag{3.1}$$

Where n the number of dimension to optimize (or number of inputs), and Ω is the search space of the problem. The search space represents all the possible solution of the problem. Sometime, depending on the problem, the elements in the vector $\vec{x}$ are ordered. In this case the value of x_i is as important as its position i in the vector $\vec{x}$. Consequently the search space is even bigger.
Bigger is the search space, more the solution may be expensive to find in term of time computation.
The goal of the EA family is to optimize a solution ($\vec{x}$), in order to have the best solution

possible for the problem, where the best solution is defined by a cost function ($f(\vec{x})$). The objective is to maximize the value of the cost function regarding an input solution ($\vec{x}$).

$$\max f(\vec{x}) \tag{3.2}$$

The EA maximizes work towards to the global optimum solution $\vec{X}$.

$$\max f(\vec{x}) \leq f(\vec{X}) \tag{3.3}$$

Where $\vec{X}$ is the global optimum solution.
Optimizing a solution $\vec{x}$ is not always enough to solve efficiently the problem. The proposed solution has to observe the constraint linked to the problem. The constraint function can be various depending on the problem. As example, one naive constraint function could be the boundary of the search space. To reach a set of m constraint function, E must be taken into consideration :

$$E = \{e_1, ...e_m\} \tag{3.4}$$

Where e_j is the j^{th} constraint function of the set E. Thus, that means the cost function F has to include the constraint set E and maximize the value of the cost function $f()$.

$$\max F((\vec{x})) = \max f(\vec{x}) - \min(\sum_{j=1}^{m} e_j(\vec{x})) \tag{3.5}$$

F is the cost function, which includes the constraints, to evaluate a solution $\vec{x}$ to optimize the problem.
The EA manage the optimization of the problem based on the cost function $F(\vec{x})$ by applying different meta-heuristics. The meta-heuristics use different methodologies to optimize an initial solution. The optimization is more or less global depending on the algorithm applied. Mainly the optimization methods are based on the generate new sets of solutions. The new sets is made by evolving the previous sets of solutions. A set of solutions is also called population. Where a population is defined as $pop = \{\vec{x_1}, ..., \vec{x_p}\}$ with p is the number of individual in the population. The solution found by the EA is not necessarily the global optimum and for some type of problem (as Np-hard) it is impossible to confirm the nature of the solution (global or local).
The risk in this case could lead to an endless optimization. To avoid the infinite optimization, the end has to be fixed depending on some stopping criterion.

3.2.3/ Stopping criterion

To determine a stopping criteria , the proposed solutions in the following part are more especially adapted to the algorithms like GA, PSO, ant colony, mimetic and other EA working with a population to optimize.

The EA are mostly efficient in the problems with many local minimums. Nevertheless, due to numbers and size of the local minimums, it can be difficult to ensure if a solution is the global optimum. To know if the global optimum is reached, the method applied must ensure that no other solution can be better. Therefore, the solution found must be always exactly the same or equivalent in term of cost. The optimization process must be repeatable (same input gives the same output).

Only the deterministic method can insure to have a global optimum solution. The convex problem has only one global optimal and no local minimum by definition.
The EA optimizes a solution to be as efficient as possible (not the optimum). Because the uncertainty of the EA optimization, it is impossible to call a calculated solution as global optimum.

Consequently, how to determine when the optimization has to be stopped? Because at some points, it is pointless to continue the optimization, the best solution has been reached or the optimization is locked in some deep local optimums.To determine the stopping criteria, three possible ways are commonly used:

- **Fixed time criterion**: stops the algorithm after a fixed number of iterations or time limit. The limit is measured in term of computation time, also the number of iteration must be fixed by the user. The main interest of this method is to control the computation time of the problem which requires a solution in a determined time. The main risk of this method is represented by a stopping criteria reached before an efficient solution computed.

- **Update criterion**: stops the algorithm (before the convergence) if no better solution is found after a predefined number of iterations. This criteria can be useful for complex problems or if many solutions can have the same quality. The advantage of this stopping criteria is that it can stop before the convergence with a solution close or identical to the one reached at the convergence. The inconvenient remains in the concept of "early shutdown". In fact, if the update criteria is not properly chosen, the optimization may stop at the beginning.
  To usage of such stopping criterion a correct number of iterations has to be selected. It must be sufficient to give time when the meta-heuristic is locked at a local minimum. A long time lock in local minimum may append mostly at the early time of the optimization due to a too sharp initialisation or in the late optimization time (when the solution is already well optimized).

- **Convergence criterion**: stops the algorithm by waiting the convergence point. The convergence is reached when the actual population is composed by a set of identical solutions. Which means that the same solution has been found by all the individuals of the population.
  The best solution found push the other to evolve in the same direction, by contagion all the individual of the population evolves to reach the convergence point.
  This solution found during the optimization process is supposed to be the best one. In this case, the population has been converging to an supposed optimized solution.

The stopping criterion presented can be organized as a combination of the three methods to have an efficient and flexible solution as presented in the following example :
The mixed stopping criteria has to combine a fixed time criteria and an update criteria. The advantage of the fixed time criteria is to avoid the case of an almost infinite optimization loop append, due to an impossible convergence.
Combined with the update criteria the other advantage is the ability to stop the optimization before reaching the complete convergence and before reaching the time criteria limit.

The combination of these stopping criterion always provide a solution optimized in a reasonable time (fast, efficient and time predictable for the worst case).

3.3/ Genetic algorithms

Among the evolutionary algorithms, one is very close to the Darwin theory by reusing the operating principle of natural selection and is also based on the genetics with the influence of the crossover and mutation (see section 3.1.2). This algorithm is called Genetic Algorithm (GA) and was introduced for the first time by Holland in 1962 [87].
The GA is one of the fundamental algorithms of the EA. The GA became popular at the late 80s and early 90s, particularly with Goldberg works [93].Since the 90s, many details were redefined and explored in the knowledge of genetics to have a huge set-up and operator available for the GA.

The following sections will try to list the most promising aspects of the GA mechanisms.

3.3.1/ Chromosomes

Like in the Biologic field, the chromosomes contains properties (with the genes and DNA) of individuals. A primordial issue the goal is to optimize a problem involving the GA is to define properly the chromosomes role, taking into account of different aspects of it.

The chromosome is used to design the problem and it has to represent a solution. To do so, it is important to know the problem and identify clearly which parts of the problem requires to be optimized and what is the range of the search area.
Depending on those previous facts, the coding can be direct or indirect.

- The direct coding: consists in direct mapping of the gene corresponding to the elements of the solution. Using the direct coding may simplifies the output of the optimization by returning it back to an element directly appropriate to use.
- The indirect coding: not directly appropriate for usage and needs conversion to be used. One example of indirect coding is the willingness to introduce redundant genes inside each chromosomes. The conversion must be done by a heuristic. The interest of the method lie inn the ability to make a strong constraint adapted to the problem as presented in [94, 95], where it is used to solve a complex scheduling problem with many constraints.

To define the chromosome, direct or indirect coding is not the only element to be taken into account. The coding structure is also important to the chromosomes design. The coding structure has to encode the solutions in the chromosome like presented in [95] and [96]. Among the possible coding structures, foud main categories can be considered as basic coding structures: the combinatorial coding, binary, the real coding (also alphabet) and tree coding. Moreover the choices for the chromosomes must also take into account the wanted solution in terms of number of dimensions to represent and their boundaries.

Binary Coding The binary coding offers the formatting of the chromosomes as a bit string in which every genes of the chromosome can be coverted on pack of bits which is a boolean. This coding method was studied since the beginning of the GA by Goldberg et al. [97] and also used in other EA like PSO in [98]. The binary coding is more efficient in the small search space or when the size of the chromosome is not too big. The advantage of the binary coding is the possibility to introduce lot of variety during the process of optimization [99]. The variety is traditionally induced by the mutation but in the case of the binary coding, the crossover introduce variety by the potential split of the gene in 2 pack of bits.

The combinatorial The combinatorial coding is commonly applied in some specific problems where the goal is to order all the elements. In this case, the position of the gene in the chromosome is primordial as in [100]. It is characteristically used to the problem as TSP [79] (Travelling salesman problem). When the combinatorial coding is used for problem that aims to optimize the order the elements of the chromosomes. With a combinatorial coding, each chromosome already has all gene of the answer. An example, in the TSP, the aim is to order various cities (in the problem of TSP every gene represents a city) and all the possible cities are included in the initial solution (chromosome). In this case, the problem represents the optimization of the combination of the elements composed by the initial solution. One evolution of the combinatorial traditional coding is to add and remove some genes that obviously affect the size of the chromosome, for example when the goal is to find the shortest distance in the tree [101].

Real Coding Real coding or integer coding is considering every genes of the chromosome as a number to optimize. This number can be a real or an integer and may have an infinity of possibilities. In fact, the value does not have an infinity of possibilities because the constraint induces by the machine and limits of the problem itself too (size of the search space). This coding is used when the search space is large and also can be efficient when many dimensions need to be optimized. But most of the time a special attention should be placed onto the operator, because in many cases the operator may be adapted or redesigned depending on the problem such as in [78], where the operators are adapted to look for close neighbors.

Tree coding The tree coding use the tree representation to take care of the hierarchy, but this method is not really popular and not flexible to all cases. The advantage of tree coding is this ability to go further than a combinatorial representation. An example in [95], the tree coding is used with the GA to optimize new phone network or gas/ water pipeline where the relations between the elements are primordial. In [95] is presented the interest of tree coding for the intrusion detection system.
The four coding methods presented are not the only potential coding. There are the most common and the roots of other coding . Many other have been developed and studied with direct or indirect coding to fit with specific problem. Thereby, in literature, a survey is dedicated to the encoding chromosome of the GA [94]. The survey [94] proposed an explanation and a search for a robust coding usable depending on the problems.

3.3.2/ Cost function

The cost function or the fitness function has an essential role in the optimization process. The aim of this function is equivalent to quantify the quality of a solution. This is primarily applied to the GA and in most of the optimization process uses meta-heuristic. The cost Function is a compass for the meta-heuristic during the optimization. The cost function is dependent on the problem and should be designed or redesigned depending on each specific problems. Once the cost function is designed for a problem, it can be used to test different other algorithms of optimization which requires also a cost function. Because the cost function is exactly the same, it becomes easier to compare the results from different algorithms as discussed in [102] and also in chapter 5.2.Once conceived, the cost function is considered as a black box by the optimization algorithm and encloses most of the complexity of the problem. Obviously, if the cost function is not designed correctly with all the constraints and the objectives, the optimization will fail.

Multi objectives

The cost functions are traditionally made-up for problems with only one objective, but in the recent years, several solutions have been adapted for multi-objective. The goal is to optimize a problem with few sub-objectives included. These sub-objectives can be at some points contradictory and a trade-off must be choosed during the optimization process, based on the grading made by the cost function.
The cost function for Multi Objective Problem (MOP) is discussed in the survey of Zhou et al. [90]. In Zhou et al. [90], one of the ways to solve the MOP is to adapt a classic mono objective algorithm in multi by customizing the cost function. The customization of the cost function proposes a way to evaluate a solution not depending on one objective but with all of them combined in the same function instead of several cost functions. Also, other solutions are discussed in [90], like using coefficients to order the objectives priority or by reducing the problem into several sub-problems.

Constraints

The goal is to satisfy the objective(s) while taking into account the constraint(s). Consequently the cost function has to take into account and subsequently integrate the constraints. Previously, in the chromosome (3.3.1), the strong constraint was established by the coding, especially by imposing limits in the real coding or using indirect coding. But it is feasible to impose some additional soft constraints to the system by adding rule in cost function. The rule corresponding to the constraint is helping the optimization to move towards a good choice not by imposing strong constraint but by affecting ”bad points” or ”good points” depending on the constraints.
The soft constraints can appear a bit irrelevant, but they can be a good trade-off between two contradictory objectives and some other strong constraints. Indeed, a soft constraint can be simpler to implement and faster in term of computation time compared to a hard constraint.

Optimization and time computation

The cost function should be designed carefully and have to pay special attention to the computation time. Indeed, the high frequency call of the function may generate some heavy slowdowns.
The cost function has a special importance in the optimization process with respect to the rating of all the individuals at every generations. If we consider a GA with 100 individuals (it is a common number of individuals not to much and generally enough) and a convergence after 100 generations (it is good minimum in order to avoid a premature convergence), in this case the cost function will be called at minima 10 000 times. Due to this important factor, it is primordial to carefully design the cost function as it is specified in [103].
It is common to have several thousand or billion calls of the cost function during the optimization process. Hence the importance to have a function timeliness and economic in resources.
Even the cost of this function must be the most accurate, in some cases like in [104], a complex calculation or noise can affect the reliability of the cost function which will be impacted on the quality of optimized solution; but despite the noise and the weakness of the cost function, the GA can optimize and give a solution. But it is advisable to have a function accurate and as reliable as possible to have an efficient solution.
To conclude with,the importance of the cost function, it is notable that it is a major piece of the GA and the choice of the design of cost function associate dto the chromosome representation will affect the result but also the setup of the GA and can generate some latches.

3.3.3/ Population

The population gathers a number N of solutions (or individuals) from the same generation. The individual can be represented into one or more chromosomes (instead of having redundancy as in [94]). Commonly each individual is composed by one chromosome. Therefore the two terms are regularly inverted in the literature.
At every generation most or all the population is renewed (by using the selection and operator). Indeed the population can have an effect on the convergence and the result of the EA and different strategy or set up exists. About the population, two main points needs to be studied : the first is the initial population and the 2^{nd} is the size of the population.

Initialization of the population Initialization of the population is one of the fundamental questions that may affect the convergence of the problems. There are mainly two common proposed solutions with one using the full random generation or using efficient and already approved heuristics.
The method based on heuristic involve a perfect knowledge of the problem and can not be applied for all the problems. To briefly state about the advantage of using heuristic, it can give very good starting individuals at the beginning the optimization and also using the heuristic may give a solution more compliant according to the constraints of the problem, especially if the problem include many constraints hard to satisfy.
Although this advantage of using a heuristic to find the first generation of the population can become a handicap and push the GA in the direction of the potential local minimum.

Indeed, using one heuristic to build the first generation can have a population too similar with not enough variety. The variety is essential to run through all the search space and allows to not converge too fast in a local minimum (see [105]).
If using a heuristic to build the initial population is not always the good solution, another solution, more versatile, is represented by the usage the randomness to find initial population. The randomness generates each individual randomly in the search space. One of the advantages is that the individual can be well distributed around the search space to cover most of it if the size of the population is big enough. The random distribution allows the algorithm to cover a wide part of the search space quickly during the first generation and the spreading of the population is a good source of variety. The random initialization is commonly used at the expense of the heuristic solution.

To initialize the population, the third method combine the full randomness with the one based on heuristic. In this case, a random solution is applied before, the heuristic is used to refine the initial solution. Otherwise, different heuristics are used randomly to generate all individuals of the first population. Other combinations are possible: see [101] for more details.

Population size Another key point remaining tis the consideration of the size of the population. The size of the population and its effect on the convergence are studied in many articles [105, 103, 106, 107, 97, 108]. The first point is to find the appropriated size depending on the problem. Indeed if the size of the population is too small the variety of the population can be too short and the algorithm can converge too fast [103]. Contrarily, if the population is too large ,the waiting time may become too long. Like that, choosing the appropriate size of the population is not trivial and remains a crucial step. To find the best size of a static population, the unique procediure is to do several experiments and compare the results, as performed in [106].

The population can also be adapted dynamically or auto-adapt its size during the optimization. Commonly, the number of individuals in the population needs to be important during the first generation but close to the convergence. The population can be reduced to obtain time economy. In [109], the size is obtained using probability distribution. It can also be fixed by a linear equation or even more complex, in function of the progress of the cost function and the variety of solution as shown in [103]. The population have an important contribution in the convergence computation, if the size of the population is static, dynamic or auto adapted, the convergence can be faster with more or less quality for the solution. The convergence is studied in the [108]. But the convergence is not only linked by the population size but also the selection, the coding and the operator choice are important aspects of the GA as shown in [106, 107].

3.3.4/ Selection mode

The selection mode is the method used to select in a population the individuals the most able to reproduce. It is an important key point corresponding to the natural selection in the Darwin theory.
The choice of the selection mode is primordial and affect greatly the quality and the speed of the optimization process. This choice needs to be done depending on the problem. A selection mode applied on a specific problem can be efficient (in term of answer quality and time convergence) but the same selection mode applied on a completely different

problem can be inefficient for it. The selection mode must be selected or adapted for each kind of problem. To choose the selection mode the testing of different method has to be performed.
One of the objective of the selection mode is to keep enough variety in the population to avoid an premature convergence. Also, too much variety in the population and especially not a the beginning may artificially delay the convergence. A good selection mode has to be a trade-off between too elitist selection (not enough variety) and a too permissive selection (too much variety).
A multitude of selection mode have been developed during the time (few of them have been listed in [96]). Making a choice among this wide list of possibility is difficult. The selection mode among the most common or representative is presented below:

Elitist selection - The elitist selection is more like a subgroup of selection mode. His particularity is the usage of a deterministic way to select the best individual depending on the cost function. The best or some of the best are selected directly for the next generation such that the best individual are preserved and there is no risk to lose a good individual during the crossover, mutation and other operations. The few best individuals selected are used to engender a new generation. This subgroup of selection is studied in [110, 105] to estimate the efficiency of convergence using this selection mode or in [92] applied in the multi-objective problem. It appeared on these article that the elitist selection is efficient, converge quickly and the best individuals generated are preserved for the next generation. But unseemly, it may sometime converge prematurely because of the difficulty of keeping enough variety with this selection mode. Consequently, some of the elitist selections have been customized to preserve the variety.

The roulette wheel - The roulette wheel selection is at the same time one of the older selection mode used since 1989 but also one of most inspiring. The roulette wheel gives many methods inspired by this one like for example remainder stochastic sampling in [111]. The roulette wheel selection had a basic operating. Every chromosome is represented in the wheel and the size of the wedge are depended them the quality of the chromosome. This quality is computed from the fitness function. With this technique, the chromosome with the best fitness function has the most luck (but not necessarily) to be selected. Once the wheel was built by the random sampling can begin to select the new individual of the generation for the next generation. The wheel turns until all the individuals are selected. This method helps the better chromosomes to be more represented in the next generation but also accepted to have some time more or less bad individual conserve for keeping the variety. More the GA work more the size on the wheel of the best solution will increase and help to converge.

Tournament selection - Tournament selection is one of the most used selection in this last decade. It is working as a tournament. The first step consider a random creation of few pools with all the individuals of the actual generation. When the pools are created, the tournament can begin and the best chromosomes of each pool (depending on the cost function) are selected to build the next generation. This selection by randomly

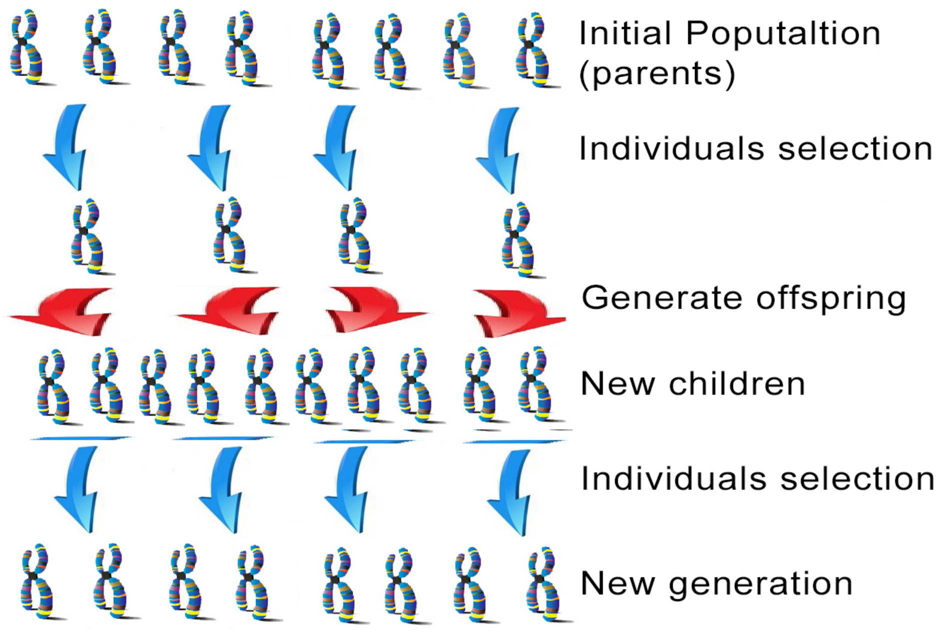


Figure 3.3: The selection implyed by GA mechanism.

building the pool could preserve the ”lesser” individuals but the best chromosomes are always used to build the next generation. The tournament selection is very efficient to keep diversity and also give a chance to have a fast convergence as it is explained in [105].
Obviously, the size of the pool has a really big influence on the convergence as is studied in[105, 104]. The conclusion of these papers about the pool size is: bigger the pool (4 or 5 individuals), less diversity; and the convergence goes faster, and can end stuck in a local minimum due to a premature convergence. Contrarily, the small pool (2 chromosomes) keeps the variety but the convergence can be slower. Finally the size of the pool is also one other parameter to take into account and the size of the pool is also strongly linked to the population size. Among the selection mode presented, the tournament selection is one of the most efficient to manage the variety and this behavior explains its wide popularity to solve engineering problems.

The last element to consider about the selection mode is its frequency of application. Indeed the selection of the individual may intervene at two occasions (see figure 3.3). The first, the more conventional, corresponds to the selection of the parent with the ability to reproduce. The second is the selection of the children good enough to be part of the next generation. The key interest of this is to be able to generate new children until the selection criteria are rich in order to have a population with acceptable children in sufficient quantity (this method is more efficient on the problem with lot of hard constraint).

3.3.5/ Operators

The operator have to aim the design of the next generation by generating the offspring. Once the parent able of reproduction has been selected (using the selection mode saw previously), it is time to engender a new children. The techniques to create a new population are numerous. Among them, the most common are: the heritage (or selection), random generation, the crossover and mutation.

Heritage or selection: The heritage or also called selection is a simple copy of the best parents to the next generation with no modification. This operator is commonly used to keep the best individual in order to ensure the salvation of the best solution if no upgrade is made by the other children. Indeed the other operator may propose degenerated children which is some time worst than parents and the heritage.
The crossover and mutation have a more interesting mechanism. They are also the basis of many other customized operators and the understanding of the basic crossover and mutation required to use it.

Crossover: The crossover operator is directly inspired by the biology. As two mammals reproduce to have progeny. Half of the genetic material of the two parents are used to create a child. The crossover operator is mixing two individuals to create a new child. The aim of this is to merge two workable solutions to have another solution potentially a bit better. To merge the two individuals, it is existing many way like studied in [101].

Mutation The mutation operator have to aim the addition of diversity by mutating some random genes of the chromosome. This mutation must be randomly chosen and is important to keep the diversity of the population. As for the crossover, different mutation are existing [101]. The mutation mechanism affect only rare gene in all the population. The genes mutated are randomly selected and will be based on randomness also.

Customized operators This two principal operators need to be chosen carefully and in most of the case redesigned. The redesign of the operator is essential in many occasions. One of the first reason to redesign part of the operator is to fit well to the problem. As an example, the mutation can be redesigned to explore the search space with a logic of close neighborhood as shown in [78].
The second reason is the linkage of the chromosome coding. Depending on the chromosome coding chosen (binary, combinatorics, real, ...), the operator have to be adapted. An possible example could be to modify the mutation and crossover to preserve the genes in the real coding. The same operator can not be used for combinatory coding or binary.
The third is infrequent but in some cases the operator have to be adapted depending on the selection mode chosen. An example is given in [112] with the crossover and the mutation for an elitist selection.
Also another reason to redesign the operators is to have operator fitting to some of the hard constraints linked to the problem. In order to have operator able to create children considering to some hard constraints of the problem.

Operators rate Another important element to take into consideration after the choice of the operator and their implementation is rate of each of them. The rate of an operator corresponds to the usage percentage of the operator on the chromosome, the higher the rate, the more this operator will be used at each generation. In the mutation, the rate can be globally understood as a chance for one gene to be muted. Finding the best rate for every operators became a real challenge. The best solution to find the appropriate operators and their associate rate for a specific problem unfortunately offers no other choice then trying combinations as shown in [99, 106, 109].

To conclude on the operators, they are an important factor to evolve generation after generation. The operator have to keep the diversity in order to have an efficient convergence. The question of the diversity introduced by the operator has been studied in [79, 104] [101]. It is revealing that a good static configuration to keep diversity [105] of chromosome is to have crossover mutation with high rate of crossover and small rate of mutation. But the rate of the operator can be adapted depending on the search space, the convergence and other elements. Some research were proposed on dynamic adaptation of the rate of the operator depending on many external factor or using probability as discussed in [100, 109, 113]

3.3.6/ Setting and set-up

The previous section introduced the different aspects of the GA and gives the key to understand the different elements and the mechanisms of the genetic algorithm. Besides the GA explication, it appears that many primordial choices to set-up properly the algorithm are necessary. Part of these choices are interdependent and the connection between the different parameters can make the set-up delicate. Also the GA have been studied for decades and many variants were developed over the time to make the GA more efficient giving even more choices but no general set-up formulated. To evaluate the performance of a set-up, the quality of the solution found is used but also the speed of convergence in term of number of generations, and also the variety of the chromosomes at each generation until the convergence. The variety is one of the factor relevant to explain the convergence speed and the solution quality. The variety is almost opposed to the convergence. It is when the chromosome are all very different in the same generation but also generation after generation. A high variety is ideal at the beginning because this allows the optimization enlarge the search space and potentially help to avoids the local minimum. As an example, if the final solution converged in a local minimum after a too quick convergence, it means that not enough variety have been introduce during the optimization.

During the choice of the ideal parameters for GA, the variety is an important element to be preserved and more precisely at the beginning of the optimization to cover the search space.

However the GA stay complex to configure because of all these parameters to have a good answer quality in a reasonable period of convergence. Many aspect need to be adapted depending to the problem, constraint, size of the search space and . . . .
Using the simple GA that mean configure a set of parameters. The parameters can be formalized as a vector like in [106]. This formulation is especially efficient test numerous settings. To set-up properly a GA few questions need to be posed as in the table 3.2 and few settings must be tested as in [99, 106, 109].

Inspiration or group	Algorithm
Coding chromosome:	Which coding choice? (binary, combinatory, real,...)
Cost function:	How to quantify the solution quality?
Population :	What size?
	Which initialization? (random or heuristic)
Selection mode:	Which choice of selection mode?
	And depending on the mode chosen which setup?
	As for tournament selection the size pool, the wheel distribution for roulette wheel or the number of parent selected for elitist...
operators:	Which operators?
	Which implementation (customized operator or not)?
	Which rate for each operators?
Stopping criteria:	Which stopping criteria to use?
	If not by convergence what are the boundaries?

Table 3.2: Important clarifications for setting up a GA.

3.4/ GA TRENDS

Whether GA is a relatively recent algorithm, it was largely studied during many years and has progressed tremendously. To follow the trends of the GA, the survey written by [114] for the Simple Genetic Algorithm (SGA) is a good point to understand the progress prior to 1994.

In this article [114] the authors explains the SGA and the signification of the natural selection with the possible modification to cumulate. Also the GA improvement in terms of performance is discussed. The SGA is customizable depending on the implementation. This behavior gives a big importance to the problem formulation and the consequences on the solution. This survey is relatively hold (from 1994) and other more recent are rather focused on the Multi Objective Evolutionary Algorithms (MOEA), which includes many different shapes of the customized GA to satisfy the multi objective problems [90]. Although, the papers are concerned about the multi objectives and many references are made to highlight the recent advance on this field with different types of adaptation such as the multi objectives evolutionary algorithm decomposition (MOEA/D) [115]. Key idea is to decompose the problem into sub-problems and each sub-problems are weighted by the neighboring relation between the sub-problems and then are aggregated.

It exists many other MOEA presented in this paper suchas Non-dominated Sorting GA II (NSGA-II) [110]. These algorithms are using an elitist selection to optimize efficiently the problem without having to deal with sorting the different solutions depending on the objectives. Some other MOEA as QGA for Quantum-inspired GA [110, 116, 117], Non-dominated Sorting GA (NSGA) or BMPGA for Bi-objective Multi Populations GA, are examined on this survey[110].
The GA have been studied for different objectives and optimization problems. These surveys give a fast view of the GA formulation and specific customizations for GA application field, as the following example.

- In [118] are interested on the problem of clustering and use GA for have a non-supervised clustering. Also in [118] show the different implementations and customization of the GA, adapted to the problem of clustering.
- In [119], the GA is applied to the problem of pattern recognition. This problem is a complex multi optimization problem. The GA is used to optimize the classification, the training and the research of a set of efficient features.
- In [96], the GA is applied into security problems to control computer access in the network and prevent attacks. In this case the GA can be used to optimize the classification of the access and this way detects the legal and authorized access then the hacking attempt.

GA have been well studied and the literature is vast about it. These last decades, the GA have been used for multi-objectives problems, but its popularity has decreased in favor of algorithm which requires less configuration or other algorithm more oriented on learning with memorization.

4 Problem modeling

Contents

The following chapter is dedicated to the in details formulation of the problem of cameras positioning . The objective here is to find the position for a camera set or waypoints. The position of this set of waypoints has to be optimized in order to cover most of the area. The area that has to be covered may be vast and composed of complex zones. A good formulation is essential to design an efficient cost function. The cost function is used to quantify the quality of the solutions. It is a crucial element for the optimization processes. This chapter present a formal definition of the problem based on the literature as well as our proposed formulation. The formulation proposed is adapted to optimize the problem of cameras or waypoints positioning depending on many constraints such as a complex and vast map or cameras mounted on UAVs.

The following sections are focused on the estimation of the area coverage depending on the camera parameters.

To estimate efficiently the area covered by a given camera set, some points needs to be clearly defined. First of all the area itself. The question of ”How could we represent the area?” is primordial. The camera definition with the question of ”How could we estimate the camera projection?” is also an important part of the problem design. The third part to take care is the constraints which must be added to the systems. These three parts are discussed in the following sections. The area definition is discussed in the section 4.1 which focus on the grid design. The camera definition is discussed in the section 4.2 and the third part about the constraints is addressed in the section 4.3.1.
Finally, when the problem is clearly defined, all the different elements are integrated to obtain an efficient cost function operable for the optimization process in 4.3.3.

4.1/ The map

The first part is equivalent to the estimation of the area that needs to be covered. Following this direction, many methods have been developed and most of them are based on an occupation grid G of the area. The occupation grid is a discretization of the area with numerous points.

$$G = \left[g_1...g_i...g_m\right], m \in \mathbb{N} \tag{4.1}$$

Where m is equal to the number of points in the grid. The occupation grid is placed on the area to cover. Each point g_i of the grid G should be covered by a camera. In the Figure 4.1 the grid G and the points g_i are represented, as well as one camera and this associate projection on to the grid. Consequently some parts of the grid is covered (red dot in the Figure 4.1).

$$\forall g_i \in G, B(g_i) = \begin{cases} 1, & \text{if } g_i \text{ is covered by at least one camera} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

B is the set of grid points which associate the value ”1” if covered by at least one of the camera. The size of B is relative to the size of G.
The design of the grid is an important element of the problem formulation. Many solutions were proposed, with different advantages depending on the situation (constraints and objectives).

4.1.1/ Designing a grid map

The following section is focused on the different grids design proposed in the literature. The usage of grid to describe the area that must be covered is common and several designs have been invented and used depending on the constraints and objectives.

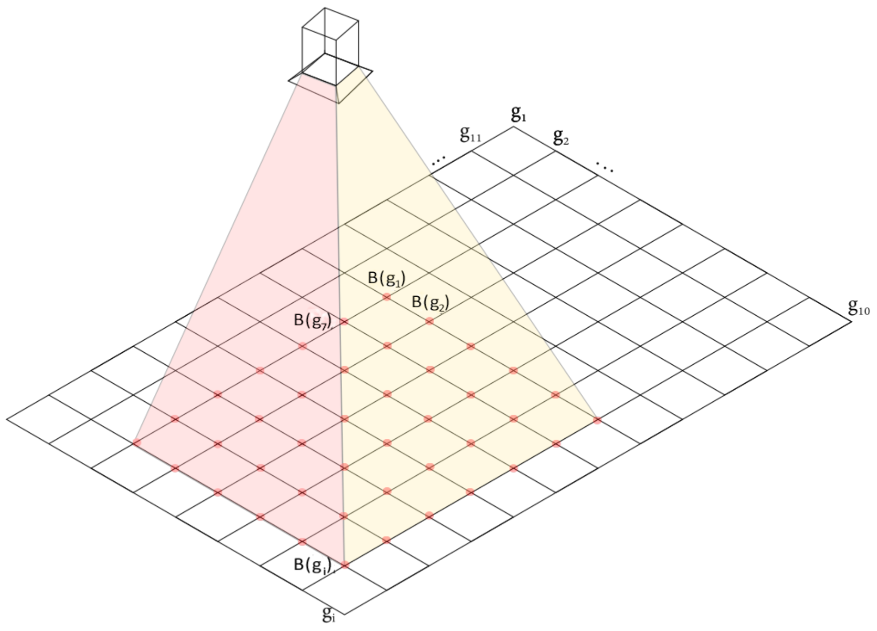


Figure 4.1: Camera projection onto a grid. The grid G is placed on the floor to discretize the area covered with numerous grid points g_i. The point covered by the camera, belonging to $B()$ are highlighted in red.

4.1.1.1/ Sampling frequency

The grid map is used to discretize the application area. The discretization of the area may vary and the area can get a high or low level of discretization. The level of sampling frequency has a bearing both on the problem formulation and on the optimization process.

High sampling frequency

The high level of discretization or the high sampling frequency of an area is characterized by a big amount of point g_i to describe an area. The big amount stands for having an important density of point g_i and consequently a high value of m.

The main advantage of this approach lies in having a better estimation of the coverage. The more the area is finely discretized, the more the estimation of the coverage will be sharp. In [14], an example of high frequency sampling is given in order to have sharp estimation of the area.
The high sampling frequency allows the cameras position to be much more accurate and requires smaller adjustments.
On the other side, a too high sampling frequency leads to high computation time. Rather to refine the solution, a too high level of discretization of the area will make the optimization longer and more complex. Indeed, to control the coverage, it is necessary to control if each point of the grid is covered by a camera. The more the size of the grid, the more unity test of coverage (see in Section 4.2) are necessary, and it impacts each step of the optimization process. Consequently the size of the grid will greatly affect the computation time.

Also the high sampling frequency will affect the positioning of the cameras pose. In fact, higher is the sampling frequency of the area, freer the pose estimation of each camera will be.

Low sampling frequency

At the opposite, a lower sampling frequency can be a good solution to upgrade the convergence speed of the optimization process as this behavior has been presented in [49]. In Zhou et al [49], a small value of m is chosen to have a real time solution for small areas with just few cameras (up to 20). On the other side, the low sampling frequency may generate a bad estimation of the area covered due to the too low density of points in the grid. The low density of points may give an approximate view of the covered area and some uncovered areas (black hole) can appear between the points of the grid. These uncovered areas can be too small to be detected due to the low density of points. In this case, the optimization cannot take into account the uncovered area sand the solution given after an optimization will be, in a real environment, of bad aspect. Concretely, the difference between the coverage estimation of the discrete area and the coverage estimation in continuous time must be high.

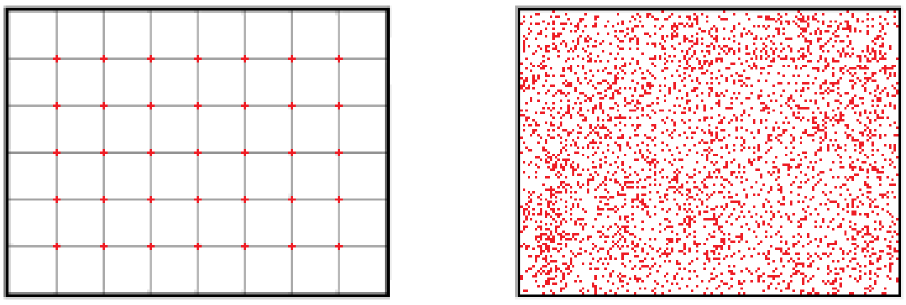
Low or high sampling frequency

Finally, the too low sampling frequency, instead of gaining computation time, may affect the quality of coverage estimation. Contrarily, the impact of a too high sampling frequency has consequences, as summarized in the Table A.1. The sampling frequency should be carefully estimated.
The density of the grid has to be adjusted depending on the goal and the precision required. One of the solutions proposed in Zhao et al [6] is a progressive refinement by increasing the grid density. Zhao et al [6], despite a low density of points at the beginning, proposef to increase the number of points slowly to obtain a better density and refined results. Also the increasing points frequency is applied to refine the solution and add more cameras at each step of the optimization leading to avoidance of uncovered area.

4.1.1.2/ Cameras pose precision

The sampling frequency of the grid is often linked to the pose precision of the camera as discussed previously. In some cases, such as in [47, 42, 75, 98], the available position of the cameras are strongly linked to the grid map and the sampling frequency. Increasing the number of points in the grid will also increase proportionally the number of possible positions for each cameras. More the area is finely defined more it is necessary to slightly adjust the cameras positions. Consequently the possible camera positions and the search space are increased to finally allow a more refined solution but also a more complex one.

On the other hand, the camera poses can be limited at some specified areas of the map. In [14, 6], the cameras can be placed against the walls. In [6], every wall of the boundaries



(a) Grid with uniform and regular distribution.

(b) Random distribution for represent the map.

Figure 4.2: Map representation using random distribution or uniform grid.

can hold a camera. In [14], the camera can be placed only on specified zone (as shown in the Figure 4.3, the green wall shows the available position of the cameras).

The camera poses precision has a similar impact and consequence on the result as the grid map. A to big number of possible poses allows an increased precision but greatly increase the complexity, while a too small number of possible poses helps to converge faster but will lead to lower precision.

4.1.1.3/ Distribution

The distribution of the points over the grid, is an important factor to manage for the design of an occupation grid.The points distribution can numerous and adapted to different specific case. Mainly two distribution are used. The grid pattern distribution is the more often seen in the literature and in a lesser extent the random distribution. These two distribution are illustrated in the Figure 4.2.

The random distribution is used to describe the area to cover as shown in [47, 14].
In Hoster et al [14], the points of the grid are randomly distributed to describe the area to cover. The advantage of this article remains the usage of random distribution to manage the density of points in some specific regions of the area. Especially, by increasing the density of points on some specific zones of the area. The increased density allocates more importance to these zones (see Figure 4.3).
Indeed, higher density will affect the optimization process. The area with more density will be comparatively more profitable to a low density area. In these cases, the zones with high density are covered in priority. This mechanism is even simplified due to the random distribution.

Hengel et al [47] have tested the random distribution and the uniform grid pattern distribution before conclusing that the grid pattern and the random distribution proposes globally observe the same result when there is no priority zones in the area. Based on this observation, authors (Hengel et al [47]) decided to use an uniform grid because of its simplicity of implementation. A hybrid distribution is proposed in Zhao et al [6]. The idea is to reduce the number of points in the uniform grid when it has a to high density. To reduce the number of points in the grid, a random selection is applied. This hybrid implementation of an uniform grid with a random selection to decrease the sampling frequency is an example of hybrid distribution.

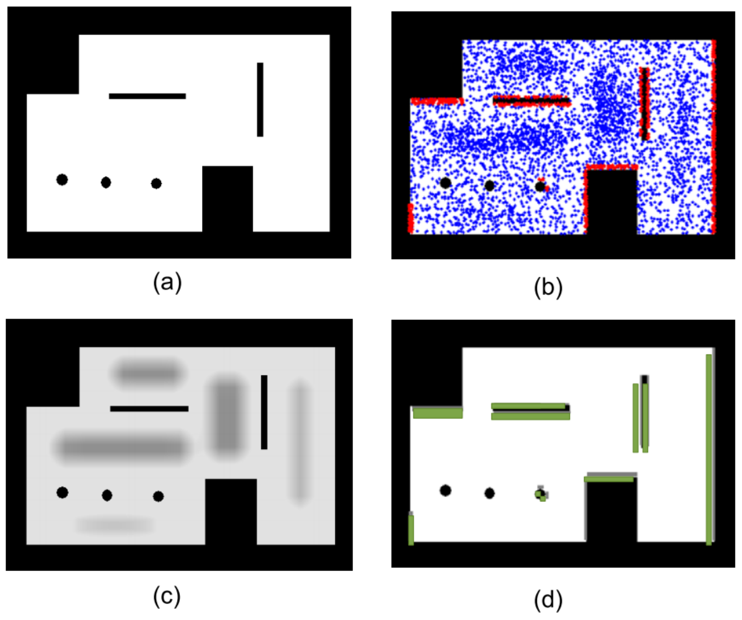


Figure 4.3: Example of map that must be covered with a randomized area sampling. Illustration from Horster et al [14]
(a) Area to cover.
(b) Area with the random points of the grid to cover
(c) Importance weighting of the area (zone of the interest).
(b) allowed position for cameras position.

4.1.1.4/ Spatial modelling (3D or 2D)

The spatial modelling has an important impact on the grid design. After imposing the necessary density and the distribution of the grid, the position of the points g_i in space which composes the grid has to be discussed. Commonly, the occupation grid is placed on the floor and calculate visibility only in 2D by computing the camera projection as in [55, 42, 49, 35, 14, 6]. But depending on the context, the grid could have interest to describe a 3D space. Hengel et al [47] calculated the visibility according to relevancy: for example, on the upper torso or head of the possible target rather than on the floor. In this case, a 2D grid is proposed inside the 3D space in order to characterize properly the volume by placing the grid at a specific height.

In Chrysostomou et al [15] or also in [120], the grid is formalized in the full volumetric space by numerous “control points” to control the area (illustrated in Figure 4.4). The points of the grid are uniformly distributed along the axis of x, y and z. The grid are formulated in the volumetric space by using an uniform 3D grid distribution. The uniform grid distribution in a 3D space increases greatly the complexity due to the important augmentation of control points (g_i) and their respective dispersion. Its implementation is unusual due to the increasing difficulty of the optimization process. To avoid the increasing complexity, the 3D grid is replaced by a 2D grid in the beginning at a specific height to optimize the coverage. Nevertheless, some solutions were discussed [43, 47] to take into account the volume of the area that must be covered which cannot be only limited to an occupation

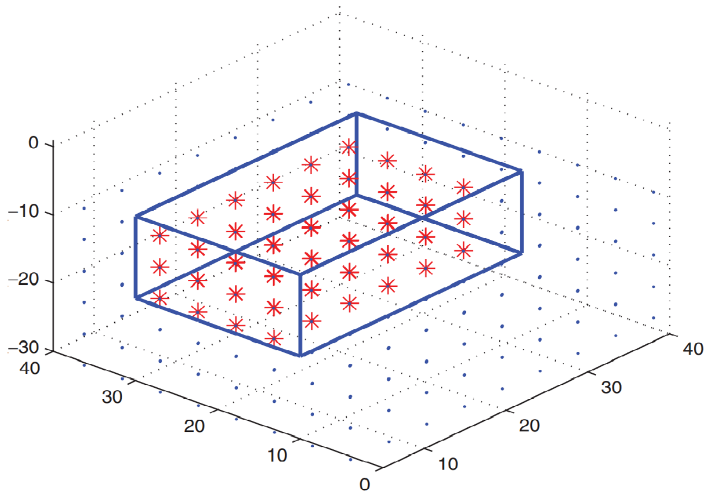


Figure 4.4: 3D grid with uniform distribution of the grid points. Illustration from [120]

grid along the axes x and y as if using a 2D grid.

In Hengel et al[47], the authors were focused on estimating the area to cover inside three floors of a building. To estimate the coverage in the building, the grid was placed at each floor. This solution is efficient in order to limit the number of grid points compared to a full volumetric description of the area. Also, this design allows to keep the three dimensional information by adding a layer at each level of the building (see Figure 4.5).

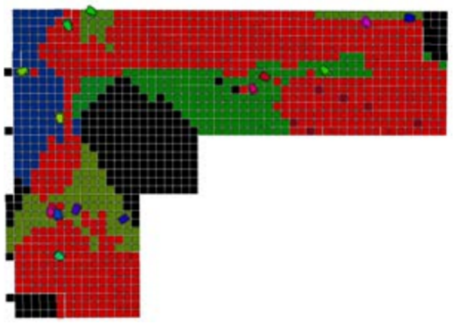
Another way to deal with a volumetric space while not using a greedy 3D grid distribution lie into adaptation of the 2D grid onto the relief. Akbarzadeh et al[43] proposed a 2D grid adapted to the relief of the area. This article is focused on covering a large outdoor area with an important relief (hills and valleys). To estimate properly the area to cover, a grid was placed following the altitude of the relief as shown in Figure 4.6. The spatial modeling is traditionally a 2D grid at fixed altitude more or less equal to the floor ground. The 2D grid distribution adapted to a 3D environment is the easiest and can be customize for numerous problems which allows the reduction of the number of control points (g_i) and thus the computation time.

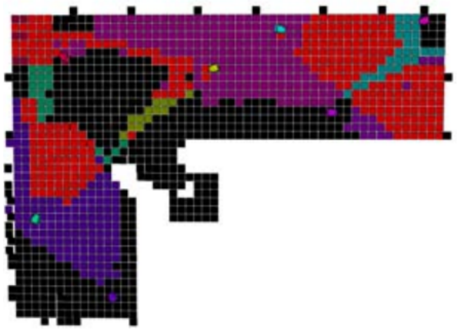
4.1.1.5/ Zones of interest

Among the areas to cover, some zones may have a particular interest to be covered. These zones can be discriminated by the grid design. Mainly three methods can be discerned to highlight the zones of interest:

- The multi-coverage zones.
- The priority zones.
- Non-interesting zones.

(a) Building view.


(b) 1st floor.


(c) 2nd floor.

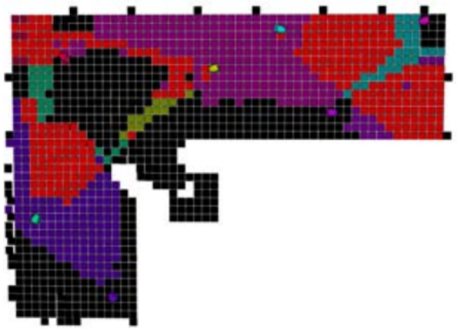

(d) 3rd floor.

Figure 4.5: Illustration of a grid layer in three floor building from [47]. In the figure (b) (d) (c) the uniform grid is visible with the black case of the grid representing the area uncovered and the colored case for the covered area.

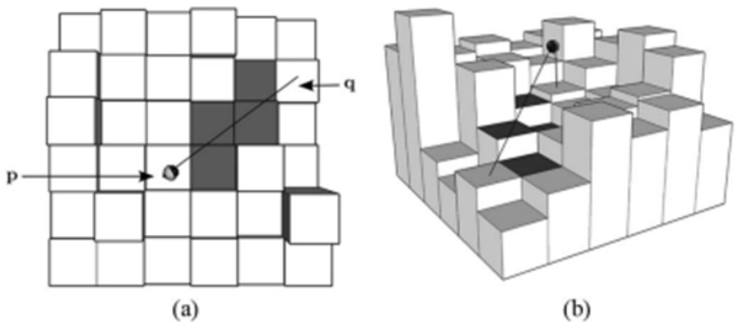


Figure 4.6: Relief grid used to discretize an area while taking into account the relief. Illustration from [43]

The multi-coverage zones The aim of the multi-coverage zones is developed to consider specific zones of the area controlled by numerous cameras. Multiple coverage may be called k-coverage such as in [26]. Where k refers to the number of cameras mandatory to cover the zones of interest. Each point of the grid G should be covered by at least one camera and for some specific zones of the area by k cameras.

$$\forall g_i \in G, B^{(k)}(g_i) = \begin{cases} 1, & \text{if } g_i \text{ is covered by } k_i \text{ cameras, } k_i \in K \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

Where k_i is the number of cameras used to cover the point g_i of the grid. K is the list of k_i associated to the number of points in the grid. The list K is initialized at one by default, excepted for the multi-covered zones, where they have to be covered by k cameras as in [15].

The priority zones The zones to cover in priority are used especially in the case where the number of cameras are not sufficient to cover the entirety of the area. This priority of coverage can be expressed by different ways.
One way is to express the priority with a grid uniformly distributed, where a weighting on the points of the grid which needs to be covered in priority is applied, as in [43, 5]. This method was implemented in [43] to optimize the position of the camera on the road passing through the area to cover.
Another way to express the weighting of the priority zone for the randomly distributed points represents the increasing of the sampling frequency of the desired zones. Increasing the sampling frequency of the priority zones is simplified due to the usage of a random distribution as in Horster et al [14] and as illustrated in the Figure 4.3b. Using this method, the zones of interest are more dense and this process push the optimization process to cover this area in priority (more density means more important).
Otherwise, the priority zone in the uniform grid distribution can be formulated as:

$$\forall g_i \in G, B^{(p)}(g_i) = \begin{cases} 1 * p_i, & \text{if } g_i \text{ is covered and } p_i \text{ is the weight, } p_i \in P \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

Where p_i is the weight of the point g_i on the grid G. $p_i \in P$ is the list of p_i which contain the weighting of the area associate to the points of the grid g_i which have to be covered in priority.

Non-interesting zones Non-interesting zones are as its name suggests, the zones without interest to be covered by the set of cameras, noted U. The set U is composed by the list of points g_i which have no interest to be covered. These zones are not strongly prohibited. Which means, the zones considered as non-interesting can be covered but their coverage or un-coverage has no impact on the coverage estimation of the area. In the case of random grid, distribution of the non-interesting zones have a very low sampling frequency or null as in [43]. For uniform grid distribution, the non-interesting zones are removed from the list G in order to create a simplified list G'. This method is currently used as shown in [6, 35, 43, 14, 5].

$$G' = G - U \ , \ \ U = \{g_i | g_i \in G, g_i \text{ are the non intresting points}\} \tag{4.5}$$

Instead of defining the non-interesting zones in a set G and excluding U to have a smaller set G', the non-interesting zones can be defined as priority zone with some p equal to 0 (in Equation 4.4). This way of defining the non-interesting zones is greedier in computation and consequently the formulation in Equation 4.5 is preferred to the 0 weight of p (Equation 4.4).

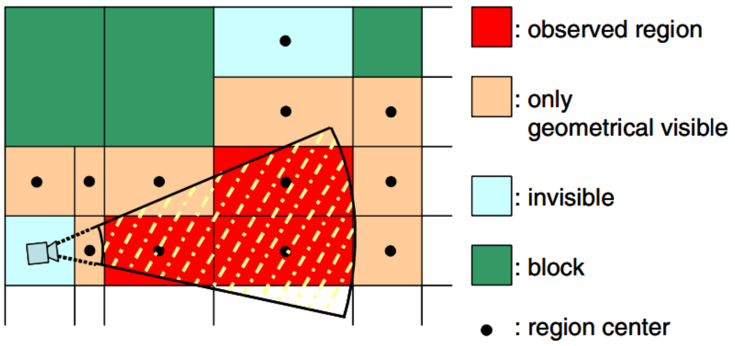


Figure 4.7: Observed regions from camera candidate. Grid representation from [35].

Finally, these methods (Equation 4.5) used to design the zones of interest are not fully independent and can be associated with the same model as expressed in [43, 14, 5]. The combination of all these zones of interest can be formulated as :

$$\forall g_i \in G', B^{(k,p)}(g_i) = \begin{cases} 1 * p_i, & \text{if } g_i \text{ is covered by } k_i \text{ and } p_i \text{ is the weight, } p_i \in P \text{ and } k_i \in K \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

Where G' is the restricted list of point to cover which takes into account the removal of the non-interesting zones (as in Equation 4.5). P and K are respectively the priority zones and the multi-coverage zones.

4.1.1.6/ Atypical designs

Previously, the method to set-up a classical occupation grid depending on the problem has been discussed. Few other solutions, more atypical, have been developed. One solution coming from the field of wireless sensor network is the topologies grid. The topologies grid is clearly explained in Chakraborty et al [42]. The interest of these methodologies is the reduction of the number of points to cover. But in this case, the number of points is not related to the resolution wished but to the sensor range. Indeed, the size between the points of the grid has been defined by the size of the minimum sensors range. The distance of the minimum sensor range is used as nodes for a topological relation.
Another atypical solution is the development of a grid composed by rectangles. Each rectangle may have different sizes adapted to the obstacles in the area. A rectangle is considered as covered if most of the area of the rectangle is covered by the cameras as presented in [35]. This method is adapted to the area with few obstacles as is shown in the Figure 4.7.

4.1.1.7/ Grid map design sum-up

The grid map may be used to represent different objectives and constraints of the problems. The map representation handles an crucial role in the area coverage. The design discussed previously are summarized in the Table 4.1. This table summarizes the more important aspect of each grid representation in different papers.

	Grid pattern	Grid randomness	Volumetric space	Sampling frequency	Zone of interest
[49] Zhou2011			2D gird	For real-time	✗
[6] Zhao2008	✔	✔	2D gird at torso hight	Incremental	Non-interesting zones
[15] Chrysostomou2012	✔		3D grid for volumetric space	Non specified	Non-interesting zones & possible k-coverage
[47] Hengel2009	✔	✔	2D grid at torso hight for each level of a bulding	Hight sampling	Non-interesting zones & k-coverage
[98] Morsly2012	✔		2D grid	Adaptable	✗
[43] Akbarzadeh2013	✔		2D grid on the relief (Figure 4.6)	Very hight sampling	Non-interesting zones & priority coverage
[42] Chakrabarty2002	✔		2D grid	Topologies sensor (low sampling)	✗
[55] Valente2013	✔		2D grid with overlap by shifting of z	Low sampling frequancy	✗
[35] Yabuta2008	★		2D grid (Figure 4.7)	Grid paterne at initialisation then zones segmented	Non-interesting zones & priority coverage
[14] Horster2006		✔	2D grid (Figure 4.3)	Hight sampling	Non-interesting zones & priority coverage

Table 4.1: Sum-up of the grid map.

4.1.2/ Our approach

Based on the designs studied, the one finally adopted is a grid G as in Eq 4.1 with an uniform distribution following the 2D grid pattern. The grid pattern has been selected due to its facility of implementation and flexibility to additional constraints. The frequency adopted is fixed depending on the size of the area, the precision required and the cameras properties. The frequency adopted has to be considered as dense (high sampling frequency). Also, due to the high sampling frequency, the 2D grid pattern is more appropriate choice and allows a reasonable number of points to describe a complex area. The grid is placed on the floor of the area to cover. Floor is always considered as flat without relief. In our case, the zones of interest can be varied and may include non-interesting zones, priorities zones and multi-coverage.
To possibly allow all these zones of interest features, the selected design of the grid is based on the formulation from the Eq:4.6. During the experiments presented in the following chapters, the priority zones and multi-coverage are not exploited. Consequently, the $k = 1$ and $p = 1$. Although some brief tests were carried out with k-coverage and priority zone constraints. The results obtained were considered irrelevant, hence the choice to focus on constant priorities and single camera coverage for the entire area to be covered.

4.2/ Camera coverage

Once the area to cover is described by the grid, the next step is to verify for each point of the grid if one or more cameras cover it, based on Eq: 4.6 with $k = 1$ and $p = 1$.
To verify if each points of the grid is covered by a camera. It is primordial to discuss the camera properties and introduce the projection model.

4.2.1/ Camera definition

The perspective projection is often used to define a camera. The camera projection has the advantage to be anamorphic Thus, the perspective projection is also the most common and more especially in the field of area coverage as presented in examples [39, 20, 49, 15, 6]. Other models of camera or vision sensors can be used such as for example omnidirectional with a $360°$ field of view [28, 42, 26]. In the following chapters ,we are only focusing on the camera perspective due to its wide use and it is also the most suitable to be embedded in UAVs for path planning.
The pinhole or in Latin the ”camera obscura” (see Figure4.8) is at the origin of the geometry model for the perspective projection.
The pinhole model is commonly composed by a box (or a chamber) hermetically closed to light, excepted by a small pinhole on the middle of the front side. All the ray of light reflected by the objects of the world and passing by the small hole are projected onto the back side of the box. Each ray of light passing by the hole is projected onto a plan (the back side of the inside box). This plan observes the reversed image of the world and can be recorded by a film or a digital sensor. Due to the simplicity of the pinhole model, the calibration and camera projection estimation is simplified. To estimate the projection, only few elements has to be known:

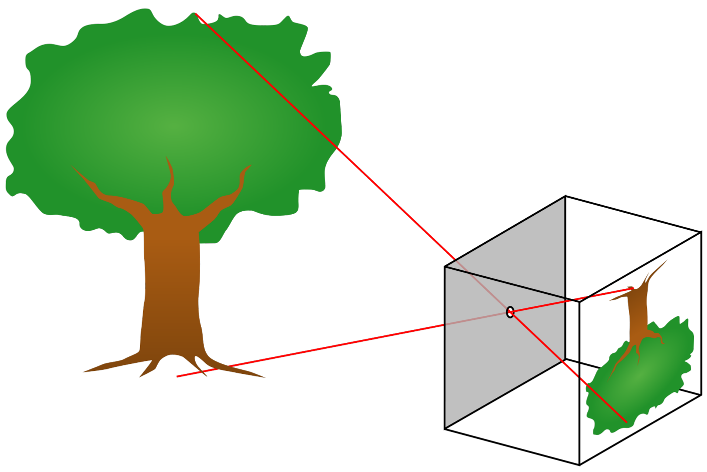


Figure 4.8: Pinhole camera model.

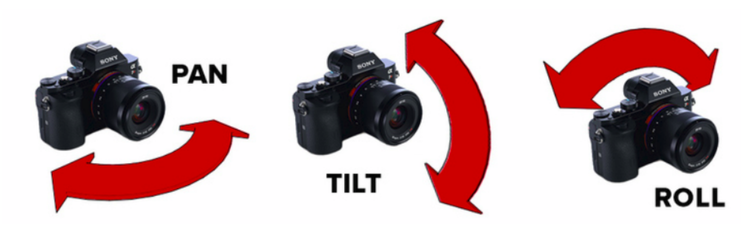


Figure 4.9: The rotation composed by three degrees of freedom on pan tilt roll (α, β, γ).

- Three degrees of freedom for the camera position : (x, y, z);
- Three degrees of freedom for the camera orientation : pan, tilt, and roll angles: (α, β, γ)
- Optical parameters including: the focal length f of the lens, u_0 and v_0 which would be ideally in the conter of the image. σ_{uv} represents the skew coefficient between the x and the y axes.

Among the parameters of the camera, only some of them are useful to estimate the projection. They can be formalized as a vector:

$$v = (x, y, z, \alpha, \beta, \gamma, f) \tag{4.7}$$

Each elements of the vector v are used to compute the camera projection on the discretized floor (the grid G).

4.2.1.1/ Coverage estimation in the literature

In order to compute the camera projection onto a grid, the pinhole model is used with the parameters of the vector v, previously defined.

The detail to estimate the camera projection onto the floor, based on the pinhole model and the parameters (v), have been detailed numerous times such as in [29, 44, 13]. In [29, 44, 13], the camera projection is used to estimate if a point g_i of the grid is visible by a given camera, and that for each point of the grid. These articles handles the classic camera projection, with the six Degrees of Freedom (DoF) in [29] and five DoF in [44]). Both are used to estimate the 2D projection of the camera onto the floor (as in Figure 4.1).
In [29], the camera projection have been computed for several rotations for all the DoF. In this case, the projection can have numerous shapes (mostly parallelogram shapes).
In [44], the model of camera projection begins to be simplified by assuming some fixed parameters. The fixed parameters allows more efficient estimation, by economizing part of projection computation of the camera at each time. The fixed parameters allows the pre-estimation all of part of the shape.
In [13], the model of camera projection is used to compute one time for a fixed pan and roll in order to have a coverage estimation usage in a 2D map. The camera projection is finally simplified by using a kind of triangular shape.
To go further, other models and formulations inspired by the pinhole model were proposed [98, 43, 7, 51]. These models are inherited from the camera projection and adapted to fit each relative problems.
In [98, 51], the camera is considered placed on the floor with a fixed pan (with the viewing direction almost parallel to the floor). Therefore, the camera projection is simplified by an isosceles triangle where the shape depends on the focal length.
In [43], the camera projection is also simplified to obtain a kind of isosceles triangle shape while considering the depth of view of the camera. In this case, the camera projection has a ”piece of pie” shape.
In [7], thanks to a fixed pan and focal length, the camera projection is simplified to have a rectangular projection onto the ground. The sweep is designed consequently to the size of the camera projection, in order to minimize the overlap and have full coverage of the area.

One of the common point of the following papers [98, 43, 7, 51, 6, 20, 29, 44, 13] is the computation of a camera projection onto a grid. The computation of the function $f()$ which has to estimate whether a point g_i of the grid is covered by a cameras v_j of the network, is not considered as really greedy (in time). But the coverage estimation using $f(g_i, v_j)$ for a point g_i by a camera v_j requires to be performed for each points of the grid and for each cameras of the network in order to estimate properly the complete area coverage.

$$\text{Coverage Rate} = \sum_{i=1}^{m} \sum_{j=1}^{n} f(v_j, g_i) \tag{4.8}$$

Where n, the number of cameras (v_j) in the network; m the number of point in the grid (g_i). The solutions proposed until now to estimate the camera projection onto the grid (as in Equation 4.8) do not take into account the occlusion made by potential obstacles. The occlusion projected by the obstacles requires to be taken into account depending on the camera projection. To detected the part of the area occluded by an object, the solution commonly proposed (as is well explain in [44]) is to check the line made between a point covered ($g_i \in Pc$) and its camera. If this line is intersected by at least by one object in the scene, the point g_i cannot be considered anymore as covered.

To take into account the potential occlusion by an obstacle Obj_l, the Equation 4.8 can be extended :

$$\text{Coverage Rate} = \sum_{l=1}^{k}\sum_{j=1}^{m}\sum_{i=1}^{n} f(v_i, g_j, Obj_l) \tag{4.9}$$

The function $f(...)$, in charge of computing a camera projection, will be called for each camera, each point of the grid and each obstacle, in order to evaluate the complete coverage of the area (with the occluded part). This numerous calls will greatly increase the computation time of the coverage. It is even more affected when numerous camera projections requires to be computed at each turn of a long optimization process.
In these conditions, the efficiency of the function $f(...)$ computing the coverage estimation, is primordial due to the repeated calls.

4.2.1.2/ Coverage estimation optimization

To design an efficient cost function, it is necessary to reduce the computation time and estimate the covered area . The camera projection model has to be simplified with some basic assumptions relative to the problems.
Considering our case, a camera is fixed on a UAV with a viewing direction orthogonal to the ground (without any rotation in α pan and β tilt). In this model, the camera projection is always a rectangle as shown in Figure 4.10. In fact, the camera is placed on a UAV and have a direct view onto the ground. The viewing direction of the cameras is perpendicular to the floor, implying the rectangle projection. The area covered by the camera is defined by the rectangle defined by $Wr \times Hr$ It is possible to compute the size of the covered rectangle for a given altitude by one camera, based on these given parameters.

Based on the useful parameters from the camera $(f, S_w \times S_h, z)$, the computation estimating the size of a camera projection (Wr,Hr) is :

$$\begin{aligned} Wr &= (\frac{1}{f}.S_w).z \\ Hr &= (\frac{1}{f}.S_h).z \end{aligned} \tag{4.10}$$

When f represents the focal length, z the altitude of the cameras and S_w, S_h the sensor size. Thanks to this model of camera projection drawing a simple rectangle, the size of the rectangle projection is directly related to the altitude. The altitude is assuming to be a coefficient of the Equation 4.10
All these simplifications helps the cost function to be fast and efficient. In fact, in the Equation 4.10, the focal length parameters and the sensor size are fixed and do not need to be recomputed for each camera position only the altitude A has to be updated.

4.2.2/ Parameters to optimization

By simplification presented previously, the parameter vector (see Equation 4.7) can be reduced. The computation of one camera projection onto a grid (as in Section 4.1.1 and the camera are placed in altitude with a viewing direction orthogonal to the floor. Based

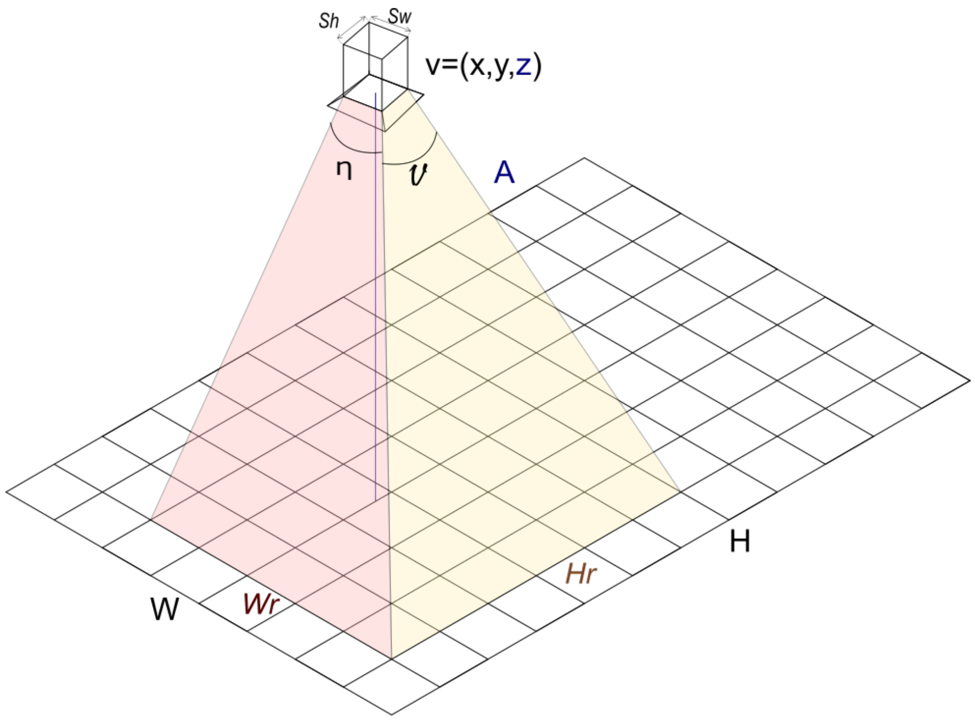


Figure 4.10: Camera projection onto a grid. The grid is placed on the floor to discretize the area covered.

on the Equation 4.10, just few parameters are necessary as shown in Equation 4.10. Thanks to the proposed assumption, Equation 4.7 can be reduced by keeping only the position of the camera (x, y, z) and the roll (γ) as:

$$v = (x, y, z, \gamma) \tag{4.11}$$

In our case, the roll γ can take two different states, portrait or landscape with respectively a rotation of $0°$ or $90°$. Reducing the number of parameters, passing from the equation 4.7 and 4.11 reduces the number of parameters to optimize.

Until now, the camera projection estimation was addressed with only one camera. But we want to compute the coverage for a camera set. The solution is based on the previous estimation for the camera projection but adapts to the location of each cameras by positioning the rectangle projection to have the center at the x and y position and compute the occupation grid.

In order to gain a bit of time, each point of the grid already covered by a camera are not tested for the next cameras. Which means that in the equation 4.8, the value of m decrease as i increase. More exactly the size of m decrease as the area coverage increase.

Parameters representation

To represent the problem, we need a set of cameras. The previous notation can be extended to have a set V of n cameras defined according to:

$$\begin{aligned} V = v_1, v_2, ..., v_n \ , n \in \mathbb{N}^* \\ \text{with } v_i = (x_i, y_i, z_i, \gamma_i) \end{aligned} \tag{4.12}$$

Where n is the given number of cameras in the network. The coordinates of a camera v_i which are the ith camera of the network is defined with x_i, y_i, z_i, in a given room and γ_i the roll rotation (portrait or landscape). The not contained parameters in V and used to compute the cameras projection are identical for all the set V, and are fixed at the beginning of the optimization.
Therefore, V represents a solution. V contains all individual positions and orientations of the camera set for a predefined focal length, sensor size and related map depending on the problem.Obviously, all the solutions V are not a ”possible solution” for our problem. Some solutions V does not complies the set of constraints noted E. E represents the set of constraints (the set E is defined in details later).

In addition, the solution V should comply a set of constraints E (see Eq.4.13). Among the constraints, few of them were already discussed, as the occlusion, the map restriction, the k-coverage, or some constraints more specific to the problems (as shown in chapter 2).
The ”possible solution” Vs must take in consideration with the set E as :

$$Vs = V \text{ , iff } E(V) = \begin{cases} 1, & \text{iff } E_i(v) = 1 \text{ , with } i = 1...Nc \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

Where $E_i(V)$ is the function applied to verify the ith constraint of the set E on the solution V. Nc is the number of constraints necessary to be satisfied to obtain an acceptable solution. Among all the possible combinations of parameters V, only the one intersecting the set of constraints E is a possible solution. If we are considering all the V and all the E as two subsets, Vs is defined as $Vs = V \subset E$.

The problem of monitoring an area and more specifically the problem of area coverage may contain many constraints depending of the environment and the context. As for example: the room shape, minimizing the altitude, have the best resolution, orientation of the camera, the possible occlusion,... All these constraints are included in the set E. The constraints requires to be defined depending on the problem and the goal.

4.3/ Cost Function

The cost function has to evaluate the quality of a given solution. To estimate the quality of a solution one of the main criteria is the coverage rate. The previous section was focused on the computation of the coverage rate.
To create an efficient cost function, other criterion has to be taken in to account as constraints. To establish the cost function, a list constraints require to be established. Each constraint of this list does not have the same importance. In order to split the constraints depending on their respective impacts on the problem, two types of constraints are introduced in the following sections before discussion.

4.3.1/ Constraint list

The constraints can be numerous and depends mainly on the problem formulation and the context. Like that, few of them were briefly introduced in the previous section as in chapter 2 and section 4.2.1.2,... This part is focused on the list of the constraints used in our case and detail their design.

Fixed number of the cameras One of the first constraint represents the establishment of number of cameras. This constraint, as some others (detailed latter), are useful to simplify and restrict the possibility of the problem. This constraint allow us to focus on the fine optimization (as in [6] where both are tested). The number of cameras is fixed at the beginning of the optimization and no more camera will be added during the optimization process.

Fixed parameters of camera and no rotations Fixed parameters of the cameras and no rotations (α and β) has been introduced previously (section 4.2.2). These constraints imposed by the use of an UAV are also an advantage for the optimization by simplifying the coverage estimation and limit the number of parameters to optimize. The parameters are fixed at each beginning of the optimization. Thanks to this constraint the optimization has to focus on the precise cameras positioning.

Fixed altitude The fixed altitude is a constraint established in order to limit the number of parameters to optimize. The usage of such constraint is linked to the reduction of the difficulty by reducing the possible search space (see section 4.4). It is also useful for other assumptions, like a camera on the celling or for a submarine [54]. This constraint is an optional constraint and is not always used in the experiments presented in the following sections.

The altitude boundaries When the altitude is not fixed, some limits must be chosen to avoid the extremely high and low altitudes. The highest altitudes will be fixed depending on the UAV ability and other restrictions as the legislation requires it. The lowest altitude has to be fixed for the safety of the users under the UAV. In practice the altitude boundaries are defined with an interval:

$$\inf z \leq z \leq \sup z \tag{4.14}$$

Where $\sup A$ is the maximal altitude of the camera and $\inf A$ the minimum altitude. A is the altitude between the camera and the grid.

The map boundaries The map boundaries represent a constraint similar to the altitude boundary. Despite the shape of an area to cover, some maximum boundaries can be established. In fact, for any shape, as complex as it is, it may be possible to encapsulate it inside a rectangle. The rectangle map boundary is defined by a width W and height H. The boundaries on x and y are:

$$\begin{aligned} 0 \leq x \leq W \\ 0 \leq y \leq H \end{aligned} \tag{4.15}$$

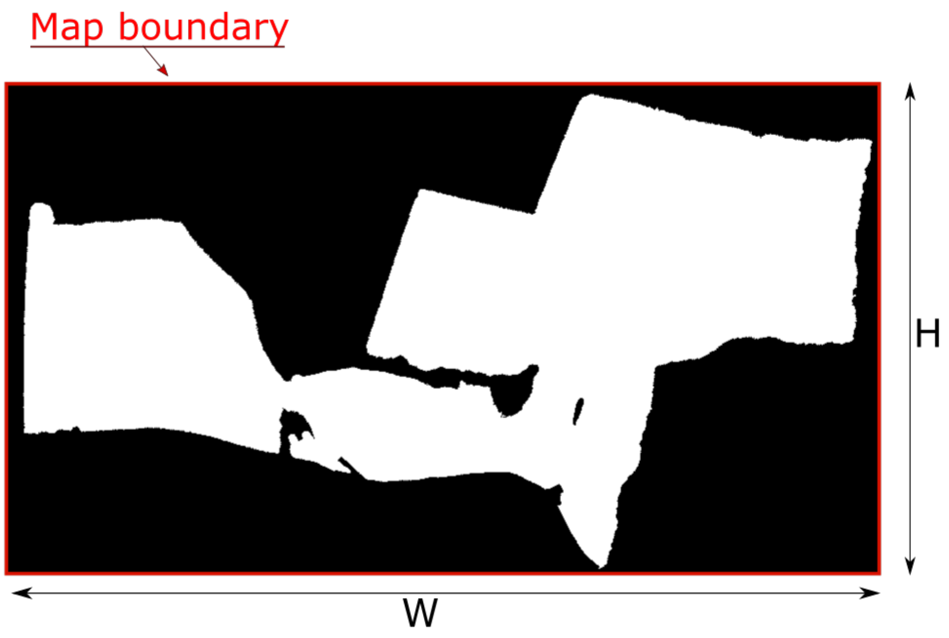


Figure 4.11: Map to cover with the map boundaries in red (W and H size) in black. The sub-part have no interest to be covered.

By associating the altitude boundaries (from Eq4.14), cubic boundaries limits the position of the camera in the three dimensional spaces.

$$\begin{aligned} &0 \le x \le W \\ &0 \le y \le H \\ &\inf z \le z \le \sup z \end{aligned} \tag{4.16}$$

Non rectangle map with possible holes The map to cover can be much more complex than a simple rectangle and may take any shape with can be composed by holes. The Figure 4.11 illustrates the map complexity. To take into account the complex shape map The grid G is reduced in order to keep only the points in the white sub-part. In the example (see Figure 4.11), each white pixel of the map is a point of the grid to cover. Concretely this implementation by using a grid map has some advantages such as the flexibility of the grid customization .
In addition, it allows the optimization to test some exotic solutions, as allowing the camera position on the black sub-part or bordering the boundaries. Obviously, the exotic solution with a camera position on the black side does not increase the coverage rate but with a correct converge, the camera are not covering the black sub-part of the map (or small part of it). The interest of leaving this freedom to the optimization is double. First, it controls the quality of the the solution obtained,. Secondly, the variety introduced during the exploration of the black sub-part of the map can be advantageous.

Resolution The resolution of the images is related to the sensor size (in px) and the distance between the camera and the object. In our case, the sensor size is dependent on the properties of the camera mounted on the UAV.The distance between the camera

and the floor (or the object) is the altitude as defined in the Section 4.3.1.
The resolution constraint has to maximize the resolution. In order to maximize the resolution during the optimization, the altitude criteria is modified in order to be as low as possible.
Considering only the minimization of the average altitude of the cameras is harmful for the coverage. Consequently, during the optimization process, a trade-off between the altitude (with the related resolution) and the coverage rate have to be deduced. The altitude of the cameras has to be minimized, the average of the altitude is used as an index to estimate the resolution. In order to manage this trade off, the average altitude of the cameras is included in the cost function and has to be minimized (see section 4.3.3).

4.3.2/ Constraint types

Among the constraints listed, different priorities and restrictions exists. Indeed the constraints can be classified depending on their priorities and impact on the problems. The constraint classes are considered mostly as hard or soft constraints.

Hard Constraint Hard constraints limits the possible solution by do not allowing the solution which does not respect it. These hard constraints are directly used during the optimization process to prohibit any solution to be out of these boundaries. These hard constraints are integrated in the optimization process in order to not generate a solution which does not complies it. Consequently the hard constraints may slow down the generation of the individuals due to the specific generation and required test.
For example, the 3D boundary as defined in Equation 4.16 is a hard constraint. Improving that, each camera must be inside the 3D boundary.

Soft Constraint The soft constraints represents the minimization of a set of errors. A solution that does not fully respects the soft constraint can be considered as acceptable if the amount of errors does not affect greatly the final answer. The soft constraint is assimilated to acceptable error.
The soft constraint leaves the possibility during the optimization process to allow some mistakes in order to learn about it. If the soft constraint is noted ϵ and the hard constraint is noted ε', the constraint set is $E = \epsilon + \varepsilon'$.

$$\max f(Vs) - \min \epsilon \ \forall Vs \subset \varepsilon' \tag{4.17}$$

The objective is to maximize the coverage of a set of cameras ($f(Vs)$) with respect the hard constraints ε' and minimize the error from the soft constraints ϵ. Concretely, the soft constraints are commonly integrated in the Cost Function as can be the resolution or the complex shape of the map thanks to the grid design.

4.3.3/ The cost function implementation

The cost function has for mission to estimate the quality of an answer. In our case, an answer is the position of a camera set. The cost function is essential in the process of optimization as it was introduced in the Section 3.3.2.

The cost function needs to estimate the area covered by a camera set. In order to do that, the area is discretized by a grid as explained in Section 4.1.
The grid customization permits to introduce some of the soft constraints such as the complex shape of the map by removing the points out of the area to cover.
The grid modification allows the camera positions to cover the area already covered and removed from the grid. The consequence of it are the reduction of the coverage rate possibility. The optimization have to minimize this error.

To evaluate the coverage of a camera set, it is essential to estimate the cameras projection of each, as detailed in the Section 4.2.1. The area cover by the j-th camera is noted as in Equation 4.3 (where $Pc \in G$). By iterations, for each camera of the set, the full area coverage is computed (as Equation4.8).

Based on the previous statements, the simplest cost function is the coverage estimation.

$$C(Vs) = \frac{\sum_{i=1}^{n} B^{(k,p)}(G')}{m} \tag{4.18}$$

Where n is the number of cameras, m represents the number of points needed to describe the grid G (as in Equation 4.1), Vs is the solution with respect to the hard constraints. The cost function $C(Vs)$ gives the quality of the solution Vs.

This version of the cost function $C(Vs)$ does not take into account the resolution constraints. The resolution is strongly linked to the camera altitude z (as show in Section 4.3.1). A criteria must be added in the cost function formula of the Equation 4.18. The average of the altitude z is used and have to be included in the cost function.

$$\bar{z} = \frac{\sum_{i=1}^{n} z_i}{n} \tag{4.19}$$

If the resolution is strongly related to the altitude, the average of it $\bar{z}$ can be considered as a part of the soft constraints (ϵ) in the Equation 4.17 and the Equation 4.18 and may be updated as :

$$C = \frac{\sum_{i=1}^{n} B^{(k,p)}(G')}{m} - \frac{\sum_{i=1}^{n} z_i}{n} \tag{4.20}$$

The Equation 4.20 is used in the cost function to add the resolution constraint. Consequently, the optimization tries to minimize the average altitude and maximize the coverage with no priority. Concretely, by just applying this Equation 4.20, the optimization will first minimize the average altitude by positioning all the cameras at the minimum altitude (with respect to the hard constraint of altitude boundary) and in second time try to maximize the position (on x and y) of the cameras.

In order to have a priority between the coverage and the altitude, a weight has to be established on the Equation 4.20. The weight requires to be chosen carefully. The weight has to be auto-adaptable depending on area covered in order to give more priority to the coverage when the coverage rate is low and add importance to the resolution when the area is already well covered. The coverage have to remains the priority, the resolution must be optimized in a second time. The best solution, is to link the weight of the resolution criteria with the coverage rate.

$$\sigma \times \sum_{i=1}^{n} B^{(k,p)}(G') \times \frac{\sum_{i=1}^{n} z_i}{n} \tag{4.21}$$

Where σ is a weighting coefficient at 0.06 to reduce the priority on the resolution criteria. Based on it, the final cost function is:

$$C(Vs) = \frac{\sum_{i=1}^{n} B^{(k,p)}(G') - \sigma \times \sum_{i=1}^{n} B^{(k,p)}(G') \times \frac{\sum_{i=1}^{n} z_i}{n}}{m} \tag{4.22}$$

Thanks to this formula, a proposed answer Vs can be evaluated and returns the quality of the solution for the problem of the coverage maximization and the minimization of the altitude in a second time. The cost function integrates all the soft constraints either by the design of the grid or by the formula of the cost function $C(...)$.

The cost function presented here is the final version; but the building of it was an incremental work and numerous version was tested in terms of weights, priorities, and constraints. The one presented here is the most balanced.
Despite that, some of the works presented in the following sections are made with the basic cost function from the Equation 4.18. In this case, the element which composes v can be reduced as only $v = (x, y, \gamma)$ depending on the need of the experimentation.

The final and complete cost function $C(Vs)$ have as input a vector Vs which is composed of all the cameras position and orientation belonging to the network. It is also composed by the map of the area to cover. Where G includes the soft constraints such as the room shape. The constraint of resolution is added by using the average altitude in the equation 4.22. The value returned by the cost function $C(Vs)$ is the quality of a solution to our problems of coverage using an UAV.

4.4/ Optimization complexity and search space

In spite of the simplification presented before, the problem remains complex. There exists many positions for each camera to cover an area with a certain amount of UAV. This number of position can be estimated for each camera as follows.
Each camera Sp defined by the position on x, y, z and γ can be placed anywhere in the search space:

$$Sp = (W \times H \times (\max(z) - \min(z)) \times 2) \tag{4.23}$$

Where W and H are the size as width and height of the area to cover, $\max(z) - \min(z)$ is the range of possible altitude. Two is to define the roll γ, as the rectangle projection is horizontal or vertical (landscape or portrait).

The problem of the search space is the propensity to increase rapidly as the area grows.

This phenomena is accentuated by the number of cameras n.

$$\binom{n}{Sp} = \frac{Sp!}{n!(Sp-n)!} = |Vs| \tag{4.24}$$

Where $|Vs|$ is the number of possible solutions for a set of n cameras in the worst case. In fact the size of the search space of one camera (as Eq. 4.23) associated to the set of n cameras (as Eq.4.24) makes an exponential number of possible solutions depending mainly on the size of the area and the number of the cameras in the network.
Obviously in the view of this behavior, the usage of a deterministic solution based on a heuristic does not seem to be a good approach to have an efficient solution. In addition, the number of possible solutions $|Vs|$ makes the computation of an optimal almost impossible due to the numerous local minimum. The formulation of the problem is primordial in order to reduce the size of the search space and gives a chance to the optimization to reach convergence. The complexity of the problem of camera positioning is at least NP-hard similarly to the AGP. In fact, if we consider the camera positioning as inherited from AGP with some extra constraints (as field of view and depth of field). Also the literature is unanimous about the NP-hard complexity as discussed in [28, 31, 52, 6, 121].

The size of the search space is even more important than the method applied. In fact, due to the problem complexity and the numerous local minimum, the classical optimization algorithms are not appropriate. The stochastic algorithms are efficient in this case due to the wide search space that increases the difficulty to converge. The stochastic algorithms are widely based on the optimized random solutions and bigger is the search space, harder it is get optimized solution.

4.5/ Our Genetic Algorithms

The GA is a complex set-up that an be made to optimize several kind of problems as discussed in the chapter 3. Among the customizable parameters, the most important are summarized in a Table 3.2. To optimize the waypoints positioning, an appropriate set of parameters has to be established to have an efficient GA. The set-up used during our experiments is summarized in the Table 4.2.
The set of parameters has been selected empirically based on several try and a deepened knowledge of the problem and the GA. To have an efficient positioning for numerous waypoints in wide areas and an important threshold rate of coverage, the GA needs to explore efficiently and quickly a wide search space. To be efficient in wide search space, the GA must introduce variety; and due to the numerous local minimum the elitist selection is inappropriate.

Important points for the GA		**Our implementation**
Coding chromosome		Discreet representation. $Vs = v \times n$ when v complies the constraint as in Eq 4.17
Cost function		$C(Vs)$ (see 4.21)
Population	Size	100
	Initialization	Random
Selection mode	Selection mode	Tournament selection
	Pool size	2
Operators	Used Operators	Crossover and mutation
	Implementation choice	Crossover adapted to merge only the corresponding genes (as x with x)
	Operators Rate	Mutation rate 1%; Crossover rate 92%
Stopping criteria		Convergence

Table 4.2: Parameters of GA for waypoints positioning.

5

Waypoints positioning experimentation

Contents

During the previous sections, the problem was discussed as an optimization problem. The formulation and the complexity of the problem has been presented in the Section 4.4. The cameras positions needs to be optimized to offer the best waypoints for the path planning. In fact, our approach has for objective to find the best path planning to maximize the coverage by using the best cameras position as a waypoints for a path. For the first experiment, different environments are proposed. The environments were designed to have various shapes and sizes. The various shapes are used to estimate the impact of such parameter on the proposed solution. The rooms are designed while knowing the exact number of waypoints in order to have a known Ground Trough (GT). The room sizes may have notable impacts on the optimization. The size of a room impact proportionally the search space. A larger room means more possibilities to place waypoints. The increased number of waypoints logically raises-up the number of dimensions

to optimize. The increasing number of dimensions is used to evaluate the robustness of the algorithms.
The rooms finally used for the experimentation are proposed in the Figure 5.1.

In addition of the rooms parameters, the number of dimensions is also tested by adding various possible altitudes for each waypoint. The waypoints can be in the simplest version placed only on x and y coordinates. In a second time, z coordinate can be optimized inside a given range of altitudes. Two sets of scenarios have been tested. One with a fixed value of z and another set of scenarios with a range of z. All this allows the evaluation of the impact of the altitude on the optimization process.

5.1/ Other algorithms used

In order to compare carefully our proposed method, two other algorithms were introduced. The following sections explains the two experimented algorithms. The Particles Swarm Optimization (PSO) and the Random Selections (RS) are described in the following sections. The PSO is used due to its importance in the literature for similar problems and the RS for its simplicity to evaluate the optimization. The RS acts as a reference point for the other optimization.

5.1.1/ Particles Swarm Optimization

The PSO (Particle Swarm Optimization) is an algorithm dedicated to the optimization problems. It is a stochastic algorithm from the family of evolutionary algorithms (see Chapter 3). The PSO is relatively young compared to the other EA. It was developed by Russel Eberhart and James Kennedy in 1995 [122].

To do so, the PSO is inspired by the behaviour of animals such as the birds flocking, fish schooling and swarming theory which are working in group (or swarm) to seek food. The direction to take is not decided by a leader, but by all individuals from the swarm, by relaying few information such as which quantities of foods they found. The swarm composed by numerous individuals becomes smarter and more efficient to reach their objectives. The algorithm proposed by Russel Eberhart and James Kennedy in [122] are directly inspired by these behaviours.

The methodology used here represents the examination of each individual, also called particles, as a solution of the problems. The problem is optimized at each iteration. Following that, each solution must be comparable and quantifiable. At each iteration, each particle has to be tested by a cost function in order to discriminate the best particles of the swarm. The cost function used in our case has been designed and detailed in the Chapter 4. When the best particle is found at the end of an iteration, all other particles of swarm tries to modify their initial directions to converge more or less quickly to the currently best particle. In such a way that the other particles can converge towards the current best, each particle have to adapt its direction according to its own velocity and previous direction. The change of direction is slowed down by the inertia of the system. Indeed the strength of this algorithm is the acquisition of a very basic individuals behaviour to guide the particles. Each particle is guided by three behaviours:

- Its own velocity V_k.

- Its own best solution P_i.
- Its best solution from the swarm P_g.

Here, the velocity represents the valuable speed of the particle to converge to the best solution. The higher the inertia, the more difficult it will be for the particle with a high velocity to converge. The behaviour of the particles X_k are modelled by the following equation to obtain the new position X_{k+1} :

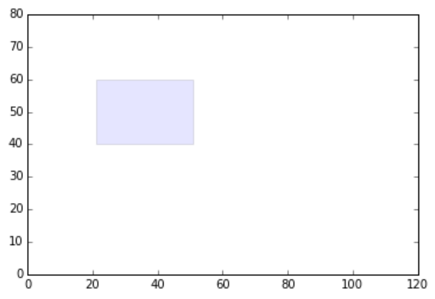
$$\begin{aligned} V_{k+1} &= \omega V_k + b_1(P_i - X_k) + b_2(P_g - X_k) \\ &\text{and} \\ X_{k+1} &= X_k + V_{k+1} \end{aligned} \tag{5.1}$$

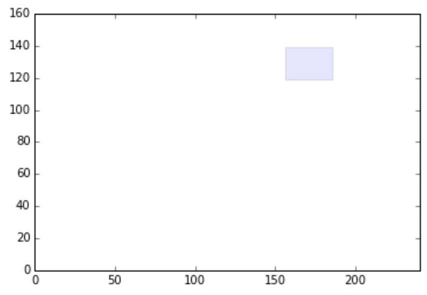
Where ω is the inertia, b_1 is a random value between 0 and ϕ_p and b_2 is a random value between 0 and ϕ_g. ϕ_g and ϕ_p are the scaling factors to search away from the particles. These factors are also called learning coefficients and their role is to push to learn or discover.

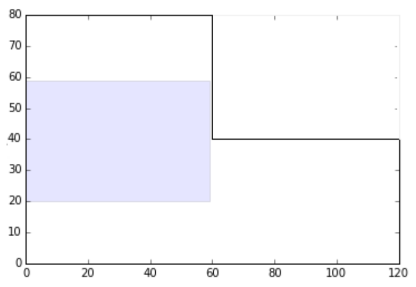
Thanks to this basic behaviour of the particles, the swarm can converge to a global solution. To have an efficient optimization, just few parameters must be set-up for the PSO. The most important are the inertia of the particles, the size of the swarm and the initial dispersion.

- The inertia will globally help the particles to keep their initial velocity. The consequence of the high inertia, is the exploration of more the search space and therefore the convergence will be longer/harder.
- The size of the swarm have an impact on the convergence time (as wall as the number of iterations). Indeed a big amount of particles in the swarm means more exploration of the search space at each iteration, but also more comparisons to find the best particles. The comparison may have a non-negligible computation time. The swarm size is commonly fixed but can be, as well as for the population in the GA (see Section 3.3.3), dynamically adjusted during the optimization process as presented in [123].
- The initial dispersion of the swarm can be as a decisive element as the population for the GA (see in Section 3.3.3). For the PSO ,the usage of a heuristic to initialize all the particles of the swarm is not recommended due to the important risk of premature convergence in local minimum. The random dispersion appears as the most appropriate initialization for a global optimization. On the other hand, the fast convergence and the PSO ability to" climb small hills" to go out of the local minimum can be used in order to refine another optimized solution. The main risk is represented by the optimization around the initial dispersion. It may not explore correctly the search space.

Finally, to summarize, the PSO is efficient in terms of optimization despite a very basic behaviour of each particle. Each particle has its own velocity defined randomly and controlled by a global parameter: the inertia. The strength of PSO is at the same time its efficiency to solve the optimization problem and its ease of use it. In fact, the PSO needs at minima few elements to works properly: A cost function, an inertia parameter and the size of the swarm. This efficiency and ease of use explain its popularity during the last decades.


(a) Simple room.


(b) Room with high ratio.


(c) Room L.

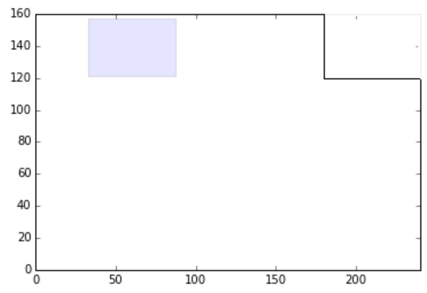

(d) Room L with high ratio.

Figure 5.1: For the experiments: (a), (b) the blue rectangle represents the field of view of one camera projected onto the ground with z=1 (30×20px) and in the figure (c), (d) with z=2 (60×40px).

5.1.2/ Random selection

The Random Selections (RS) is a very basic algorithm. It serves as a reference point for the comparison of algorithms. The RS does not take a complex meta-heuristic and is perfect to compare the efficiency of the other algorithms.

The random selection works by randomly generating numerous solutions. Among the solutions randomly generated, the best solution is kept as the optimized global solution. The RS allows to visualize through the search space by randomly try many possible solutions. The search space exploration by the RS is only made by random sampling without any other optimization method. Again, the RS is invoked as a reference points for the other algorithms. If the RS gets a similar result with the same number of cost function calls, the algorithm compared can be considered as efficient as a simple random solution. To use the RS as reference point, several optimization requires to be performed to obtain an average solution.

5.2/ Algorithm comparison

To solve the problem of camera positioning (or waypoints positioning), the usable algorithms are various as discussed in the Chapter 2 (see the sum-up Tables 2.1 and 2.3.4). Among the algorithms studied in the literature, the EA family appears as the more suitable to have an appropriate solution despite the numerous constraints of our problem. The EA is a vast family of algorithms. Among this family, the most used for our problem

is the PSO (see Table 2.3.4). The PSO gives a good and fast result in many cases. In the EA family, the GA is one of the founders and was one of the most popular due to its great flexibility and efficiency. After more investigation, the GA is under-estimated for the problem of camera positioning unlike the PSO (see [20, 49, 5, 50, 29, 51] as discussed in section 2.3) .
The conscientious comparison of GA and PSO was proposed by Boeringer et al in [124]. The conclusions of this comparison in [124] is open. In fact Boeringer et al. [124] concludes after several comparison applied to several optimization problems, Neither PSO nor GA offers a solution that is objectively better in all cases. Due to the average similarity of results between these two algorithms, this survey highlight the importance of evaluating them for each new problem. Therefore, there is no real tool to estimate the intrinsic performance of such algorithms on a given problem. Consequently a set of experimentation has been performed to find the best algorithm for the problem of camera positioning in complex environments.

5.2.1/ Design of experiments

To find the best solution to optimize the coverage of an area, many experiments have been performed to compare PSO and GA. PSO is easier to implement and runs faster, but GA is more flexible and generic thanks to the many tunable parameters. To compare and evaluate their respective performances, we tested them in different scenarios. Each algorithm was tested before having an appropriate set-up of the parameters. The comparison is applied on different scenarios listed below with different sizes and shapes of the area and altitudes.

- z is the height of the camera (within the range $[1/z; z]$).
- Figure 5.1a is an area of size 120×80 (named Simple Room).
- Figure 5.1b is an area of size 240×160 (named Room with high ratio).
- Figure 5.1c is an area of size 120×80 (named Room L).
- Figure 5.1d is an area of size 240×80 (named Room L with high ratio).

The size of the room can be expressed by a ratio between the size of a camera projection (at $z = 1$) and the size of area. The biggest rooms called ”high ratio” needs more waypoints to be covered and also the increased size of the area due to a wider sampling allows more fine positions to place the waypoints.

The design of the experiments in Table 5.1 has been set-up to identify the most efficient algorithm for the positioning of a camera set with maximum coverage depending on the numerous cases. The Design of Experiments (DoE) has been made to take into account shapes, sizes and constraints as well as the fixed altitude and various size for the set of waypoints. The DoE has been established to highlight the impact of the constraints on the optimization process with the GA and the PSO.

The Ground Truth (GT) is the minimum number of waypoints required to fully cover a given area. The sizes and the shapes of the areas were selected so that the GT can be determined. NW is the maximum Number of Waypoints (or cameras) used for the experiments. For each experiment, a solution is computed for a number of cameras from

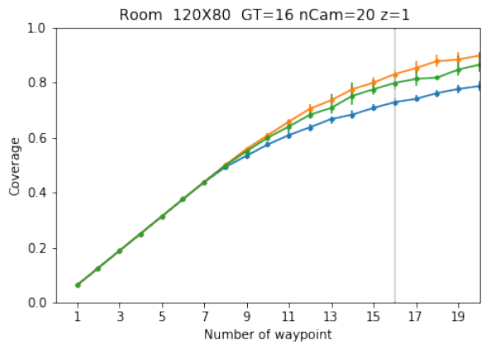
z=1		**GA / PSO / RS**	
		GT	**NW**
Room	**120x80**	16	20
	240x160	64	70
Room L	**120x80**	12	20
1≤ z≤2		**GA / PSO / RS**	
		GT	**NW**
Room	**120x80**	4	10
	240x160	16	20
Room L	**120x80**	3	10
	240x160	15	20

Table 5.1: Design of the experiment to compare the efficiency of PSO and GA in different conditions. (GT is Ground Truth and NW is the maximum Number of Waypoints).

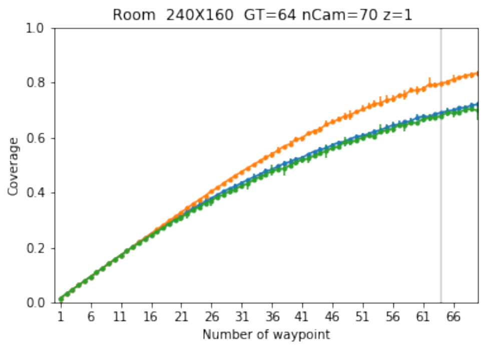
1 to NW. To compare the different algorithms fairly, only 10 000 calls of the cost function are allowed for each optimization (this procedure has been inspired by [124]). The optimization has been executed eight times for each optimization process. The number eight has been chosen to have an operable average despite the potential of high volatility of the algorithms tested. The volatility is due to the usage of randomness. The eight has been fixed empirically due to a profound knowledge of the algorithms applied to the problem and its is also recommended by a tutorial to make a appropriate DoE (see http://3dc.asso-web.com/actualite-12-creer-un-plan-d-experience-methode-rapide-et-efficace-sans-anova.html).

5.2.2/ Analysis of the results

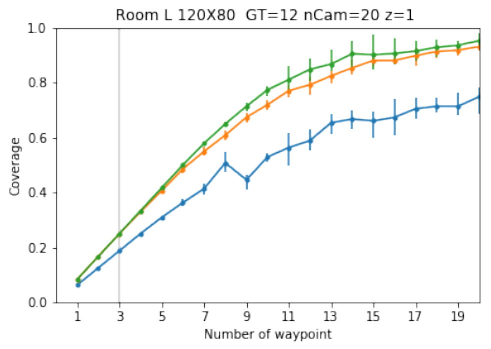
In the experiments described in the Table 5.1, it appears that the GA and PSO algorithms are close in performance in numerous cases (as shown in the Figure 5.2a, 5.2c, 5.2e and 5.2g). Among several experiments of the DoE, some particularities appears despite the globally close results between GA and PSO. Also, as expected, the RS is always the worst solutions (cf. the blue line in Figure 5.2). In the following section, the experiments are used to illustrate the interesting specificity and advantage between the GA and PSO applied to our problem.



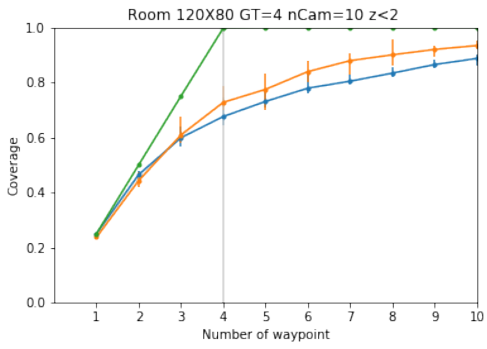
(a)



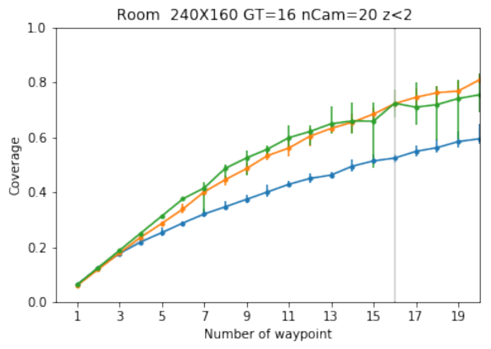
(b)



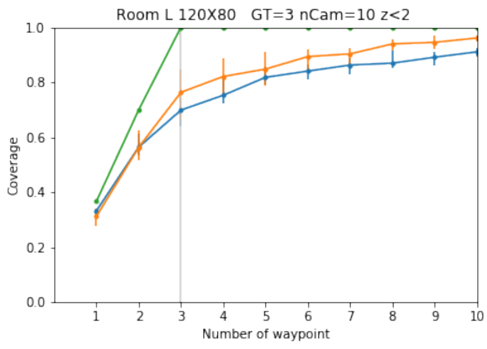
(c)



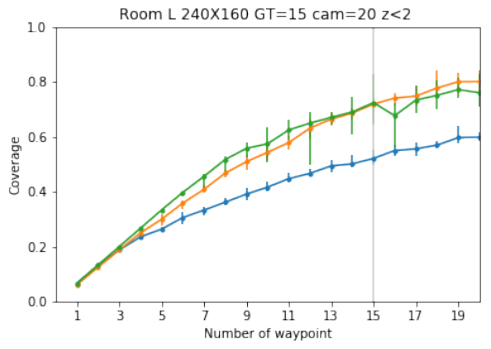
(d)



(e)



(f)



(g)

—	Average with RS
|	Difference between min and max with RS
—	Average with GA
|	Difference between min and max with GA
—	Average with PSO
|	Difference between min and with PSO

(h)

Figure 5.2: Graph of experiments based on the DoE presented in Table 5.1. Each graph represents the comparison result between GA PSO and RS.

In the case of large search space and numerous dimension requires an optimization, the GA appears globally more efficient as shown in Figure 5.2b (or in a lesser case in the graph 5.2a and 5.2e when the number of cameras is over the 16 cameras). In these cases, when the room with high ratio and z=1, the PSO gives a solution close to simple RS which is not an acceptable solution. Instead, PSO is more effective for optimizing small areas as shown in Figure 5.2f and 5.2d. In small rooms disposed as L shape with a z between $1/2$ and 2, the PSO reaches quickly (quicker than the 10'000 calls) to the optimal solution. Where, here, the optimal solution is known and equal to GT (respectively GT=4 and GT=3). In the same cases, the GA proposes an optimized solution (almost comparable to the RS) but can still be considered far from the PSO performance.

This efficiency can be explained by the slight variation of the solution introduced by the PSO. However, this slight variation is not enough to find an optimized solution in a big search space which occurs when many cameras are required or when the local minimum is deeper. The PSO appears really efficient in a relatively small search space while the number of dimensions to optimize is not too high. On the other hand, the variety introduced by the GA allows to shift from local minimum. This variety is valuable in the big search space in order to explore quickly a wide part of it. The variety introduced by the GA can as well become a handicap for a finer optimization. This explains the relatively bad results obtained during the experiment in the small rooms. The variety of the GA negatively affects the accuracy of the solutions and may require a further optimization step to refine. To summarize the comparison, it is difficult to rank the two algorithms in all the environments. GA and PSO have both advantages depending on the area and the number of waypoints to estimate pose. GA is better in the vast search space and for several dimensions to optimize. Whereas PSO is efficient to refined faster the solution. To conclude, we cannot generalize the choice of algorithms for all types of problems. On the other hand, experiments tend to promise better performance when the choice is indexed to the type of problem to be solved.

5.3/ Hybrid GA PSO

Thanks to the experiments done and presented in the previous section (see Section 5.2), the GA and PSO are two algorithms efficient and complementary to solve the problem of camera positioning in complex and potentially vast areas. Thanks to the GA addressed to optimize numerous dimensions in vast search space and PSO ability for have a refined the results, the hybridization can be the key to optimize the cameras positions in all the conditions. The hybridization aims to exploit the best of both algorithms.

5.3.1/ The different hybridizations

Different hybridizations of the GA and PSO can be made. Each hybridization of GA and PSO has advantages and disadvantages. This following section is focused on the main hybridization strategies of GA and PSO.
Premalatha al et [125] propose three different solutions to hybridize the GA and the PSO:

- GA and PSO are employed in parallel. The best solution between both algorithms at each step is used into the other algorithm. For example, if the best solution at the end of the first generation is from PSO, this solution is used as a new individual for the crossover on the GA. Or if the best solution is from the GA, this good individual is employed in PSO as best particle for the next draw. This operation continues until such time as the convergence of both algorithms.

- The GA is used to introduce variety on the PSO, when the PSO is stagnating. Stagnated states are reached when the algorithm isn't able to find and upgraded solution after a predefined number of iterations. In this case, the GA introduces variety by proposing other solutions for PSO. This hybridization has to be managed carefully due to its high risk of non convergence.
- The GA is used until the convergence point. When the GA converges to a solution, the PSO is used to refine with one more optimization. This solution is costly in time due to its double optimization and its double convergence. Finally, this hybridization uses the GA optimization as an initialization for the PSO.

The last hybridization, using GA as initialization for PSO is one of the most suitable for the problem of cameras positioning in a vast and complex area. The experiments made until now (see Section 5.2.1) confirms the mechanism described by the last hybridization of GAPSO. In fact, for our problem, GA is efficient to run through all the search space. On the other hand, the PSO ability to refine the solution is also confirmed. In this case, the GA can be a very good initial guess for the PSO.

In Shi et al. [107], the hybrid PSO GA (same as GA and PSO employed in parallel) was studied for 6 problems listed as F1 to F6. The six problems have a known global optimal. In this article [107], the different problems are used to demonstrate the efficiency of hybrid PSO GA and search the appropriate set-up for their parameters. One of the interesting aspects presented is the importance given to find the best set-up for each algorithms. The set-up of the algorithms has to be adapted to the hybridization and the problem. As ir was discussed in Shi et al [107], numerous tests have to be done to find the good set of parameters for the algorithms.

In our case, the GAPSO is used within a first time then a GA and a PSO are performed next to refine the solution. In this case, the GA has to introduce even more variety in order to be more efficient. Consequently the GA has to be modified to have a mutation ratio slightly higher.

5.3.2/ Experimentations

To compare the efficiency of the hybridized GAPSO to the GA, one more experimentation is proposed. The experimentation followed the rule fixed during the comparisons as shown in Table 5.1. Based on our knowledge and the future use (waypoint positioning in vast area), it appears that the room in L shape with high ratio shown in Figure 5.1d is the most suitable to test the hybridization. The L shape room proposes a big search space which can require an important amount of cameras to cover it and the results obtained in Figure 5.2b can be improved. Also the room in L shape is closer to a realistic configuration. The proposed experiment uses the GA for a maximum of 100 generations and the deduced solution as an initialization for the PSO. Also the PSO is locked at 100 iterations.

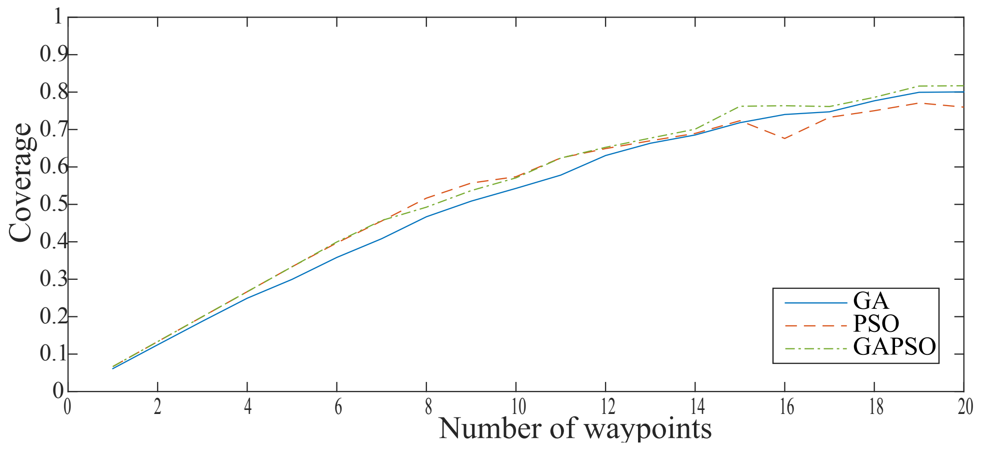


Figure 5.3: Comparison between GA PSO and the hybridization of GAPSO.

The limit of number of generation is fixed to respect the 10’000 calls of the cost function per algorithms.

The set-up of the GA has been slightly modified by increasing the mutation ratio and the PSO is also adapted by reducing the inertia (by passing from 0.5 to 0.4). Before the reduction of the inertia of the PSO, few tests were made, especially with a dynamic inertia. A dynamic inertia can be efficient and allows the PSO to start with a bigger inertia to explore more the search space at the beginning and time after time the reduction of it helps the PSO to converge faster. In this case, a small decrease of the inertia is applied at each iteration (around 0.001 at each iteration). Finally, this method does not provide a significant gain. The dynamic inertia is finally not really useful in the context of hybrid GAPSO. The solution preferred for the PSO set-up has a slightly lower inertia parameter (around 0.4).

5.3.3/ Results and comments

The room in L shape with high ratio, where the comparison between simple GA and PSO was performed, is also used to compare the GAPSO.

In the Figure 5.3, it is appearing that the hybridization of GA and PSO increases slightly the percentage of coverage. This graphic can be separated into two parts, the left side with a relatively low number of waypoints (or cameras) to estimate pose (until 15) and the right part with more waypoints. In these experiments, each waypoint is defined according to x, y and z. In example, for 15 waypoints to estimate pose, 45 dimensions have to be optimized. Both sides of the graphic (Fig 5.3) highlights the different algorithms behaviour and confirms the comparisons results obtained with GA and PSO (in Section 5.2). The PSO is more efficient at first when the number of dimensions to optimize is reduced and the GA is efficient in the big search space with an important amount of dimensions to optimize. The GA became better than the PSO in the right part of the Graph 5.3. The solution proposed by the GAPSO on the left side of the Graph 5.3 is slightly better than the PSO. On the other side, GAPSO proposed a solution more refined than the simple GA. This refinement is due to the PSO ability to optimize the solution from the first optimization (GA).

Finally the main advantage of GAPSO is to propose at almost any time the best solution.GAPSO can reduce the limitations of the GA and helps to go deeper in the optimization process in order to have a refined solution. GAPSO is efficient despite the number of dimensions to optimize and is more robust depending on the size of the search space. The main drawback of the GAPSO is caused by the double convergence, leading to increases in the computation time.

5.4/ Going further, more experiments

The previous experiments made were focused on several simple area. The next step is to increase the difficulty with more complex scenes investigated, by:

- Adding more non-interesting zones.
- Adding hole in the area.
- Increasing the size of the area.
- Increasing the search space by adding more parameters (such as the roll degree of freedom).

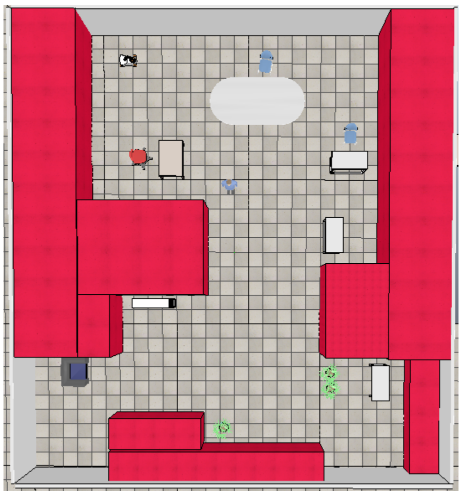
The following sections presents the results obtained by increasing step by step the difficulty. In the following section only the more significant steps will be presented.

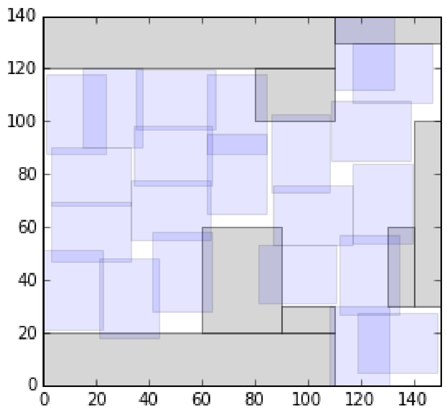
5.4.1/ Uninteresting rectangular zones

The environments of these experiments tries to be more realistic, consequently the room is designed with more ”obstacles”. Here the obstacles are the non-interesting zones which haven’t interest to be covered. The non-interesting zones are added in the map to cover in order to make the area more complex as explained in Section 4.3.1. The following areas are designed to be more challenging and more realistic without a perfect ground truth (GT is not an integer). The room presented until now has a shape designed to have a known number of waypoints as shown in Figure 5.1 where the GT is given.

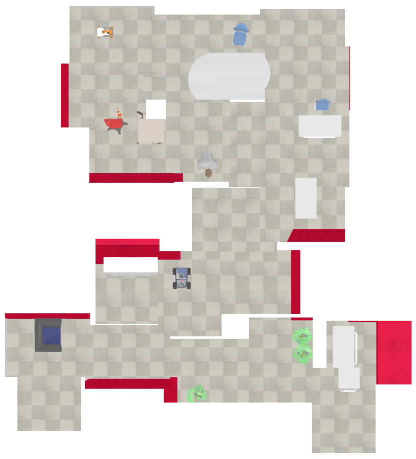
The simulated room is 15×14 m^2 which corresponds more or less to a large lecture hall. The areas in red (see Figure 5.4a) represents the zones which do not require coverage. Every camera can cover a 4×3 m^2, when z is equal to one. The z factor can be equal to $[0.5, 1, 1.5]$, and the cameras can turn at $90°$ to have the image in portrait or landscape. All of these parameters are taken into account in order to compute the waypoints position.The optimization of the waypoints position is made using only the GA, but this time it is applied until the convergence (no limits in the number of cost function calls).

After running the single GA, a well optimized waypoints positioning is given (see Figure 5.4b). Not perfect but good enough with a limited number of cameras. At each waypoints, an image is captured in order to offer a mosaic image of the scene with a restricted number of small black holes (see Figure 5.4c). The solution obtained is comparable to the experiments made previously (see Section 5.2.1) despite the increasing number of non interesting zones and the increased size of the search space The GA converges to a suitable solution. This confirms again the ability of adaptation of the EA optimization and more exactly the single GA. These results encouraged us to go further.


(a) Room in V-rep.



(b) Poses of every image captured in the room.


(c) Reconstructed room based on the waypoints positions.

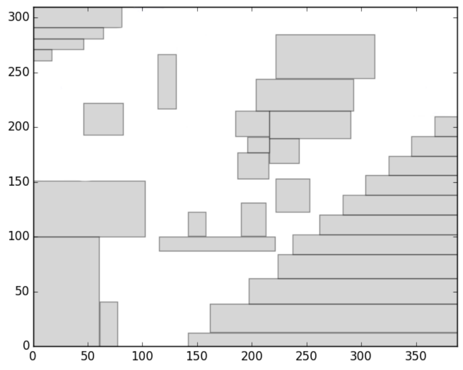
Parameters	**Value**
z	[0.5;1;1.5]
γ	portrait or landscape
Size of the area	150x140 px
Maximum size of camera projection	30x22 px
Number of waypoints	22

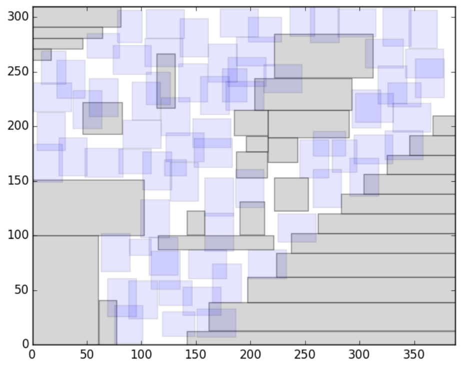
Figure 5.4: Indoor area coverage using V-rep to simulate a realistic environment.

5.4.2/ Uninteresting rectangular zones with holes

The previous experiments shown the efficiency of the single GA despite numerous uninteresting rectangular zones and slightly bigger room (room with a higher ratio between size of the map and size of a camera projection). The following experiments try to push a bit more the optimization process using a single GA. The increased complexity of map were made by adding much more obstacles with some holes in the middle of the map. In fact, until now, all the obstacles were added around the bounding of the area. Adding obstacle in the middle to create a hole in the area will increase significantly the complex-

(a) The area to covered taken from satellite images.


(b) The mask of the area to cover.

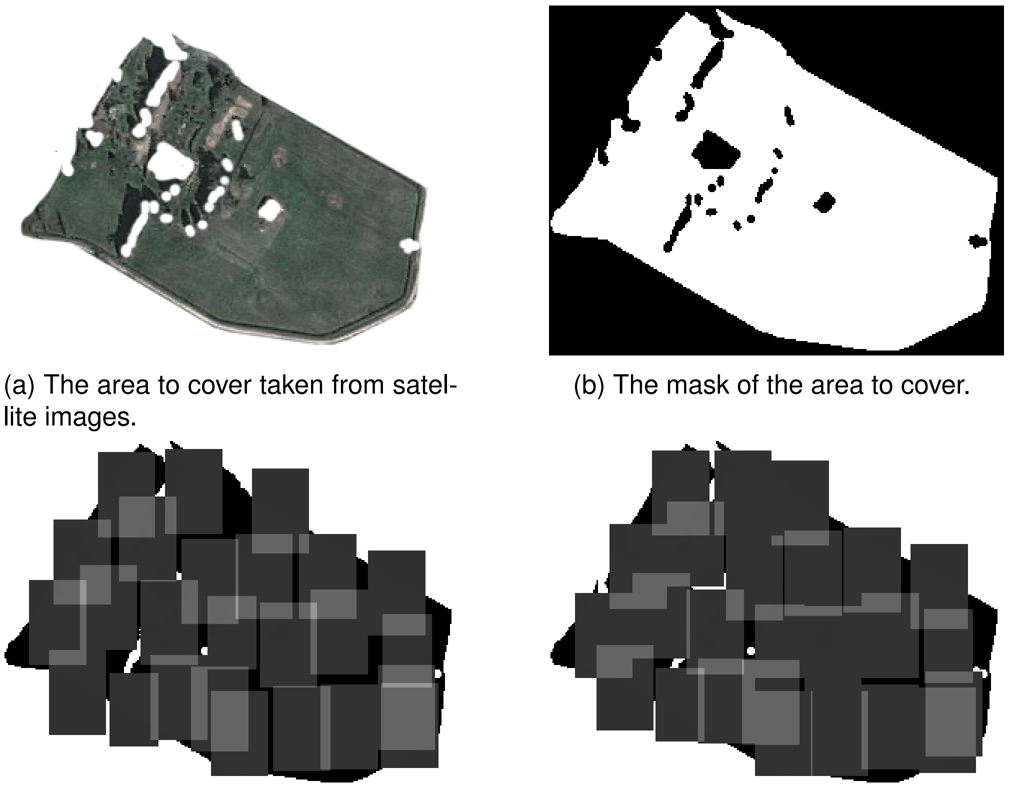

(c) Area covered by the network of cameras using the GA.

Parameters	Value
z	[0.5;1;1.5;1.75]
γ	portrait or landscape
Size of the area	380x310 px
Maximum size of camera projection	42x54 px
Number of waypoints	75

Figure 5.5: Coverage area from satellite images with 75 cameras for a coverage of 76.39% using the the GA.

ity of the coverage estimation. To simulate a realistic environment the map is designed manually based on a satellite image (see Figure 5.5a). Each rectangular obstacle has been placed to reproduce the buildings as shown in the satellite images.

The result obtained by the GA optimization (see Figure 5.5c) shows, one more time, the adaptation power of the single GA to complex scene. The total coverage of the area is around 76.5% for 75 waypoints. The answer proposed is not perfect and can be improved in order to reduce some overlaps and black hole. The important number of dimensions to optimize (75 due to the addition of z and γ) and the increased size of the area (twice bigger than previously) highlight a limit of the simple GA optimization.
The most interesting aspect of this experiment is the highlight of the efficiency of the single GA in a real complex environment. The solution proposed can be considered as good for the purpose of the challenge (obstacle, hole, numerous waypoints to pose estimate, vast area, etc.).


(a) The area to cover taken from satellite images.

(b) The mask of the area to cover.

(c) Area covered by the network of cameras using the GA 90.76%.

(d) Area covered by the network of cameras using the GAPSO 92.81%.

Parameters	**Value**
z	[1;1.5;1.75]
γ	only portrait
Size of the area	278x214 px
Maximum size of camera projection	35x52 px
Number of waypoints	25

Figure 5.6: Coverage area from satellite images with 25 cameras for 90.76% of coverage using the GA and 92.81% of coverage using the GAPSO.

5.4.3/ Using mask to describe the area

Based on the limitation of the map design and the necessity to go one step further, the areas design has to evolve. The solution proposed representing the areas until now was the addition of rectangles which represents uninteresting zones by removing the corresponding points of the grid (as explained in Section 4.1.1.5). The primary advantage of using uninteresting rectangular zones was impacting the coding implementation. It is easier and fast to remove rectangular shape from the grid map. This facility becomes a lock for more complex areas. In addition, it is revealed not ”user friendly”.
The solutions chosen for the following experimentation are using a binary mask of the area to cover. The mask represents in white the area to cover and in black the non-interesting zones (see Figure 5.6b). This solution is finally more ”user friendly” for the complex and realistic area (the user just have to give a mask of the area to cover) and do not change the fundamental concept of the grid map used until here. Each white pixel of the mask is a point of the grid to cover.

For this first experiment with the binary mask to describe the area to cover, a smaller area is selected. A smaller area involve a smaller amount of similar waypoints necessary to fully cover it. In the Figure 5.6a, the area to cover is extracted from a satellite images to have a mask (see Figure 5.6b). The single GA is performed with 25 waypoints to estimate pose. The solution obtained by the single GA is re-injected in the PSO. Each waypoint has to be placed on $x; y; z$. In order to test this new paradigm, the rotation γ is removed from the parameters to optimize.

To initialize, the single GA was performed. The results are visible in Figure 5.6c. The optimized waypoints positioning cover 90.76% of the area with 25 waypoints and the single GA converge after just 67 generations. The solution given by the single GA is already good enough despite the complexity of the map. The solution obtained is conformed to the expected result. The solution of the single GA gives a well optimized waypoints poses, despite the new area design for complex map.

Among the experiments presented until now (see Sections 5.4.2 and 5.4.1), only the single GA was employed for the optimization. On the last experiments (see Sections 5.4.2) with a bigger and more complex area composed by hole, the limitations of a single GA appears slightly. This observation is confirmed and is getting bigger when addressing more complex areas; more complex than the one designed according to the satellite images with mask (as shown in Figure 5.6a and 5.6b). The solution proposed being to apply a GAPSO. The PSO will allow the refinement of the GA solution. The PSO is used with an initialization out-coming from the first optimization (using the GA solution). Finally, the result presented in the Figure 5.6d shows a much more refined coverage with significant reduction in the amount of black holes and overlaps (coverage is over 92.8%).

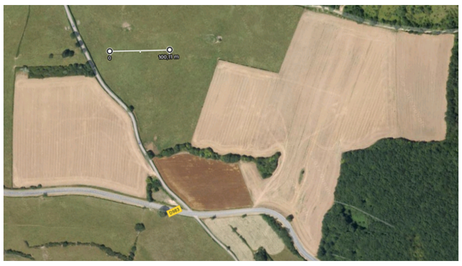
The overlaps of the camera projection is reduced by the use of GAPSO. In fact the coverage with simple GA has an overlap of 26.43% of the area covered when the solution obtained with GAPSO has 23.39% of overlapping.

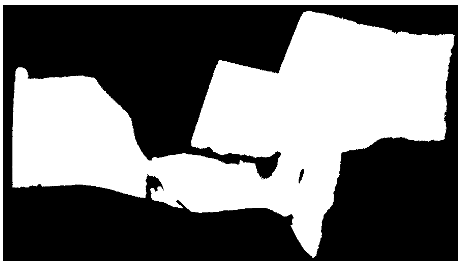
The main result of this experiment was to evaluate if the area design modification have a significant impact on the waypoints positioning. The conclusion of this experiment is that the impact is significant, but it is compensated by the usage of GAPSO. Despite the increased complexity due to the area shape (possibly also due to the mask), the usage of the hybrid GAPSO permits to compensate it. The experiments allows also to evaluate the improvements made by the GAPSO. The GAPSO hybridization is robust and flexible despite the strong constraints due to the non-geometric area.

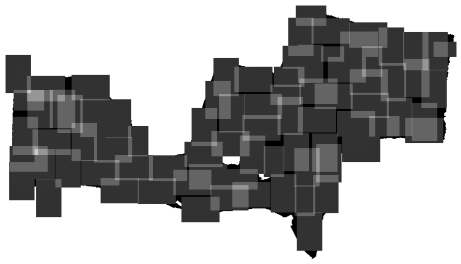
Thanks to these tries, the size of the area to cover and the number of waypoints can be increased. The next experiment has to test the limitation of the GAPSO optimization in terms of size and number of cameras using a mask to describe the area.

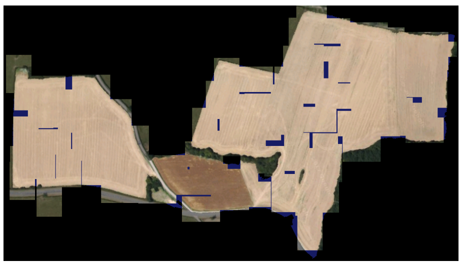
5.4.4/ Using mask for area with high ratio

Based on the last experiment (see Section 5.4.3), a much bigger area with much more waypoints that needs to be pose estimated are presented here. The goal of the following sections is to see the limitation of the GAPSO optimization when the parameters of the experiments are pushed to the maximum. The maximum in term of area size, number of waypoints and shape complexity.


(a) The area to cover taken from satellite images.


(b) The mask of the area to cover.


(c) The result of the coverage with the waypoints position.


(d) The final mosaic images.

Parameters	**Value**
z	[0.6;0,8;1]
γ	portrait or landscape
Size of the area	895x502 px
Maximum size of camera projection	75x50 px
Number of waypoints	80

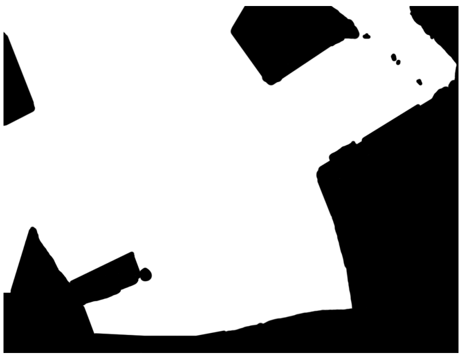
Figure 5.7: Optimization of the waypoints poses with a vast outside area and just a few black holes.

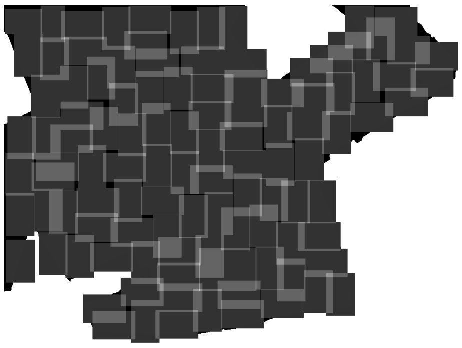
5.4.4.1/ Vast and complex outdoor

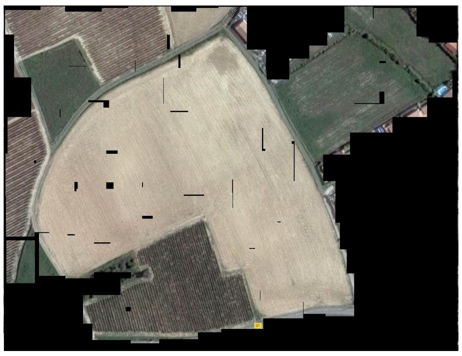
In this experiment, a much bigger and complex area is presented. A bigger area involving the increasing of the search space. In the following example shown in Figure 5.7b, the satellite images are used to define the area to cover (in white). The size of the area has been increased to have a grid composed by almost half million of points with nearly 200 thousands points to cover. The objective is to increase the difficulty in this experiment. In addition to the increased size of the area to cover, the shape of it has become complicated. To do that, an area with several sub-parts composed by small spaces and holes has been selected.
During this experimentation, a high coverage rate is required. We expect more than 98% of coverage rate. More than 95% of coverage rate is a high requirement and push the optimization in term of precise positioning to the limit of the GAPSO. The risk to ask a very high coverage rate is to need a lot of waypoints with several overlap and consequently a long time before convergence. The convergence time is linked to the number of dimension to optimize (80×4) and thus the size of the search space.

In order to cover the area with high ratio, the solution can be to use a bigger focal length or higher altitude to have a wide area covered at each waypoint, thus keep few waypoints to control the area. The other solution is to increase the number of waypoints. The increased number of waypoints can be a source of difficulty for the optimization. Despite

(a) The area to cover taken from satellite images.


(b) The mask of the area to cover.


(c) The result of the coverage with the waypoints position.


(d) The final mosaic images.

Parameters	**Value**
z	[1;1.5;1.75]
γ	portrait or landscape
Size of the area	1113x848 px
Maximum size of camera projection	69x104 px
Number of waypoints	110

Figure 5.8: Optimization of the waypoints pose with a big outside area: (a) is the area to cover taken from a satellite images,(b) is a mask of the area to cover, (c) is a result of the coverage with the waypoints position, (d) is the representation of the black hole.

the difficulty to manage more waypoints, the GAPSO associated to the adapted cost function allows to manage it as shown in example Figure 5.7c. In the Figure 5.7c and 5.7d, the area is covered by 80 waypoints for 98.48% of coverage. To reach this coverage rate, the GA convergence is achieved after 4'856 generations. The important number of necessary generation before the convergence of the GA and a similar increasing time computation for the PSO, allows us to glimpse the limits of the GAPSO for the too big search space with numerous waypoints to pose estimate. Despite these potential future limitations, the solution of the GAPSO is relatively fast (4'856 generations) and efficient (cover 98.48% of the area). The gain offered by the GAPSO compared to the simple GA is 0.079 which allows to perform from 98.40% to 98.48% of coverage. The overlap is 12.79% for the "all area" while the overlap is 30.59% for the covered area.

5.4.4.2/ Biggest map with numerous waypoints

The previous experiments (see Section 5.4.4.1) showed the efficiency and flexibility of the GAPSO to the big map with lot of waypoints in a really complex area. Despite an important increasing computation time, the GAPSO can be used to go even more further and try to touch the boundaries.

The grid is composed by almost 1 million of points, with more than 616 thousands points to cover. The map proposed here is the biggest tested. The number of waypoints to cover this vast area have been increased to reach the 110 waypoints. The number of waypoints to estimate pose is among the most important compared to the literature as shown in [49, 20, 15, 50, 4, 25, 13, 29, 44, 3]. The GAPSO is executed with success and the final coverage is over 98.26%. The GAPSO allows to upgrade sightly the GA optimization (98.19% of coverage) by increasing the coverage by 0.069 points. The overlap is 15.80% for the "all area" while the overlap is 24.74% for the covered area.

To reach this coverage rate, the GA optimization converge after 170'501 generations.
The important number of generations before reaching the convergence with the GA in a first time, and proceeding to a PSO optimization for a second time, reveals a long time computation before the final solution given by the GAPSO. This shows the great efficiency of GAPSO to optimize the position for numerous waypoints in a big search space while proposing a really good solution. This show the limit of the GAPSO due to the important number of generations and consequently an important computations time (a few hours with a core i7).

To nuance the really important time required reach the double convergence (GA and PSO), the context of the experiment has to be highlighted. In Fact the area to cover is important but also the coverage rate required is also an important factor because more this coverage rate is important more a fine tuning of the numerous parameters of the solution must be done. Reducing the coverage rate required and consequently the number of camera allows a much faster optimization. For example a similar experiment with 100 waypoints reach 92.659% of coverage with the GAPSO after only 16'545 generations instead the 170'501 generations.

5.4.5/ Waypoints positioning limitations

Among the numerous experiments performed, the GAPSO appears as a good solution to optimize the positions and orientations of numerous waypoints (or cameras) in a vast and complex map. In the first time, the GA appears efficient enough for the optimization in rooms with high ratio (see Figure 5.4 or 5.5); but after more experimentation, the single GA appears weak to finally refine the solution (see Figure 5.6). The contribution of PSO was essential to have a more refined solution and also allows more flexibility especially when the number of waypoints is restricted.
During the different experiments proposed ,some limitations appears. Among the limitations, the most important is mostly the consequences of a high number of waypoints to have a high coverage rate. The size of the area and moreover the number of waypoints to estimate pose has an important impact on the time convergence of the GAPSO. This time convergence increases even more when the number of waypoints is high to result in a high coverage rate.
To illustrate this phenomenon, the GAPSO ran on the last experiment (same map) but

with less waypoints(only 80). Consequently, the coverage rate is also smaller around 85.5% of coverage for 80 waypoints. In these conditions, the number of generation for the GA before convergence is just around 200 generations. This 200 generations for the GA and a similar number of iterations for the PSO appears giving a huge difference between the 98.26% of coverage of the 110 waypoints in 170'501 generations. This huge difference is due at the same time to the number of dimensions to optimize and the fine refinement due to the high coverage rate. Thanks to these observations, it is appearing that the main limitation of the GAPSO is not the big or complex area, but is mostly the high number of waypoints (or number of dimension to optimize) and the high coverage rate which involve a fine refinement of each waypoints position.

6

Coverage path planning problem

Contents

This chapter is devoted to the Coverage Path Planning (CPP) problem. The introduced method uses GA and is followed by a more flexible GAPSO. These algorithms gives an efficient result to estimate the pose of a set of fixed cameras. To find a solution to the CPP problem, we propose for the first time to use the algorithms developed for the camera positioning to find the set of useful waypoints, instead of the conventional methods based on sweeping trajectory. The optimized poses of the cameras are considered as waypoints of the complete path. The details of the proposed solution to optimize the CPP problem and few experiments made are presented in these sections below. The sequential method is proposed at first followed by experimentation in different environments. Finally a global CPP method will be presented.

6.1/ Sequential method

To optimize the CPP problem, we perform a simple and innovative method called sequential method. The method can be decomposed into three principal interconnected parts.

- Number of waypoints : A crucial step is to estimate the number of waypoints. A wrong estimation of a too high or low number of waypoints will cause a bad area coverage or a too long computation time.

- Waypoints positioning : As already discussed numerous solutions has been studied to optimize the pose of the cameras depending on several constraints. Based on it and the experiments made until now, an efficient algorithm (GAPSO) has been chosen and adapted to estimate pose for each waypoint (see Section 5.3).
- Path plan computation : When the number and the positions of the waypoints has been computed, the last step is to find the shortest route passing by all the waypoints. The route must start and finish at the same position (similar to the TSP cf. Paragraph 2.4.2.1).

The proposed method has the advantage of optimizing independently all the waypoints and the path plan. This method is also not systematically based on sweep and allows to have a shorter path adapted to a complex map. Contrary to the conventional solution (based on sweep), our proposed algorithm offers the ability to return to the starting points.

6.1.1/ Number of waypoints estimation

It is difficult to estimate properly the minimum of waypoints which are necessary to cover a complex area. To do so, a two-steps procedure has been implemented. The procedure is based on the pose optimization for a fixed number of waypoints introduced in the previous Chapter5. The first step is to find the minimum number of waypoints depending on the area to cover like formulated in Equation 6.1.

$$\frac{A_{room} - \sum_{i=1}^{n} A_{walli}}{A_{cam}} \times \text{Threshold Rate} = \text{NWayPoint} \tag{6.1}$$

With the parameters:

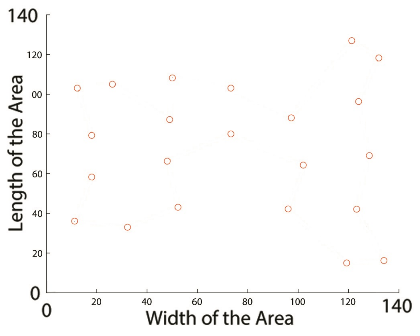
- A_{room} : area of the Room
- A_{Wall} : area of the obstacle like walls
- A_{Cam} : area covered by the camera in the maximum size of z
- NWayPoint : number of waypoints
- Threshold Rate : objective threshold rate
- S : one solution of waypoints set
- $evalCost$: cost function

The second step is the computation of an optimization until the threshold is reached. At the convergence of each optimization, if the threshold rate is not reached, one more waypoint is added and a new optimization start with an extra waypoint. The algorithm used to estimate the number of waypoints is explained in the ”Algorithm 1 Estimation of the number of waypoints”.

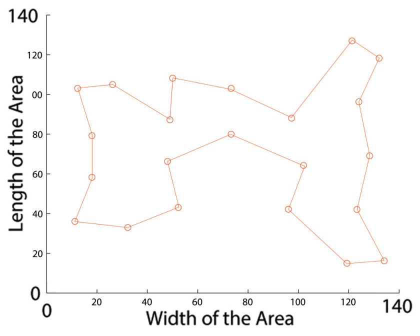
- S : one solution of waypoints set
- $evalCost$: cost function

Algorithm 1 Estimation of the number of waypoints

```
procedure NmbWaypoint($A_{room}, A_{Wall}$, Threshold Rate, $A_{Cam}$)
    $S \leftarrow 0$
    $NWayPoint \leftarrow \frac{A_{room} - \sum_{i=1}^{n} A_{walli}}{A_{cam}} \times$ Threshold Rate (Equation 6.1)
    while $evalCost(S) \leq ThresholdRate$ do
        $S \leftarrow GA(NWayPoint)$
6:      $NWayPoint \leftarrow NWayPoint + 1$
    return $NWayPoint$
```



(a) Every pose after the optimization of the waypoint positioning.



(b) Path compute with GA multi objective for TSP.

Figure 6.1: Optimization of the path planning.

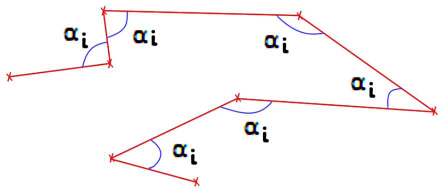
At the end of these steps, we have the number and a good set of waypoints poses from the last GA convergence. The waypoints poses can be directly used, or can also be refined with the GAPSO (as shown in the GAPSO part cf. 5.3). Once the number and the efficient poses of the waypoints are found, the next step is to compute the path planning. The path planning has to pass by all the waypoints found before returning to the starting point.

6.1.2/ Sorted waypoints and path planning.

In the previous sections, the methods to obtain the list of waypoints positions to have a desired coverage has been detailed. Now, the list of waypoints have to be sorted in order to compute an efficient path with the shortest travelling distance passing by all the waypoints. In order to create an efficient path passing by all optimized waypoints, the problem is formalized as a TSP.

Traveling Salesman Problem.

The sorted path can be formulated as Travelling Salesman Problem (TSP). The TSP is inspired by a question asked by a salesman "**What is the shortest path passing by each city only one time and returning to the starting city? Especially when i know a list of cities and the distances between each pair of cities.**". The TSP problem is a well known NP-Hard and NP-complete problem (see in [71]) and different solutions exists to optimize it depending of the context. Ponnambalam et al [126] proposed to



(a)

Figure 6.2: Extraction of the curve angle in the trajectory.

optimize the TSP by using a multi-objective GA. The GA proposed in [126] is given with an appropriate set-up promoting high mutation rate and advocate the population size, crossover and selection mode. Davies et al [80] proposed also to use the GA to solve the TSP applied on robotic with obstacle constraints. The idea is to propose the shortest path passing by all the given waypoints and avoiding the collision with obstacles.
Based on the literature, GA seems efficient to optimize the TSP problems, but it has to be set-up properly depending on the specific TSP problem (see Section 3.3.6). The know the more appropriate set-up to optimize the TSP, several studies has been made as [78, 79, 127]. The conclusion of these studies is to use a simple GA for combinatory problem. That mean the operator has to be adapted (using swap for example) with a high mutation rate and a very elitist selection. To illustrate the GA ability, one example using an adapted GA (high mutation rate, elitist selection and combinatory formulation) solution is provided in the Figure 6.1.

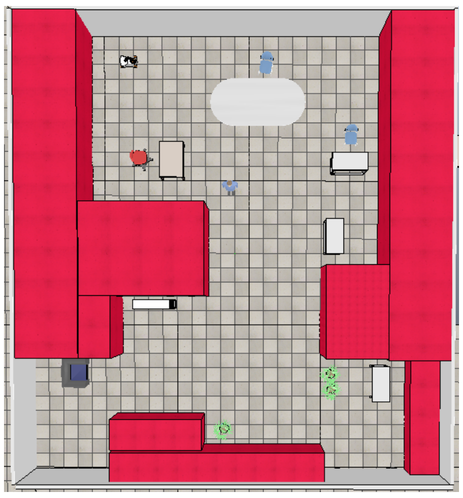
Complexity of trajectory.

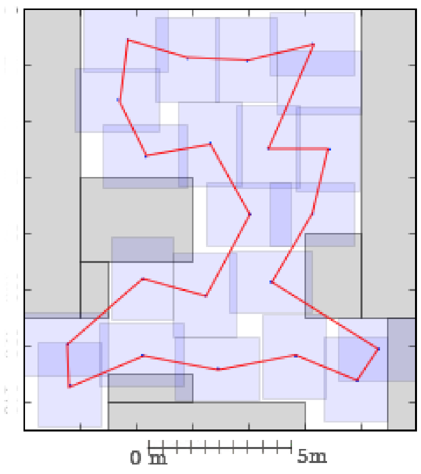
Once the waypoints position estimated and the shortest path passing by all the waypoints computed it is important to evaluate the trajectory complexity. The trajectory complexity is a comparison between our solution and the more classical sweep trajectory.

To estimate the trajectory, two indicators are used to evaluate properly the path complexity. The first indicator is the distance of the trajectory. The distance allows the evaluation of the optimization when addressing the path planning part. This indicator is directly included in the optimization process as discussed previously (see Section 6.1.2). To estimate the complexity in terms of curve for the UAV evolving in the 3D space, the angles at each node are studied. The complexity of trajectory indicator is computed as follows:

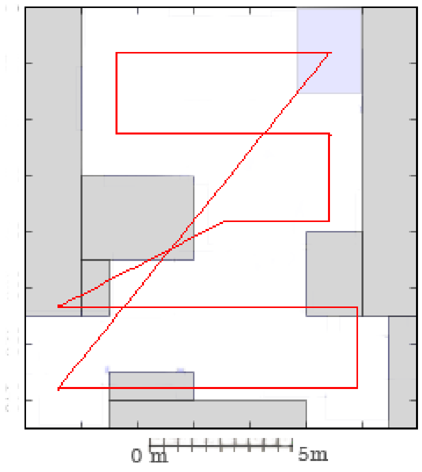
$$\text{Trajectory complexity} = \frac{\sum_{i=1}^{size(\alpha)} 180 - \alpha_i}{size(\alpha)} \tag{6.2}$$

Where α_i is an angle of curve in the trajectory as shown in Figure 6.2.
$Size(\alpha)$ is the number of curves in all the trajectories.
This method provides an idea of the global trajectory complexity despite its simplicity. The two indicators presented (the distance and the trajectory complexity) are used during the following sections to evaluate the advantages of the proposed method.


(a) Room Coverage



(b) Pose of every images with the path planning



(c) Room in V-rep

	Coverage rate	**Path length (in px)**	**Trajectory complexity (Eq.6.2)**
Pattern Method	100%	1137px	97.81°
Our Method	90%	881px	65.21°

Figure 6.3: Optimization of the path planing

6.2/ Experiments

The proposed method for the problem of CPP was tested during different experimentations. The experimentations brings out the advantages and the limits of the developed methods. The experiments are structured in sections which will show the methods and algorithms in more and more complex experimentations.

6.2.1/ Rectangle obstacle

In order to experiment the proposed method for CPP, a simple indoor experiment is proposed to begin. The experiments are made in a simulated room $(15 \times 14m^2)$. The area to cover is shown in the Figure 6.3a. The camera parameters are the same, $4 \times 3m^2$ when z is equal to one and the z factor can have various range such as $[0.5; 1; 1.5]$. On the other hand, the sweep is computed with the camera at the maximum altitude to have the biggest area coverage possible.

Thanks to this first experiment, the path planning proposed appears much more appropriate (see Figure 6.3). Indeed the path planning proposed returns a shortest path, of 881 pixels length when the path by sweeping is at 1'137 pixels length. The path proposed by the sweep is longer notably due to the return to the starting point, but not only. The junction between the start and finish of the swept path is 245 pixels long (the path without the return to the start is 892 pixels long).Moreover, the proposed path planning, in addition to the shorter path proposes also a better trajectory complexity with 65.22° instead of 97.81° for the path with sweep. Finally the proposed solution allows a better path planning thanks to its optimized waypoints. Despite a better path plan efficiency, few points (g_i) in the area are not covered contrarily to the traditional swept method.

6.2.2/ Using mask to describe area.

The gain for the indoor area is important. The gain is mostly due to the well optimized position of the waypoints. The experiment must be extended for a bigger and more complex area with a higher coverage rate requirement.

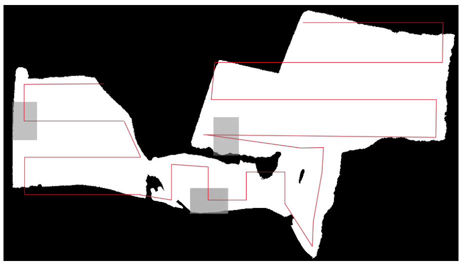
6.2.2.1/ Vast and complex area.

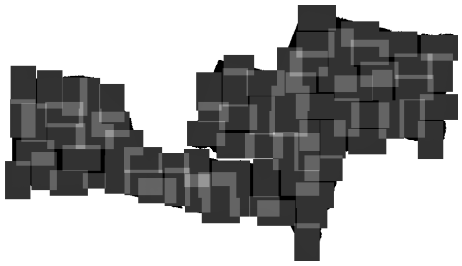
The CPP and more precisely the path planning part is compared to a standard method (the sweeping). We compared the CPP in terms of coverage, path plan distance and trajectory complexity. The standard methods applied for CPP uses an adapted pattern to cover an area. The patterned method application is based on several articles as [73, 22, 61, 54, 128]. In these articles, different kind of sweeps and spirals have been applied to cover the basic area shape. To split the complex area in basic area shapes, the method commonly used is the cellular decomposition (see in Section 2.4.2.1).

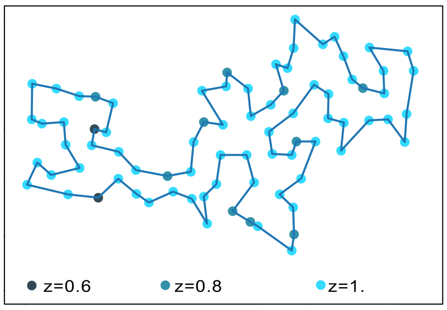
To focus on a bigger areas, the experiment for outdoor waypoints positioning in vast and complex area using a mask is reused as in Section 5.4.4.1.

To compute the sweep, the area is firstly decomposed into cells to apply an appropriate sweep. In the second time. The appropriate sweep is placed in each cell. The size of the sweep has been designed for fixed altitude depending on the size of the camera projection on to the floor.The path plan using a sweeping method is visible in the Figure 6.4a. In order to cover completely the area, several overlaps and non-interesting regions (the black region in Figure 6.4a) has to be covered as well.

The solution proposed by optimizing in a first time the waypoints positions and in second time the path passing by the waypoints is applied. The optimization is made for a set of 80 waypoints which must cover over the 98% of the area. This threshold is reached after 3'101 generations (see in 6.4b). Once the waypoints are optimized (in $x; y; z$ and roll), the path plan is computed using the method based on the TSP (presented in the Section 6.1.2). The final CPP is visible in the Figure 6.4c with the different altitudes represented in colors.


(a) Path planing using the pattern method.


(b) 80 waypoints for 98.28% of coverage.



(c) Path with 80 waypoints for a distance of 4015.6 px.

	Coverage rate	Path length (in px)	Trajectory complexity (Eq.6.2)
Pattern Method	100%	4362.66px	82.14°
Our Method	98.28%	4015.6px	65.78°

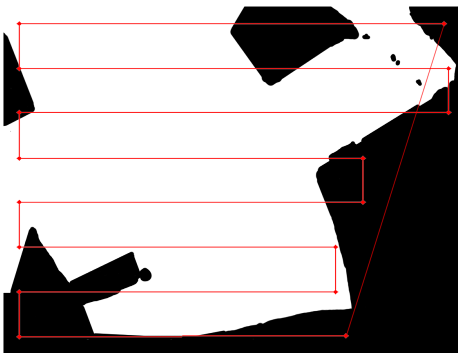
Figure 6.4: Experimentation of coverage path planning in outdoor area.

One more time and despite the increased size and complexity of the map (which increases greatly the complexity and the number of the waypoints optimization), the proposed solution gives a better result. The path plan is shorter: 4'362px for the pattern method while we obtain 4'015px for our method. The path is also easier to compute in terms of trajectory 82.14° against 65.78°. In fact, the optimization of the waypoints, despite a not completely covered area, offers good waypoints for the path planning.

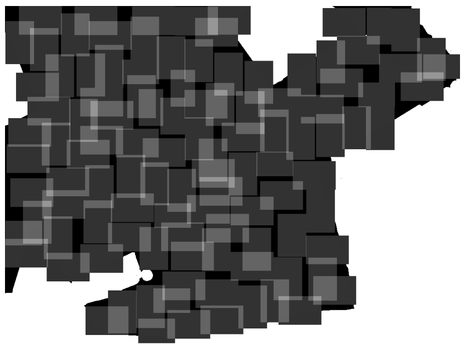
6.2.2.2/ Biggest map with numerous waypoints

To continue the experiments, a biggest area can be selected which requires much more waypoints. The big area selected to test the coverage path planning is the same area used in Section 5.4.4.2. The simple sweep path is computed based on the camera projection at the highest altitude (cf. Figure 6.5a). Our method is computed with 110 waypoints to estimate the poses. After 3'634 generations for a coverage area of almost 95% (see Figure 6.5b), the path can be computed (as explained in the section 6.1.2). The final CPP is shown in the Figure 6.5c.

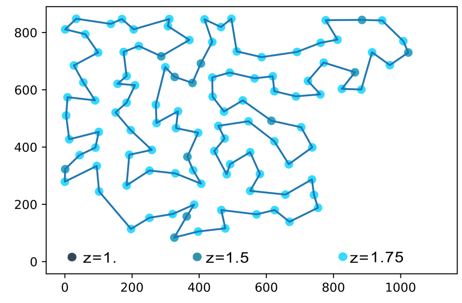
The solution proposed gives a shorter path compared to the swept method (8'173px against 8'582px) for a high level of waypoints coverage. The differences between the 2 paths are not so significant in term of distance. Nerveless the difference of trajectories complexity gives an important advantage for the proposed method.



(a) Path planing using the pattern method.



(b) 110 waypoints for 94.862% of coverage.



(c) Path with 80 waypoints for a distance of 4015.6 px.

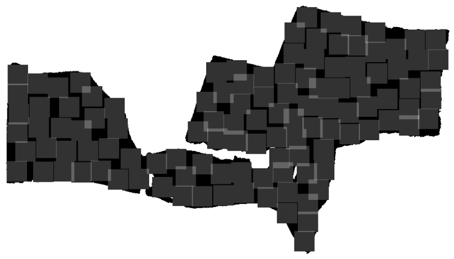
	Coverage rate	Path length (in px)	Trajectory complexity (Eq.6.2)
Pattern Method	100%	8582px	90°
Our Method	94.86%	8173px	69.75°

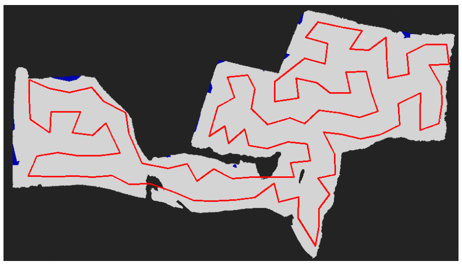
Figure 6.5: Experimentation of coverage path planning in outdoor area.

Thanks to these experiments, the limit of our method begins to appear. The main limitation is due to the difficulty to optimize numerous waypoints. The increased number of waypoints associated to a high level of coverage rate required pushes the GAPSO optimization to its limits. Numerous generations are necessary to reach convergence. On the other hand, the path plan computation appears efficient despite the increased number of waypoints and globally the solutions proposed gives a better CPP in terms of distance and trajectories complexity for a higher coverage rate.

6.2.2.3/ Camera contraint by the trajectory

The previous indoor or outdoor simulations assumed to have an UAV with a camera stabilization for the roll (portrait or landscape) independently of the UAV trajectory. During this simulation, we assume the UAV not able to control the roll independently of the direction and the UAV is sufficiently agile to follow the path. The solution proposed in this section is adapted to the UAV trajectory to constrain the camera directions. In the following experimentation, the roll and the altitude are fixed for the waypoints positioning. The camera size is reduced as a 40x40px square shape projection onto the ground, corresponding to the smaller side of the camera projections. The camera is represented as a reduced


(a) 115 waypoints poses after optimization for a coverage of 94.3%.


(b) Coverage path planning for 99.71% of coverage with a distance of 4785px.

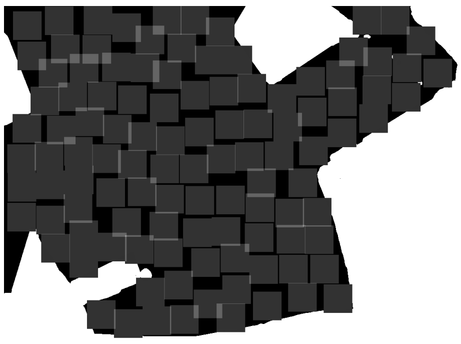
Waypoints coverage rate	**Stream coverage rate**	**Path length (in px)**
94.3%	99.7%	4785px

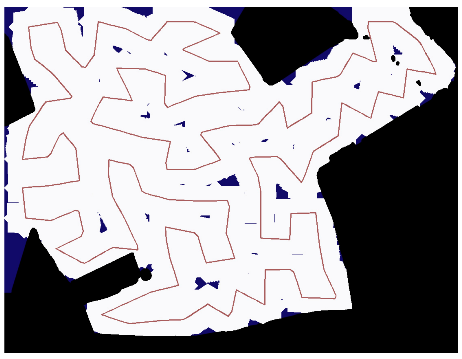
Figure 6.6: Outdoor simulation of the coverage path planning. The camera projection FoV is $40 \times 60px$ and fixed altitude(with a square projection of 40px wide for the positioning optimization).

square used only for the optimization process. The reduction of the camera estimation, the fixed roll and also a fixed altitude allows us to make a coverage at minima of the area. The optimization of the waypoints are computed (as in Chapter 5) with these new constraints. Once the waypoints positioned the path planning can be computed as presented in Section 6.1.2 to have the sorted waypoints.

To estimate the final coverage of the area with a fixed camera on the UAV, the UAV direction have to be estimated prior. Based on the path computed with the TSP formulation, the UAV orientations are deduced by estimating the trajectory to go from one waypoint to another. The next step is to estimate the coverage of the area with the camera stream record and the cameras FoV oriented depending of the trajectories (see Figure 6.6b). The final stream coverage estimation is done with the camera of size 60x40 pixel.
Obviously, the final coverage path planning is better than the estimation established during the waypoints positioning, cf. the following example Figure 6.6a. The camera projection for the optimization is 40 pixel by 40 pixel with 115 waypoints for coverage pose estimation of 94.3% and 99.71% for stream coverage of the full room with the simulation of path planning and a camera projection to 40x60 pixel (see Figure 6.6b). This gain is not taken into account during the phase of waypoints positioning optimization and in consequence does not impact the number of waypoints.To confirm the gain of stream coverage, another experiment has been made with a similar conditions but in a different map. The experiment is made with the map introduced in Section 5.4.4.2 with the camera projection of 70 pixel by 70 pixel with 110 waypoints for a coverage estimation at 88.38% (see Figure 6.7a) and 98.64% (see Figure 6.7b) for the full room after the simulation of fly.

This experiment demonstrates the ability of our proposed method to do coverage path planning in a large area with a non convex shape with different constraints. The solution proposed being adaptable and flexible to the different conditions. This experiment also shown the non-negligible gain of coverage between the waypoints positioning optimization and the final CPP with the stream coverage of the area.


(a) 110 waypoints for 88.38% of coverage fixed altitude .


(b) Path planing using the pattern method. Coverage path planning for 98.6% of coverage with a distance of 8114px.

Waypoints coverage rate	**Stream coverage rate**	**Path length (in px)**
88.38%	98.6%	8114px

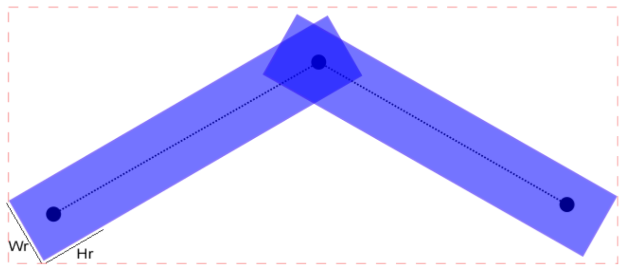
Figure 6.7: Outdoor simulation of the coverage path planning. The camera projection FOV is $40{\times}60px$ and fixed altitude (with a square projection of 40px wide for the positioning optimization).

6.3/ CPP GLOBAL OPTIMIZATION ATTEMPT

The experiments made previously were successful despite the different optimization steps. Indeed to compute an efficient CPP, several optimization have been used (see in Section 6.2). First, number of waypoints have to be estimated and the waypoints poses have to be optimized. When the coverage rate is reached, the optimized waypoints can be refined with a PSO to have an appropriate coverage. The final optimization lie in the usage of the TSP paradigm to compute a path passing by all the waypoints.
Despite the success of this greedy method a logical improvement clue maybe to reduce the number of successive optimizations. In the experiments proposed previously, the waypoints positioning optimization shown some beginning of limits due to the too high number of dimensions to optimize (see Section 5.4.5). In the experiment made for a camera constrained by the UAV trajectory, the streamed coverage area increases significantly compared to the waypoints coverage. This increased area covered was not taken into account until now. The idea is to try to limit the number of waypoints by using a global method to optimize the CPP problems.
The method explored in the following sections is merging the optimization of the waypoints positioning and the sorted waypoints process to have only one global optimization for the CPP problem.

6.3.1/ ADAPTED FORMULATION

Until now, the waypoints positioning and the path planning computation were made by using two different GA (and/or GAPSO) during two different optimization processes. The idea here is to combine the cost function from the waypoints positioning and the path planning computation to have an appropriate and global cost function. The cost function



(a)

Figure 6.8: Covered area between waypoints following the path planning. Where Wr and Hr are the size of the camera projection as defined in the Section 4.2.1.2.

requires:

1) Estimation of the covered area. The area has to take into account not only the area covered by each waypoints but also the covered area during the path (stream coverage).

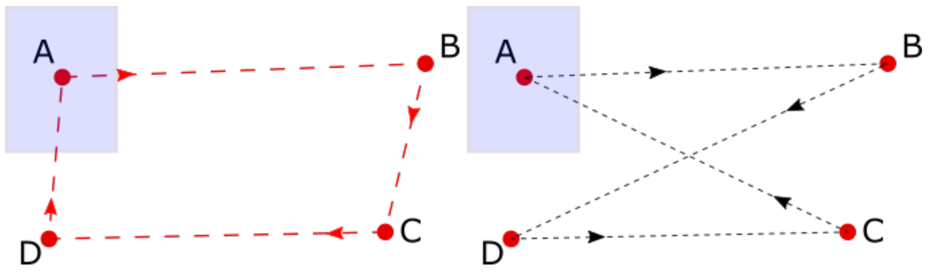
2) Path distance estimation. The path distance was already used to estimate the path passing by all the given waypoints during the path computation (as in Section 6.1.2). In this case, it implies having beforehand the sorted waypoints for the path.

Coverage path plan estimation

To formulate properly the CPP problem, it is essential to estimate the area covered by the UAV during the path passing by all the optimized waypoints. For that, the waypoints and the path must be known. Once the path is known, the coverage estimation can start.
The idea is to consider all the points of the grid G (see Section 4.1 for the grid definition) between two waypoints as covered. To compute the points covered during the fly between two waypoints, the camera fixed on the UAV has to be known in terms of projection size onto the floor and the direction. The direction is directly deducted from the waypoint positions and the size of the camera projection from the camera parameters as its altitude. Based on the projection size onto the floor, the area covered between the waypoints is deduced as illustrated in Figure 6.8.

Which means that the roll of the camera is not optimized but taken into account depending on the trajectory to follow. Moreover, in this case, the altitude of the UAV is fixed in order to simplify the optimization computations and simplify the cost function.

The proposed methodology to estimate the area covered by the path requires to have a ordered set of waypoints. In the previous algorithms, the TSP paradigms with a GA optimization was used to order the waypoints, to have the shorter path passing by the set of optimized waypoints. In this new method, the path has to be computed in the same optimization step as the waypoints positioning. The idea is to use the position of each genome inside the chromosome such that it is important in the estimation. In this case, a genome is the set of parameters of a waypoint. Which means that the same genome at different position in the chromosome does not have the same impact in the cost function. Consequently the optimization is not only focused on finding the best coverage for a set of waypoints but also find the sequence of waypoints positions. This formulation add some



(a) ABCD (b) ABDC

Figure 6.9: Influence of the waypoints order in the new cost function

combinatory constraints inside each chromosome and particle. To illustrate, an example is provided in the Figure 6.9 :
The solution shows ABCD (see Figure 6.9a) where A; B; C and D are the optimized positions of the waypoints. We display also another solution ABDC (see Figure 6.9b) which has found the same waypoints but not in the same order. In the previously proposed method, only the coverage at each waypoint was used to estimate the coverage. Therefore, the cost function estimates the two solutions as identical. In this new estimation of the coverage, by taking into account not only the coverage of a set of waypoints but the entire path plan coverage the order of the position, gives the path to follow. Consequently the solutions are completely different, ABCD (see Figure 6.9a) is much better due to its shorter path for the equivalent coverage.

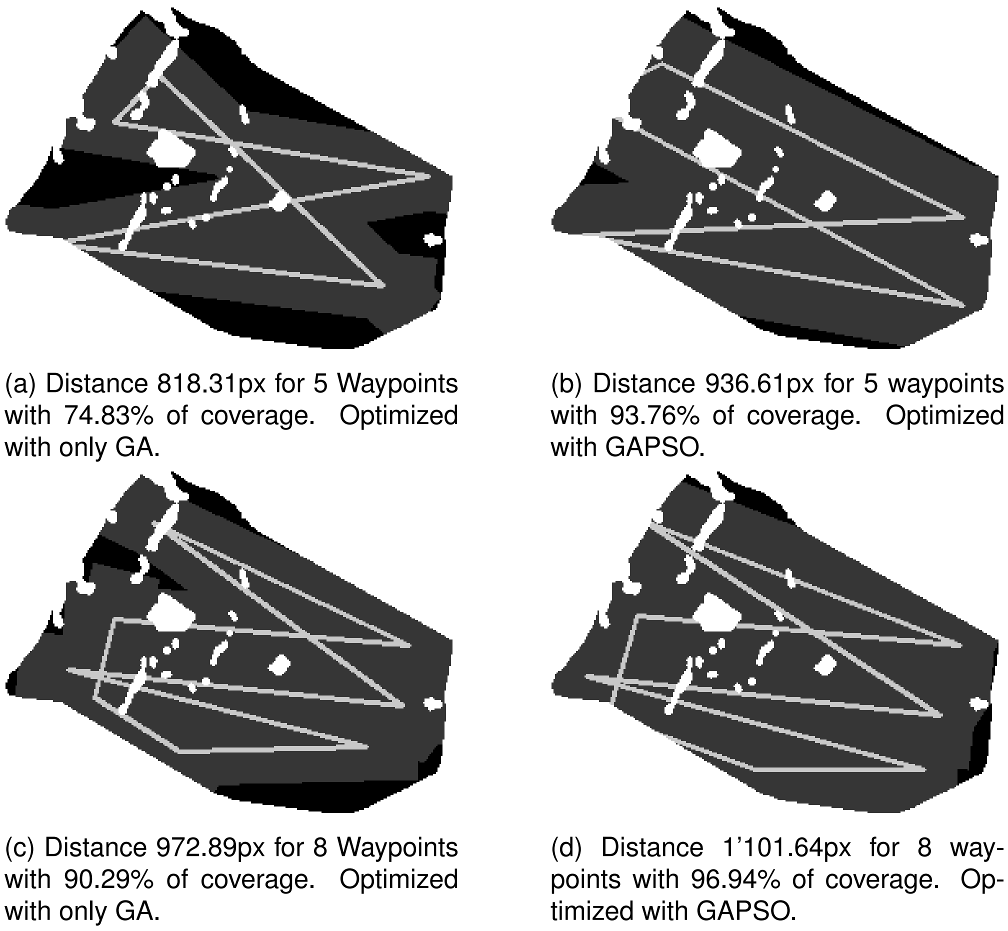
Finally, the new cost function will integrate the covered area by the path passing by all the waypoints and the distance of this path. The goal is to have a cost function that maximizes the area covered with the shortest path at same time. The proposed cost function can be written:

$$f = \sum_{i=1}^{n} B^{(k,p)}(G') + \frac{\sum_{i=1}^{n} B^{(k,p)}(G')}{(\frac{Distance}{n}) \times 5} \tag{6.3}$$

Where $\sum_{i=1}^{n} B^{(k,p)}(G')$ is the number of points cover by the path planning (as in Equation 4.6) and the $Distance$ is the size of the path. The 0.5 is an empirical coefficient to minimize the distance importance in favor of the coverage.

Optimization

Once the cost function is redefined to be able to take in account the coverage and the path distance. One more time the GA is used associated to the PSO for the optimization step. The GA was the best algorithms to solve the TSP problem and also well adapted to the waypoints positioning (even more with an GAPSO). The last reasons are that the knowledge grab about the GA PSO until now make the implementation fast and easy. The solution adopted is to use the GAPSO as introduced in 5.3 associate to the newly adapted cost function (see Paragraph 6.3.1).



(a) Distance 818.31px for 5 Waypoints with 74.83% of coverage. Optimized with only GA.

(b) Distance 936.61px for 5 waypoints with 93.76% of coverage. Optimized with GAPSO.

(c) Distance 972.89px for 8 Waypoints with 90.29% of coverage. Optimized with only GA.

(d) Distance 1'101.64px for 8 waypoints with 96.94% of coverage. Optimized with GAPSO.

Figure 6.10: Path plan for maximizing the covered area with a camera projection equal to 40x60px. The area is covered with 5 and 8 waypoints

6.3.2/ Results and consequences

Based on the new problem formulation and the usage of an appropriate GAPSO directly inherited from the waypoints positioning ,few experiments has been made. The experiments proposed here are based on the map presented in the Section 5.4.3. Once the map is selected, the optimization begins with three waypoints and the number of waypoints is slightly increased until reaching 8 waypoints. For each number of waypoints, three optimizations have been made. In the Figure A.1, some result examples have been shown with the GA followed by GAPSO. The results obtained offers a relatively high coverage for a minimum of waypoints. Despite this good coverage, the path is worst than the solution obtained by the method proposed earlier.

Experiments and discussions

The experiments made allowed to see the advantages and limits of this optimization. The main advantage of this formulation is that it optimizes the CPP problem in a single optimization and reduces the number of waypoints compared to optimizing waypoint positioning. In fact, for the map proposed, only 5 waypoints are required to cover the area (see Figure A.1d) compared to the waypoints positioning optimization that requires 25 waypoints (see Figure 5.6d). Reducing the number of waypoints needed to cover an area is essential to have a better and faster optimization process. The reducing amount of number of waypoints is even more important when the problem is complicated by the position order of the waypoints in a solution. On the other side, despite a good coverage of the area, the path plan appears too long with many overflight (overlaps). Consequently, and despite the good coverage, the path appears not optimized. About the optimization, we can observe that the average number of generations of the GA required is around 47 generations for a set of experiment between 3 and 9. The average number of generations are comparable to the number of generations needed to optimize the waypoints positioning (see in Section 5.4.3), where for 25 waypoints to estimate pose, 67 generations was required. It shows also some limits by using the unique optimization for the problem of CPP. In the other hand, the formulation despite the inconclusive but promising result appears interesting and the cause of the inconclusive results are explored.

Why inconclusive results?

Despite inconclusive results during the first experiments made, it is an interesting track to follow which must be more explored. The inconclusive results of the experiment can be induced by :

- The algorithms: As that was previously introduced, the GAPSO with a similar set-up as the previous experiment for the waypoints positioning are used. Few inconclusive tests has been done to slightly modify the set-up of the GAPSO by adding more mutation for example. The inconclusive preliminary result pushes to consider the GAPSO proposed previously good enough.

- The cost function: The cost function presented in the Equation 6.3 is the more appropriated due to our experiments (some summarized results are presented in the following Table 6.3.2) . Despite that, the cost function has some coefficients which were applied empirically. The coefficients could be an interesting track which must be meticulously set in order to reduce their uncertainty.

Cost function	**3**	**4**	**5**	**6**	**7**	**9**
$\sum_{i=1}^{n} B^{(k,p)}(G') + \frac{400\times\sum_{i=1}^{n} B^{(k,p)}(G')}{(\frac{Distance}{n})\times 2.}/400$	56.23% 458px	60.54% 652px	65.15% 817px	65.85% 1114px	69.41% 1077px	90.67% 1396px
$\sum_{i=1}^{n} B^{(k,p)}(G') + \frac{\sum_{i=1}^{n} B^{(k,p)}(G')}{(\frac{Distance}{n})\times 5.}$	56.15% 444px	59.34% 645px	64.93% 859px	70.14% 1051px	76.59% 1001px	86.77% 1381px
$\sum_{i=1}^{n} B^{(k,p)}(G') + \frac{\sum_{i=1}^{n} B^{(k,p)}(G')}{(\frac{Distance}{n})\times 0.1}$	57.78% 426px	58.33% 638px	61.84% 845px	70.96% 962px	69.64% 1099px	82.50% 1318px

- Complexity and search space: The last potential reason of the inconclusive results obtained can be the increased size of the search space and complexity. The search space estimation was made for the problem of optimizing the waypoints positions and can be reused by part for the problem of CPP for a unique optimization. In this case, the search space is simplified (see the equation of Sp in eq: 4.23), only the waypoints position in two axes are taken into consideration. In fact, the altitude is fixed and identical for all the waypoints as wall as the pan and tilt. The roll of the camera projection is depending on the trajectory thus this removal from the criteria to optimize is reducing the search space. Despite the important decreasing size of the search space for each waypoint, the global search space for the CPP increases dramatically due to the number of waypoints and the order constraint. The global search space is defined as $A_n^{Sp} = \frac{Sp!}{(Sp-n)!} = |Vs|$. Where $|Vs|$ is the number of possible solution for a set of n cameras. Due to the importance of the waypoints order in the optimization process, the number of possible solutions increase significantly. Obviously, due to the complexity of the problem and the increased size of the global search space for CPP problem, it can be one of the important reason of the inconclusive result.

For all these reasons, the optimization of Coverage Path Planning in one optimization by ordering the waypoints does not appear efficient enough and the solution proposed before appears more much more efficient and flexible.

7 CONCLUSION

Contents

7.1/ The origin!

The initial project was a partnership between UBFC and UTP with the aim of developing a smart system composed of cameras mounted on self-organized UAVs. Such project can have applications to video surveillance of vast areas. Addressing such problem is ambitious and requires a wide range of skills since it has numerous constraints as optimal positioning, robust control or flying time limits. Among which we focus our work on the maximization of area to be monitored.

7.2/ What was done?

A vast area can be monitored in static, by ingeniously positioning a set of camera, or in dynamic, by finding an efficient path plan. Both methods are similar in numerous points. Especially on the minimization of the camera/ waypoints number or the path plan size which is a real challenge.

In this thesis, coverage path planning problems are solved using new adaptive methods based on all the work previously carried out. A naive approach for the coverage path planning would have been to apply a sweeping over the area to cover. Even if it can be efficient in numerous cases, it will have disadvantages as a longer path size or a lower robustness to additional constraints. Indeed, sweeping based methods will provide workable results yet not optimized and rigid to constraints.
The proposed solution is a two steps optimization, by splitting the problem of coverage path planning into two sub-problems :

- The waypoints positioning
- The path planning passing by previously found waypoints

7.2.1/ Waypoints positioning

The first sub-problem consists of finding the best waypoints positions to cover the area. The best position for a set of waypoints can be led by several indicators and constraints depending on the application. In our case, the main interest is the area coverage rate. Based on it, a cost function has been developed to estimate the coverage rate and also integrate some predefined constraints.
In the state of the art, the camera positioning, which is strongly related to waypoints positioning, is mainly solved by applying different optimization algorithms. Among them, the evolutionary algorithm family is the most commonly used and the most promising, with notably the PSO (Particular Swarm Optimization) and the GA (Genetic Algorithms). The PSO is offering a fast solution which is flexible and can easily be tuned for different constraints. The GA is also providing fine results for our problems, yet has been underestimated for the optimization of waypoints positioning. For this reason, it has been extensively investigated during this thesis. The 2 algorithms have been compared under several conditions to conclude on the advantage of using the GA when dealing with wide and complex areas. On the other hand, for small areas, the PSO is faster and provides a result close the optimal solution. To conclude, the best solution is obtained with a third methods that combines the advantages of these two methods called GAPSO.

The GAPSO implies a longer computation time, but provide significant advantages such as better efficiency and higher flexibility to the area size and to the number of camera to optimize.

An innovative solution is introduced to improve optimization of the waypoints positioning problem thus allow the first sub-problems to obtain a more efficient solution.

7.2.2/ Path planning

The second sub-problem is to find a path passing by all the waypoints positioned using the GAPSO. The problem of estimating the shorter path passing by a set of given waypoints is closely related to a well known problem. The TSP (Traveller Sell-man Problem) has the same goal and numerous works have been done to find the best algorithms to solve it. The TSP is one of the NP-complete problem and cannot be completely solved. The best way to get an acceptable solution for the TSP is to use a GA for combinatory problem as advocated in the literature. The idea is to find the good order to pass by all the waypoints before to come back to the starting one, while minimizing the length path. In this thesis this two sub-problems have been addressed and an efficient answer has been proposed. The efficient answer for the path planning sub-problem is to apply the TSP formulation and solution with a genetic algorithms adapted to combinatorial problem.

7.2.3/ Solution and experiments

Finally, our main contribution is based on an original problem splitting to offer 2 independent optimization phases. Where some advanced algorithm has been applied to solve more efficiently the problem of waypoints (or cameras) positioning, an important work has been made on the algorithms development to optimize the waypoints positioning. This allows us to affirm that an appropriate GA is more efficient than the PSO for large spaces and for a large number of waypoints despite previous works in the literature. Moreover the GAPSO allows an even more robust solution to constraints and give more flexibility to the proposed solution.
The proposed solution can design an efficient path planning and, at the same time, a shorter and a more adaptable solution to the constraints. During the presented experiments, few constraints have been added and the proposed solution has shown its adaptability and efficiency. From the constraint studied, the size and the shape of the area have crucial importance, as the altitude boundary and his consequences on the images resolution. The advantage of the solution proposed is the adaptability and flexibility to new constraint. In fact the addition or removal of constraints do not much affect the obtained results during the experimentations.

7.3/ Future prospects

The obtained results are promising and can be a starting point for further research. The solution proposed must be adapted on UAVs for smart agriculture and precision agriculture. The advantage and flexibility provided by our algorithm can be useful for it. In this case the next step is to work on the image acquisition from RGB or multi-spectral images, which can precisely characterize the fields and segment of the zone depending the vegetation requirement.

Another prospective can be to use this work as preliminary work for a smart integrated tracking system. In fact the efficiency of the GAPSO for waypoint positioning allows an adequate coverage of an important area. This area can be covered by using several UAVs which split the area. Few UAVs are executing the path plan efficiently computed by our algorithms. The path plan can be split in sub-part for each UAV. Once a target is detected, one of the UAV is dedicated to the target tracking. Consequently, the others have to re-adapt the path to continue the area monitoring. In this prospective some unexplored constraints can be added to the solution proposed. The one concerning the creation of the path is probably the most interesting. Indeed, many constraints of trajectory can be added to obtain an optimized path for different kind of UAVs. This type of constraints must be added according to physical property of the UAVs used and deserve to be experimentally validated by fly. The experimental validation is also an important track of research.

The novel optimization algorithm for coverage path planning proposed by this thesis can be extended to various applications related to unmanned robots. The proposed solution, even if is not optimal, allows flexibility and robustness to additional constraints that are frequently met during real experiments.

Bibliography

[1] Jose Luis Alarcon Herrera and Xiang Chen. Online configuration of ptz camera networks. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–6. IEEE, 2012.

[2] Cristian Soto, Bi Song, and Amit K Roy-Chowdhury. Distributed multi-target tracking in a self-configuring camera network. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1486–1493. IEEE, 2009.

[3] Chong Ding, Bi Song, Akshay Morye, Jay A Farrell, and Amit K Roy-Chowdhury. Collaborative sensing in a distributed ptz camera network. *IEEE Transactions on Image Processing*, 21(7):3282–3295, 2012.

[4] Jian Zhao, Ruriko Yoshida, Sen-ching Samson Cheung, and David Haws. Approximate techniques in solving optimal camera placement problems. *International Journal of Distributed Sensor Networks*, 9(11):241913, 2013.

[5] Yi-Chun Xu, Bangjun Lei, and Emile A Hendriks. Camera network coverage improving by particle swarm optimization. *EURASIP Journal on Image and Video Processing*, 2011(1):458283, 2011.

[6] Jian Zhao, Sen-Ching Cheung, and Thinh Nguyen. Optimal camera network configurations for visual tagging. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):464–479, 2008.

[7] Yan Li, Hai Chen, Meng Joo Er, and Xinmin Wang. Coverage path planning for uavs based on enhanced exact cellular decomposition method. *Mechatronics*, 21(5):876–885, 2011.

[8] Christof Hoppe, Andreas Wendel, Stefanie Zollmann, Katrin Pirker, Arnold Irschara, Horst Bischof, and Stefan Kluckner. Photogrammetric camera network design for micro aerial vehicles. In *Computer vision winter workshop (CVWW)*, volume 8, pages 1–3, 2012.

[9] Stefanos Nikolaidis, Ryuichi Ueda, Akinobu Hayashi, and Tamio Arai. Optimal camera placement considering mobile robot trajectory. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1393–1396. IEEE, 2009.

[10] Robert Bodor, Paul Schrater, and Nikolaos Papanikolopoulos. Multi-camera positioning to optimize task observability. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 552–557. IEEE, 2005.

[11] Xiao-Shan Lu, Hai-Jun Huang, and Jiancheng Long. A bi-level programming model for network traffic surveillance of optimal camera location. In *Computational Sciences and Optimization (CSO), 2011 Fourth International Joint Conference on*, pages 1035–1039. IEEE, 2011.

[12] Robert Bodor, Andrew Drenner, Michael Janssen, Paul Schrater, and Nikolaos Papanikolopoulos. Mobile camera positioning to optimize the observability of human activity recognition tasks. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1564–1569. IEEE, 2005.

[13] Yibo Jiang, Jinwei Yang, Weijie Chen, and Wanliang Wang. A coverage enhancement method of directional sensor network based on genetic algorithm for occlusion-free surveillance. In *Computational Aspects of Social Networks (CASoN), 2010 International Conference on*, pages 311–314. IEEE, 2010.

[14] Eva Hörster and Rainer Lienhart. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120. ACM, 2006.

[15] Dimitrios Chrysostomou and Antonios Gasteratos. Optimum multi-camera arrangement using a bee colony algorithm. In *Imaging Systems and Techniques (IST), 2012 IEEE International Conference on*, pages 387–392. IEEE, 2012.

[16] Liang Liu, Xi Zhang, and Huadong Ma. Dynamic node collaboration for mobile target tracking in wireless camera sensor networks. In *INFOCOM 2009, IEEE*, pages 1188–1196. IEEE, 2009.

[17] Dalei Wu, Song Ci, Haiyan Luo, Yun Ye, and Haohong Wang. Video surveillance over wireless sensor and actuator networks using active cameras. *IEEE Transactions on Automatic Control*, 56(10):2467–2472, 2011.

[18] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for self-organization of a wireless sensor network. *IEEE personal communications*, 7(5):16–27, 2000.

[19] Liang Liu, Xi Zhang, and Huadong Ma. Optimal node selection for target localization in wireless camera sensor networks. *IEEE Transactions on Vehicular Technology*, 59(7):3562–3576, 2010.

[20] Krishna Konda Reddy and Nicola Conci. Camera positioning for global and local coverage optimization. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–6. IEEE, 2012.

[21] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks: self-configuring localization systems. In *International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK*, 2001.

[22] Carmelo Di Franco and Giorgio Buttazzo. Coverage path planning for uavs photogrammetry with energy and resolution constraints. *Journal of Intelligent & Robotic Systems*, 83(3-4):445–462, 2016.

[23] Wang Meiting, Tan Shili Ding Junjian, and Yan Liwen. Complete coverage path planning of wall-cleaning robot using visual sensor. In *Electronic Measurement and Instruments, 2007. ICEMI'07. 8th International Conference on*, pages 4–159. IEEE, 2007.

[24] Aaron Mavrinac and Xiang Chen. Modeling coverage in camera networks: A survey. *International journal of computer vision*, 101(1):205–226, 2013.

[25] Chang Wang, Fei Qi, and Guang-Ming Shi. Nodes placement for optimizing coverage of visual sensor networks. *Advances in Multimedia Information Processing-PCM 2009*, pages 1144–1149, 2009.

[26] Zhao Zhang, James Willson, Zaixin Lu, Weili Wu, Xuding Zhu, and Ding-Zhu Du. Approximating maximum lifetime k-coverage through minimizing weighted k-cover in homogeneous wireless sensor networks. *IEEE/ACM Transactions on Networking*, 24(6):3620–3633, 2016.

[27] Nabajyoti Medhi and Manjish Pal. Sixsoid: A new paradigm for k-coverage in 3d wireless sensor networks. *arXiv preprint arXiv:1401.0200*, 2013.

[28] Uğur Murat Erdem and Stan Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103(3):156–169, 2006.

[29] Yi-Ge Fu, Jie Zhou, and Lei Deng. Surveillance of a 2d plane area with 3d deployed cameras. *Sensors*, 14(2):1988–2011, 2014.

[30] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[31] Eli Packer. Computing multiple watchman routes. *Lecture Notes in Computer Science*, 5038:114–128, 2008.

[32] Joseph O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[33] Marcelo C Couto, Pedro J de Rezende, and Cid C de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.

[34] Shachar Fleishman, Daniel Cohen-Or, and Dani Lischinski. Automatic camera placement for image-based modeling. In *Computer Graphics Forum*, volume 19, pages 101–110. Wiley Online Library, 2000.

[35] Kenichi Yabuta and Hitoshi Kitazawa. Optimum camera placement considering camera specification for security monitoring. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 2114–2117. IEEE, 2008.

[36] Craig A Tovey. A simplified np-complete satisfiability problem. *Discrete applied mathematics*, 8(1):85–89, 1984.

[37] Mahdi Moeini, Alexander Kröller, and Christiane Schmidt. Une nouvelle approche pour la résolution du problème de la galerie d'art.

[38] Yuhao Wang, Ge Dang, Yang Si, and Huilin Zhou. An inversion propagation model using ga for coverage prediction of a single urban cell in wireless network. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 4894–4898. IEEE, 2008.

[39] Haluk Rahmi Topcuoglu, Murat Ermis, and Mesut Sifyan. Hybrid evolutionary algorithms for sensor placement on a 3d terrain. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 511–516. IEEE, 2009.

[40] Huan Ma, Meng Yang, Deying Li, Yi Hong, and Wenping Chen. Minimum camera barrier coverage in wireless camera sensor networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 217–225. IEEE, 2012.

[41] Raghavendra V Kulkarni and Ganesh Kumar Venayagamoorthy. Particle swarm optimization in wireless-sensor networks: A brief survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(2):262–267, 2011.

[42] Krishnendu Chakrabarty, S Sitharama Iyengar, Hairong Qi, and Eungchun Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE transactions on computers*, 51(12):1448–1453, 2002.

[43] Vahab Akbarzadeh, Christian Gagné, Marc Parizeau, Meysam Argany, and Mir Abolfazl Mostafavi. Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage. *IEEE Transactions on Instrumentation and Measurement*, 62(2):293–303, 2013.

[44] Meizhen Wang, Xuejun Liu, Yanan Zhang, and Ziran Wang. Camera coverage estimation based on multistage grid subdivision. *ISPRS International Journal of Geo-Information*, 6(4):110, 2017.

[45] Na Li and Jason R Marden. Designing games for distributed optimization. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):230–242, 2013.

[46] Bi Song, Cristian Soto, Amit K Roy-Chowdhury, and Jay A Farrell. Decentralized camera network control using game theory. In *Distributed Smart Cameras, 2008. ICDSC 2008. Second ACM/IEEE International Conference on*, pages 1–8. IEEE, 2008.

[47] Anton van den Hengel, Rhys Hill, Ben Ward, Alex Cichowski, Henry Detmold, Chris Madden, Anthony Dick, and John Bastian. Automatic camera placement for large scale surveillance networks. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–6. IEEE, 2009.

[48] Weicai Zhong, Jing Liu, Mingzhi Xue, and Licheng Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):1128–1141, 2004.

[49] Pu Zhou and Chengnian Long. Optimal coverage of camera networks using pso algorithm. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 4, pages 2084–2088. IEEE, 2011.

[50] KB Maji, Atreye Ghosh, R Kar, D Mandal, and SP Ghoshal. An evolutionary algorithm based approach for vlsi floor-planning. In *Science and Technology (TICST), 2015 International Conference on*, pages 248–253. IEEE, 2015.

[51] Xiao Fu, Ru-chuan WANG, Li-juan SUN, and WU Shuai. Research on the three-dimensional perception model and coverage-enhancing algorithm for wireless multimedia sensor networks. *The Journal of China Universities of Posts and Telecommunications*, 17:67–72, 2010.

[52] Gopalakrishnan Vijayan and R-S Tsay. A new method for floor planning using topological constraint reduction. *IEEE transactions on computer-aided design of integrated circuits and systems*, 10(12):1494–1501, 1991.

[53] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[54] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[55] João Valente, David Sanz, Jaime Del Cerro, Antonio Barrientos, and Miguel Ángel de Frutos. Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precision agriculture*, 14(1):115–132, 2013.

[56] David W Casbeer, Derek B Kingston, Randal W Beard, and Timothy W McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351–360, 2006.

[57] Chaomin Luo, Simon X Yang, Deborah A Stacey, and Jan C Jofriet. A solution to vicinity problem of obstacles in complete coverage path planning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 612–617. IEEE, 2002.

[58] Tae-Kyeong Lee, Sang-Hoon Baek, Se-Young Oh, and Young-Ho Choi. Complete coverage algorithm based on linked smooth spiral paths for mobile robots. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 609–614. IEEE, 2010.

[59] Simon X Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):718–724, 2004.

[60] Pablo J Zarco-Tejada, José AJ Berni, L Suárez, and E Fereres. A new era in remote sensing of crops with unmanned robots. *SPIE Newsroom*, pages 2–4, 2008.

[61] Haiyang Chao, Marc Baumann, Austin Jensen, YangQuan Chen, Yongcan Cao, Wei Ren, and Mac McKee. Band-reconfigurable multi-uav-based cooperative remote sensing for real-time water management and distributed irrigation control. *IFAC Proceedings Volumes*, 41(2):11744–11749, 2008.

[62] DS Long, SD DeGloria, and JM Galbraith. Use of the global positioning system in soil survey. *Journal of soil and water conservation*, 46(4):293–297, 1991.

[63] Antonio Barrientos, Julian Colorado, Jaime del Cerro, Alexander Martinez, Claudio Rossi, David Sanz, and João 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*, 28(5):667–689, 2011.

[64] Camille CD Lelong, Philippe Burger, Guillaume Jubelin, Bruno Roux, Sylvain Labbé, and Frédéric Baret. Assessment of unmanned aerial vehicles imagery for quantitative monitoring of wheat crop in small plots. *Sensors*, 8(5):3557–3585, 2008.

[65] DS Long, SD DeGloria, and JM Galbraith. Use of the global positioning system in soil survey. *Journal of soil and water conservation*, 46(4):293–297, 1991.

[66] James F Roberts, Jean-Christophe Zufferey, and Dario Floreano. Energy management for indoor hovering robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1242–1247. IEEE, 2008.

[67] Wei-pang Chin and Simeon Ntafos. Optimum watchman routes. *Information Processing Letters*, 28(1):39–44, 1988.

[68] Xuehou Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77(1):27–33, 2001.

[69] Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph SB Mitchell. Touring a sequence of polygons. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 473–482. ACM, 2003.

[70] Jan Faigl. Approximate solution of the multiple watchman routes problem with restricted visibility range. *IEEE transactions on neural networks*, 21(10):1668–1679, 2010.

[71] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[72] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1):113–126, 2001.

[73] Marina Torres, David A Pelta, José L Verdegay, and Juan C Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.

[74] Paulo A Jimenez, Bijan Shirinzadeh, Ann Nicholson, and Gursel Alici. Optimal area covering using genetic algorithms. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–5. IEEE, 2007.

[75] Young-Ho Choi, Tae-Kyeong Lee, Sang-Hoon Baek, and Se-Young Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5788–5793. IEEE, 2009.

[76] Ping-Min Hsu, Chun-Liang Lin, and Meng-Yao Yang. On the complete coverage path planning for mobile robots. *Journal of Intelligent and Robotic Systems*, 74(3-4):945–963, 2014.

[77] Vatana An and Zhihua Qu. A practical approach to coverage control for multiple mobile robots with a circular sensing range. In *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pages 112–117. IEEE, 2013.

[78] Heinz Mühlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Workshop on Parallel Processing: Logic, Organization, and Technology*, pages 398–406. Springer, 1989.

[79] Martin Serpell and James E Smith. Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation*, 18(3):491–514, 2010.

[80] Trevor Davies and Amor Jnifene. Multiple waypoint path planning for a mobile robot using genetic algorithms. In *Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on*, pages 21–26. IEEE, 2006.

[81] John Ware and Nicholas Roy. An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1507–1514. IEEE, 2016.

[82] Yu Liu, Xiaoyong Lin, and Shiqiang Zhu. Combined coverage path planning for autonomous cleaning robots in unstructured environments. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 8271–8276. IEEE, 2008.

[83] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[84] Hans J Bremermann. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

[85] Richard M Friedberg. A learning machine: Part i. *IBM Journal of Research and Development*, 2(1):2–13, 1958.

[86] George EP Box. Evolutionary operation: A method for increasing industrial productivity. *Applied statistics*, pages 81–101, 1957.

[87] John H Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314, 1962.

[88] Jarmo T Alander. An indexed bibliography of genetic algorithms papers of 1996 (in proceedings). Technical report, Report 94-1-96PROC, University of Vaasa, Department of Information Technology and Production Economics, 1995.(ftp://ftp. uwasa. fics/report94-1/ga96PROCbib. ps. Z) ga96PROCbib, 1994.

[89] Ingo Rechenberg. Cybernetic solution path of an experimental problem. 1965.

[90] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[91] Qingfu Zhang and Hui Li. Moea d a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[92] G Soremekun, Z Gürdal, RT Haftka, and LT Watson. Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers & structures*, 79(2):131–143, 2001.

[93] David E Goldberg. Genetic algorithms and walsh functions: Part i, a gentle introduction. *Complex systems*, 3(2):129–152, 1989.

[94] Simon Ronald. Robust encodings in genetic algorithms: A survey of encoding issues. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 43–48. IEEE, 1997.

[95] Godfrey A Walters and David K Smith. Evolutionary design algorithm for optimal layout of tree networks. *Engineering Optimization*, 24(4):261–281, 1995.

[96] Suhail Owais, Vaclav Snasel, Pavel Kromer, and Ajith Abraham. Survey: using genetic algorithm approach in intrusion detection systems techniques. In *Computer Information Systems and Industrial Management Applications, 2008. CISIM'08. 7th*, pages 300–307. IEEE, 2008.

[97] David Edward Goldberg. *Optimal initial population size for binary-coded genetic algorithms*. Clearinghouse for Genetic Algorithms, Department of Engineering Mechanics, University of Alabama, 1985.

[98] Yacine Morsly, Nabil Aouf, Mohand Said Djouadi, and Mark Richardson. Particle swarm optimization inspired probability algorithm for optimal camera network placement. *IEEE Sensors Journal*, 12(5):1402–1412, 2012.

[99] Alden H Wright et al. Genetic algorithms for real parameter optimization. *Foundations of genetic algorithms*, 1:205–218, 1991.

[100] Agoston E Eiben, James E Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[101] HAJ-RACHID Mais, Christelle Bloch, Wahiba Ramdane-Cherif, and Pascal Chatonnay. Différentes opérateurs évolutionnaires de permutation: sélections, croisements et mutations. *LABORATOIRE D'INFORMATIQUE DE L'UNIVERSITE DE FRANCHE-COMTE*, 2010.

[102] Olivier François and Christian Lavergne. Design of evolutionary algorithms-a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, 2001.

[103] Jaroslaw Arabas, Zbigniew Michalewicz, and Jan Mulawka. Gavaps-a genetic algorithm with varying population size. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 73–78. IEEE, 1994.

[104] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[105] K Matsui. New selection method to improve the population diversity in genetic algorithms. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 1, pages 625–630. IEEE, 1999.

[106] John J Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128, 1986.

[107] XH Shi, YC Liang, HP Lee, C Lu, and LM Wang. An improved ga and a novel pso-ga-based hybrid algorithm. *Information Processing Letters*, 93(5):255–261, 2005.

[108] Raphaël Cerf. An asymptotic theory for genetic algorithms. In *European Conference on Artificial Evolution*, pages 35–53. Springer, 1995.

[109] Hans-Paul Schwefel. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. *Annals of Operations Research*, 1(2):165–167, 1984.

[110] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International Conference on Parallel Problem Solving From Nature*, pages 849–858. Springer, 2000.

[111] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[112] Dirk Thierens and David Goldberg. Elitist recombination: An integrated selection recombination ga. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 508–512. IEEE, 1994.

[113] Mandavilli Srinivas and Lalit M Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667, 1994.

[114] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. *computer*, 27(6):17–26, 1994.

[115] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[116] Kuk-Hyun Han and Jong-Hwan Kim. Genetic quantum algorithm and its application to combinatorial optimization problem. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 2, pages 1354–1360. IEEE, 2000.

[117] Kuk-Hyun Han and Jong-Hwan Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE transactions on evolutionary computation*, 6(6):580–593, 2002.

[118] Rahila H Sheikh, Mukesh M Raghuwanshi, and Anil N Jaiswal. Genetic algorithm based clustering: a survey. In *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*, pages 314–319. IEEE, 2008.

[119] Yuan-Kai Wang and Kuo-Chin Fan. Applying genetic algorithms on pattern recognition: An analysis and survey. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 740–744. IEEE, 1996.

[120] Vali Nazarzehi and Andrey V Savkin. Distributed self-deployment of mobile wireless 3d robotic sensor networks for complete sensing coverage and forming specific shapes. *Robotica*, 36(1):1–18, 2018.

[121] Wing Ning. Strongly np-hard discrete gate-sizing problems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(8):1045–1051, 1994.

[122] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[123] Wen Fung Leong and Gary G Yen. Pso based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B Cybernetics*, 38(5):1270–1293, 2008.

[124] Daniel W Boeringer and Douglas H Werner. Particle swarm optimization versus genetic algorithms for phased array synthesis. *IEEE Transactions on antennas and propagation*, 52(3):771–779, 2004.

[125] K Premalatha and AM Natarajan. Hybrid pso and ga for global maximization. *Int. J. Open Problems Compt. Math*, 2(4):597–608, 2009.

[126] SG Ponnambalam, H Jagannathan, M Kataria, and A Gadicherla. A tsp-ga multi-objective algorithm for flow-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 23(11-12):909–915, 2004.

[127] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1134–1139. International Association of Engineers Hong Kong, 2011.

[128] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

List of Figures

List of Tables

I

Appendices

A

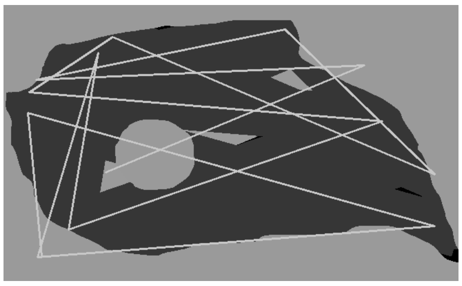
Appendices

A.1/ Advantage and disadvantage of sampling frequency

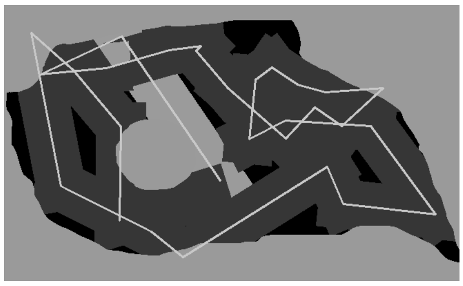
The impact of the sampling frequency of the grid is of great importance to the computation of the cost function and consequently on the optimization process. A sum-up table for the impact of the low and high sampling frequencies for the grid map is proposed in A.1.

	Advantage	**Disadvantage**
High sampling frequency	Best estimation of the area to cover	Time consuming
	Give more precision on the cameras poses	
Low sampling frequency	Faster computation	Bad coverage estimation

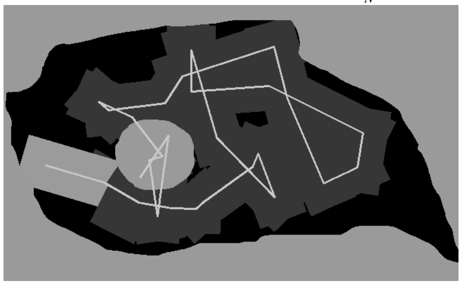
Table A.1: Sum-up of the low and high sampling frequency advantages or disadvantages.



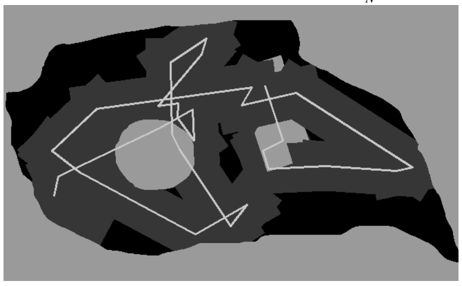
(a) Distance 5708px for 15 Waypoints with 96.22% of coverage. Optimized with GAPSO. $\sum_{i=1}^{n} Pc_i + \frac{400\times\sum_{i=1}^{n} Pc_i}{(\frac{Distance}{N})\times 5}/400.$



(b) Distance 2955px for 30 Waypoints with 76.15% of coverage. Optimized with GAPSO. $\sum_{i=1}^{n} Pc_i + \frac{400\times\sum_{i=1}^{n} Pc_i}{(\frac{Distance}{N})\times 5}$



(c) Distance 2074px for 30 Waypoints with 73.08% of coverage. Optimized with GAPSO. $\sum_{i=1}^{n} Pc_i + \frac{400\times\sum_{i=1}^{n} Pc_i}{(\frac{Distance}{N})\times 3}$



(d) Distance 2233px for 30 Waypoints with 72.77% of coverage. Optimized with GAPSO. $\sum_{i=1}^{n} Pc_i + \frac{400\times\sum_{i=1}^{n} Pc_i}{(\frac{Distance}{N})\times 0.01} \times 100$

Figure A.1: Path planning for maximizing the covered area. The area is covered with 15 and 30 waypoints and several cost functions have been tested

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the `tex-upmethodology` package suite — http://www.arakhne.org/tex-upmethodology/

Abstract:

The goal of this paper is to optimize the coverage of a vast and complex area such that its mosaic image can be created. To find the best waypoints, two methods have been investigated: Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Our investigation proved that GA is a better method due to its performance and adaptability. After having performed experiments to compare the algorithms, a hybridization of GA and PSO is investigated. The proposed method can be applied on large areas with irregular shapes, such as agricultural fields, and it provides a minimized number of waypoints that must be flown over by the Unmanned Aerial Vehicle (UAV). The experiments were made to simulate the flight of the UAV in an indoor environment, and the images generated during the simulated flight have been used to show the final mosaic. The proposed method is also applied in the vast outdoor area using satellite images to visualize the final result of the coverage path planning. The experiments validate the efficiency of the proposed method for finding the number and the poses of the waypoints. The solution proposed to approach the problem of coverage path planning is rather different than the state of the art by dividing the Coverage Path Planning on independent sub-problems to optimize and then using GA and later on GAPSO.

Keywords: optimization, UAV, evolutionary algorithm, PSO, path planning, coverage