HORIZON 2020

D3.5 Dynamic and Adaptive Swarm Optimization V1

Augmented Reality Enriched Situation awareness for Border security
ARESIBO – GA 833805

Deliverable Information

Deliverable Number: D3.5
Date of Issue: 29/09/2020
Document Reference: N/A
Version Number: 3.1

Work Package: #3

Nature of Deliverable: Report

Dissemination Level of Deliverable: Public

Author(s): Savvas Apostolidis, Dimitrios Koutras, Georgios Orfanidis, Panagiotis Michailidis, Konstantinos Ioannidis, Athanasios Kapoutsis, Stefanos Vrochidis, Ioannis Kompatsiaris and Elias Kosmatopoulos

Keywords: swarm, control, path-planning, coverage, mission, coordination, object detection, dynamic data driven assimilation, DDDAs, active sensing

Abstract: This report summarizes the first release of the functionalities of T3.3 – Collective intelligence for swarming robots and optimised human-robot collaboration and T3.6 – Sensing optimisation. Overall, this document describes the algorithms and methodologies that are developed and used in the context of the ARESIBO project for the high-level swarm control, the missions' management, the visual object detection and localization, the Dynamic Data Driven Assimilation (DDDAs) and the active sensing.

Document History

Date	Version	Remarks
24.05.2020	0.1	**Initiation of deliverable**
25.06.2020	1.0	**All content added**
30.06.2020	2.0	**Final issue of deliverable**
27.07.2020	2.5	**Updates on visual detection service**
28.07.2020	2.9	**Revision of content and minor changes**
28.08.2020	3.0	**Final version**
29.09.2020	3.1	**Corrections performed after internal review**

Document Authors

Entity	Contributors
1	**Savvas Apostolidis**
2	**Dimitrios Koutras**
3	**Georgios Orfanidis**
4	**Panagiotis Michailidis**
5	**Konstantinos Ioannidis**
6	**Athanasios Kapoutsis**
8	**Stefanos Vrochidis**
9	**Ioannis Kompatsiaris**
10	**Elias Kosmatopoulos**

Table of Contents

Acronym	Meaning
AP	Average Precision
CNN	Convolutional Neural Network
COCO	Common Objects in COntext
CPP	Coverage Path Planning
DARP	Divide Areas algorithm for optimal multi-Robot coverage Path planning
DoF	Degrees of Freedom
fps	frames per second
HITL	Human In The Loop
mAP	mean Average Precision
MST	Minimum Spanning Tree
NED	North East Down coordinate system
RC	Resource Controller
RCNN	Region proposing Convolutional Neural Network
ROI	Region of Interest
RPN	Region Proposal Network
RT-SAM	Real-Time Situational Awareness Maximization
SSD	Single Shot Detector
STC	Spanning Tree Coverage
UxV	Any type of unmanned vehicle
WGS84	World Geodetic System 84 coordinate system
YOLO	You Only Look Once

1 Executive summary

This report presents the tools that are developed for the ARESIBO project in order to provide high-level guidance for the multiple vehicles participating in the missions and the methodologies used for the visual object detection. The tools for the multiple vehicles' guidance are developed in the context of two different tasks. Specifically, for T3.3 - Collective intelligence for swarming robots and optimised human-robot collaboration, is developed the module Resource Controller, a module responsible to translate the high-level objectives defined for the missions into low-level commands and actions for all the involved assets and vehicles. This module takes over a double role, offering multiple path planning solutions for the vehicles, depending on the user-defined objectives, as well as acting as a coordinator for the missions, managing different aspects and issues during their execution. Additionally, for T3.6 - Sensing optimisation that deals with the Dynamic Data Driven Assimilation (DDDAs) and the active sensing, is developed the Real-Time Situational Awareness Maximization (RT-SAM) module. RT-SAM is a tool for real-time, vision-based path planning, that continuously provides the best monitoring positions for a swarm of UxVs, with the objective of maximizing both the number and accuracy of the detected objects of interest in a region, ensuring simultaneously that each of the UxVs deals with unique information in the region of interest (ROI), by punishing overlapping detections. RT-SAM is developed with a modular design that supports the use of any object detection solution. In the context of T3.6 a state-of-the-art object detection algorithm is selected and customized to deal with the specific needs and challenges that will be faced in the project. The modules developed for tasks 3.3 and 3.6 are extensively described in the following sections. It is worth noting that the object detection solution will be used for both the RT-SAM module and as a standalone module for the needs of the project. At the time that this deliverable is written, the development of all modules is still in process. This means that some of the features of the final versions have not been developed yet and some of the implementation details are not included in this version of the deliverable. The final version of the modules, along with all the details for the integration with the rest of the project are expected to be included in the second version of the deliverable (D3.6 – Dynamic and Adaptive Swarm Optimization V2).

2 Resource Controller

2.1 Introduction

The Resource Controller (RC) module is developed in the context of T3.3 - Collective intelligence for swarming robots and optimised human-robot collaboration. The objective of this module is to provide a tool that will create and manage missions with multiple, heterogeneous assets, demanding minimum cognitive load from the operator and providing autonomy, safety and increased operational efficiency. RC module receives the high-level operator defined objectives and translates them to low-level commands and actions for all the involved assets and modules, participating in a mission. After the definition of a new mission through the ARESIBO Mission Editor tool, the RC receives and handles the defined, high-level information, in order to coordinate all the participating assets and manage the execution of the mission. RC module's fundamental functionality is to provide the UxVs with a plan to follow for every mission. Moreover, it is likely that it will act as a coordinator for the tasks that will be performed from other modules.

Regarding the path planning functionality, RC is going to support the following modes:

- Follow paths
- Coverage Path Planning
- Persistent coverage

The aforementioned modes will be explained in the following sub-sections.

2.2 Communication and message exchange

Since the integration of the project has not started at the time that this deliverable is written, the exact pipeline of data and exchange of messages for the initiation and the management of a mission has not been finalized yet. However, below follows a list of the expected inputs/outputs of the module:

Input:

- A mission description from "Task 4.2 DSL-based specification of autonomous robotic missions"

Output:

- A message containing information to visualize missions
- A mission plan for the UxVs and participating assets in a mission to follow
- Messages for the activation/triggering of different modules participating in a mission

2.3 Follow paths mode

This is the simplest of the modes that RC supports for path-planning. In this mode the operator of the platform defines a set of waypoints (strictly defined path) for one or more vehicles to follow. RC in this case does not need to calculate any paths, but has to just forward the user-defined ones, for visualization and execution. A defined mission is likely to contain paths for multiple vehicles or groups of vehicles simultaneously. Since RC in the follow paths mode does not need to compute any paths for the mission, its role is restricted to the management and coordination of them.

2.4 Coverage Path Planning

When the description of a mission contains as objective the coverage of a Region of Interest (ROI), RC is responsible to calculate paths for one or more vehicles, inside a polygon ROI, in order to cooperatively, completely cover this ROI.

RC expects to receive the following information:

- WGS84 coordinates of the polygon ROI,
- number and names of the vehicles participating
- scanning density in meters (will be explained later)
- initial positions of the involved assets
- percentages of the complete ROI that should be assigned to each of the vehicles

The Coverage Path Planning (CPP) problem deals with the generation of paths inside a given *ROI,* for one, or more vehicles, having in mind *specific coverage capabilities of the sensors* used, with the objective to *completely cover the ROI*, taking into account *constraints that may exist for the vehicles, or the ROI itself*. Most of CPP methods, try to be optimal for specific kind of missions, by trying to minimize turns, operational time, etc.

There are many research works available in the field of CPP. The state of the art on CPP methods is summarized in [1] and [2]. CPP methods can be divided in categories, based on different aspects that are taken into account when trying to solve the problem. Some of these categories are presented below:

- cellular decomposition/grid-based methods (discretization method for the ROI)
- single/multi-robot
- on-line/off-line (depending on the capability to alter missions in real-time)
- energy-aware or not (depending on the energy-efficiency of the generated paths)
- categories regarding the patterns of (lawnmower patterns, spiral patterns, Spanning Tree Coverage patterns (STC) [3]).

Having extensively studied the most important works in CPP so far, for the project was developed a novel one, integrating all these features that would make it appropriate for this specific use. In the following subsections, the CPP method developed to calculate the paths will be extensively described. This method is based on a previous work from ConvCAO – CERTH's lab, the DARP algorithm [4] that was developed to divide areas for multi-robot CPP, however, a set of optimizations and innovative ideas were used in order to be optimal for real-world use. The provided solution incorporates energy and resource efficient features, supports the definition of obstacles inside the ROI and unlike other methods is able to support both convex and concave polygons, making it ideal for the scenarios that will be faced in the ARESIBO project.

2.4.1 ROI Representation on Grid & Path Planning

The data needed as input for the CPP method to work are:

- The defined *ROI*, along with potential obstacles, formatted in the WGS84 coordinate system,
- A *scanning density in meters*, representing the distance between two sequential designed trajectories,
- the *number of UxVs* for themission,
- the *initial positions of the UxVs*.

The output produced of the CPP method is a set of paths, one for each UxV, so as to completely cover the ROI.

1) Field Representation on Grid

The CPP method developed for the project is a grid-based method. This means that a physical ROI, selected on a map, has to be represented with the use of a grid, in order to be used as input for the CPP method.

i) Transformation of Coordinates

As a first step, all of the coordinates that will be used in the method (ROI, obstacles, initial positions) have to be transformed from the WGS84 system[1], to a local NED system[2], with a common reference point. NED coordinates are used, because of their convenience they offer in the node placement and optimization procedures that take place. The paths are also calculated in NED coordinates and are transformed later, back to the WGS84 coordinate system.

ii) Nodes Placement

The next step is to represent the polygon ROI with the use of a grid. The grid size should be analogous to the scanning density that user has selected. The centres of the grid's cells are used as nodes for the construction of MSTs, a step that is critical for the generation of the paths.

iii) Grid Representation

Once the grid is initiated, each node obtains a state to describe it. The possible states are:

- *Obstacle*, used to represent areas that are outside of polygon, or inside obstacles.
- *Free Space*, for nodes that will be available for the path generation.
- *UxV*, used to represent the initial UxVs' position.

2) Node Placement Optimization

One of the very significant limitations that grid-based methods face when they are applied in complex-shaped ROIs, is that the representation of the ROI ends up being a lot different from the actual region, resulting in incomplete coverage.

In this work, this issue is faced by the implementation of an optimization intending to provide optimal node placement for complete coverage. The key idea is that by *transforming* the polygon ROI over a grid that respects the selected mission constants, there are states that provide better coverage than others. The optimization procedure implemented has a significant effect on both the provided coverage and the qualitative features of the generated paths.

3) Task Allocation & Path Calculation

Having an optimal representation of the ROI on grid, DARP algorithm takes over to divide the complete region to sub-regions, exclusive for each UxV, and provides independent paths for all of them utilizing STC. The generated paths have energy efficient characteristics, as they respect UxVs' initial positions, avoiding redundant movements that do not contribute to the coverage process and reduce the number of unnecessary turns.

i) DARP - Area Division

DARP algorithm [4] deals with the path planning problem of a team of mobile robots, in order to cover an area of interest, with prior-defined obstacles. This technique transforms the original integer programming problem into several single-robot problems, the solutions of which constitute the optimal mCPP solution, alleviating the original mCPP explosive combinatorial complexity. n the heart of the proposed approach lies the DARP algorithm, which divides the terrain into a number of unique areas, each corresponding to a specific robot, so as to guarantee complete coverage, non-backtracking solution, minimum coverage path, while at the same time does not need any preparatory stage.

DARP’s initial implementation provides a fair area division of the region, meaning that all vehicles are assigned with the exact same percentage of the complete ROI. In the context

[1] https://en.wikipedia.org/wiki/World_Geodetic_System
[2] https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates

of this work, an extension of the algorithm is implemented, in order to also allow a proportional area division, based on the different energy and operational capabilities of the vehicles.

The following figures show the allocation of an area to five robots over the execution time of DARP, from the beginning, till the point that the algorithm has converged to a solution. The black dots in these figures represent the robots' initial positions. Different colours are used to visualize all the exclusive sub-regions, that will be assigned to different robots. The colour for every sub-region remains the same at all time-steps of the execution of DARP. It should be noted that the timesteps cannot be directly matched with the execution time of the algorithm, which depends on the hardware used. In any case, area allocation must be performed in a way that every exclusive sub-region includes the initial position of the robot that will be responsible for it, so as to avoid redundant movements of the robot that do not contribute to the coverage procedure. More details about the implementation of DARP algorithm, along with a complexity analysis, can be found in [4].

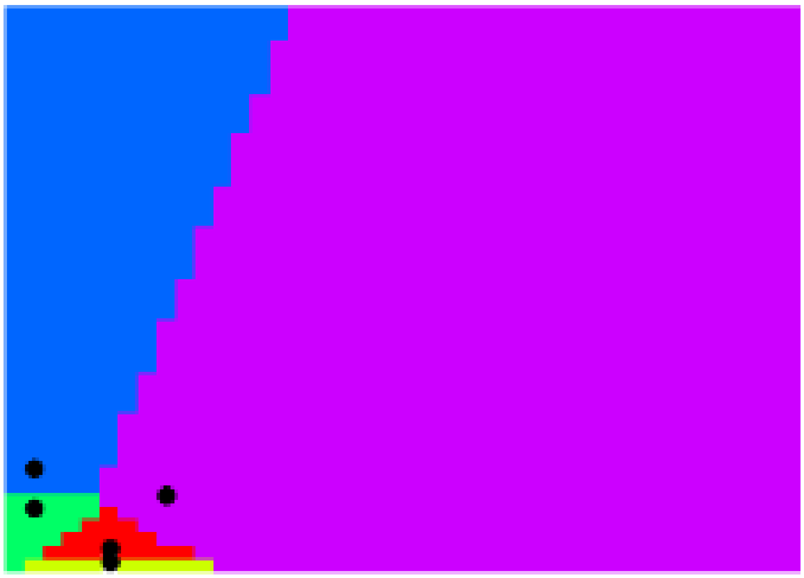


Figure 1: DARP – Progress 0%

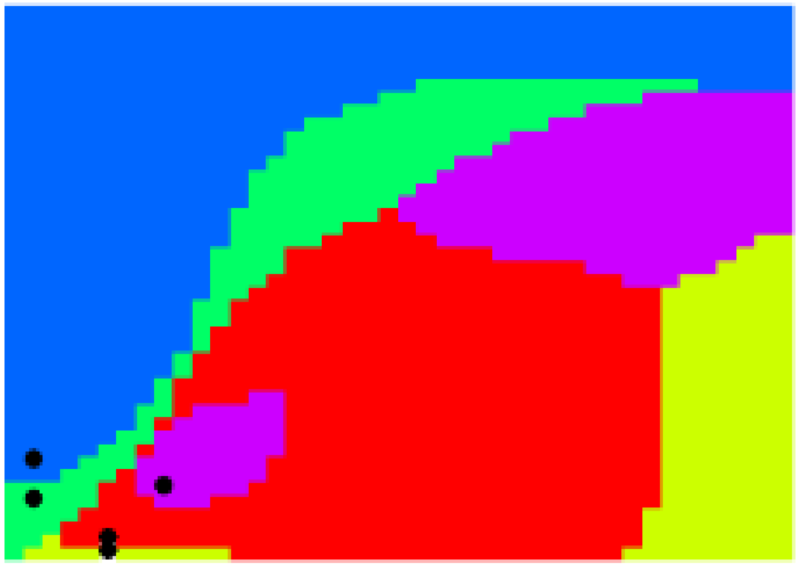


Figure 2: DARP – Progress 20%

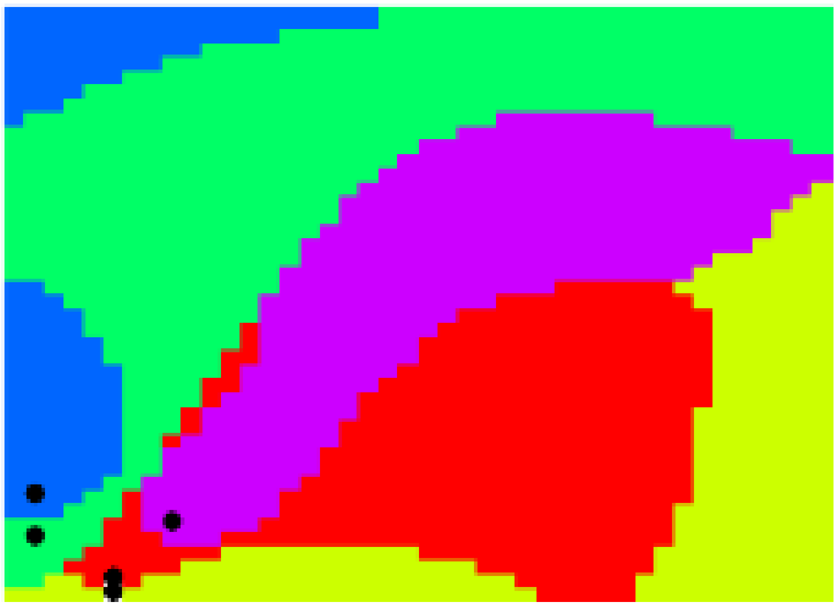


Figure 3: DARP – Progress 40%

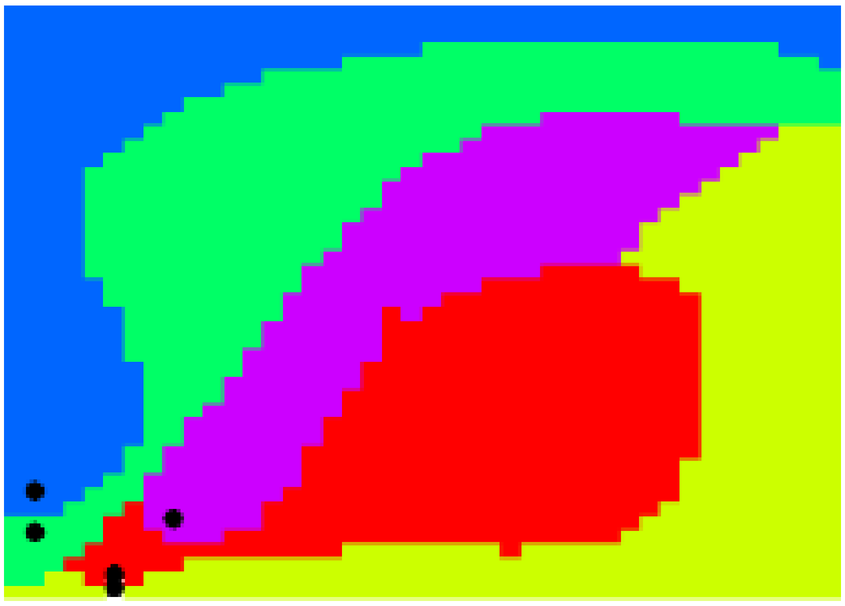


Figure 4: DARP – Progress 60%

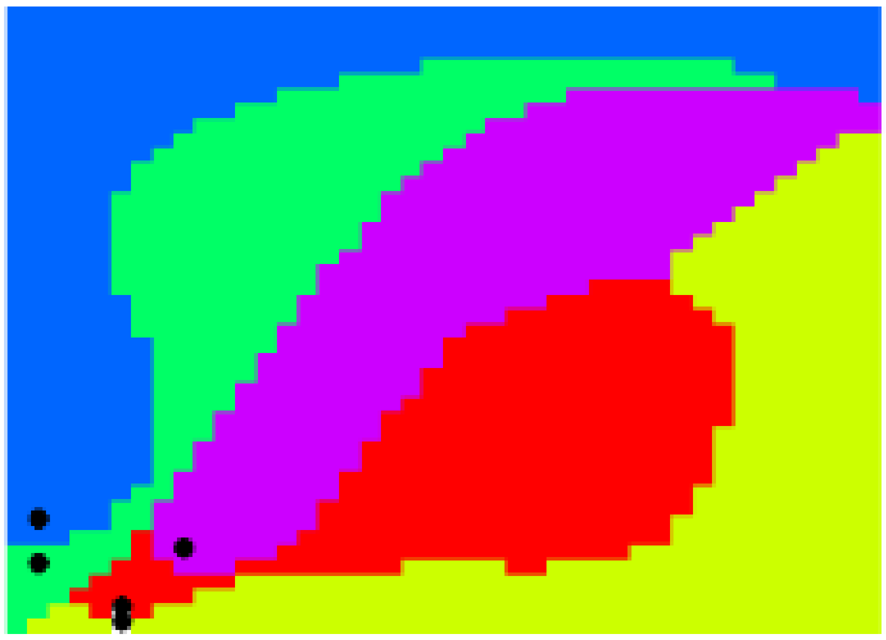


Figure 5: DARP – Progress 80%

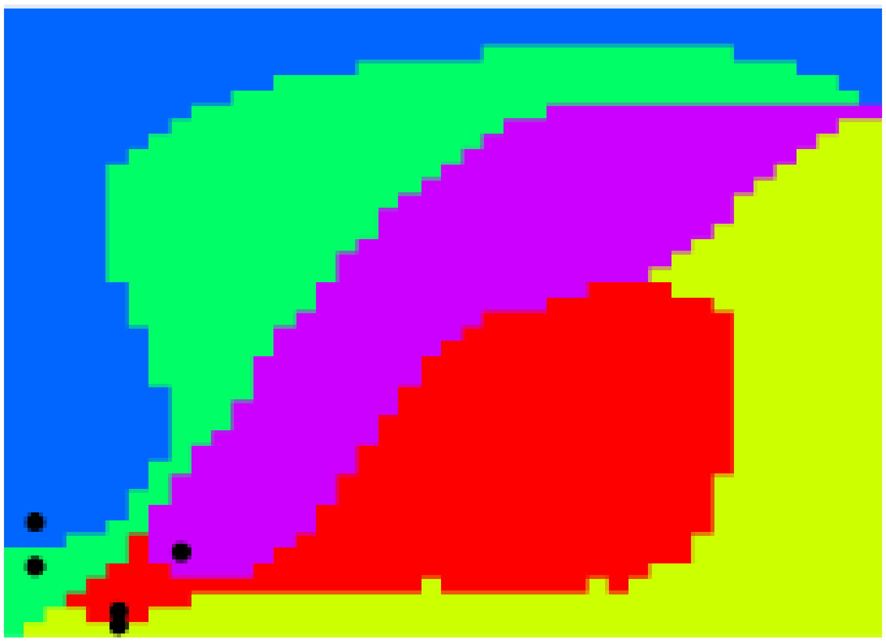


Figure 6: DARP – Progress 100%

ii) Reduced Turns MSTs & Path Generation

As mentioned above, from the time that each robot gets assigned with an exclusive part of the overall ROI, a single-robot STC problem is solved for all of them. For the path generation procedure, a MST is constructed in each sub-region and a circumnavigating path is generated around it afterwards. As mentioned above, the spanning-tree nature of the algorithm is likely to create complex shaped paths with a lot of unnecessary turns. In order to alleviate this issue in this work, during the MSTs generation are taken actions in order to significantly reduce the number of turns for each vehicle. In the following figure is presented an example for turn-reduced and non-turn-reduced path in the same polygon ROI.

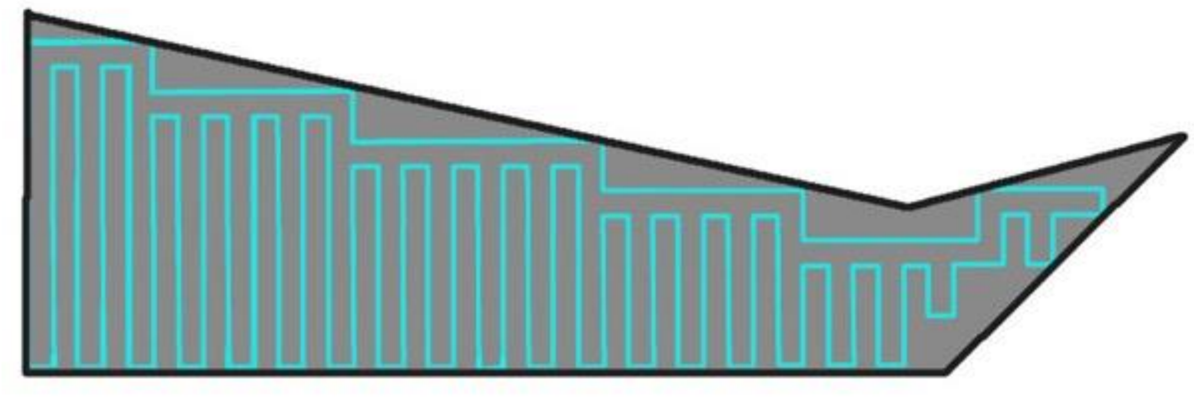


Figure 7: Without turn reduction: 94 turns

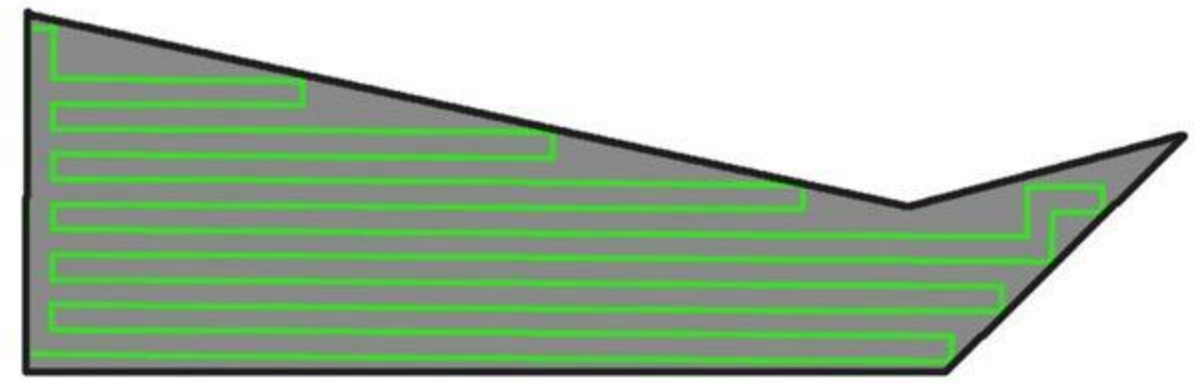


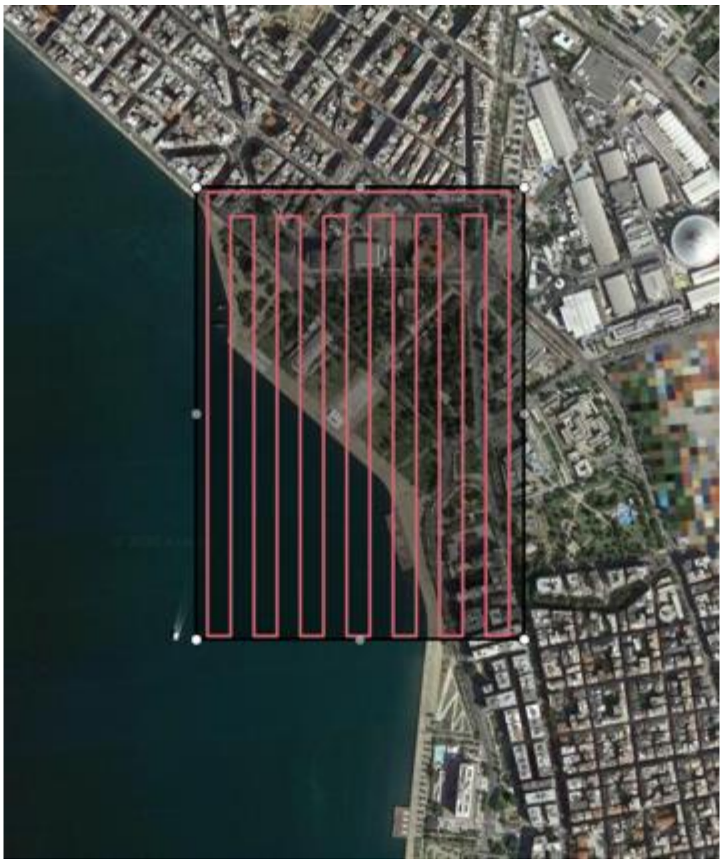
Figure 8: With turn reduction: 30 turns

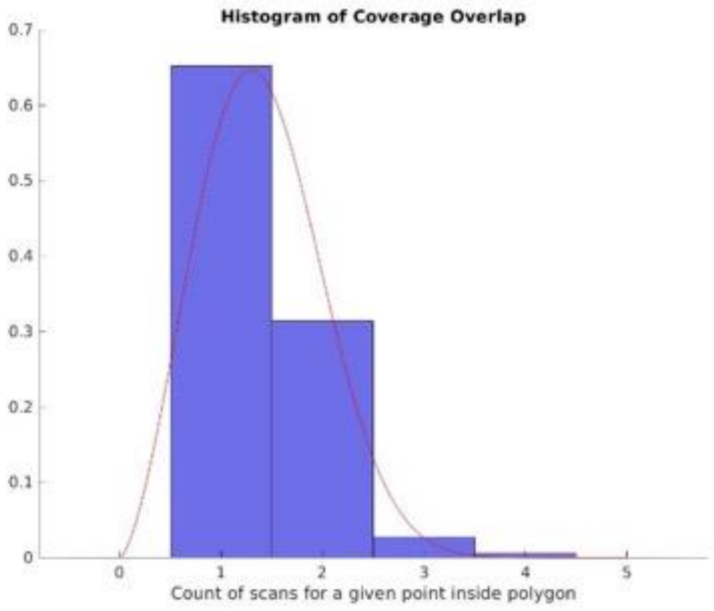
2.4.2 Simulated Evaluation

The evaluation of the CPP method will be performed through a series of examples. It should be noted that the efficiency of the generated paths regarding the coverage capability, is independent of the number of UxVs participating. So, a multi-robot coverage example is included just to demonstrate the capability. For all the examples is included an image of the generated paths, a heatmap of coverage showing the percentage of the area that was covered and how many times and a histogram of overlapping coverage. The last figure is helpful to understand if operational resources were wasted in order to provide complete coverage of the region. For all the experiments was selected an overlap percentage of 30-50% for the coverage between two sequential scans, which is usually required in order to have effective operation of the modules utilizing computer vision. Moreover, for all the presented examples is included the number of waypoints, the value of the optimization index J, the actual percentage of coverage and the actual percentage of overlapped coverage. The ROIs presented were selected randomly and were designed for execution with small, coptered, aerial vehicles. However, the mCPP algorithm can provide paths for any type of unmanned vehicles, taking into account the motional constraints and coverage capabilities of them.

1. Rectangle ROI

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
30	0.8982	99.998976 %	34.859423 %




Histogram of Coverage Overlap
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
Count of scans for a given point inside polygon

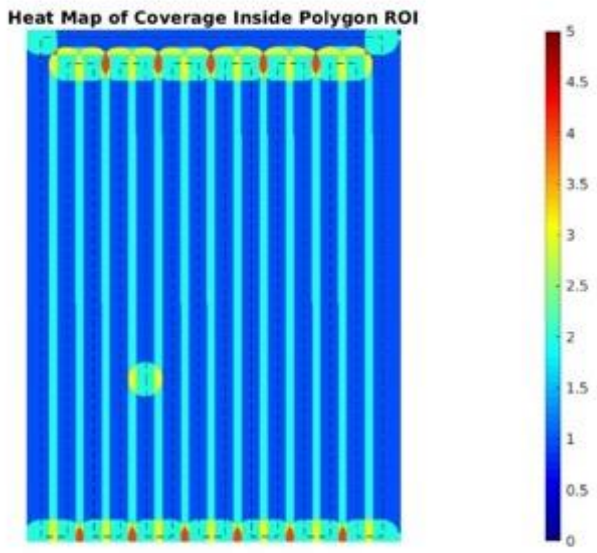


**Figure 9: Rectangle ROI
Paths & evaluation**

2. Rectangle ROI with Obstacle

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
45	0.810835	99.980993 %	84.838557 %

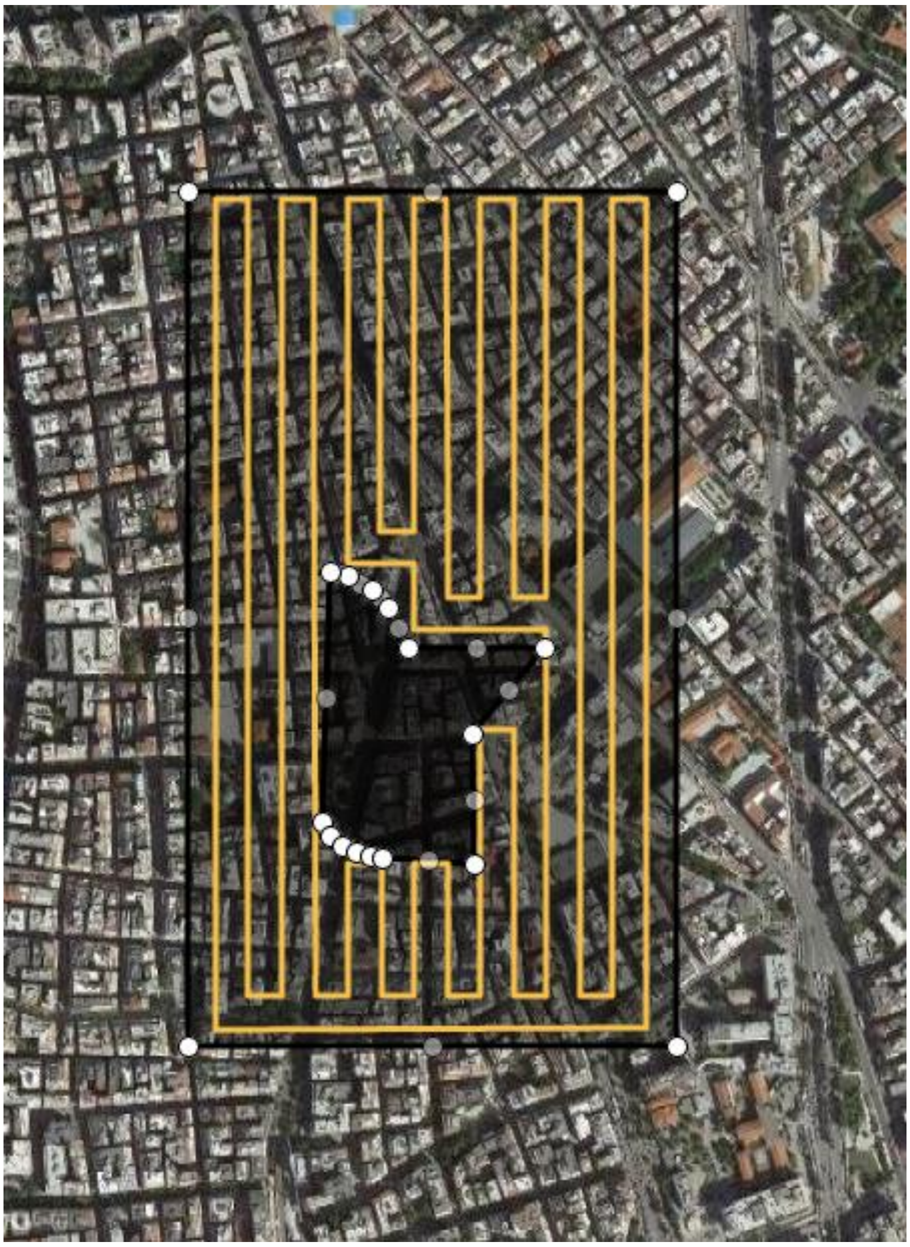
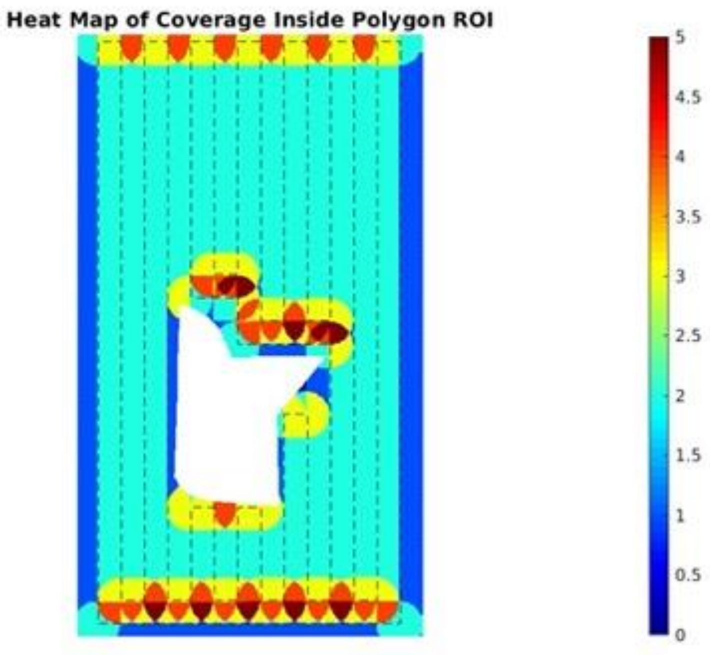


Figure 10: Rectangle ROI with Obstacle



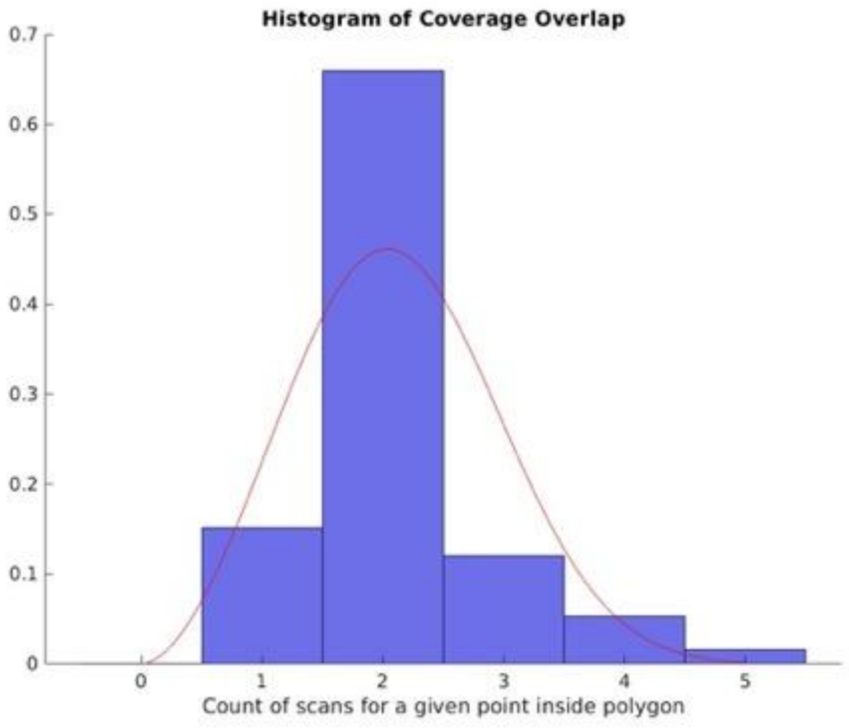
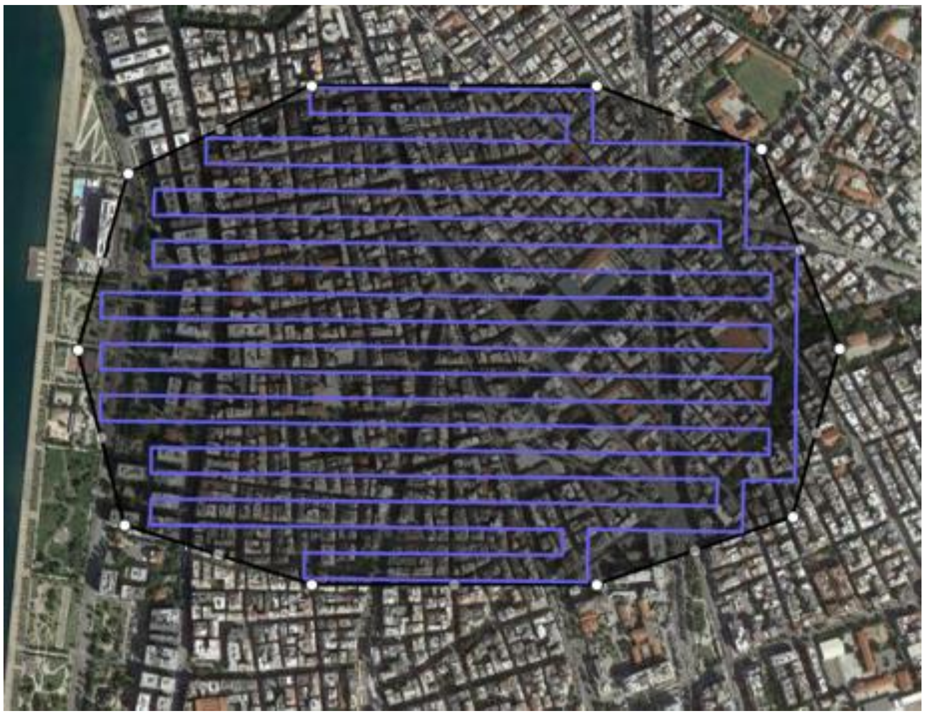
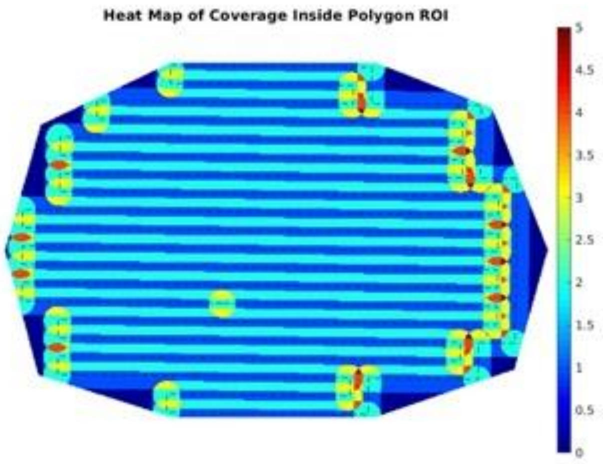


Figure 11: Evaluation of Rectangle ROI with Obstacle

3. Convex Polygon ROI

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
59	0.770574	93.676319 %	51.022816 %





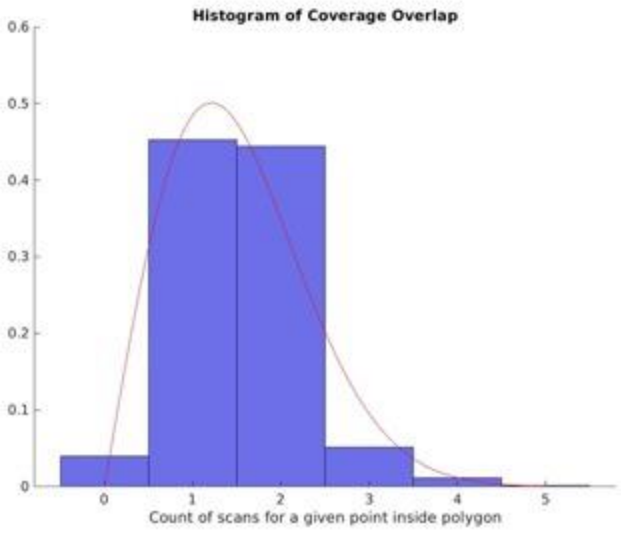
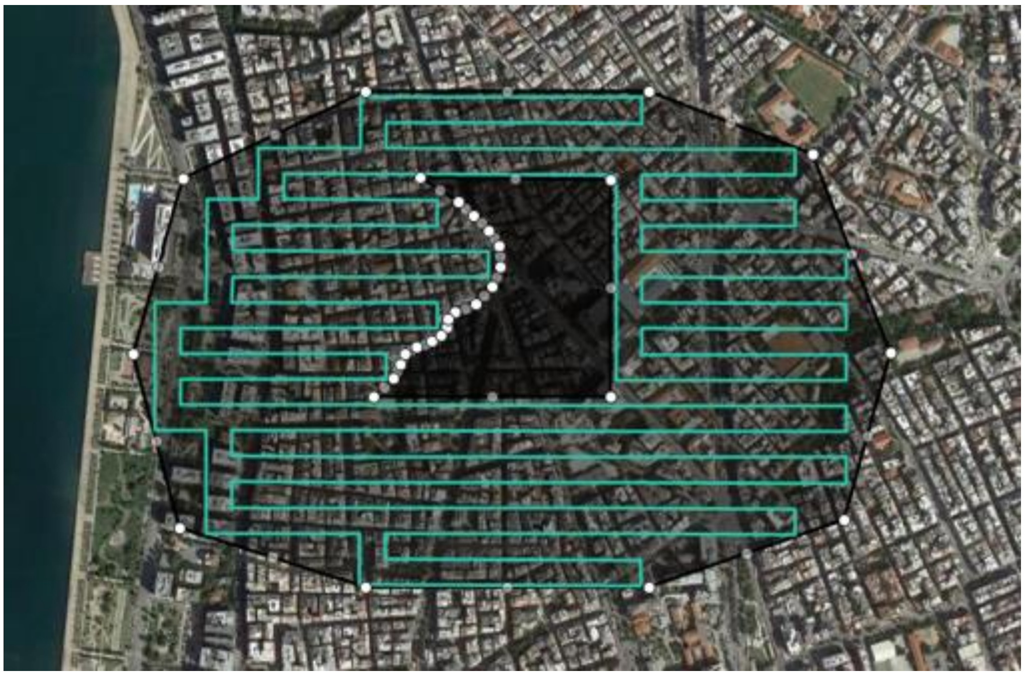
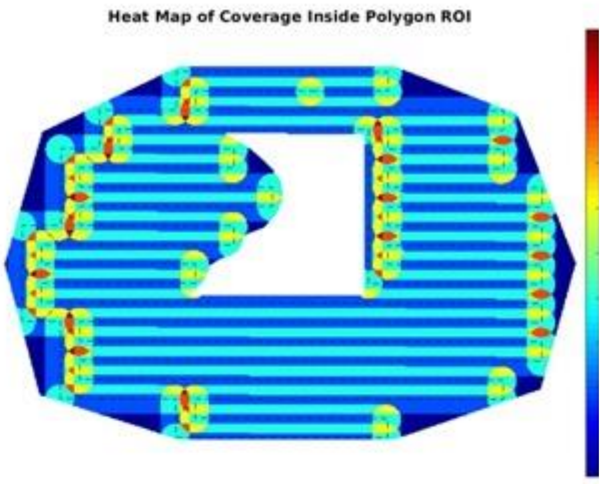


Figure 12: Convex Polygon ROI Paths & Evaluation

4. Convex Polygon ROI with Obstacle

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
70	0.775232	95.174975	52.911855 %





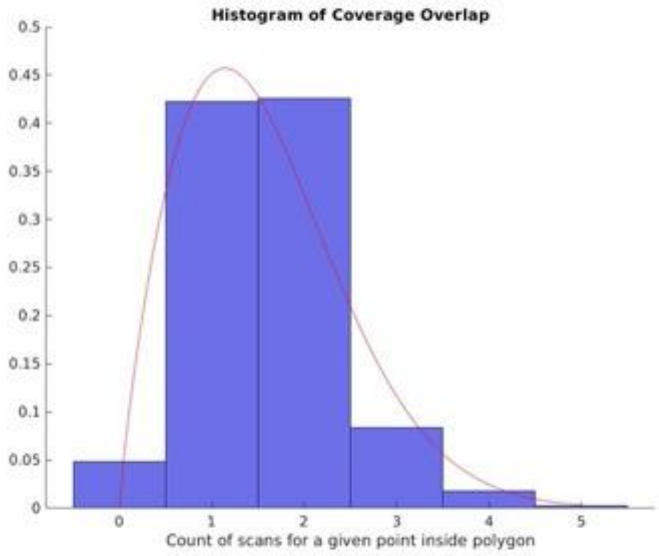


Figure 13: Convex Polygon ROI with obstacle - Path & Evaluation

5. Concave Polygon ROI

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
59	0.770574	93.676319 %	51.022816 %

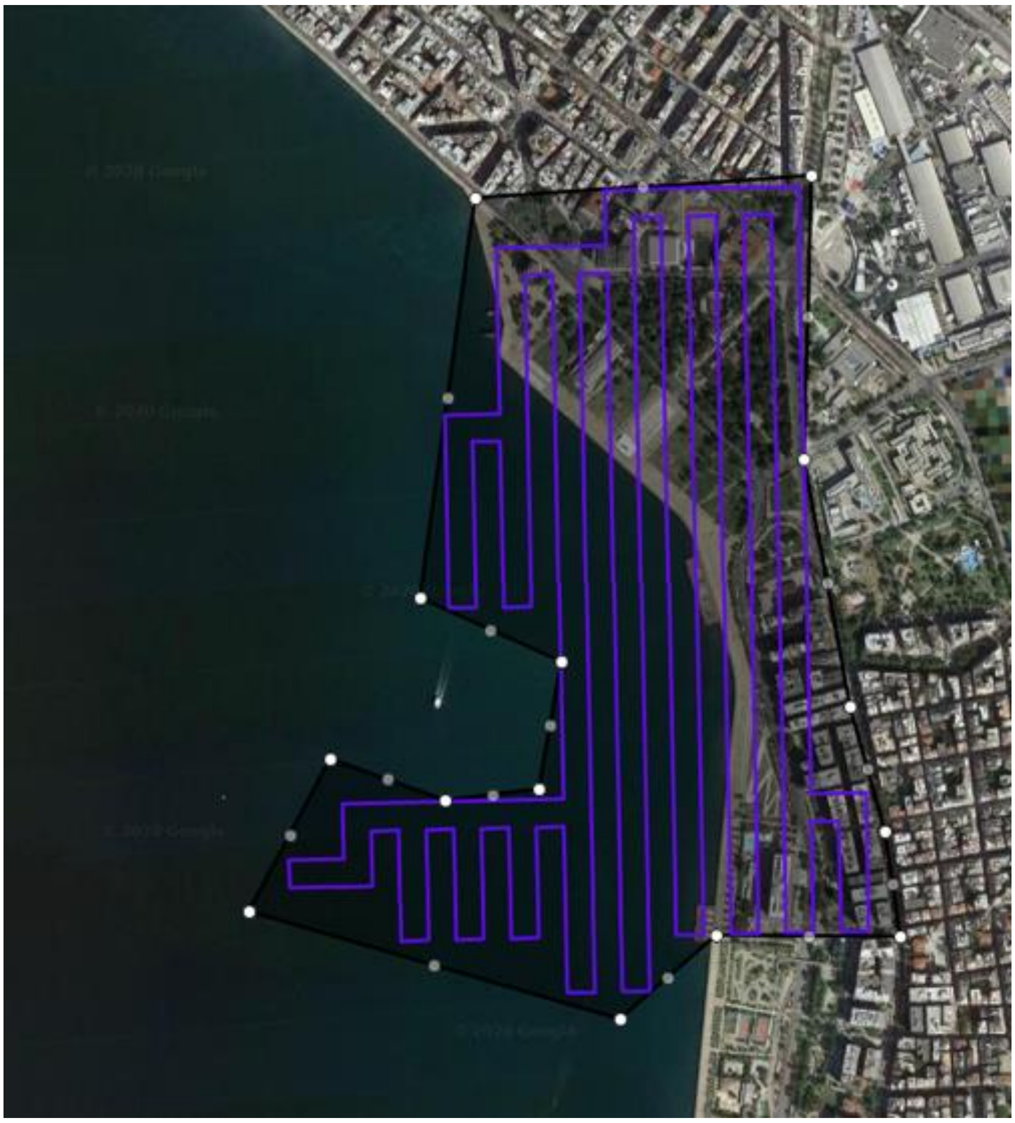
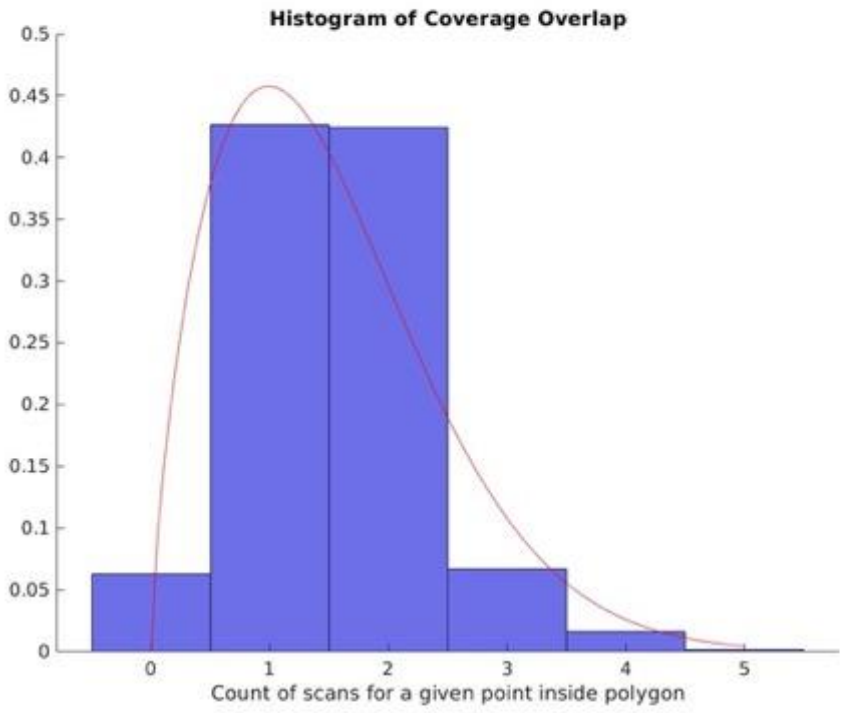


Figure 14: Path in concave polygon ROI



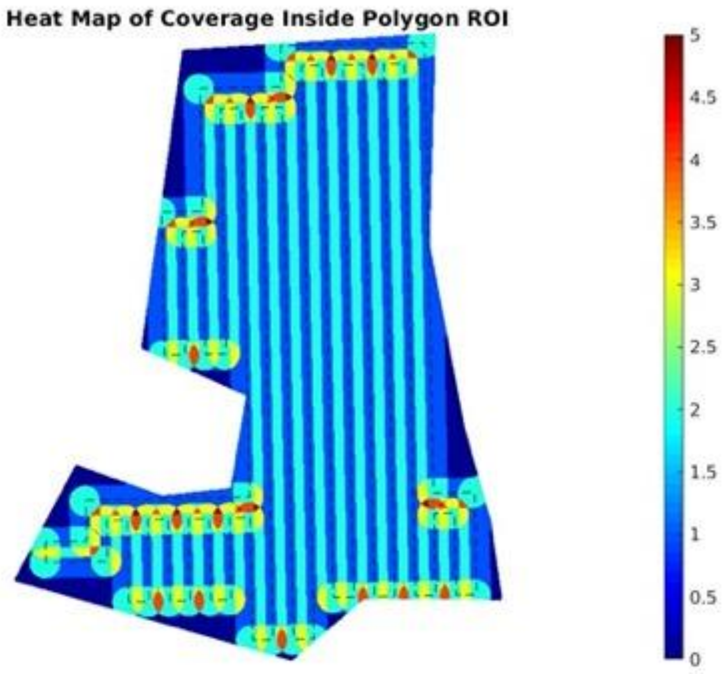


Figure 15: Evaluation of path in concave polygon ROI

6. Concave Polygon ROI with obstacles

# Waypoints	Optimization Index	Percentage of Coverage	Percentage of Overlap
78	0.756781	91.302181 %	52.928507 %

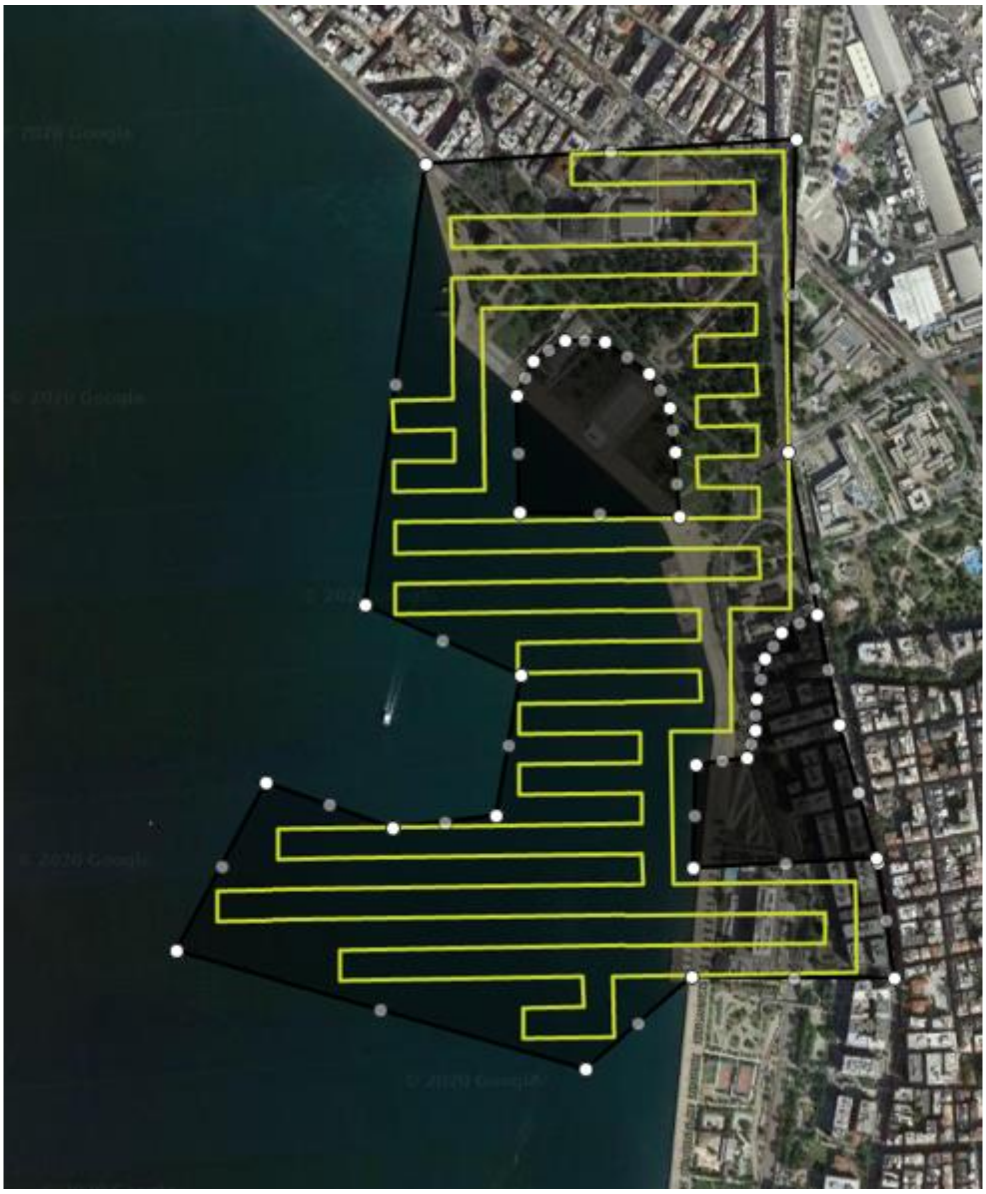
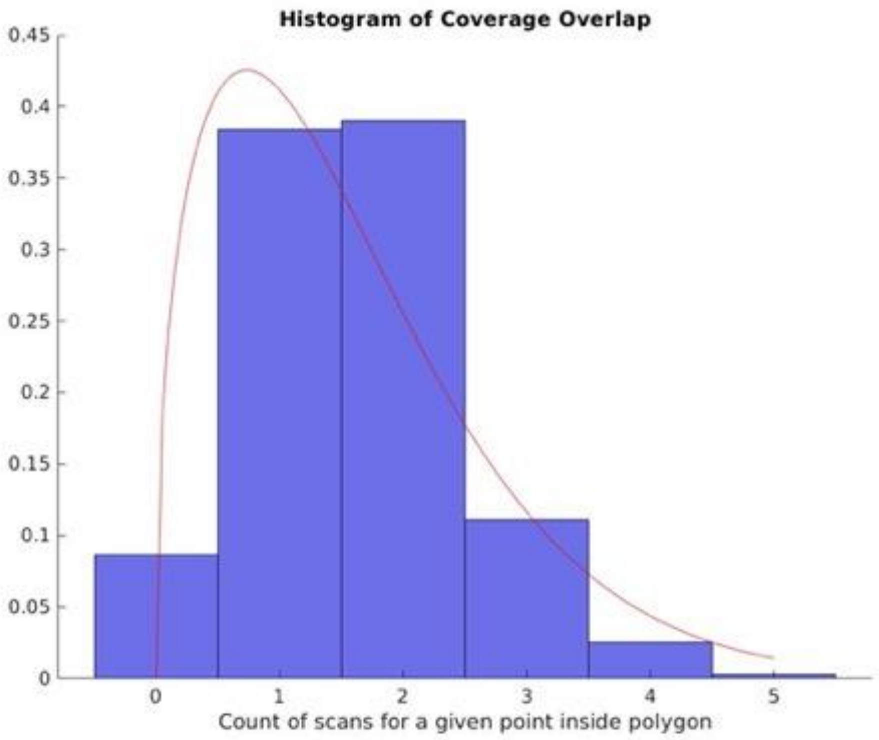


Figure 16: Path in concave polygon ROI with obstacles



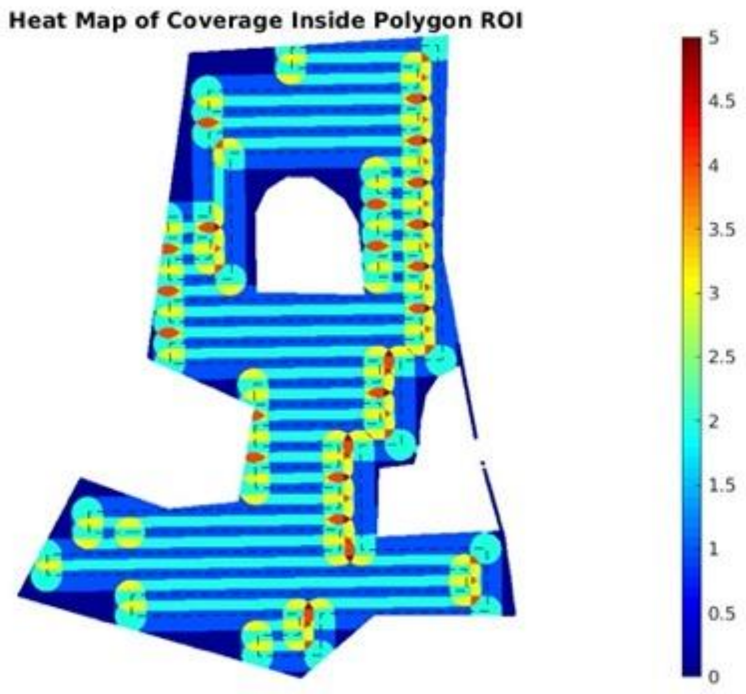


Figure 17: Evaluation of path in concave polygon ROI with obstacles

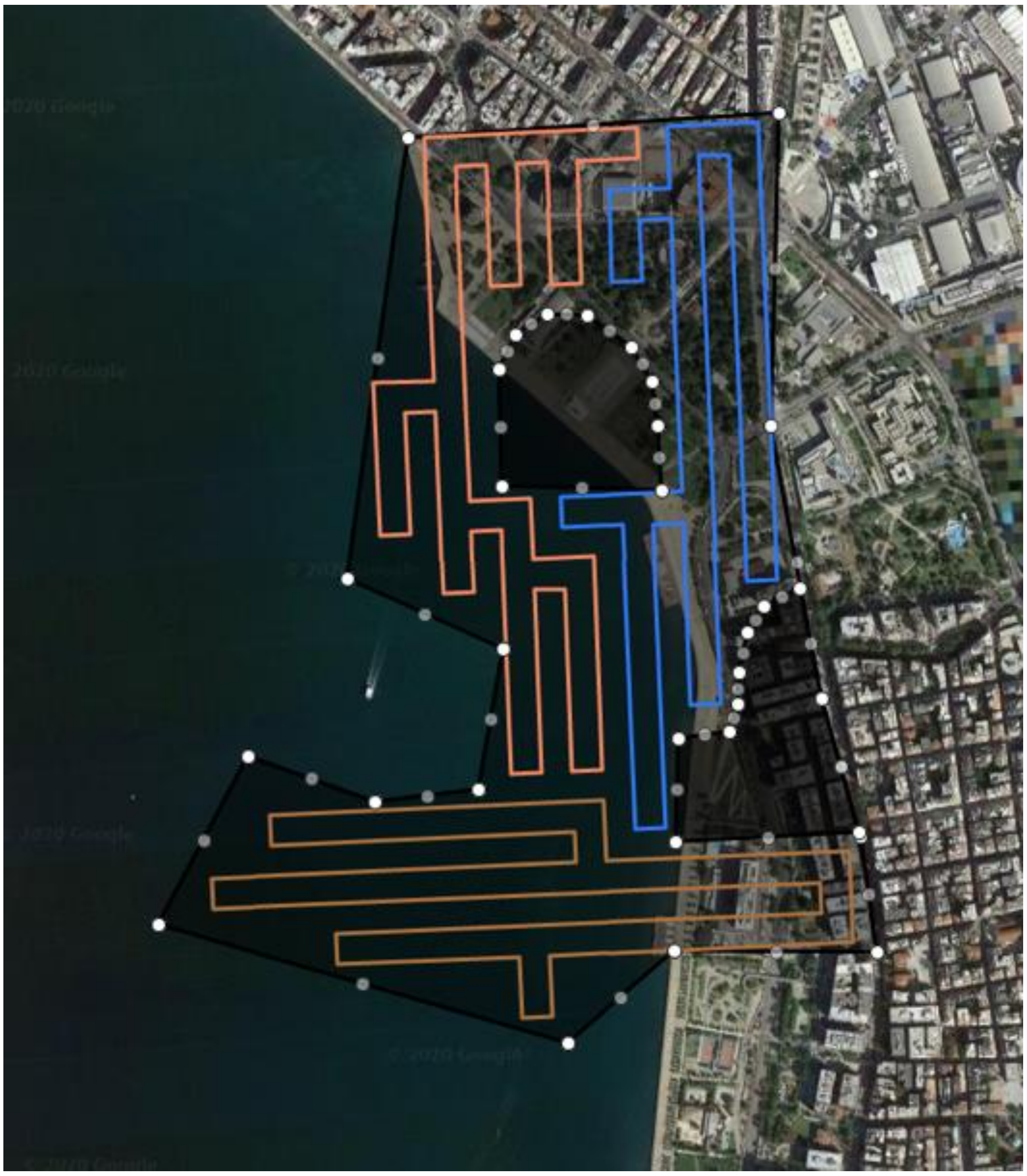


Figure 18: Multi-robot coverage paths in the same concave polygon ROI with obstacles

7. Summary of the mCPP method

As shown in the previous examples, the CPP method developed for the RC module of the ARESIBO platform offers a really high percentage of coverage in the defined ROI, manages wisely the operational resources offering energy and time efficiency and is capable of handling complex concave polygon ROI's with obstacles without any problem. In simple rectangle ROI's the method provides paths that completely cover the ROI, while even in the most unfavourable concave polygons with obstacles, that most of existing CPP methods cannot face at all, this method manages to provide paths with a percentage of coverage larger than 90%. It should be noted that in cases where the complete coverage of the ROI is critical, the method also provides the mode for better coverage that allows the paths to get outside of the polygon. Moreover, smaller scanning density or larger percentage of overlap may be used as well. The features provided by this CPP method makes it ideal for the need of the ARESIBO project.

2.5 Persistent coverage

In this path planning mode RC will provide paths for one or more vehicles inside a polygon ROI, in order to continuously patrol this ROI. For this path planning mode, the CPP method described above will be used as a base and a set of optimizations to minimize the time demanded to visit all of region’s points again will be used. At the time that this deliverable is writer, the persistent coverage mode of the RC module is at the starting point of development. A detailed description of persistent coverage mode will be provided in the second version of the deliverable.

3 RT-SAM: Real-Time Situational Awareness Maximization

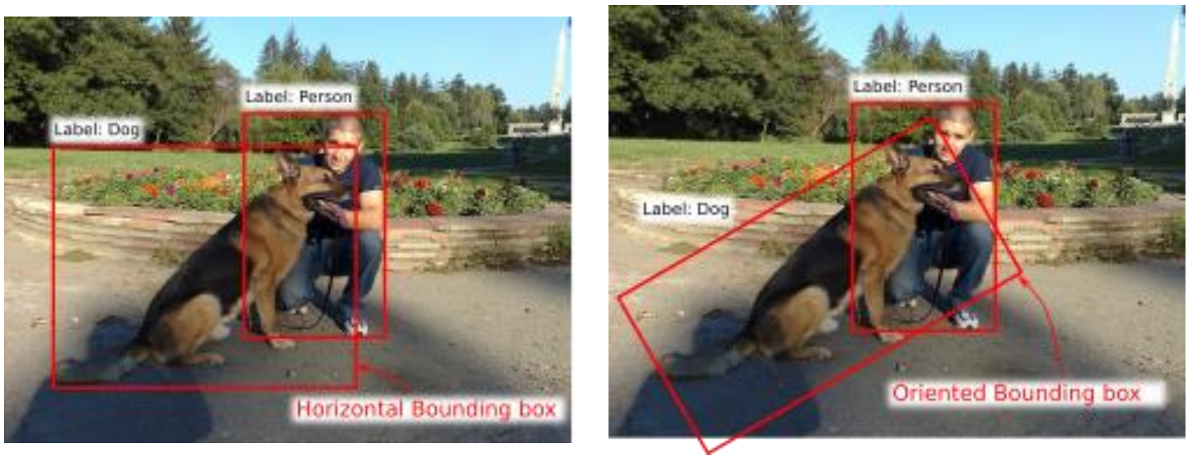
3.1 Introduction

The RT-SAM module is developed in the context of T3.6 - Sensing optimization that deals with the Dynamic Data Driven Assimilation (DDDAs) and the active sensing. The main focus of this module is the development of a self-learning cognitive tool, which will be able to autonomously guide all of the surveying assets, so as to increase the situational awareness on the field in real-time. In order to achieve the aforementioned goal two tasks are of a paramount importance: a) a state-of-the-art object detector and b) real-time adaptive path planning for the UxVs.

3.1.1 State-of-art object detector

Recent technological advancements in terms of hardware [5] have enabled the development of deep learning architectures [6] capable of dealing with extremely difficult [7] and abstract problems [8]. While deep learning techniques have been applied to a variety of task, the tasks at which the more impressive impact has been observed are the ones involving image processing. The key innovation which allowed this breakthrough is the Convolutional Neural Network (CNN) concept. The fundamental task in this field is image recognition or classification, where the goal is to classify a set of images into separate categories using the optical characteristics of each input image. They have successfully been used in simple consecutive mode as in VGG [9] or more elaborated implementations as in [10] where residual neural networks where introduced.

A closely connected task to image recognition is object detection. The latter task involves two interconnected layers: image classification and a bounding box (bbox) regression part. The purpose of object detection is to identify and localize object of interest inside an input image. Thus, an additional step (regression phase) is inserted in the process of image classification in order to achieve the overall objective.



(a) Horizontal bounding box **(b)** Oriented bounding box

Figure 19: Object detection and localization

The most common approach involves a tight bounding box but other options also exist. In Figure 19 (a), an example of horizontal bounding box is presented around two objects, a dog and a person. It can be observed that although the bounding box strictly surrounds the required object, there is a lot of background information also included within the box area. This effect is more intense in the case of the dog in comparison to the person box. An oriented bounding box, i.e. a bounding box which can be rotated also besides being stretched to encompass the object, around the dog seems to better encapsulate the actual object and contain less background information as can be seen in Figure 19 (b). Although, there are works that utilizes the notion of oriented bounding boxes, as in [11], the typical approach is to use horizontal bounding boxes. One of the main reasons for this selection is the additional parameters which should be learned for the oriented bounding boxes, apart from the 4 variables representing a horizontal box, which eventually increase the overall complexity.

An object detector will produce numerous bounding box candidates, with this number being at size class of 200-300 per image. The majority of the estimated areas are not useful candidates. Thus, it is necessary to filter the results and only retain what is considered as a good candidate. As the model performs an image classification for every bounding box, a corresponding confidence score can be assigned to each candidate area. Therefore, all rectangles can be ranked and treated separately considering this score. It is expected that a well depicted object will acquire a higher score than a blurred, occluded or generally not clearly presented instance. Some examples are present in Figure 20 where the occluded cars are denoted with lower scores than the fully represented and even a misclassified bounding box is displayed, which in fact contains a fraction of a truck. Producing a lower score can be efficiently filtered without resulting any false positive detection. The image was acquired from VisDrone dataset [12] which contains various small objects and is entirely recorded using Unmanned Aerial Vehicles. A typical value to distinct the useful from the noisy candidates is 0.5 although this can be adapted depending on the problem or the model. Applying this threshold to Figure 20 would lead to elimination of all bounding boxes without a full visible car inside.

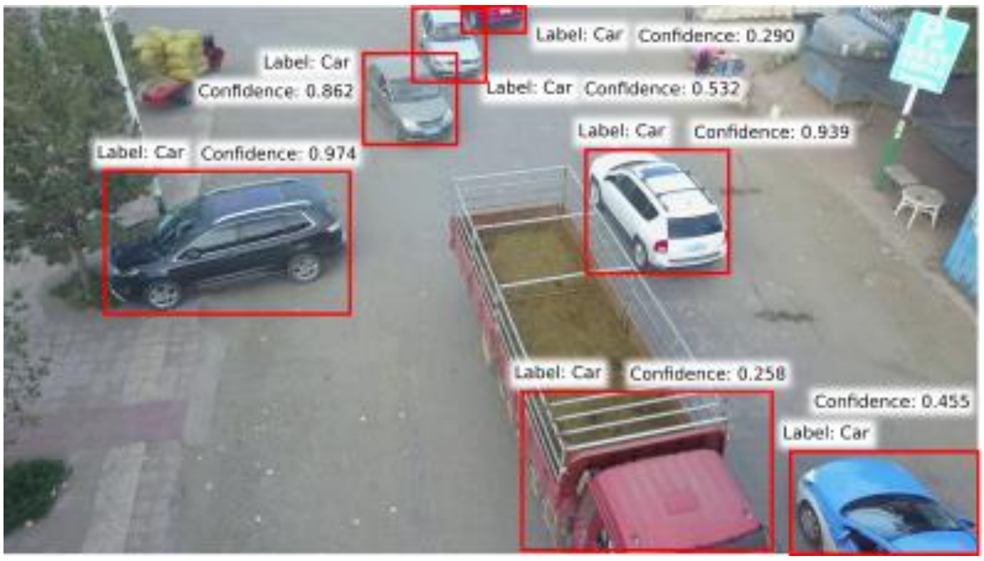

Label: Car Confidence: 0.290
Label: Car Confidence: 0.862
Label: Car Confidence: 0.532
Label: Car Confidence: 0.939
Label: Car Confidence: 0.974
Label: Car Confidence: 0.258
Confidence: 0.455
Label: Car

Figure 20: An example containing multiple instances of car objects from VisDrone.

UAV object detection

A general-purpose object detector initially focused on objects being captured on a first-person perspective. Nonetheless, current interest has been shifted to more sophisticated perspectives and acquisition angles such as from a UAV which captures objects from above, with or without an angle and typically from a distance. The following figures depict car instances along with their corresponding bounding boxes. In addition, all other objects of interest are discarded as it is applied in Figure 23 where the spatially dominant bus is excluded. It is obvious that the visual characteristics of the object change according to the perception. The first image has been retrieved from Pascal Voc [13], a general-purpose dataset that also includes aerial imagery, while the second has been extracted from UAV123 [14] which is a dataset for object tracking evaluation. Nonetheless, the latter database can also be utilized for object detection objective as both models are relevant to the final task. In addition, Figure 23 presents an another example where car instances are depicted and were taken from database COCO [15] which comprises an another general-purpose dataset. In all cases, the main object of interest involves cars but it is clear that different views of this object are depicted in every case. As it is obvious, the acquisition perception changes significantly between the three images, but also other quality differences can be spotted: the typical size of objects depicted in UAV imagery is smaller than that of a general-purpose dataset. The example selected for the UAV perspective can be considered similar. In ARESIBO, the perspective is expected to resemble the second example both in aspect of point of view as well as the object size, and even smaller object can be expected to appear, and thus, special care should be taken for the appearance and recognition of small objects. Regarding the orientation of the bounding boxes in this case, another factor that horizontal boxes are more suitable is that since the objects are usually small and the most important issue is the actual localization and recognition of the objects, in other words the correctly identification of an object and derivation of its position, the amount of included background information inside the bounding box seems a secondary oversight.

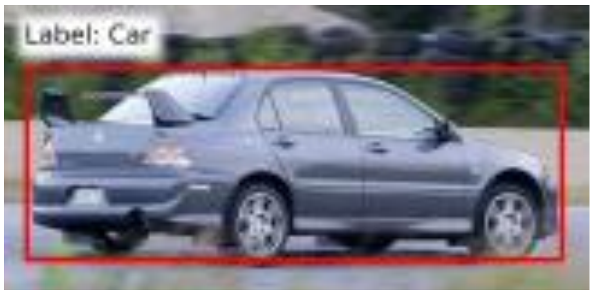


Figure 21: Typical example of an image containing car from Pascal Voc 2007

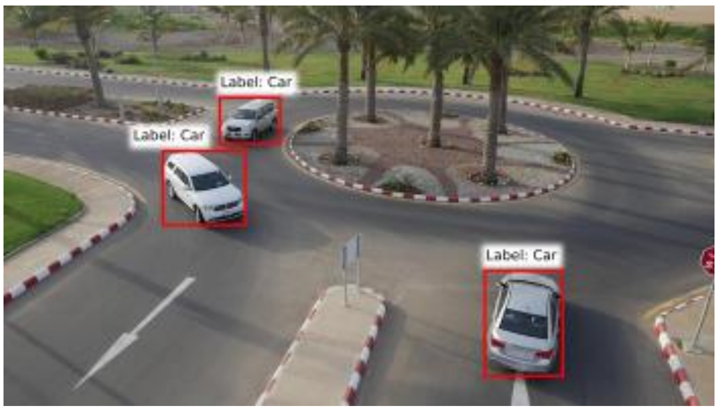


Figure 22: Typical example of an image containing cars from UAV123

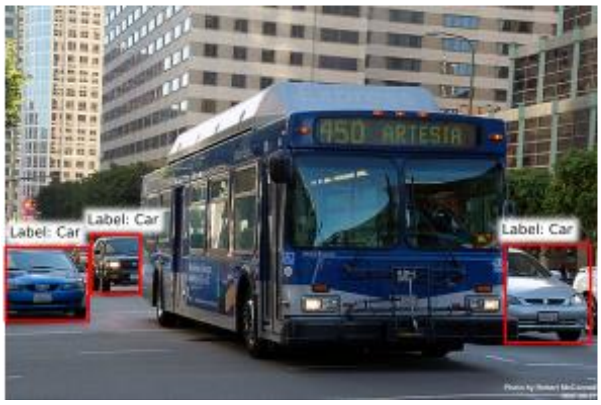


Figure 23: Typical example of an image containing cars from COCO

As it has been stated the perspective from which an image is captured is of high importance for the achievement of effective detection results. Though, the ideal case would be to exclusively use datasets with UAV point of view, this is not always easy to be fulfilled. Although, nowadays there are UAV perspective specific datasets which could be utilized for the purposes of the project, there is no guarantee that every object of interest will be covered by the available ones. Usually, the available datasets cover the most typical classes quite satisfactory, with many images and perspectives, but provide less, limited or even minimal images for cases of more specialized objects. The latter cases are a large issue when the coverage of these classes is also a requirement. The unbalance of objects of interest occasionally causes problem with the proper detection of misrepresented classes. This is further enhanced when the provided samples do not cover all required perspectives.

Related work & selected object detector

First approaches of Object detection with deep learning was a direct extension of region proposal network working along with convolutional neural networks (CNNs) as in Fast RCNN [16]. This approach had some efficiency issues and an end-to-end approach using CNNs was introduced: Faster RCNN [17] which proven to be quite more efficient while being effective also. In general, object detector can be roughly separated into two categories regarding the use of a Region Proposal Network (RPN): two phase detectors which make use of RPN and single-phase ones which attempt to simultaneously apply classification and regression. Monumental works of the first category is the aforementioned Faster RCNN along with R-FCN [18] while for the second category (single phase detectors) are Single Shot Detector (SSD) [19] and Yolo [20]. From these works Faster RCNN focuses more on improving the detection performance, Yolo primarily on creating an efficient model and SSD on making a compromise between the two latter cases. Many modifications have been proposed of the previous models and some of them have provided improved version(s) of the initial model also, as in the case of Yolo which now has 5 distinct versions so far. Besides these models, various architectures have been also proposed which either attempt to further improve the effectiveness or the efficiency. For example, in [21] multiple feature maps are being used with Faster RCNN to improve the model.

If a comparison is to be made between the two categories, the two phase detectors exhibit a robustness and higher performance because the RPN is usually quite effective at filtering the non-relevant bounding boxes and, thus, provide the model with good box candidates. Single phase detectors on the other hand, mainly focus on efficiency, since they lack the additional step of RPN, while trying to achieve good performance. It must be highlighted that the former category typically requires less extensive augmentation to achieve decent results. In the case of ARESIBO, the typical object size is expected to be small compared to the whole image as the footage will be captured by UAVs hovering on a distance. Thus, a robust object detection system is considered to be an efficient selection in order to be able to deal with small objects.

The concurrent developments in the field of robotics have allowed the introduction of such methodologies to ground and aerial and also the development of special-purpose algorithms to detect and/or track objects of interest with one [22] or more UxVs [23]. For the needs of the project, a state-of-the-art detection algorithm will be used, leveraging all of the aforementioned advancements in the relative field, that will be tailored to the needs of ARESIBO platform, trained on public (and if necessary private) datasets and will be developed in the context of the same task (T3.6).

The dataset through which a model is trained constitutes a key role in the effectiveness of the model. A general rule is for a model to successfully cover the usually unforeseen inference images, to use a quite large dataset. Nevertheless, in many cases, the collection of a large training set is not easily accomplished. Thus, in those cases, and in order to fulfill the requirements posed in the larger degree we opt for a model which can adopt and generalize easier. This is interpreted as the ability of the model to recognize misrepresented classes or even unforeseen perspectives of objects of interest. Faster RCNN has proven to be highly robust and, thus, to produce a well generalized effective model. Thus, in this context, a specially trained Faster RCNN implementation has been selected to meet the project requirements and to accurately detect objects of interest. The actual model that was used is Faster RCNN with a Resnet101 [10] backbone which has a reported inference time of 106ms per image on COCO dataset. In the following figures, some examples of this specially trained Faster RCNN model are presented where the under-consideration classes were identified based on an initial assessment of the first version of the end-user requirements.

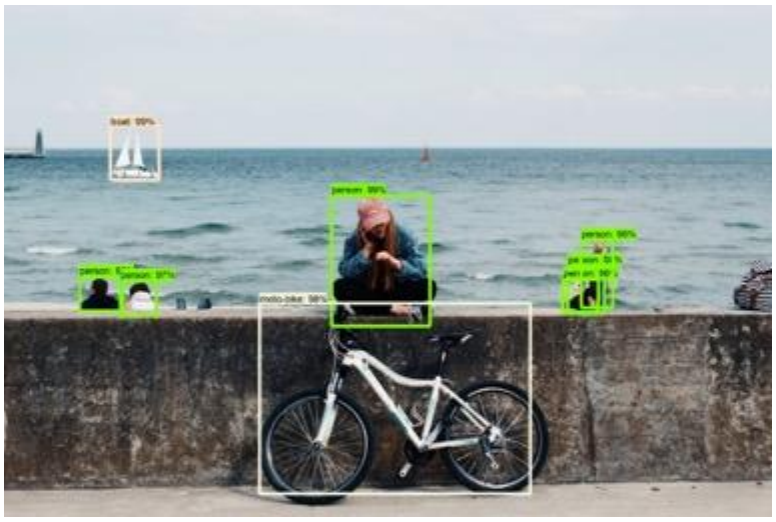


Figure 24: Inference example of Faster RCNN

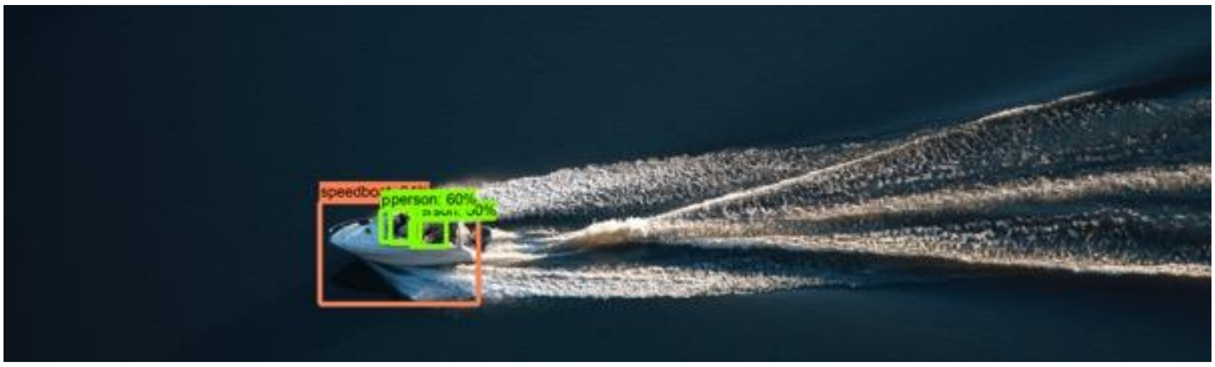


Figure 25: Inference example of Faster RCNN (UAV perspective)

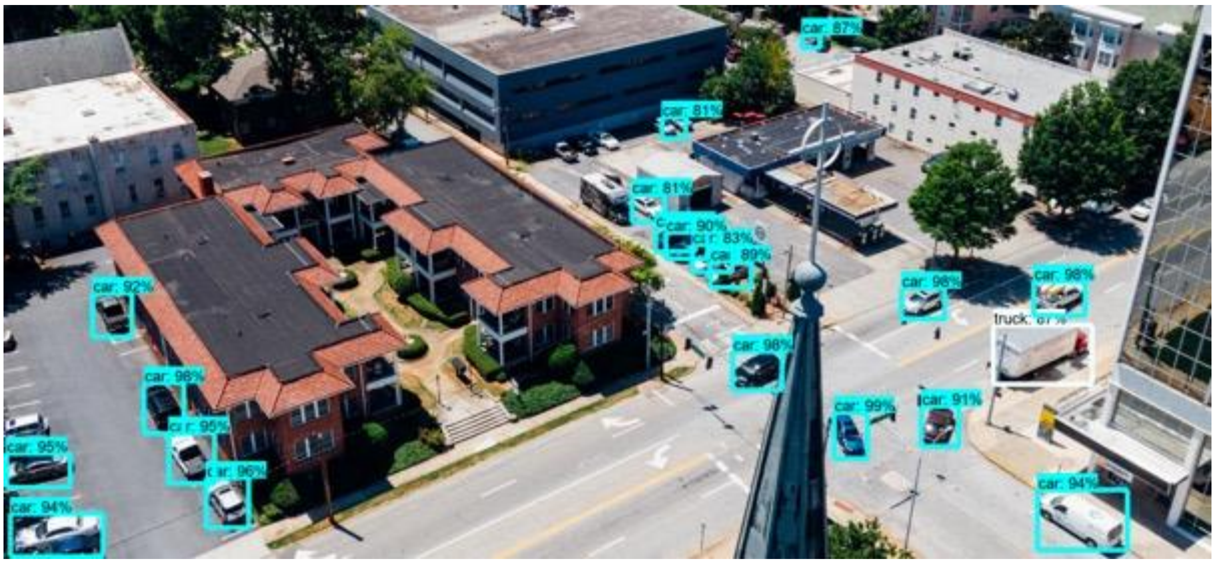


Figure 26: Inference example of Faster RCNN (UAV perspective)

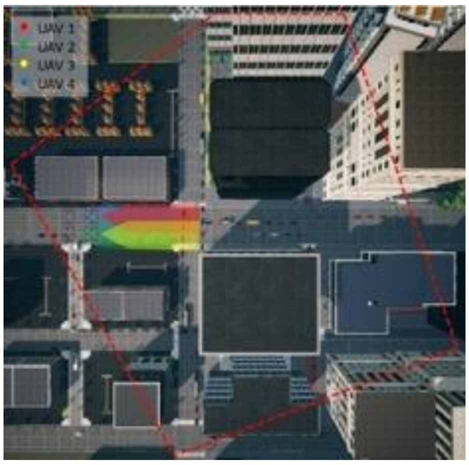
As mentioned before, a Faster RCNN model was trained taking into consideration the initial end-user requirements available at the time. As a first choice publicly available datasets were chosen to provide training samples for the model. In this aspect, a number of dataset provided for object detection or similar tasks was used which includes UAV123, VisDrone and UCF aerial action dataset [24]. However, in order to include samples for all object of interest, defined in the end-user requirement, private datasets were also used along with the aforementioned datasets. The overall training set included more than 20k images. Additionally, an evaluation dataset was created to examine the performance of the model and its ability to learn the provided training set. This later dataset, as expected did not include any training image and it was specially chosen so as the different classes have a balanced representation in it. Table 1 contains the evaluation results of the model on this evaluation set. The model was tested on a system equipped with a GTX-1070TI-A8G GPU card which achieved a processing time of 5 fps for a typical image size of 720p. The classes that were used include most typical land and maritime vehicles along with a human (person) class. It worth mentioning that the final class list is expected to be further modified according to the final end-user requirements.

Table 1. Detection results of Faster R-CNN

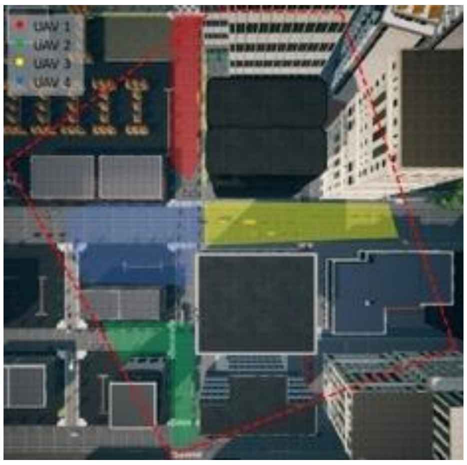
Object category	**Average Precision (AP)**	**mAP (mean AP)**
Boat	0.56543	0.66271
Bus	0.70576	
Car	0.75568	
Inflated boat	0.41560	
Motorcycle/Bicycle	0.76698	
Person	0.84015	
Ship	0.73174	
Speedboat	0.53311	
Truck	0.64993	

Real-time adaptive path planning for the UxVs: refers to the on-line trajectory generation for each robot of the swarm, based on the feedback provided for the relevant task. Despite many methodologies are capable of detecting objects of interest in real-time, most of them rely on either predefined or human-controlled paths for the UxVs. Moreover, such approaches lack the active feedback required for a real-time adaptive path planing and the coordination among different UxVs is usually achieved by assigning spatially exclusive areas to each member of the swarm, significantly restricting the UxVs cooperation. The novel module we developed tackles all of the above issues and it is capable of optimizing in real-time the monitor positions of the UxVs to increase the overall situational awareness by allowing each member of the swarm to move freely inside the operational area. In addition, no prior knowledge about the “geolocation” of information-rich areas is needed, thereby no human-controlled predefined path is required.

Figure 27 portraits an indicative example of such a setup, in which a swarm of 4 UAVs has as an objective to increase the accuracy and number of detected pedestrians and vehicles. The left-hand side of Figure 27 illustrates the initial monitoring positions for the swam of 4 UAVs and their corresponding fields of view (transparent polygons) of the all the UAVs, moments after their take-offs. The right-hand side of this figure illustrates the converged configuration for the swarm of UAVs. A direct outcome, that can be derived sorely from the bird’s-eye view of the operational area, is that the proposed algorithm spread the UAVs taking into consideration the combined field of view. UAVs have been placed either in close vicinity of a crossroad (UAV 2 & 4), to exploit the fact that these spots usually have the biggest flow of cars/people or in positions that allows them to monitor the whole length of a road segment (UAV 1 & 3). With the converged positions, the UAVs can monitor multiple road-parts from a single monitoring spot.



(a) Initial configuration



(b) Final configuration

Figure 27: Illustrative example: A swarm of 4 UAVs is deployed having as objective to find the monitoring poses (position & orientation) inside the operational area (red, dashed polygon), that maximize the overall situational awareness. In both instances [sub-figures (a) & (b)], the fields of view (transparent polygons with the corresponding colour) of all the UAVs (top bird's-eye view) are depicted.

3.2 Technical Elaboration

3.2.1 UxV control

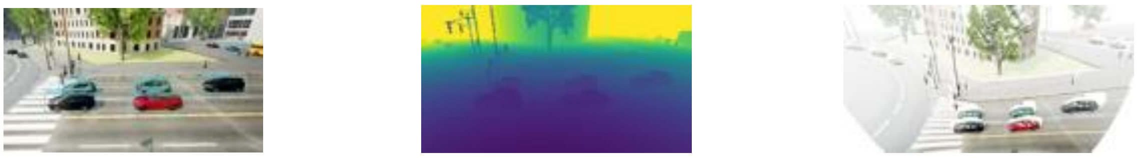
Consider a team of UxVs which work cooperatively to increase the situational awareness over an unknown, dynamic environment. At each moment, each UxV is allowed to configure its own DoF (e.g pose by changing its own position and orientation). Moreover, a number of constraints is applied to the UxV controllable variables for a number of reasons. These are:

i. The rate of change of every controllable variable is restricted to a maximum value, so as to ensure that the HITL (human in-the-loop) will be able to monitor all of the UxVs and any cancel in time any abnormal behavior.
ii. The UxVs should always remain inside the operational area.
iii. The UxVs should avoid any collisions with all the obstacles (stationary or moving).

Constrain (i) is met by the setting *.json file provided by Mission Planner, (ii) via telemetry measurements provided by each UxV individually and (iii) by leveraging the on board mounted sensors on each UxV.

3.2.2 Measurements

After moving towards a new position, each UxV will be able to enable its sensors and thereby perceive a part of the operational area. This perception includes RGB and Depth image. The RGB image is processed by the corresponding module, which will provide a list with all the detected objects, their confidence interval and the relative location of the corresponding bounded box. The center of mass, in each one of the detected objects, is extracted by utilizing the formation of the pixels that lie inside the bounding box of that object. Combining also the information from the depth image, the dimension of center of mass is expanded to 3D space.

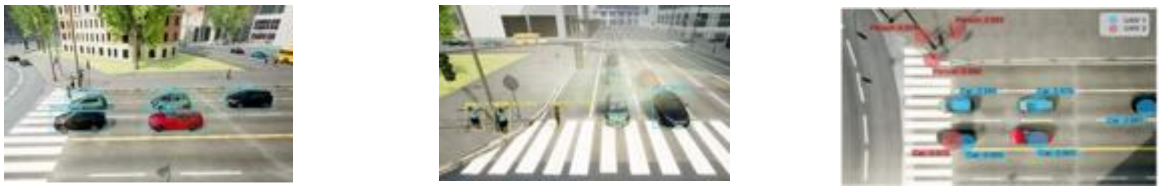


(a) RGB Image (b) Depth Image (c) 3D Representation

Figure 28: Sensor information received and 3D representation

3.2.3 Objective function

Having projected all these objects in a global 3D frame, we can now retain only the unique objects, by checking the distance between the detected objects of each UxV, in this common 3D frame. In the case where an object is identified by more than one UAV, the object with less confidence in its detection is discarded. By doing so, we want, on one hand, to "force" UxVs to detect unique objects, and, on the other hand, to reward UxVs that have achieved to detect objects with a high level of certainty. Thereby, the objective function of the swarm is defined as the summation of the confidence interval for every uniquely detected object of interest.



(a) UAV 1 perception (b) UAV 2 perception (c) Combined perception

Figure 29: Illustrative example of the objective function. Images taken from each UAV with the corresponding bounded boxes (a) & (b). Their combined perception (c)

3.2.4 Navigation

Initially the Centralized System will calculate the contribution of the last action each UxV performed. Then, it will forward these contributions to the corresponding UxVs. Each UxV will keep track of these contributions and it will use them to update its estimator. It will then create a number of perturbations around its current position and it will evaluate each one of them utilizing the aforementioned estimator. Finally, each UxV will move towards the candidate position that maximizes its contribution and thereby the maximization of the objective function will be achieved.

3.3 Algorithm Evaluation

3.3.1 Implementation details

In order to evaluate the the developed algorithm, we run a number of experiments using the AirSim simulator [25]. AirSim is an open-source, cross-platform, simulator for drones, cars and more that supports hardware-in-loop with popular flight controllers for physically and visually realistic simulations. All the experiments were carried out in the CityEnv of the aforementioned simulator, which is a vast, realistic environment that simulates both the static structures and the highly non-linear and dynamic behavior of the moving assets (e.g. cars, trucks, pedestrians, etc.). The operational area (Figure 30Figure 27) is defined on top of the "central" roundabout of the environment. Robots were allowed to move inside a circle centered at the center of the roundabout and with a radius of 70 m, covering an area of 15393.8 m^2.

The swarm consists of simulated multirotor UAVs instantiated with the AirSim built-in flight controller (called simple flight) and equipped with stationary cameras. Each stationary camera is located at the front and rotated downwards 45 degrees on the pitch axis. Each UAV can move inside the xy plane of the previously defined operational area and also has a 360 degrees yaw movement. The operational height of the first UAV is at 14m from the ground, while each additional UAV is deployed at a height 0.5m higher than the previous one. Our algorithm can also facilitate different types of UxV and each UxV may have more DoF, but as a proof of concept the aforementioned UAVs are sufficient.

The calculation of detected objects, in each one of the received RGB images of the UAVs, was achieved by employing YOLOv3 [26] detector pretrained on COCO dataset [27]. YOLOv3 is indeed a state-of-the-art object detector capable of producing reliable predictions in real-time, however, it has not been trained for data coming from a simulator, nor for top-view images (typical UAV's images.). Moreover, COCO dataset is a general-purpose large-scale object detection. Although this unsuitability of the object detector caused several problems (e.g. objects completely missed if they were observed from different angles, radical changes in the confidence levels of detected objects caused by slight changes in the pose of the UAV, etc.) in the evaluation of the UAVs' configuration, we chose to apply the object detector "as is", to highlight that the proposed navigation algorithm does not utilize any information related to these choices, therefore it is modular with respect to alternative systems.

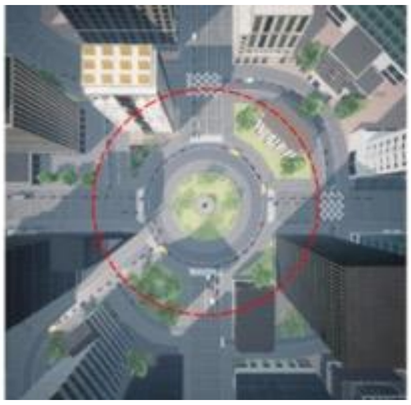


Figure 30: The operational area in which the developed algorithm was evaluated

3.3.2 Performance Comparison with Centralized Semi-Exhaustive Search

Before continuing with the analysis of the evaluation results, let us define an algorithm that has little practical value, however, its achieved performance can provide us with valuable insights, when compared with the developed navigational algorithm. This algorithm is a centralized, semi-exhaustive methodology that works as follows: At each time-step, first, it generates a subset (semi-exhaustive) of candidate UAVs' configurations (centralized) out of all possible ones. Then, all these candidates are evaluated on the AirSim platform, i.e. the UAVs have to actually reach that candidate monitoring positions, and, for each one of them, is calculated also the objective function. Finally, the next configuration for the swarm is the candidate maximizes the objective function value. This procedure is repeated for every time-step of the experiment.

As the number of candidate UAVs' configurations (that are evaluated) increases, the performance of this algorithm will approximate the best possible one. The biggest asset of this approach, to be able to evaluate configurations of UAVs before deciding about their next movement, is also its major disadvantage, as it renders this decision-making scheme unfeasible and unsuitable for any kind of real-life applications.

Figure 31 presents a comparison study between the semi-exhaustive search and the proposed algorithm. 4 UAVs were deployed in the same operational area as defined in Figure 30. For both algorithms, the experiment lasted 300 time-steps. The semi-exhaustive algorithm evaluated 60 different UAVs' configurations before deciding about the next monitoring position. The proposed algorithm was capable of producing a similar behavior with the semi-exhaustive algorithm, having almost the same convergence rate. The final converged value (average value after 150 time-steps) was 47.64 for the proposed algorithm and 50.97 for the semi-exhaustive algorithm. All in all, this analysis highlights that the proposed algorithm is capable of achieving a performance that is equivalent to having the "luxury" to spare 60 different combinations of UAVs, before deciding about the next configuration of the swarm.

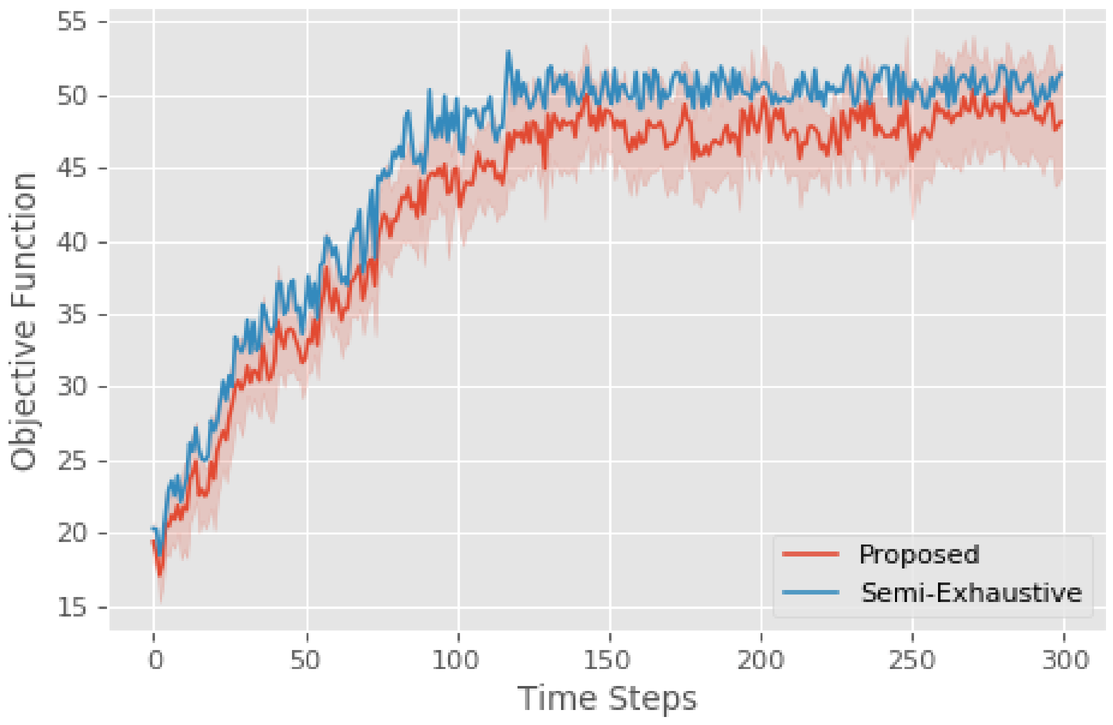


Figure 31: Performance comparison between the developed algorithm and the centralized, semi-exhaustive search

3.3.3 Trajectory – Areas of Convergence

In this subsection, we look deeper into the results of Figure 31 for the evaluation of the developed module and provide with extra insights and visual results about its operation and robustness. Figure 32 graphically illustrates the trajectories of all the UAVs from their taking-off positions to their converged ones, for a single instance, as calculated by the module. This visual representation of the resulting trajectories validates the fast convergence of the proposed navigation algorithm, as it was initially derived from Figure 32. The fluctuations of the UAVs around the converged positions is an essential ingredient of module, that allows rapid adaptation of the swarm to cope with changes in the environment (e.g. sudden changes of the traffic-flow, congestion, etc.). We can see that the final positions of the UAVs are close to a crossroad, where the traffic is usually higher. Moreover, in the final configuration of the swarm, robots are placed in such a manner (fixed positioning and orientation) so as to be able to detect object from multiple road-parts.

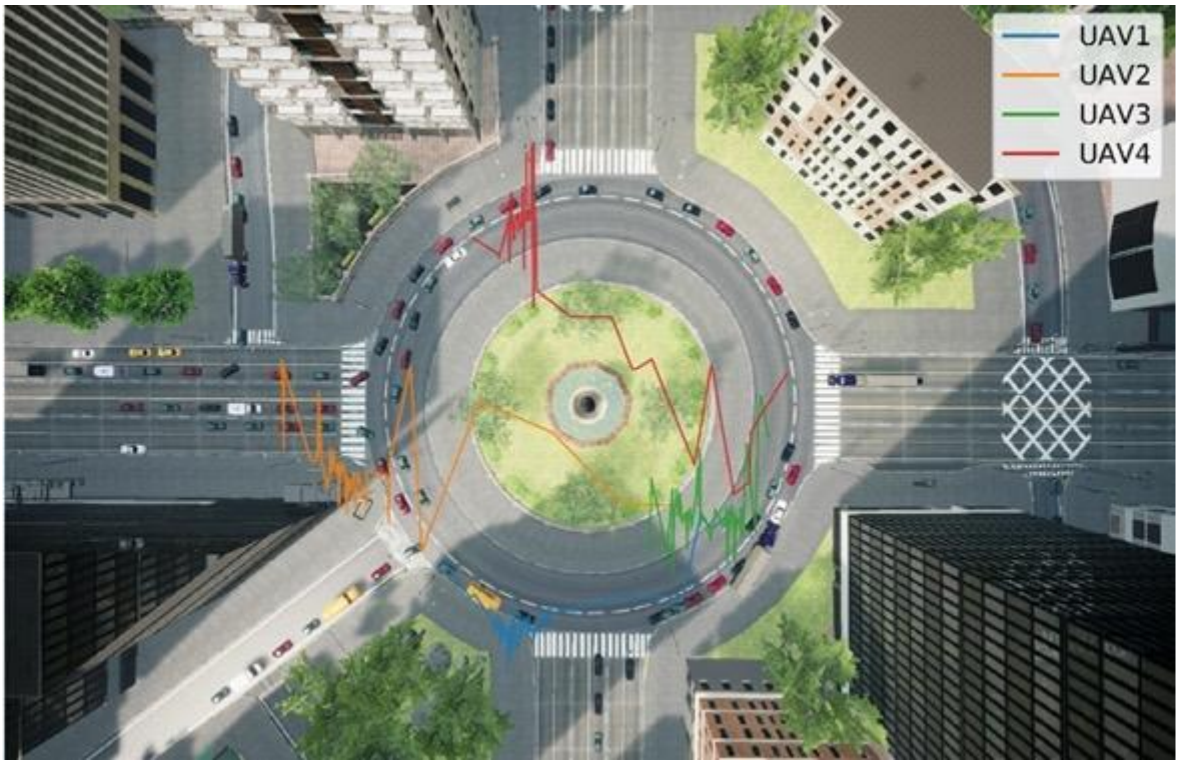


Figure 32: Evolution of the trajectories as calculated in real-time by the proposed algorithm

4 Conclusions

The modules described in the previous sections provide the operator of the ARESIBO platform with some powerful tools for the planning, management and execution of missions, in order to acquire and process data, to obtain complete operational. Thanks to them, the operator of the platform is able to create and execute missions involving multiple vehicles, with the minimum demanded cognitive load possible, without the need of knowing specific technical details, or having specialized training courses. The state-of-the-art algorithms developed for this purpose, ensure increased operational efficiency, high accuracy in the collected information and most importantly safe multi-robot autonomous operations. The state-of-the-art object detection solution provided, facilitates in the overall situational awareness, the assessment of threats and the high-level guidance of vehicles in order to fulfil the missions' objectives. It is expected that in the second version of this deliverable (D3.6 - Dynamic and Adaptive Swarm Optimization V2) will be presented further results for the evaluation of both modules described above, and will be resolved the possible issues that will come up during the integration and testing with the rest modules of the project.

5 Acknowledgment

This deliverable has been completed thanks to the support of ARESIBO partners who provided their contributions to be integrated in the document, as well as participated in the review of the document's content.

References

1 Cabreira, T. M., Brisolara, L. B., & Ferreira Jr, P. R. (2019). Survey on coverage path planning with unmanned aerial vehicles. Drones, 3(1), 4.
2 Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. Robotics and Autonomous systems, 61(12), 1258-1276.
3 Gabriely, Y., & Rimon, E. (2001). Spanning-tree based coverage of continuous areas by a mobile robot. Annals of mathematics and artificial intelligence, 31(1-4), 77-98.
4 Kapoutsis, A. C., Chatzichristofis, S. A., & Kosmatopoulos, E. B. (2017). DARP: divide areas algorithm for optimal multi-robot coverage path planning. Journal of Intelligent & Robotic Systems, 86(3-4), 663-680.
5 H. Sarbazi-Azad, Advances in GPU Research and Practice. Morgan Kaufmann, 2016.
6 Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” nature, vol. 521, no. 7553, pp. 436–444, 2015.
7 V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602, 2013.
8 D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” Science, vol. 362, no.6419, pp. 1140–1144, 2018.
9 Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
10 He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
11 G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, “Dota: A large-scale dataset for object detection in aerial images,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3974–3983.
12 Zhu, P., Wen, L., Du, D., Bian, X., Ling, H., Hu, Q., ... & Ma, W. (2018). Visdrone-det2018: The vision meets drone object detection in image challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 0-0).
13 Everingham, M., Van Gool, L., Williams, C. K., Winn, J., \& Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338.
14 Mueller, M., Smith, N., & Ghanem, B. (2016, October). A benchmark and simulator for uav tracking. In *European conference on computer vision* (pp. 445-461). Springer, Cham.
15 Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.
16 Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).
17 Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).
18 Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In Advances in neural information processing systems (pp. 379-387).
19 Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
20 Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

21 T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie,“Feature pyramid networks for object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp.2117–2125.
22 Y. Bazi and F. Melgani, “Convolutional svm networks for object detection in uav imagery,” Ieee transactions on geoscience and remote sensing, vol. 56, no. 6, pp. 3107–3118, 2018.
23 J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, “Multiple moving targets surveillance based on a cooperative network for multi-uav,” IEEE Communications Magazine, vol. 56, no. 4, pp. 82–89, 2018.
24 University of Central Florida, UCF aerial action dataset, November 2011.<http://server.cs.ucf.edu/ vision/aerial/index.html>
25 S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in Field and Service Robotics, 2017. [Online]. Available: https://arxiv.org/abs/1705.05065.
26 J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” arXiv, 2018.
27 T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in European conference on computer vision. Springer, 2014, pp. 740–755.