Vehicle agnostic area coverage maximisation for campaign planning

Andrea Munafo*, Atanas Laskov, Patrick Opgenoorth, Ben Whetton, Pierre-Yves Mignotte, Scott Reed
SeeByte ltd, Edinburgh EH4 2HS, UK
*Corresponding author. Email: andrea.munafo@seebyte.com

Synopsis

The level of maturity of maritime autonomous vehicles makes the deployment of networks of vehicles cost-effective with an increasing number of applications that today require the usage of autonomous robots. To maximise operational effectiveness, one critical decision variable that has a direct impact on both the cost and performance, lies in how to properly place the different robots/sensors in the target environment. The number of assets, the dedicated communication networks, can contribute a significant portion of the overall cost and their placement determines the overall mission time and the associated coverage area. The ability to quickly allocate the available vehicles to one specific area, to monitor the mission performance, and to understand the quality of the data gathered would represent a key capability to speed up the uptake of the technology even more. This paper aims at bridging this gap, and provides a solution to efficiently schedule multi-vehicle, multi-days and large areas campaigns. The output of the campaign planner is a set of goals (e.g. areas to survey, targets to reacquire), which can then be allocated to the vehicles for execution. Finally, to maximise vehicle interoperability this paper also report an automated translator from high-level mission plans to vehicle to vehicle specific commands. Results are reported through simulations.

Keywords: Mission planners, Autonomous Systems, Autonomous Underwater Vehicles, Autonomous Surface Vessels, Marine systems

1 Introduction

The achieved maturity of unmanned underwater and surface vehicles (UUVs and USVs - UxVs) makes the deployment of multi-vehicle networks practical and cost-effective. For example, many scientific, civilian and military applications nowadays require the usage of autonomous underwater vehicles (AUVs). Networks of AUVs have been deployed in scenarios ranging from surveillance to mine-counter measurement operations or oceanographic systems. See, among others, Ferri et al. (2017); Yordanova et al. (2017); Caiti et al. (2013); Salavasidis et al. (2019) for some recent examples.

From an end-user perspective the seamless ability to scale from one asset to multiple vehicles able to coordinate their activities means maximising their benefits, creating a better match between the capabilities and strengths of the robots with those of the human operators. Moving in this direction, Caiti et al. (2012) presents a simulative approach to assess the level of underwater security in civilian harbor installations to explicit the link between system specifications and operative performance. The system aimed at using autonomous systems but no explicit planning was included.

The problem of planning for multiple vehicles typically deals with designing algorithms that can generate a collaborative travel plan for a set of autonomous agents. Recent examples include Kala and Warwick (2011) where heterogeneous multiple ground vehicles are coordinated together using RRT-connect. Coupled and decoupled multi-AUV motion planning approaches for maximizing information gain is presented in Wu et al. (2019). Each AUV is equipped with a video camera and a side-scan sonar and the planners are used to generate multi-AUV trajectories that capture close-up video footage of a site from a variety of different viewpoints.

One interesting approach to deal with the command and control complexity of multi-vehicles deployments is presented in Marques et al. (2017). The work focused on offline planning and on how to optimise the coordination of multiple autonomous vehicles through some degree of automated decision making. SeeByte has developed its own multi-vehicle planner, called Neptune (Miguelañez et al. (2011)). Neptune is a goal-based planner that is able to optimise and adaptively plan the execution of a mission. It acts at the command and control level to produce optimised offline multi-vehicle plans and at vehicle level to automate the mission execution using a modular, and open behaviour-based architecture (see Figure 1).

These automated planners have represented a real paradigm shift from the early days where UxV missions were a set of simple pre-planned waypoints to today where vehicles are able to plan and re-plan based on data measured in the field. At the same time, planning missions for multiple autonomous vehicles requires operators to monitor multiple parameters at once, including vehicle and environment state. Information such as vehicle maintenance and preparation status, battery levels, suitability for a task, presence of obstacles or weather conditions are key variables that affect mission requirements and execution for each single deployment. When mission planning involves multiple vehicles and multiple days these problems are exacerbated even more with an exponential increase in planning complexity and hence on the operator's cognitive load. As a result, most of the available command and control systems for autonomous marine vehicles tend to deal with missions that are relatively short in both time and space (e.g. up to 1 day, and up to tens of square kilometers).

 https://doi.org/10.24868/issn.2515-818X.2020.048

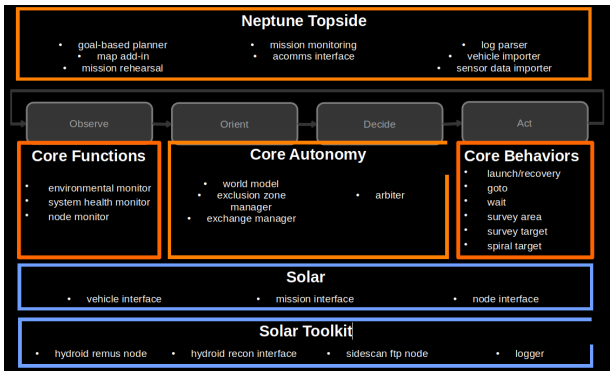

Figure 1: High level design of SeeByte's Neptune Goal-based planner.

Within this context, this paper describes a research and development project that SeeByte has recently started to bridge the gap from daily operations to long endurance and large area campaign planning of large group of heterogeneous vehicles. The approach is based on separating the complexity of higher level campaign planning from the day-to-day mission requirements and execution. This resulted in the development of two new key components: an automated scheduler (or campaign planner) that is able to optimally assign mission goals to vehicles, and a vehicle-agnostic translator that is able to map goal-based mission requirements into vehicle specific commands. The automated scheduler connects to a unified asset and task manager to collect campaign relevant information, but it abstracts away from the details of mission execution, to finally optimise the placement of the available robots in the target environment. This is one critical decision variable that has a direct impact on both the mission cost and performance. The number of sensors and the dedicated communication networks can contribute a significant portion of the overall cost, while the placement of the vehicles in the region of interest determines the overall mission time and the associated coverage area. The ability to quickly allocate each vehicle to one specific area, to monitor the mission performance, and to understand the quality of the data gathered would represent a key capability to speed up the uptake of the technology, overcoming critical weaknesses of most existing systems. The output of the campaign planner is a set of goals (e.g. areas to survey, targets to reacquire), which can then be allocated to the vehicles for execution. This is done shifting responsibility from the campaign planner to SeeByte's Neptune system. At this level, Neptune can potentially access a finer degree of details (e.g. dynamic models of vehicles, very fine weather forecast or environmental data, etc.) and further optimise the mission if needed. Neptune does this both pre-deployment and/or in-mission, with the vehicles that are able to change behaviorus based on the data that are collected in real-time.

Finally, because vehicles differ in the level of abstraction of their control interface, it becomes necessary to develop a consistent way to translate the desired higher-level goal-based commands into vehicle-specific lower level concepts. This is done at Neptune level, using an automated translator that converts Neptune plans into a sequence of vehicle commands. The availability of this translator enables the development of completely hardware-agnostic behaviours, reduces maintenance costs and alleviates the backward compatibility problem for interaction with legacy vehicle interfaces.

This paper presents the newly developed campaign planner, from the theoretical framework that has been developed to solve the campaign planning as an area coverage maximisation, to its implementation into the broader SeeByte's software ecosystem. Details on how the scheduler is linked together with the Neptune system are provided, and more specifically how the automated translator is used as a bridge between the high-level plans and the vehicle-specific interfaces. Results are shown through numerical simulations to demonstrate how the system is able to plan multi-asset and multi-days missions effectively.

The rest of the paper is organised as follows: Section 2 describes the problem formulation and the underlying assumptions. Section 3 goes into the implementation details including the translator from the resulting mission plans to vehicle specific commands. Section 4 presents the results, and finally Section 5 draws conclusions and highlight future work.

2 Problem formulation and assumptions

We consider the problem of surveying an area $\mathscr{A}$, with a fixed maximum number of robots n, over a desired period of time T (e.g. a week).

Each robot k can only work for a specific amount of hours T_k (e.g. 8 hours), after which it has to go back and recharge its batteries. Each robot is also equipped with a specific sensor S_k (e.g. side scan sonar, camera, etc.) that determines its maximum travelling speed $v_{max,k}$ and surveying trajectories.

 https://doi.org/10.24868/issn.2515-818X.2020.048

The problem can be approached from the general perspective of camera (or sensor) placement, with the objective of maximising coverage subject to having a fixed number of sensors, a maximum survey time available, and potentially other application specific constraints (e.g. communications).

Following Zhao et al. (2013), the problem can be tackled discretising the space of sensor configurations as $\{\Gamma_i : i = 1...N_s\}$, and the target space as $\{\Lambda_j : j = 1...N_p\}$ and using two sets of binary variables: set $\mathscr{S} = \{b_i : i = 1...N_s\}$ where $b_i = 1$ if a sensor is placed in Γ_i; the set $\{x_j : i = 1...N_p\}$, where each $x_j = 1$ indicates that the point Λ_j of the target space is observed given the current sensor allocation $\mathscr{S}$.

Using this formulation, the problem can be formally stated as:

$$\begin{aligned} &\text{maximize} \quad f(x_1, ...x_{N_p}) \\ &\text{subject to} \\ &\sum_{i \in N_s} b_i \leq n, \\ &x_j = \{0,1\}, \quad \forall j \in \Lambda, \\ &b_i = \{0,1\}, \quad \forall i \in \Gamma \end{aligned} \tag{1}$$

where the objective function $f(x_1, ...x_{N_p})$ measures the area coverage.

2.1 Sensor Acoustic Footprint and Area Coverage Function

Key for problem (1) is how to determine that a particular point Λ_j of the target space is visible when a sensor is placed at location Γ_i. It is easy to see how this, in general, can be quite complicated as it depends on a number of factors such as the acoustic environment, obstacles, robot/sensor geometry, etc. However, given a robot with a specific sensor configuration, it is possible to calculate the maximum area coverage A_k that the robot can cover in T_k hours before its battery runs out. For example, in the case of a robot using a side scan sonar with a swath width w that can travel at the maximum speed v_{max}, and work for T hours, the maximum area coverage is: $A = wTv$ (see Figure 2).

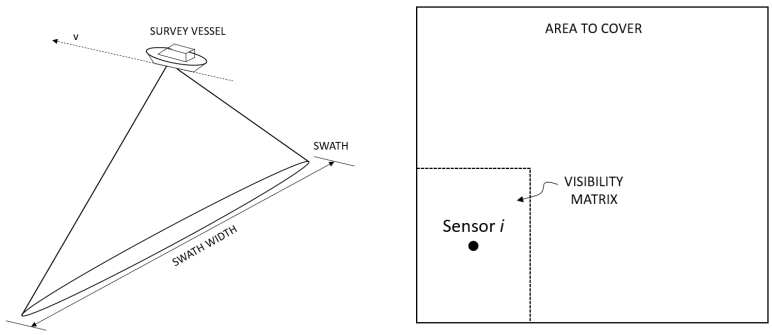

Figure 2: Left: Sketch of swath width and associated total area coverage. Right: Visibility matrix associated to a sensor located in one specific point of the area of interest.

Once each robot k is associated to its maximum coverage area A_k, the problem can be simplified, hiding most of the complexity behind a sensor visibility matrix V for any given position of the sensors (Zhao et al. (2009)). The visibility matrix can be pre-computed and each element $v_{ij} = 1$ implies that a sensor placed in Γ_i is able to insonify a target point Λ_j.

Figure 2, right, shows an example of visibility matrix associated to a sensor k positioned in one specific point of the area.

Using the visibility matrix, the final area coverage problem (1) can be reformulated as a linear problem as:

 https://doi.org/10.24868/issn.2515-818X.2020.048

$$
\begin{aligned}
&\text{maximize} \quad \sum_{j \in N_p} x_j \\
&\text{subject to} \\
&\sum_{i \in N_s} b_i \leq n, \\
&\sum_{i \in N_s} v_{ij} b_i \leq 1, \quad j = 1 \ldots N_c, \\
&x_j = \{0,1\}, \quad \forall j \in \Lambda, \\
&b_i = \{0,1\}, \quad \forall i \in \Gamma
\end{aligned}
\tag{2}
$$

where the objective function $f = \sum_{j \in N_p} x_j$ aims at maximising the overall coverage and the additional constraint ensures that there is one sensor that can insonify the target point Λ_j.

2.2 Optimising over multiple days

Problem (2) can be solved using Binary Integer Programming (BIP) (Chinneck (2015)). The output is a vehicle-area allocation for a single day of operation (i.e. up to T_i hours for robot i). Once one day of activity has been selected, the optimisation is run again with the additional constraint that some target points have been already assigned. This is done adding the result of the previous optimisation step as an additional constraint. The procedure is then repeated until the total area is covered or the maximum time allocated to complete the survey is reached. Note that this has the advantage of reducing the computational complexity, as each day is calculated independently, and makes it possible to obtain a solution even in the case where no sufficient resources are allocated, with the system providing the best possible area coverage with the available time and robots.

3 Implementation

The scheduler described in Section 2.1 has been implemented in C# and integrated with SeeByte's SeeTrack v4 (SeeByte (2020)).

The scheduler receives as inputs the areas to be surveyed, the assets that are available and their characteristics (e.g. speed, sensor type, battery endurance,etc), the maximum time available to complete the survey, and environmental information (e.g. seabed type, etc.) and outputs a vehicle-to-area allocation for each day of operation. To maximise flexibility all the information is encoded through a modular XML-based schema. During in-field operations, the output of the scheduler is then sent to the Neptune goal-based mission planner for translation into vehicle-specific executable plans.

More specifically, when a new area is received by the scheduler, it is first discretised into a grid of 1km x 1km units (user selectable). A boolean allocation matrix is then created for all the assets that are available. The optimisation algorithm described in (2) is then applied for each available day in a recursive manner, so that areas that remain unallocated are carried over to be allocated in the subsequent deployments. The result is a vehicle-area allocation per day which is again encoded as an XML file.

Figure 3 shows the full pipeline from receiving input data to producing vehicle plans using the Neptune mission planner.

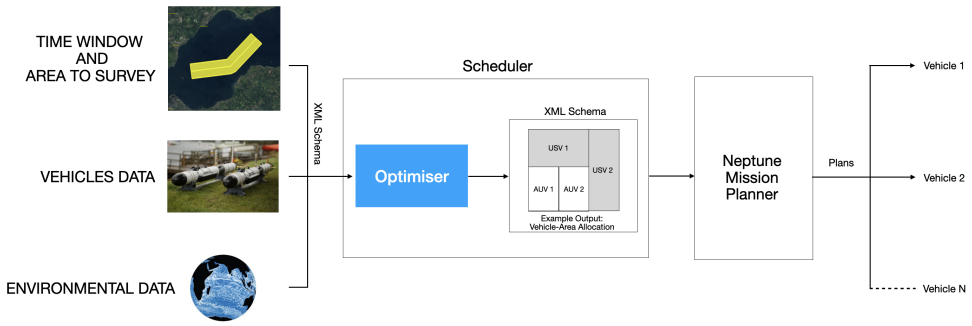

Figure 3: Scheduler workflow: the scheduler receives as inputs the areas to be surveyed, the assets that are available and their characteristics and outputs a robot-area allocation that is then sent to Neptune for day-to-day mission planning.

 https://doi.org/10.24868/issn.2515-818X.2020.048

3.1 Translating high-level goals to vehicle-specific commands

The output of the Scheduler defines a vehicle-area assignment, while Neptune translates this high-level information into vehicle plans that can be executed. To maximise interoperability and to simplify plan conversion, Neptune provides a vehicle-agnostic translation service that interprets high-level plans as vehicle-specific commands. This makes it possible to separate the specificity of a vehicle interface that might only support a limited set of commands, from the higher level mission objectives and goals. The translation service defines a consistent way to convert plans into lower level concepts that are defined at vehicle level and performs a (potentially lossy) conversion into basic elements that can be executed. A simple example would be the translation of a series of tracks into a series of "Go To Waypoint" requests. The translator serves as a bridge between existing vehicle interfaces that only supports waypoints, and higher level behaviours that issue sequences of tracks. Translations can not necessarily be reversed, and there may be more than one way to translate any one complex command. More in general, if a plan is composed of a sequence of goals: <A,B,C,D>, but a vehicle only supports commands <D,E> a translation between the two plans need to be performed. A translation rule is defined as a function $T(c) : R \rightarrow R^n$ that defines how to break down a command c. Translation rules are statically defined and they are applied when the Neptune plan is converted into a vehicle specific one. This makes it possible for operators to verify and validate the resulting plans before they are sent to the vehicles for execution.

The pseudocode of the plan translation algorithm is reported below:

```
For each command c_i of the plan P:
    Apply rule T_j and calculate a new subset of commands ĉ = T_j(c_i)
    Define the new plan P̂ = {P − c_i} ∪ ĉ
    Stop if P̂ == P otherwise assign P = P̂
```

For example, to translate the plan <A,B,C,D> into the new plan <D,E>, rules to translate each of the initial commands A,B,C,D into a combination of the target commands D,E must be applied. In this case, a valid set of rules might be: C $\rightarrow$ A, A $\rightarrow$ D, B$\rightarrow$ Z and Z $\rightarrow$ E.

The application of the translation algorithm would result in this case to the sequence of translations: <A,B,C,D> $\rightarrow$ <A,B,D> $\rightarrow$ <B,D> $\rightarrow$ <D,Z> $\rightarrow$ <D,E>.

Note the use of an intermediate command (Z) which is neither in the source set of types nor in the target set, is used to chaining rules together and makes it possible to translate complex plans.

4 Results

This section reports simulative results. In the first scenario, three assets are available, two identical UUV and one USV. The maximum speed for all the vehicles is the same and equal to $2m/s$. Each asset is equipped with a side scan sonar able to provide a swath range of about $150m$. The overall area to survey is a 4km x 8.5km, and the maximum available time is five days.

Figure 4 (left) reports the result of the vehicle-area allocation. In the figure, different colors correspond to different vehicles: UUV 1 is orange, UUV 2 is purple and USV is light red. The entire area can be surveyed in less than 5 days. The area-asset allocation calculated for each day can then be passed on to Neptune for daily executions. This is shown in Figure 4 (right) for the first day of the mission. In this case, the first three areas to be surveyed are assigned to the three available vehicles. Neptune can now define the mission in greater details specifying the track-lines that each vehicle is expected to follow during the area survey.

In a second scenario, four vehicles are used to survey the same area. In particular one additional USV is added to the group. The new USV (light blue in Figure 5) is able to travel up to a speed of $3m/s$ and it is equipped with a side scan sonar with a shorter swath width of $100m$. A different area-vehicle allocation is obtained in this case, with the two USVs covering the majority of the area. Only four days are needed to completely survey the area with four vehicles. The obtained vehicle-area allocation is shown in Figure 5 (left). As before, the first day is then passed on to Neptune for execution (Figure 5 (right)).

5 Conclusions and further research

This paper presented a new campaign planner that has been designed to plan multi-vehicles, multi-days and large area missions for autonomous vehicles. Details on the scheduling algorithms are provided, together with implementation details, including the translation from vehicle agnostic mission plans to vehicle-specific commands. This allows to abstract from the vehicle details and to maximise interoperability. Experimental validation of the proposed scheduler is planned for the next few months. This would be important to characterise the ability of the scheduler to correctly capture the actual mission time of each vehicle, and to validate the offline results. From a development perspective, additional constraints will be also added to include communication constraints, and transit times.

 https://doi.org/10.24868/issn.2515-818X.2020.048

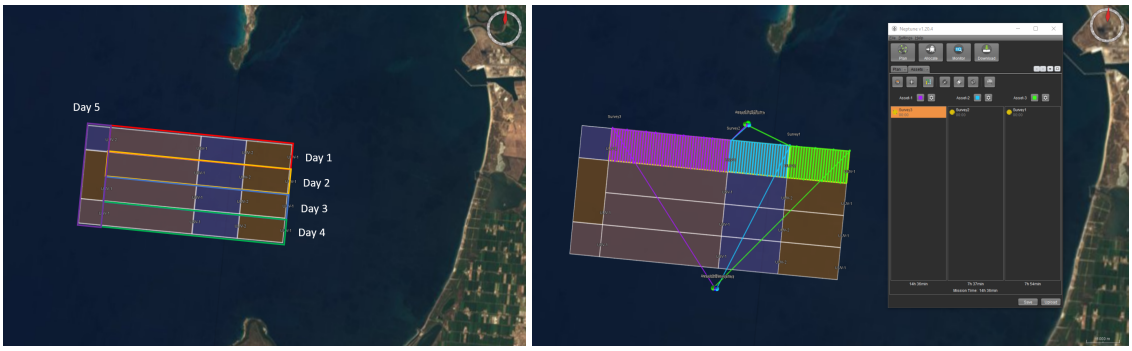

Figure 4: Left: Results of the asset-area allocation for scenario 1 as shown in the SeeTrack map. The entire area can be surveyed in less than 5 days with the available vehicles. The same color correspond to the same vehicle. White daily labels are added manually in the figure. The scheduler has a dedicated time management window which is not shown in the picture. Right: the first day of the mission is passed on to Neptune Daily Mission planner which defines the mission in greater details (e.g. track lines).

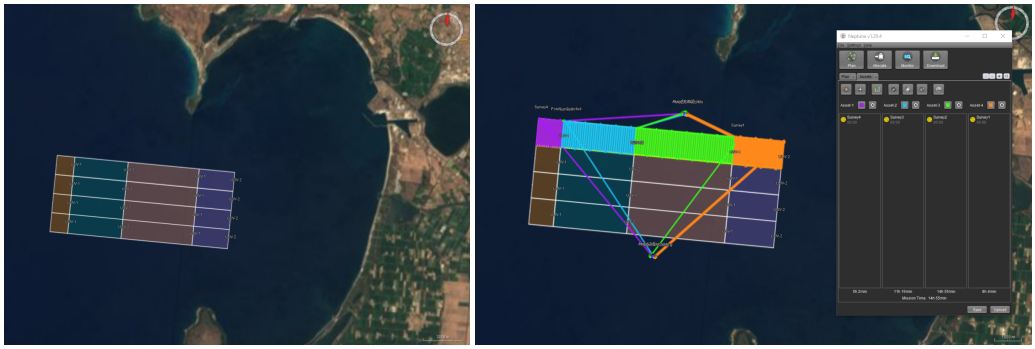

Figure 5: Left: Results of the asset-area allocation for scenario 1 as shown in the SeeTrack map. The entire area can be surveyed in less than 5 days with the available vehicles. Different colors correspond to different vehicles. Right: the first day of the mission is passed on the Neptune Daily Mission planner which defines the mission in greater details (e.g. track lines).

References

Caiti, A., Calabrò, V., Munafò, A., Dini, G., Lo Duca, A., 2013. Mobile underwater sensor networks for protection and security: Field experience at the uan11 experiment. Journal of Field Robotics 30, 237–253.

Caiti, A., Munafo, A., Vettori, G., 2012. A geographical information system (gis)-based simulation tool to assess civilian harbor protection levels. IEEE Journal of Oceanic Engineering 37, 85–102.

Chinneck, J.W., 2015. Practical optimization: a gentle introduction. Available on line at `http://www.sce.carleton.ca/faculty/chinneck/po.html`.

Ferri, G., Munafò, A., Tesei, A., Braca, P., Meyer, F., Pelekanakis, K., Petroccia, R., Alves, J., Strode, C., LePage, K., 2017. Cooperative robotic networks for underwater surveillance: an overview. IET Radar, Sonar Navigation 11, 1740–1761.

Kala, R., Warwick, K., 2011. Multi-vehicle planning using RRT-connect. Paladyn 2, 134–144.

Marques, T., Pinto, J., Dias, P., Tasso de Sousa, J., 2017. Mvplanning: A framework for planning and coordination of multiple autonomous vehicles, in: OCEANS 2017 - Anchorage, pp. 1–6.

Miguelañez, E., Patrón, P., Brown, K.E., Petillot, Y.R., Lane, D.M., 2011. Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles. IEEE Transactions on Knowledge and Data Engineering 23, 759–773.

Salavasidis, G., Munafò, A., Harris, C.A., Prampart, T., Templeton, R., Smart, M., Roper, D.T., Pebody, M., McPhail, S.D., Rogers, E., Phillips, A.B., 2019. Terrain-aided navigation for long-endurance and deep-rated autonomous underwater vehicles. Journal of Field Robotics 36, 447–474.

SeeByte, 2020. SeeByte Website. `www.seebyte.com`. Accessed: 2020-05-14.

Wu, J., Bingham, R.C., Ting, S., Yager, K., Wood, Z.J., Gambin, T., Clark, C.M., 2019. Multi-auv motion planning for archeological site mapping and photogrammetric reconstruction. Journal of Field Robotics 36, 1250–1269.

Yordanova, V., Griffiths, H., Hailes, S., 2017. Rendezvous planning for multiple autonomous underwater vehicles

 https://doi.org/10.24868/issn.2515-818X.2020.048

using a markov decision process. IET Radar, Sonar Navigation 11, 1762–1769.
Zhao, J., Cheung, S.c.S., David, H., 2009. Optimal visual sensor network configuration. Multi-Camera Networks Chapter 6.
Zhao, J., Cheung, S.c.S., David, H., 2013. Approximate techniques in solving optimal camera placement problems. IEEE International Journal of Distributed Sensor Networks .

 https://doi.org/10.24868/issn.2515-818X.2020.048