Analysis and Design of an Intelligent Oceanographic Probe with Autonomous Sampling Capabilities.

Report on Project ANERIS: Subproject ANERIS-L

Albert Vilamala, Enric Plaza, Josep Lluis Arcos
IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB, E-08193, Bellaterra, Spain
{vilamala, enric, arcos}@iiia.csic.es

Editor: IIIA-CSIC

Abstract

Recent studies in marine regions have determined the existence of phytoplankton and zooplankton assemblages near the surface level. Such patches, named thin layers, are of interest for biologists due to their important impact on the ecosystem where they lay in, such as phytoplankton growth dynamics or predation, just to name some. Therefore, new tools to ease the task of biologists are being under development.
The purpose of this project is to develop a system capable of detecting and characterizing thin layers from hyperspectral optical sensors on a first phase and, secondly, use this information to make wise decisions about the best locations where retrieve water samples.
Specifically, a Particle Swarm Optimization technique has been used to detect thin layers and the Case-Based Reasoning methodology, which takes advantage of previous experience in order to adapt it for new situations, has been applied to successfully characterise them. Moreover, an extended Case-Base Reasoning has also been used to predict the places where thin layers are present.

Keywords: Case-Based Reasoning, Particle Swarm Optimization, Hyperspectral Optical Sensors.

1. Introduction

For many years, the observation of the flora and fauna in oceanic environments carried out by biologists was directly performed in the field by sampling and posterior examination in a laboratory. This approach was not only costly and time consuming for the researchers but it could also change the environment of interest due to the invasive nature of such traditional techniques, increasing the difficulty of this sampling which would even lead the project to failure if those samples were not gathered with caution.

However, in the last few years, advances on hyperspectral optical sensoring have broadened up the possibilities for these biologists to proceed with their examinations in a more neat way. These new techniques take advantage of the optical properties present in the matter that can be found in underwater systems allowing for the monitoring of their dynamics.

As a result, new tools able to work in-situ without much environment disturbance are being developed, with the advantages that those features represent. Phytoplankton is one of the most recurrent components that are sought using optical sensors. The reason for this fact is the well-known optical property of its main component —the chlorophyll— when daylight comes into contact with it.

On the other hand, increasingly more biologists believe in the presence of thin plankton patches near the surface of marine regions contributing to important impacts on the ecosystems where they have been found. This fact has grown the interest of the community in performing research.

The current research project arises from coupling the two concepts explained above in an attempt to use the new techniques available in the field of marine sampling applied to identify optical significant thin phytoplankton assemblages, namely *thin layers* (Dekshenieks et al., 2001; Stramska and Stramski, 2005).

This project report describes the study of two Artificial Intelligence techniques applied on noisy hyperspectral optical data from marine water columns for detecting and characterising thin layers. Moreover, a technique to build an online decision component able to continuously identify the best locations of interest in a water column is also examined. The final purpose of those studies is to embed these techniques in a prototype autonomous oceanografic probe, which should be able to identify thin layers and recollect water samples from proper places.

In order to accomplish the main purpose of this project, a system composed of two big modules has been developed. The first one is focused on the probe's descent; that is, identifying thin layers when the probe is sinking. In order to approach this challenge, we can use complete data.

The second big module is in charged of making online decisions about where to retrieve the water samples during the probe's ascent. Notice the incompleteness nature of data available for such purpose, since the decisions must be made while the probe is passing through.

More precisely, the first module performs in a two step basis: the first phase locates such layers using Particle Swarm Optimization and the second one uses Case-Based Reasoning to determine some of their different features. The main challenges these techniques must overcome are: first, the capability of properly work on data highly affected by noise and second, due to the early stage of the field under treatment, the difficulty of building a tool wise enough to confront its duties using the currently shallow existing knowledge in the domain.

The second module uses an extended Case-Based Reasoning technique, which is not only able of reusing previous experiences to decide the best places to recollect samples, but also takes into account the local information gathered during the probe's descent in order to improve its capability.

According to the ANERIS' project definition, the aim of the subproject presented in this report can be defined by its main objective, which in turn, is split into 8 different milestones. All these milestones are desirable to be attained in order to achieve the main objective.

Main objective: *Analysis of Artificial Intelligence systems for high level decision making.* From a database of simulated scenarios, a study regarding the feasibility of learning and

data mining techniques in order to perform a high level decision making and data retrieval will be performed.

1. *Analysis of requirements from data generated by the simulator.* The possibility of directly working with data generated by the simulator or applying preprocessing techniques will be exposed.

2. *Analysis for determining two crucial issues*: a) the proper set of reduced feature attributes for an automatic learning system, and b) the typical scenarios or situations which require a specific distribution class of recollection data levels.

3. *Exploration of possible learning and data mining techniques to be used over data* to determine: a) typical situations, b) their applicability range, and c) a reduced vocabulary for describing the situations.

4. *Selection of a learning technique L* in order to acquire the data decision strategies.

5. *Analysis of the learning method L usage in simple scenarios* (e.g., unique observable layers and discrete regarding depth).

6. *Analysis of the learning method L usage in a more complex scenarios.* The sufficiency of the set of scenarios described in activity 5 will be validated and its possible extension studied.

7. *Revision and generalization of the learning method L for more realist scenarios imitating the general probe's cycle.* In this case, the data observed during the probe's ascent will be deviated regarding the observed during the probe's descent. The performance will be assessed regarding the results obtained by the prototype from objective 6 and respect to the cost in calculation time.

8. *Integration of the learning method L code with the automatic probe prototype code.*

This monograph is composed of the following sections: first a brief overview on oceanographic concepts and sensors is carried out to introduce the problem to solve; next, some cutting edge AI methods are reviewed followed by a section where these techniques are applied and deeply explained to approach the identification of thin layers using complete data; later, the experiments to evaluate this module performance are commented. Then, the problem of making an online decision component is presented, the techniques applied are explained and evaluations of its suitability are performed. Finally, some conclusions are stated as well as new future research lines.

2. Marine background

This section introduces some concepts on oceanography such as the definition of the term thin layer and a review of the hyperspectral optical sensors currently available, together with the theory that supports them.

2.1 Thin layers

Thin layers are plankton assemblages present in water column near the surface level, usually in the photic zone in coastal waters (Sieburth and Donaghay, 1993; Johnson et al., 1995; Carpenter et al., 1995). They are composed by either phytoplankton or zooplankton, or a mixture of them. It has been observed that their thickness[1] may range from approximately

1. Thickness is the distance between the thin layer's upper level and the bottom one.

10cm up to nearly 3.5m (Donaghay et al., 1992; Cowles and Desiderio, 1993; Hanson and Donaghay, 1998).

As stated in Nielsen et al. (1990); Bjørnsen and Nielsen (1991); Rines et al. (2002), their horizontal size is very variable and it can even extend for several kilometres. Regarding distribution, it varies among space, meaning that different compounds can be found at different places within a thin layer. This same variability is found in the time dimension, that is, thin layers may persist for several days in a certain location.

Such structures have an important impact for the ecosystems where they live in, since they may alter the biological structure and dynamics of the system (Nielsen et al., 1990; Donaghay and Osborn, 1997; Cowles et al., 1998), as well as its acoustical (Holliday et al., 1998) and optical (Zaneveld and Pegau, 1998) properties.

According to Dekshenieks et al. (2001), thin layers occur more often in certain locations than others, related to physical conditions which are still not known in depth. Nevertheless, even lacking a strong theory about those conditions, it is known that in turbulent waters such as tidally mixed regions or those having mixed wind on surface, thin layers do not happen or very rarely. Contrarily, they are often located in pycnoclines, according to its depth and strength.

2.2 Radiative transfer theory

The sensors presented in this project take advantage of the optical properties of each element in matter existing in the water column that is to be examined. In particular, they are based on the *radiative transfer theory*.

Radiative transfer theory explains how a sunlight ray that has come into contact with a surface is modified according to the nature of the matter it has impacted. That is, each compound owns a set of properties called *inherent optical properties* (IOPs), which depend exclusively on the target matter itself and is completely independent from the light field. Hence, whenever a light beam comes into contact with an element, this light can be either *absorbed* by the matter or *scattered*. The addition of these two components results in what is known as *attenuation*. Such properties are measured in form of coefficients directly tied to the element.

On the other hand, if the observer does not focus on the IOPs but the resulting light field after the light beam has come into contact with the matter, the *apparent optical properties* (AOPs) arise from the properties of this light. In that sense, we can distinguish the *irradiance*, either downwelling (E_d) or upwelling (E_u) (i.e., whether the examined field is on top of the matter of observation or under it), the *reflectance*, which is the ratio between E_d and E_u and the *diffuse attenuation coefficient* (K), being the difference of two consecutive measures of E.

Despite IOPs and AOPs are the main properties used by hyperspectral optical sensors to characterise the environment, matter has other optical properties worth to mention such as *fluorescence*[2] or *luminescence*[3]. These properties can also be measured using optical sensors, but they will not be set in this project though, given the fact that not all the matter we work with owns them.

2. Fluorescence is the property of re-emitting the absorbed light in a different wavelength.
3. Luminescence is the property of releasing energy in form of light.

An important point regarding IOPs and AOPs is their capacity to be used as proxies to determine the quantity of certain matter in a water column. Thus, this phenomenon will be applied in this project to detect and characterise thin layers.

2.3 Hyperspectral optical sensors

Hyperspectal optical sensors make the most of the radiative transfer theory to successfully perform their task. The main feature of those sensors, used by the literature to classify them, is whether they own a light source or not. The performance of each sensor type and the target optical properties depend on this distinct feature. Such classification is done by splitting them up into two main groups namely *active* and *passive* sensors.

Considering Perry (2003), active sensors are the ones that own a light source —either a LED or a lamp— in order to accomplish their function, which normally is to measure IOPs and fluorescence. The advantage of this kind of sensors is that they are capable of measuring in very deep waters, where the sun light barely arrives or not at all. However, their price is a bit more expensive compared to passive sensors and the power consumption is also higher.

Passive sensors are characterised for lacking an emitting light source since their task is performed by using the environmental light provided by the sun. They are mainly used to measure AOPs and bioluminescence. In contrast to active sensors, these ones can only work within the photic zone (i.e., $10m$ to $150m$ under the sea level) and only during daylight. The good point is that they are cheaper and they need less power consumption to work. These two facts are strong enough for deciding to use these sensors instead of the active ones for the current project.

3. State of the art

This section presents two main Artificial Intelligence techniques that have been applied in the current project development from their general form to the different contributions that researchers have done. More precisely, an evolutionary technique called Particle Swarm Optimization and a machine learning and reasoning approach called Case-Based Reasoning will be introduced.

3.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a technique that aims at finding local maxima solutions in a search space by combining a cognitive model with a social interaction model. This technique is inspired from the natural model of a flock of birds searching for corn. Even though each one of the birds has its individual purpose, they interact with each other in order to achieve a certain common goal. These birds and interactions among them follow specific rules. The next subsection introduces the PSO algorithm.

3.1.1 Basic algorithm

The PSO basic algorithm —developed by Kennedy and Eberhart (1995), who were influenced by the work done by Heppner and Grenander (1990)— is defined as a set of *particles* randomly initialised over a search space that keep moving around searching for local max-

ima until they have converged. At each algorithm iteration, the particles compute a fitness value which assesses the current position.

Notice that the particles are not moving blindly in the search space, but they are heading towards the best position that they know from their own experience. To achieve this behaviour each particle needs to keep information about its current position, its current velocity and the best position it has found so far. This set of features has been called the *cognitive model*, according to literature.

Furthermore, the *social model* also plays its role in order to improve the performance achieved by the self-driven particles. Specifically, the particles swarm keeps the global best position —according to the fitness function— its members have found so far. Notice that each particle communicates with the particles in its neighbourhood and is driven according to the best position found. While a single particle may not be able to find best global positions, the interaction among the particles of a swarm is what gives power to this algorithm.

The original PSO equations to update the velocity and position of each particle is the following one:

$$\vec{v_i} = \vec{v_i} + \vec{U}(0, \phi_1)(\vec{b_i} - \vec{p_i}) + \vec{U}(0, \phi_2)(\vec{g} - \vec{p_i})$$

$$\vec{p_i} = \vec{p_i} + \vec{v_i}$$

where,

$\vec{p_i}$ is the current position of the particle i;
$\vec{v_i}$ is the velocity of the particle i;
$\vec{b_i}$ the best position found by the particle i;
$\vec{g}$ the global best solution found by the particles; and
$\vec{U}(0, \phi_i)$ represents a vector of random numbers uniformly distributed in $[0, \phi_i]$.

However, research on this preliminary PSO algorithm leaded Shi and Eberhart (1998) to increase the algorithm performance by introducing a new parameter called *inertia weight*. Such parameter allows to have deeper control on the search scope and downplays the importance of the maximum velocity (V_{max}) —in the original PSO, each $\vec{v_i}$ is kept within [$-V_{max}$,$+V_{max}$]. Intuitively, inertia weight parameter may be seen as the fluidity of particles on the search space. Therefore, a good strategy is to give high values to this parameter at the beginning of the algorithm in order to rapidly reach the regions of interest in the search space and then gradually lower the value of this parameter in order to focus on a specific space region.

Finally, Clerc and Kennedy (2002) proposed an improved method which is the canonical PSO known nowadays. In their study, they proposed to introduce some *constriction coefficients* which are useful to ensure fast convergence and eliminate the domain-specific maximum velocity parameter. One way of implementing the constriction coefficients is by adding the constant multiplier χ (Equation 1). According to their experiments, best results are achieved when χ is set to 0.7298 and the *acceleration coefficients* (ϕ_i) are bounded between 0 and 2.05.

Formally, the movement of every particle in the canonical PSO is driven following the next two equations:

$$\vec{v_i} = \chi(\vec{v_i} + \vec{U}(0,\phi_1)(\vec{b_i} - \vec{p_i}) + \vec{U}(0,\phi_2)(\vec{g} - \vec{p_i})) \quad (1)$$
$$\vec{p_i} = \vec{p_i} + \vec{v_i}$$

where χ is the constant multiplier that ensures the convergence.

In order to successfully work with the PSO algorithm, according to Poli et al. (2007), there are several settings that are crucial. One of them is the population size (i.e., the global number of particles) which is usually experimentally set. There is also the need to focus on ϕ_i, which are used to balance the forces between the particle best position attraction and the global best position attraction. Choosing the proper initial position of each particle and its initial velocity is also a vital decision.

Another important factor is the selection of particles to update at each iteration. Standard PSO updates all particles in turn, but other strategies just select some of them to be updated (e.g., the best particles in order to easily reach its purpose or the worst ones to improve the performance of the system as a whole).

Also, particle dynamics is another issue that researchers pay attention to. Different approaches such as spherical particles, sliding particles, quantum effects or repulsion among particles have been applied with varied results. Moreover, sometimes particles move to a region where there is nothing of interest. In this case, methods to drive the affected particles out of this region are needed, such as moving the particles to the edges or re-initialising them.

The memory of particles is important as well. In the standard PSO, every particle has its own record about the best position and takes into account the global best position found. However, different approaches have been proposed. For instance, using just some of the surrounding particles' memory to calculate the best maxima in the neighbourhood —the neighbouring particles might be fixed or might change dynamically.

Finally, improvements in the field of swarm adaptation —self addition or suppression of particles and parameter tuning— have recently being performed. Ongoing research along this path aims at finding simple systems able to self-adapt for optimality.

3.1.2 PSO variants

Using the canonical PSO explained above as the starting point, many researchers have applied different strategies for improving the overall PSO performance or redesigning it for specific purposes. We present the most relevant ones for our current work.

The traditional attempt is the so-called *global best* topology, where all the particles follow the single best position found by all the particles searching in the space. In contrast to this idea, there is the *local best* approach (Eberhart and Kennedy, 1995). Here, every particle is only influenced by the n closest particles, leading to a successful algorithm which searches for multiple local maxima at a time. A third algorithm is proposed by coupling these two techniques —start the algorithm using local best and changing it to the global search at the end of the run (Suganthan, 1999).

In Mendes (2004) PhD thesis, the author presents the *fully informed particle swarm* (FIPS). Using this technique, each particle is influenced by a stochastic average of all the neighbours. The results presented report fast convergence of particles, implying that the

solution can be found more rapidly than using canonical PSO. As a drawback, Mendes points out the high dependency on the particles topology.

Different methods for searching the best neighbours that should drive a given particle include computing a *distance ratio*, which compares the fitness of the current particle with the fitness of the neighbours, together with a weighted Euclidean distance (Peram et al., 2003); selecting *random size subpopulations* of neighbours (Liang and Suganthan, 2005); organising the particles in a *hierarchy-like* structure where the most influent particles move up the hierarchy driving the particles that lay beneath them (Janson and Middendorf, 2005) or splitting the particles into several differently sized *tribes*, where the weakest particles in a competing tribe can be expelled and then they can join other less competitive tribes and vice-versa (Clerc, 2006).

Other research lines have pushed researchers to borrow techniques from several AI areas. Examples are Angeline (1998), who used *genetic algorithms*; Wei et al. (2002) who computed the velocity of each particle by means of an *evolutionary algorithm* reporting very good results; and Robinson et al. (2002) who switched from PSO to a genetic algorithm as the execution was being performed.

More research was conducted coupling PSO and *hill-climbing* (Poli and Stephens, 2004); PSO and ACO^4 (Hendtlass, 2001; Holden and Freitas, 2005); and PSO and *genetic programming* Poli et al. (2005a,b).

Just before finishing this section, special attention must be paid to the *multiple particle swarm optimization* (mPSO). This technique presented by Blackwell and Branke (2006), proposes to use several swarms to search for multiple ever changing peaks —in location and height.

The existence of multiple swarms rambling in a search space, approaching different peaks, ensures the capability of the algorithm to find different local maxima, eventually finding the global maximum, avoiding to get stuck in a particular local maximum.

Moreover, defining an *exclusion operator* is a must in order to avoid several swarms approaching the same peak (Fernandez-Marquez and Arcos, 2009). This operator defines an exclusion radius that surrounds the current local maximum found by the swarm, keeping the particles from other swarms out of this circle. Also an *anti-convergence operator* is needed to keep two particles far enough for ensuring diversity in a given swarm.

It is worth mentioning the importance of choosing the proper swarm size (i.e., the number of particles that the swarm is composed of).

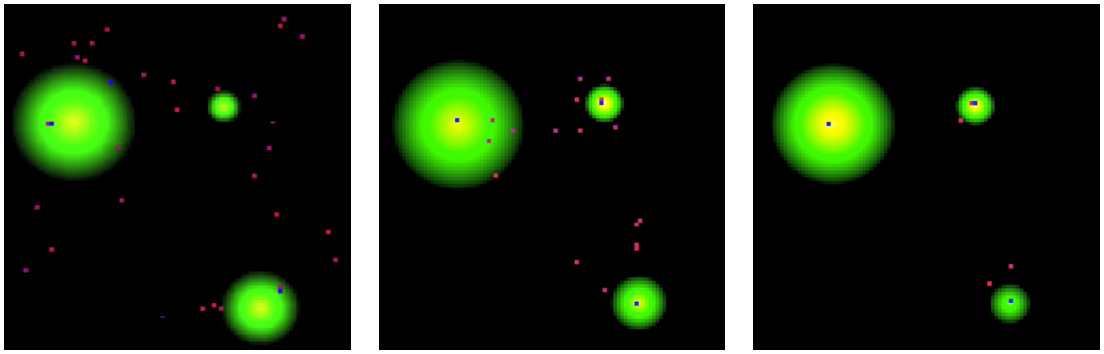
A snapshot of a working mPSO can be seen in Figure 1, where three swarms of ten particles each search for the three peaks existing in a sample domain.

3.1.3 PSO in noisy environments

PSO has been shown to be a robust method in front of noise according to Parsopoulos and Vrahatis (2001), who tested the algorithm by applying Gaussian distributed random noise to different well-known optimisation fitness functions.

Many researchers have attempted to improve PSO performance when influenced by noise: Pugh et al. (2005) propose a variant where each particle computes multiple evaluations for the same candidate solution to assess a fitness value. Despite increasing the

4. Ant Colony Optimization was initially proposed by Dorigo (1992).



(a) (b) (c)

Figure 1: Evolution of mPSO in a three-peak environment.

algorithm performance, this method incurs in a higher computational cost when performing such multiple evaluations.

Other methods use a statistical sequential selection procedure to improve the accuracy of the function estimation (Bartz-Beielstein et al., 2007). Another one, called Partitioned Hierarchical PSO (Janson and Middendorf, 2006), organises the swarm neighbourhood in a tree hierarchy that is monitored to detect possible changes in dynamic and noisy environments. And finally, Fernandez-Marquez and Arcos (2009) use an evaporation mechanism to approach similar environments.

3.1.4 Applications

PSO in all its forms has been applied to a large number of real applications of different nature. According to a survey conducted by Poli (2007), who took the IEEE Xplore database in 2007 as a source of information, nearly 700 papers were of PSO applications. The most relevant categories which PSO has been applied are image and video analysis (7.6%); design and restructuring of electricity networks and load dispatching (7.1%); control applications (4.0%), applications in electronics and electromagnetics (5.8%); antenna design (5.8%); power generation and power systems (5.8%) and scheduling (5.6%). Other applications include from robotics to clustering passing through biological and medical applications, prediction and forecasting, fuzzy systems or music generation and games.

3.2 Case-Based Reasoning

Case-Based Reasoning (CBR) (Kolodner, 1993; Leake, 1996; Mantaras et al., 2005) is a *lazy learning* technique (i.e., stores instances in the learning step, delaying the induction for the problem solving phase) that uses experience from problems tackled in the past in order to seek for solutions that fit the current problem to be solved. CBR is based on the assumption that similar problems can be solved by reusing similar solutions.

3.2.1 CBR Basics

CBR defines a *case* as the item of information that contains both the *problem* to be solved and the *solution* to overcome such problem. Sometimes a case might also contain a value assessing the 'goodness' of the proposed solution.

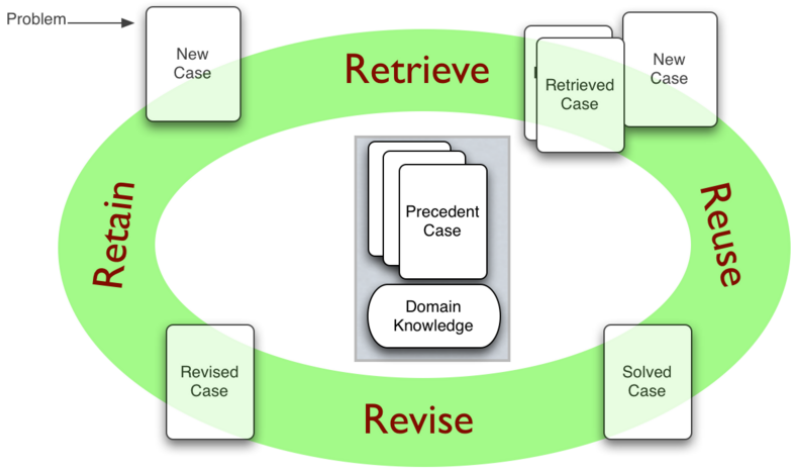


Figure 2: Case-Based Reasoning cycle

Upon time, different problems in a specific domain are faced and solved, storing this information as cases in a *case base*. Once this case base is informative enough, the system can start using this experience to solve new problems by applying the CBR cycle (Fig. 2), which is made up of 4 processes:

First, in the *retrieve* process, a set of cases whose problem is similar to the problem at hand are sought in the case base and retrieved to be used as possible solutions.

Next, the *reuse* process comes into play. The purpose of it is to adapt each one of the cases' solution to fit the current problem, select the best one and actually apply it to solve the problem.

Assessing the performance of the solution applied to the current problem is called the *revise* process. This can be done either manually or automatically, depending on the domain.

Finally, the *retain* process is performed. Here, if the solution applied to the current problem is useful to be taken into consideration in the future —for its goodness, or on the contrary as a counterexample— the pair is stored in the case base in form of a new case.

The CBR cycle is repeatedly executed and its performance, in principle, increased over time (i.e., the more cases it stores, the better it performs). Nevertheless, different modifications have been applied to this cycle to fit the specific needs of every situation.

3.2.2 Foundations

CBR stands in the junction between several fields as different as *cognitive science*, some AI subfields such as *representation*, *reasoning* and *machine learning* as well as some influence of *mathematics*. In this section we review the contributions of each field in the foundations of CBR.

As explained in Mantaras et al. (2005), CBR deals with capturing and using specific experiences, borrowing some knowledge from cognitive sciences, mainly those involving the words experience, memory and analogy.

The work done by Tulving (1972) made a distinction between episodic and semantic memories in human reasoning, which coupled with the Smith and Medin (1981) exemplar view —a concept is defined as the set of exemplars that conform it—, together with the dynamic memory theory (Schank, 1982) —memory is organised in packets— set the basis of CBR.

Moreover, further research noticed that even though CBR shares many properties with human analogical reasoning, there is a big difference among them: while human analogical reasoning reasons on several domains, CBR reasons only on one specific domain.

Another computational model of human analogy reasoning worth to mention for its impact on CBR is that of Falkenhainer et al. (1990).

CBR has taken some techniques from knowledge representation and reasoning in AI rather than developing specific ones for its purposes. Among them, examples are *description logics* (Díaz-Agudo and González-Calero, 2001), *feature terms* (Plaza, 1995), and some well-known *knowledge representation techniques* (Koton, 1988; Aamodt, 1994).

It is also important to use efficient data structures and indices to properly store case information and retrieve cases in a fast enough way. For such purpose, literature explains that techniques used in the database field have been adopted.

CBR has made the most of machine learning techniques to deal with its needs regarding *concept learning*, that is, inducing a general concept from a set of positive and negative examples. According to Kibler and Aha (1987), concept learning can be achieved by storing the exemplars presented to the learner and whenever an unknown example is found, assign to it the concept class that better matches a specific criterion.

Mathematical methods are broadly used in CBR, mainly aiming at computing *similarity measures* and *learning*. Measures are mathematical functions to assess utility. The main class of similarity measures are linear functions with several domain specific coefficients.

To sum up, CBR is made of several different-nature well-established fields, but further research that studies the relationship among the disciplines should be done for a major understanding and improvement of the techniques used in CBR.

3.2.3 Representation

In 3.2.1 the basic composition of a case —a problem and a solution— has been explained. However, Kolodner (1993) proposes five pieces that, from her point of view, a case must contain: a *situation* and its *goal*; the *solution* and how it has been *derived*; the *result* of carrying it out; *explanations* of the results; and *lessons* that can be learned from the experience.

According to Bergmann et al. (2006) traditional ways to represent cases in CBR fall into three main categories: *feature vector representations*, *structured representations* and *textual representations*. Furthermore, some advanced and task specific approaches have also appeared.

Feature vector representation consists of a set of exemplars that represent a category. Therefore, different categories are represented by their particular exemplars. A new case

is classified to belong to a category whenever the case that best matches the current one belongs to that category. An example can be appreciated in the PROTOS system (Porter et al., 1990).

Plaza (1995) explains how cases can be represented using feature logics, a form of *frame-based representations*.

Borrowing ideas from data modelling, *object-oriented case representations* mimic the object-oriented paradigm to represent cases as a set of objects (Bergmann, 2002).

Finally, the *textual representations* (Lenz and Burkhard, 1996) use documents or texts as cases. In this approach a case is built from text by analyzing it to find the relevant information entities. These cases are organised into a retrieval net whose nodes are linked according to their similarity.

More sophisticated approaches include using *hierarchical representations* where different levels of abstraction are used to represent a case; *generalised cases* where a case covers a subspace of the representation space rather than a single point in that space; and specific representations for particular tasks such as *planning* (e.g., CHEF by Hammond (1989)) and *design* (e.g., the FABEL project by Gebhardt et al. (1997)).

3.2.4 Retrieve

The retrieval phase of CBR has traditionally focused on resemblance of cases by calculating similarity between problems, since it is based on the statement that similar problems have similar solutions. Nonetheless, some authors have questioned this assumption.

Some approaches assess the similarity between cases by means of their *surface* features. This includes computing a similarity measure and picking the k most similar cases[5]. Another way to compare two cases consists in computing the similarity of every feature and finally calculating the global similarity measure by weighting the partial measures. Smyth and McKenna (1998, 2001) proposed a method that is made of two stages: the first one retrieves a subset of possible cases —using a lightweight similarity measure— and the second one refines the search by actually retrieving the interesting ones.

Other authors have reported results on measuring *structural similarities* between cases in certain domains. An example of it is the work conducted by Börner (1983), who defined the structural similarity as the most specific graph structure that two problems have in common and a set of transformation rules needed to determine this structure. Similarity among hierarchical-structured cases can be assessed by searching the cases that are closer in the hierarchy.

Brown (1994) proposed to represent case memory as an interconnected network of nodes capturing case attribute-value combinations.

Bunke and Messmer (1994) presented similarity measures for domains where cases are represented as graphs. They proposed to use graph editing operations as the metrics to calculate the similarity.

Sometimes computing a similarity measure for every pair of cases on runtime is not feasible. For those situations it is wise to store similar cases close to each other and index them.

5. k-nearest neighbour (Cover and Hart, 1967).

In order to improve similarity measures, literature provides us with different attempts: The measure can be amended by applying an *adaptive learning process* or on the contrary, by using *knowledge-intensive* measures defined by a domain expert. Nevertheless, the designer of the system must evaluate the pros and cons of each approach.

Right now, the focus has been on similarity measures that aim at comparing the problem part of cases. Notwithstanding, there exist other strategies, as the one presented by Bergmann et al. (2001) who argues that not only the most similar cases to the target case should be retrieved, but the usefully similar. Following the same line Smyth and Keane (1994, 1995a, 1996, 1998) explain that sometimes the most similar cases are also the most difficult to adapt. Therefore, the retrieval should pay attention to this fact and retrieve those cases similar enough and at the same time easy to adapt. Leake et al. (1997) propose an approach to this problem with a system that uses CBR to predict adaptation effort.

3.2.5 Reuse

Reuse is the stage where previous experience of solving a problem is adapted and applied to the problem at hand. In some scenarios such as classification, there is no need to adapt the solution but it is directly applied. In some others, though, adaptation is a must. The methods to commit this purpose can be split into three main categories: *substitution* adaptation, where some parts of the retrieved solution are reinstantiated; *transformation* adaptation, which alters the structure of the solution (Kolodner, 1993); and *generative* adaptation, that replays the method of deriving the retrieved solution on the new problem.

CHEF (Hammond, 1990), a menu-planning system, is a clear example that applies both substitution and transformation adaptation in its reuse phase. The former is used to substitute ingredients in the retrieved recipe to match the ingredients required in the target menu, whilst the latter is used to add or remove steps in the proposed recipe due to the ingredient substitution.

Gómez de Silva Garza and Maher (2000) proposed to use evolutionary methods for the adaptation phase in an architectural design system. According to their approach, the retrieved designs are the initial population for a genetic algorithm. Mutation is a substitution adaptation that randomly alters part of the design and crossover is a transformation adaptation that alters the structure of a design.

Generative adaptation has been applied to Prodigy/Analogy (Veloso and Carbonell, 1994), a general purpose planning system. In this system, when the retrieved solution presents a faulty element, the system recalls how the element was computed and replays a calculation for the new problem.

3.2.6 Revise

The phase where the solution proposed by the CBR system is evaluated is called the revise process. It is usually conducted by an external agent, because the purpose of this process is not only determining the correctness of the proposed solution but finding a correct solution in case the system has obtained an incorrect solution. It is important to notice that, by doing this, the system is able to acquire new knowledge which might be useful in the future.

In most cases, revise is manually done by a human expert on the domain. In other scenarios, a simulator in a virtual environment is in charge of evaluating the solution and

providing an alternatively correct one, if needed. The last option is to directly apply the proposed solution to the real world and check its consequences.

3.2.7 Retain

This is the last process in the CBR classic cycle, where the knowledge gained from the last treated episode is incorporated to the system. While some systems straightforwardly keep the problem specification and the solution applied, others retain a more in depth detailed information regarding the last target case, such as how well the solution suited the system's goal (Goel et al., 1991) or how the solution was derived from the problem specification (e.g., Veloso and Carbonell (1994)).

It is important to realise that, even if at first sight, storing as many cases as possible seems a good approach —the more cases stored, the more the coverage and hence, the less the cost of adaptation— it might not be the case, since as the case base grows, the retrieval time also increases. Therefore, there is a need for a good trade-off between these two.

Specifically, when growing the case base, new cases are more probable to overlap with the existing ones offering very little improvement on adaptation but significantly increasing the retrieval cost. From then on, the performance of the system starts to degrade.

Different approaches include selecting the proper cases to add into the case base by eliminating directly redundant cases, eliminating noisy cases or even using a metric to assess the cost of adding a case compared to its expected savings (Minton, 1990). Even more, other authors propose to properly select some cases as representative of a particular subspace (Smyth and Keane, 1995b).

To conclude this section, notice the importance of maintenance in CBR systems. Along the system lifetime many different sources might degrade its performance. It may be due to the fact that indexes are out of date and ought to be updated (Muñoz-Avila, 2001; Craw et al., 2001); some cases that were relevant in the past are no longer needed; etc. Consequently, a maintenance policy is very important for the success of the system. Maintenance polices may be triggered *periodically* (e.g., at every case addition), *conditionally* (e.g., when retrieval time increases to a pre-specified threshold) or *ad-hoc* (e.g., by a random human maintainer). Reinartz et al. (2001) propose to add two more phases in the classical CBR cycle named *review* —to monitor the quality of the system knowledge— and *restore* —to select and apply maintenance operations.

3.2.8 Time-aware CBR

Case-Based Reasoning has commonly been used to solve problems that can be described using discrete representations on a particular moment of time. However, few studies have approached those problems claiming for a different treatment due to the continuous nature of the time dimension. Among them, *Continuous CBR* (Ram and Santamaría, 1993) uses continuous representations to constantly reasoning and adapting the system according to the current performance, applied on the autonomous robot navigation domain.

More recently, Martin and Plaza (2004) have extended this idea of CBR in continuous environments by proposing a system able to deal with a sequence of interleaving events simultaneously occurring from many different sources. *Ceaseless CBR* has been applied in

the intrusion detection domain aiming at identifying the responsible sources of real threats when alerts are fired.

4. Analysis using complete data

The first of the two main blocks of the system focuses on the identification of thin layers during the probe's descent. The data analysis is performed at the end of the probe's sink, when all data corresponding to the descent have been already measured (complete data). The information gathered will be used by the second block to make wise decisions regarding the best places to catch samples. This section explains all about identifying thin layers during probe's descent.

4.1 Approach

Identifying thin layers is carried out by a two step approach: The former is *detection*, where the depth at which the thin layer is located has to be determined, and the latter is called *characterisation*, that aims at finding the algal group which the target thin layer belongs to as well as predicting its real thickness and concentration (Fig. 3).

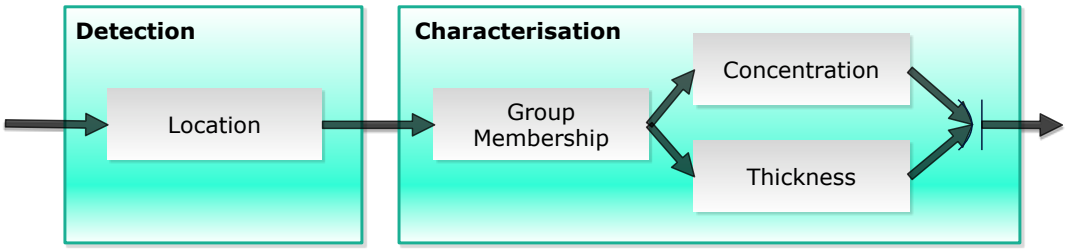


Figure 3: Thin layer identification approach.

The first challenge we face when dealing with the detection step is the large dimensionality of the data to be treated. Therefore, we must select an algorithm that exploits the nature of the data efficiently, leading to detect the location of the main thin layers. Moreover, we already know that data supplied by sensors might suffer from highly noise levels. According to literature, Particle Swarm Optimization is a technique able to successfully meet all these requirements.

Regarding the characterisation phase, it can be split into two sub-steps. The first one is determining the algal group membership of the main alga present in the thin layer. Then, the second one is calculating the thin layer thickness and concentration. The first step is especially crucial for the correct behaviour of the system because algal type directly determines the meaning of the hyperspectral optical data. For instance, the same value of K_d might mean a high level of algal concentration in a thin layer for a certain algal group, whereas it might mean a very low concentration level for another group.

Another important factor that must be taken into consideration for the design of this system is the lack of information about the domain at hand. For this reason, it is very

important to build a flexible system capable of adapting to unseen phenomena. Case-Based Reasoning is a technique that perfectly tackles the mentioned challenges with the advantage that can incorporate these new phenomena by learning from its own experience.

In this section we present the data we have used for building the system and explain the different decisions that we have taken to apply the before-mentioned AI techniques to fit our specific domain.

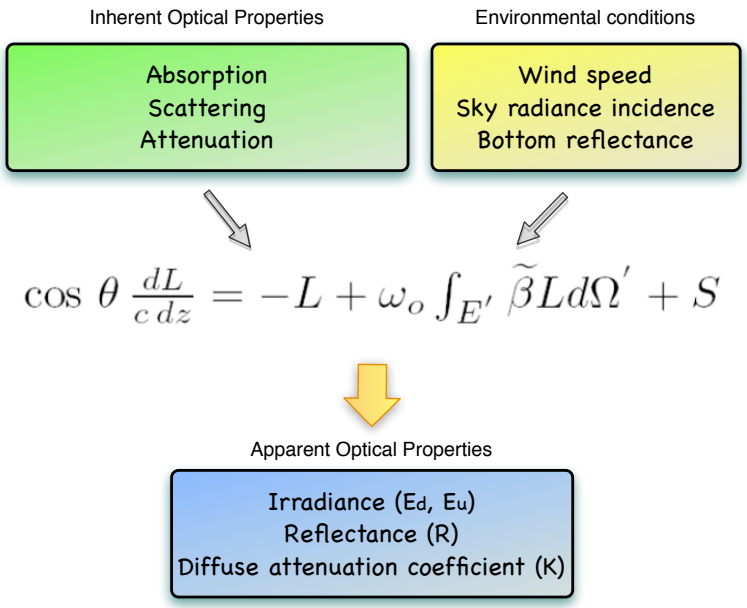


Figure 4: Radiative transfer equation.

4.1.1 Data

Ideally, we would use passive sensors to measure the AOPs owned by the thin layers we want to detect. Nevertheless, to that end we can not use data from sensors given the fact that these sensors are not available for this project, yet. Therefore, working with real data is not feasible right now. In order to overcome this problem, we model our techniques upon synthetic data generated using the *radiative transfer equation*. The theory behind this equation states that it is possible to deduce AOPs from IOPs and vice-versa (Gordon et al., 1975) (Fig. 4).

To generate these synthetic data (Fig. 5), we have used the Hydrolight-Ecolight 5.0 radiative transfer model (Mobley and Sundman, 2008) which uses the radiative transfer equation to achieve its purpose. In our case, we have defined a set of scenarios composed by several thin layers at different depths, ranging in concentration and thickness and we have set specific IOPs depending on the compounds of these thin layers.

Moreover, we have also specified standard environmental conditions such as wind speed or cloud coverage, to name few. This information has been used to feed the model which, in turn has used the radiative transfer equation to provide us with the AOPs associated to

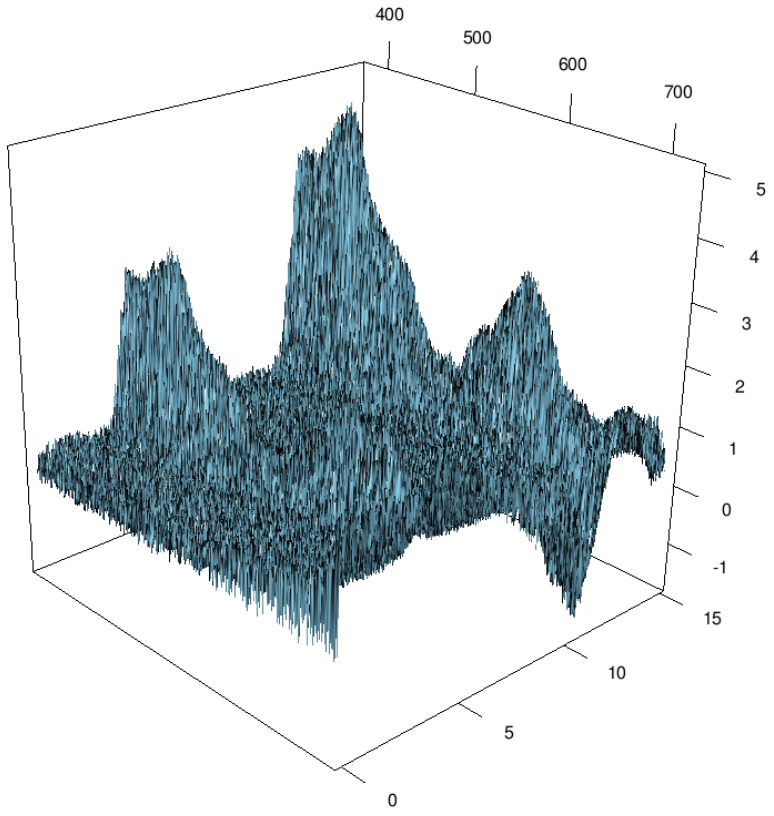


Figure 5: Data for a two thin layers scenario. The plot shows the K_d values at every sampled depth and wavelength. Data is perturbed by 30% noise.

our predefined thin layers. Such AOPs come in form of three-dimensional data, having the depth where the simulated AOP is measured in one dimension, the wavelengths along the visible spectrum in the other dimension and the actual AOP in the third dimension.

Notice that even if the model provides a set of different properties (e.g., irradiance, radiance, attenuation coefficient) we have used the downwelling irradiance (E_d) to calculate the downwelling attenuation coefficient(K_d) following the next equation:

$$K_d(z,\lambda) = -\frac{ln[E_d(z_2,\lambda)/E_d(z_1,\lambda)]}{z_2 - z_1}$$

where z_1, z_2 are the previous and subsequent depths to the target z.

The reason for using this approach has been purely empirical meaning that our system performed better with the K_d calculated by hand rather than using the values from the model.

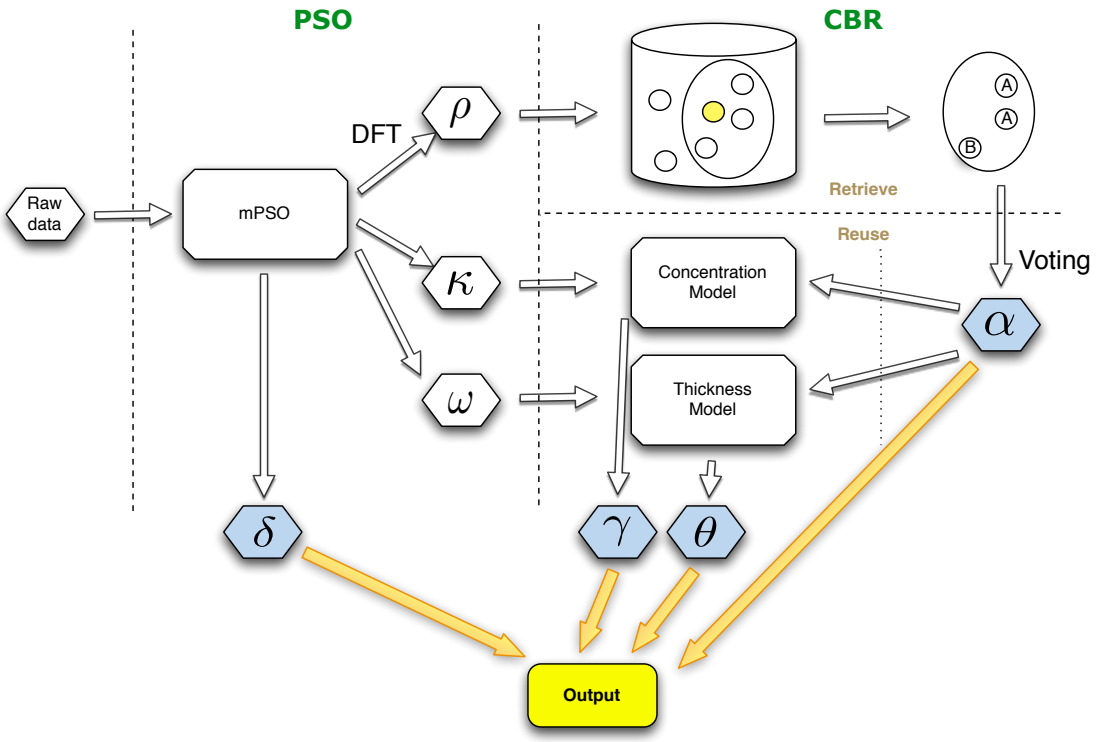


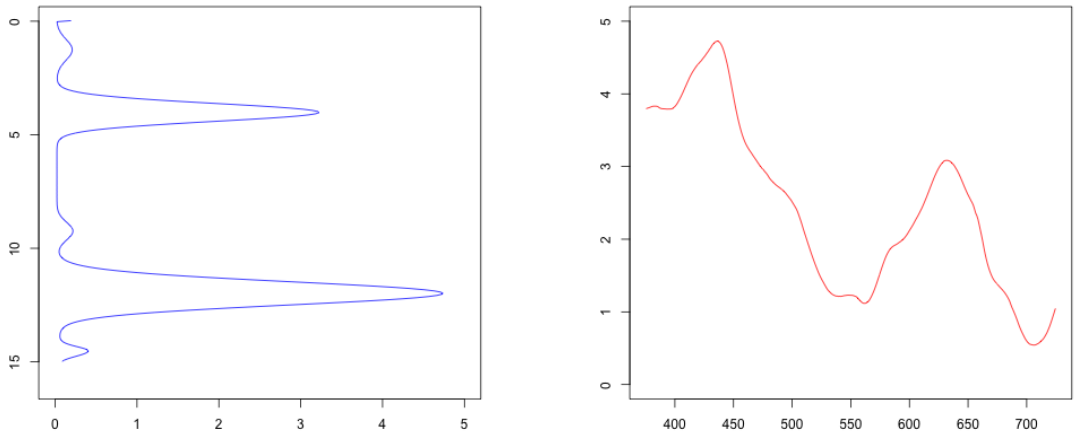
Figure 6: The application workflow.

4.1.2 Thin layer detection

In the first stage of the application (Fig. 6), mPSO is executed to discover irradiance peaks over the K_d landscape. As stated in 4.1.1, the inputs received are the irradiance coefficients for different depths along the visual spectrum. By applying the mPSO, we have been able to determine not only the depth (δ) under the sea level where the thin layers are located —position containing the maximum phytoplankton concentration— but also the local maximum K_d value (κ) which will be used as an estimator of the thin layer concentration.

For determining the 'real' phytoplankton concentration from the κ estimator, knowing the algal type we are facing is of vital importance because there is a direct relation between algal type, κ and real concentration. After several failed attempts of using only κ as algal group discriminator, a good method to successfully determine the algal group has been to focus on the contour defined by K_d at δ along the spectrum —*spectral signature*—, since all the members in an algal group share similar spectral signature, differing with other groups (Fig. 7(b)).

Notice that such signature is a series of values, meaning that managing various signatures to perform operations over them —such as comparing and contrasting in order to assess the membership of a given thin layer to a certain algal group— is tedious due to its large dimensionality. According to Agrawal et al. (1993), an efficient dimensionality reduction with minor information loss can be achieved by computing a Discrete Fourier Transform



(a) K_d value (X axis) at different depths (Y axis) for the best wavelength.

(b) K_d value (Y axis) at different wavelength (X axis) for the best depth.

Figure 7: Example of a vertical profile and spectral signature.

(DFT) to the contour, keeping just the first significant coefficients. The advantage of this technique is twofold: it aids at reducing the data's dimensionality and it defines out of the box operators between signatures since those coefficients can be described as a point in an Euclidean space.

Particularly, in this application only the four first DFT coefficients have been sufficient to determine the contours. Therefore, a contour is expressed as a point (ρ) in an 8-dimensional space[6].

Additionally, the most suitable wavelength among the visible spectrum can be obtained from the mPSO output. This wavelength is especially important because it allows to track the vertical profile of the water column (Fig. 7(a)). Such profiles clearly represent the matter assemblages as Gaussian curves.

The morphology of these curves are used to infer an estimator (ω) for the thin layer's real thickness (θ). ω corresponds to the *full width at half maximum* (FWHM) of the Gaussian curve defined by the values around δ; being FWHM the distance between the two points whose K_d value is half the maximum K_d value in δ (Fig. 8).

4.1.3 Thin layer characterisation

To that end, we have got δ as the 'real' depth where the thin layer is located, ρ as the group indicator and κ and ω as the concentration and thickness estimators respectively. What is left is the 'real' thin layer's algal group (α), the 'real' thin layer's concentration (γ) and the 'real' thin layer's thickness (θ). Thus, α, γ and θ will be computed by feeding the CBR module with ρ, κ and ω.

6. DFT coefficients are complex numbers.

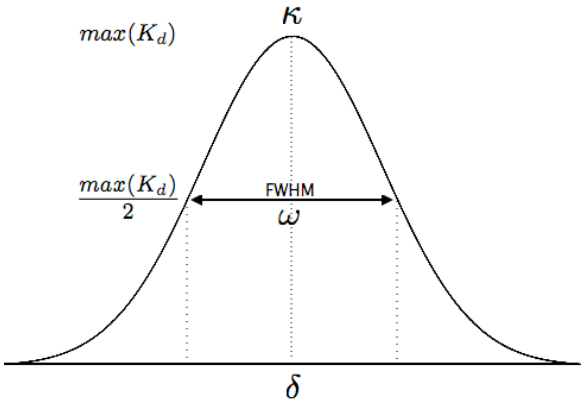


Figure 8: Depth, concentration and thickness estimators.

Taking into consideration the module's goal explained above, a modified version of the classical CBR has been implemented in this project. As a reminder, the most important concept in CBR systems is the case definition. In this particular domain, a case has been defined as follows:

$$Case =< P, S >$$

$$P =< \rho, \kappa, \omega >$$

$$S =< \alpha, \gamma, \theta >$$

where,

ρ are the first DFT coefficients;
κ is the maximum attenuation coefficient;
ω is the full width at half maximum (FWHM);
α is the algal group;
γ is the concentration; and
θ is the thickness.

Taking into account this case definition, the case base needs to be built. For this purpose, the experience accumulated through time is retained in form of cases that are being added iteratively into the case base.

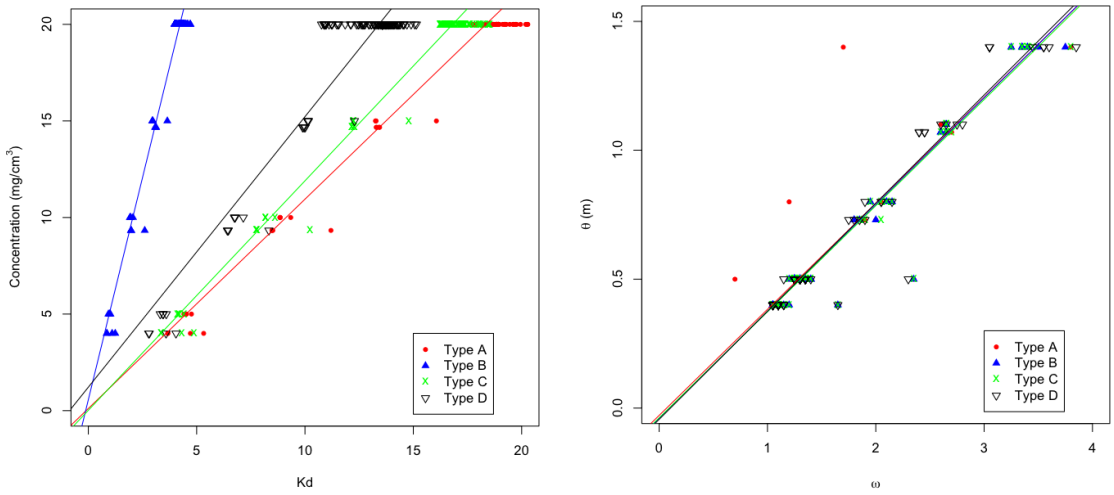
Then, whenever a new problem arrives to the module, a proper associated solution must be obtained. Notice that in this context we have to achieve two different tasks which have to be evaluated in a specific order. First, we have to determine α and secondly, from this information, we can predict γ and θ. These two tasks will be achieved in the retrieve and reuse processes respectively.

In the *retrieve* process, the system focuses on the ρ part of the problem and applies a k-Nearest Neighbour approach over the case base by performing an Euclidean distance, retrieving the most similar cases.

Next a majority voting on the algal groups (α) from the retrieved cases is performed in order to decide the appropriate α for the current case, which is kept as part of the solution.

Furthermore, α is given to the *reuse* process in order to calculate γ and θ. More precisely, α and κ are used to run the *concentration model* which predicts γ. Similarly, by using α and ω, the *thickness model* can be applied obtaining a prediction of θ. Right now, the three components of the solution have been already computed and are embedded in the final solution of the CBR.

Notice that the general form of CBR specifies a *revise* and a *retain* process which can be either automatic or manual. In the current approach, we have decided that a marine expert should manually evaluate the solution proposed by the system and decide if it is useful enough to be stored in the system. The reasons for the unfeasibility of automating this task are the lack of knowledge on the domain and the uncertainty in front of real scenarios.



(a) Concentration model: estimated (κ) vs real (γ). (b) Thickness model: estimated (ω) vs real (θ).

Figure 9: Linear regression models (for each algae type) between estimated and real values.

The models used for γ and θ predictions have been built by performing several linear regressions over the entire case base. Regarding concentration, a linear regression for each one of the owned algal groups has been done (Fig. 9(a)). This same approach has been applied for modelling thickness behaviour (Fig. 9(b)).

Finally, coupling the PSO and CBR outputs, the actual output of the system made of a thin layer's classification into one of the possible algal groups (α), a prediction of the real thin layer's location (δ), concentration (γ) and thickness (θ) is given to the user, accomplishing the proposed application's goal (Fig. 6).

4.2 Experiments

This section reports the experiments that have been carried out in order to proof the system performance. We have tested not only the suitability of the methods but also robustness, accuracy and precision. That is, we have performed three different sets of tests, each one of

them aiming at validating a certain module along the application workflow. In that sense, we have designed tests to proof the detection of thin layers, which clearly correspond to validate the applied mPSO; we have also designed experiments for assessing the algal group membership identification and for testing the concentration and thickness prediction. These two last sets of experiments have been performed to evaluate the thin layers characterisation by means of CBR.

4.2.1 Benchmark

For testing our system, we have used a controlled environment made of 300 water column scenarios generated by experts using the Hydrolight-Ecolight 5.0 (Mobley and Sundman, 2008) radiative transfer model. Each one of these scenarios contains a set of data that correspond to the simulation of a hyperspectral optical sensor measuring the attenuation coefficient (K_d) from the surface level to $15m$ deep. The measures are taken every $5cm$. Moreover, due to the hyperspectral dimension, each measure is composed by a set of frequencies —between $400nm$ and $650nm$— with a spectral resolution of $1nm$.

Regarding the water composition, we have designed the scenarios for owning two thin layers per water column. This means that we have got up to 600 thin layers, each one ranging in depth, concentration, thickness and algal group membership. Thin layers are located between $4m$ and $12m$ deep, with a concentration ranging from $4mg/cm^3$ to $20mg/cm^3$ and their thickness is included in the interval that range from $0.4m$ to $1.4m$. Furthermore, every thin layer has been designed to be of one of the four most usual algal types.

The last important fact that must be taken into account when defining the benchmark considering the present data is the gradually application of random white noise levels up to 50% in order to test robustness (Table 1).

	Depth	**Concentration**	**Thickness**	**Spectrum**	**Noise**
Min.	$4m$	$4mg/cm^3$	$40cm$	$400nm$	0%
Max.	$12m$	$20mg/cm^3$	$140cm$	$650nm$	50%

Table 1: Composition of the 600 thin layers used for testing the system. Each thin layer belongs to A, B, C or D algal group.

4.2.2 Thin layer detection

Detection set up

The first parameters that have to be set when launching an mPSO algorithm are the ones regarding the number of swarms and particles. In this case, we used four swarms containing ten particles each. The reason for using ten particles has been purely empirical. However, it is a plausible choice since we need a good trade-off between having enough particles to take advantage of the cognitive part of the module and not overloading the system which would lead it to achieve poor performance. The decision of using only four swarms, apart from being the quantity which report better results, is because the data we are dealing with contain two clear parallel ridges along the spectrum, each one at depth where a thin

layer is present. Hence, we have got two swarms approaching the main peak in each ridge, another one searching for new unknown peaks (e.g., they might randomly appear even in a controlled environment) and the last that has been left to allow some degree of failure.

Next, it is important to set a proper exclusion radius parameter because it will be useful for avoiding two swarms approaching the same local maximum. According to Blackwell and Branke (2006), the optimal exclusion parameter must be set as:

$$r_{excl} = 0.5 d_{boa}$$

where, d_{boa} is the linear diameter of the basin of attraction of any peak.

In our case, though, there is no such basin of attraction and in turn, no linear diameter for the basis. The cause for this statement is the configuration of the data retrieved from the sensors. As said, there is a ridge in each depth where the core of the thin layer is located, lasting along the visual spectrum. Therefore, the need to differentiate between the two dimensions (i.e., wavelength and depth) is a must. Having said that and coming back to the exclusion radius parameter, it is obvious that we need to set a r_{excl} to keep other swarms away in the depth dimension —$excl_d$— and another one in the spectral dimension —$excl_s$— (Fig. 10).

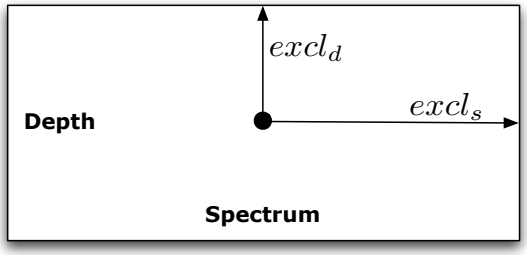


Figure 10: Exclusion distance on depth and spectrum dimensions.

Regarding depth dimension, the diameter of the basin formed by the thin layers of our data set range between $2m$ and $8m$ averaging $4m$, so the $excl_d$ parameter has been given the value of 2.

In the spectral dimension, there is no such basin, but a ridge along it instead. Then, we have considered that another swarm should not search in the same depth, at any wavelength, unless the current swarm is searching near the limits of the search space. Thus, the value of the $excl_s$ has been set to 150.

Similarly, a parameter to determine when the particles of a swarm have converged needs to be set depending on the domain. In this project we have split the parameter into two to have deeper control over the convergence as it has been done for the exclusion radius. Here, though, after performing the appropriate tuning, both convergence parameters have been given the value 2. Despite this coincidence, we think that it is important to differentiate these parameters because they might be useful for further studies when using other data sources. In other words, such coincidence has been purely chance.

The last parameters worth to mention for setting the mPSO benchmark are those used to calculate the velocity of each particle. Reminding Equation 1, the χ and ϕ are crucial to obtain proper velocities. Standard PSO values have been given to those parameters according to Blackwell (2007). Such values are $\chi = 0.729843788$ and $\phi = 2.05$.

The set of tests consists in launching the mPSO algorithm 20 times for each scenario[7] and average the results. Besides, different levels of random white noise have been applied to the raw data to test robustness. Specifically, we have applied noise levels of 10%, 20%, 30%, 40% and 50% performing a detailed study for each one of them. The purpose of applying such high levels of noise is because we have to develop a very robust system that will be able to work in real scenarios, still unknown, in a near future.

Detection results

One of the strong points of using mPSO to approach this kind of problems is its capacity to access just few positions within the search space, yet achieving quite high performance. In our case, the algorithm has accessed 13% of the existing 75,000 measurements.

The first issue that worried us about detecting thin layers was the possibility that the system could miss a thin layer when both ones present in a water column were too close to each other. Intuitively, this idea seems pretty plausible, since it is hard even for humans to determine when a thin layer finishes and the other one starts when we are facing two too close thin layers. From the available data, however, this was not an issue. The scenarios that present closest thin layers were those containing two layers $3.5m$ separated from their core. These situations have been demonstrated to be innocuous regarding the performance of our system.

Contrarily, the problems arouse for certain situations where a thin layer was not detected. This happened exclusively when the upper thin layer in the water column contained a very high level of concentration, whilst the bottom one presented very low concentration. In such scenarios, the bottom thin layer was not always detected.

Noise	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**
Failures	1.38	1.62	1.64	1.67	1.75	1.82
Error	2.8	3.3	3.4	3.9	4.1	4.2
Std.Dev.	3.0	6.0	7.4	8.3	9.8	10.8

Table 2: Peak detection failures(%), error(cm), and standard deviation(cm), given noise levels ranging from 0% to 50%.

Due to the fact explained above, our tests focused on first calculating the failure rate —both false positives and false negatives— over different levels of noise. Table 2 shows this rate which goes from 1.38% in a no-noisy data to 1.82% of failures on data affected by 50% of noise. Clearly, even if this failure on detecting certain thin layers exists, it rarely occurs according to our tests.

For those thin layers that have been properly detected, we need to measure the accuracy on locating such thin layers. In this case, Table 2 also shows the good response obtained

7. A scenario consists of a water column profile.

by the system, since the error rate ranges from $2.8cm$ when no noise is found to the system up to only $4.2cm$ when the maximum level of noise in our experiments has been applied. Notice here, that even for the highest error level, it is smaller than the depth resolution —$5cm$. This means that the real error rate of the system might be even less if we use data with a higher resolution. Giving these low error rates, we can infer that the depth where the thin layer is located at does not affect the module final result.

Even if the average error rate for the 20 runs gives good results, it is interesting to calculate its standard deviation in order to assure the real response of the system and that we are not being fooled by the results. In that sense, notice the low deviation — from $3.0cm$ to $10.8cm$ depending on noise— which leads us to trust the accuracy just shown and claim the robustness of the thin layers detection module.

4.2.3 Thin layer characterisation

Determining algal group membership

In order to assess the performance of the CBR module, we have started by focusing on the algal group identification part. Specifically, the DFT coefficients retrieved by the mPSO module corresponding to the contour defined by K_d along the spectrum for a thin layer's depth (ρ) have been used as an n-dimensional point in order to determine the main algal group (α) —either A, B, C or D. A 10-fold cross validation technique has been applied on every pair of values consisting in a DFT point and the algal group.

Several tests for tuning the parameters regarding the k-Nearest Neighbour approach have been performed. We have used different values of k (i.e., 1, 3 and 5) obtaining the best results when k=1, where its accuracy is over 97% for noise levels under 30%. From that point, the system accuracy decreases steadily until achieving 95% of accuracy for noise levels up to 50%. For all other values of k, the accuracy is lower, but always reaching more than 90% accuracy, as it is in the worst case. Such results are shown in Figure 11.

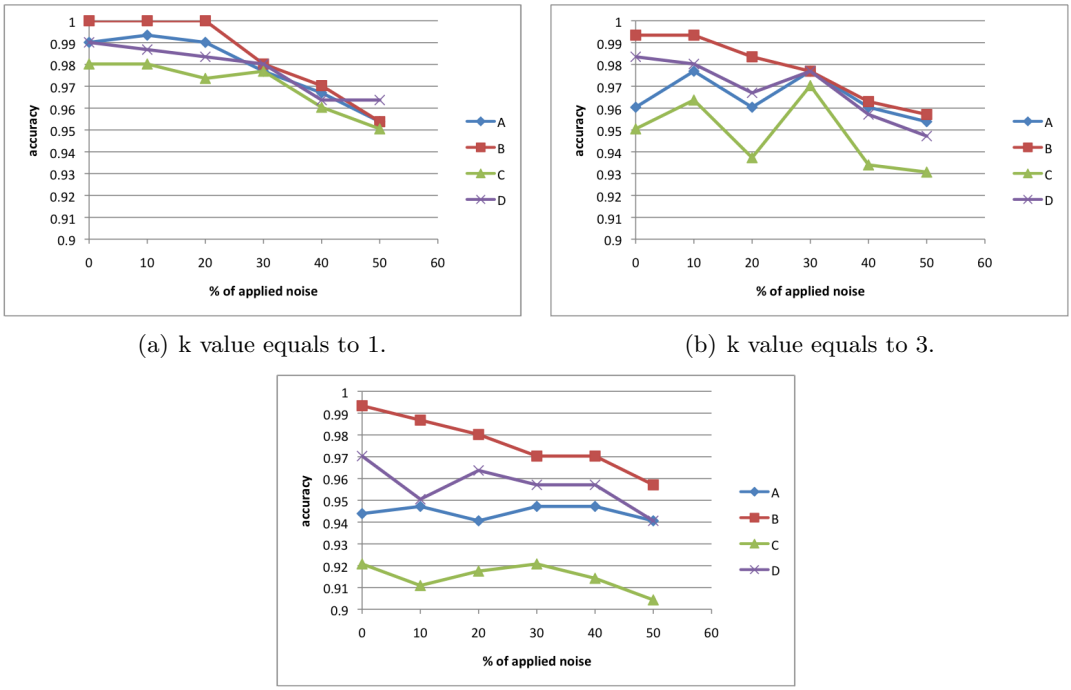
At the same time, different voting methods such as majority voting (Equation 2) and weighted voting (Equation 3), which takes into account the distances between target point and their neighbours, have also been conducted. In spite of the different methodologies, the results obtained have been exactly the same. Therefore, we have decided to use the majority voting due to its lower computational cost.

$$\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \mu(v, f(x_i)) \tag{2}$$

$$\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \frac{1}{\|x_q - x_i\|} \mu(v, f(x_i)) \tag{3}$$

where,

$f(x)$ is the membership function;
V is the set of algal groups; and
$\mu(a, b) = 1$ if a=b; 0 otherwise.



(a) k value equals to 1.

(b) k value equals to 3.

(c) k value equals to 5.

Figure 11: Accuracy in algal group identification when applying noise levels up to 50%.

Concentration and thickness set up

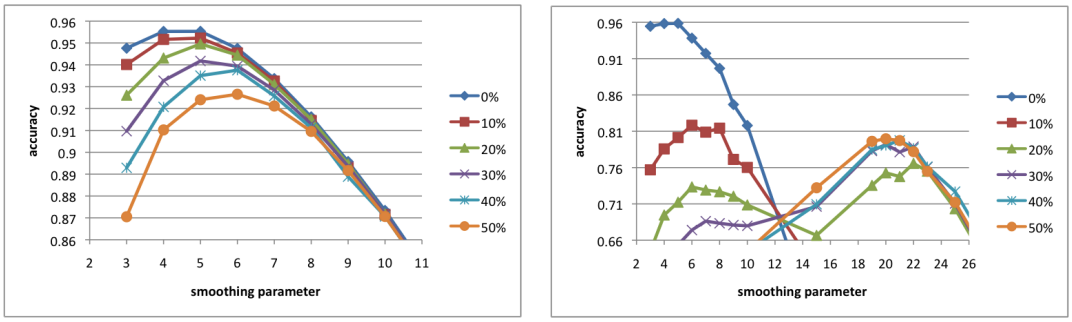
The second block of the thin layers characterisation aims at evaluating the prediction of real concentration (γ) and thickness (θ) in the CBR module, which corresponds to the last phase of the system.

It has been stated that the prediction of γ and θ is the result of applying linear regression on the concentration and thickness estimators, which are κ and ω respectively. Hence, the importance of calculating proper estimators is a must for ensuring precise and accurate predictions. This condition still holds whenever more noise is applied to raw data. Because we want to build a robust system able to deal with data highly influenced by noise, we had to apply a smoothing technique to remove the noise on the data and extract reliable κ and ω estimators.

The smoothing technique used on the system consists in calculating the mean value of every measure in its neighbourhood along the depth axis by defining a smoothing grade which is the number of neighbouring measures immediately before and after the current measure in order to soften the existing peaks derived by the noise.

Another important parameter regarding the estimators worth it to comment is the sliding window. This parameter specifies the interval around a measure where to search for the highest peak. It is especially important for noisy environments since it only focuses on the highest peak within the window, ignoring other lower peaks that might arise from the noise.

Good κ and ω estimators in noisy data are obtained by joining the smoothing and sliding window techniques. Nevertheless, tuning those parameters is a compulsory task for achieving good system performance. The results shown in Figure 12 have been achieved by applying different values for the smoothing and sliding window parameters to tweak first concentration and then thickness on all the scenarios for different levels of noise.



(a) Concentration tuning.

(b) Thickness tuning.

Figure 12: Accuracy on smoothing parameter tuning for data perturbed by different levels of noise.

Notice that the best performance for concentration has been reached setting the smoothing parameter to 6 (Fig. 12(a)) when dealing with highly noisy data —noise levels greater than 40%— and using 5 as the value for the parameter when facing data perturbed by low noise levels —under 30%. In all those cases, though, either of the previous values for the parameter can be used since the differences in accuracy are very low.

Regarding thickness, Figure 12(b) shows the need for using very different parameters depending on the noise affecting our data. Whilst data affected by low noise levels —up to 10%— perform well using a smoothing parameter equals to 6, those samples perturbed by 20% reach their best performance setting the parameter to 22. Withal, data affected by more than 30% of noise need to use 20 as the value for the smoothing parameter. Otherwise the accuracy drops rapidly. From these statements we can generalise that 6 is a good value for low-noisy data and around 20 is good for high-noisy data. Nevertheless, data-specific parameter tuning should be done in each case to ensure good performance.

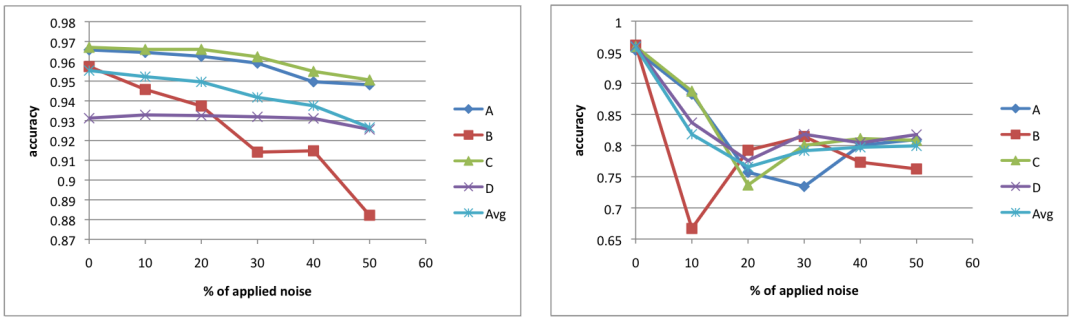
The most suitable value for the sliding window parameter has been found to be 1. Even if the implementation of this technique is compulsory for the smoothing to work, tuning its parameter is not as critical as the smoothing one. Hence, we have used a sliding window big enough to avoid finding two different peaks as being the maximum for a single thin layer due to noise variations or sampling errors and at the same time small enough to focus only on one thin layer at a time. After several trials, sliding window equals to 1 have given the best results.

Concentration and thickness results

Apart from those tests for tweaking the parameters, other experiments to evaluate the suitability of the methods used to predict γ and ω have also been conducted. Figure 13 exposes the accuracy variability on concentration and thickness at different noise levels

for the best parameters possible in each case. The high values achieved demonstrate the suitability of using linear regression techniques for predicting γ and ω. Nonetheless, we can not ignore the fact that B type algae are the ones that perform worst in average, so further study must be carried out.

Focusing on Figure 13(b) B type algae accuracy, the thickness prediction technique reaches its minimum —slightly under 70% accuracy— for noise levels of 10%. The reason for this is that the smoothing parameter has to be set to 6 for low noise levels yet must be set to 22 for high noise levels. In this case, noise levels of 10% seem to be the threshold between low and high noise perturbation, together with the B type algae problematic issues lead the system to such lower performance.



(a) Concentration accuracy.

(b) Thickness accuracy.

Figure 13: Accuracy in concentration and thickness prediction for noise levels ranging from 0% to 50%.

Notice that these studies have evidenced the weaker performance of ω compared to κ. Those differences on accuracy rely on the nature of each estimator. In other words, noise perturbations on a Gaussian-like function affect more significantly the FWHM rather than its maximum value.

Algal Types	**A**	**B**	**C**	**D**	**Avg.**
Concentration	0.95	0.88	0.95	0.93	0.93
Thickness	0.81	0.74	0.82	0.82	0.80

Table 3: Concentration and thickness accuracy on the four algal types for data perturbed by 50% of noise.

Finally, other tests to emphasise the differences among algal types regarding concentration and thickness prediction have been carried out. As can be seen in Table 3, algal type B is the one that achieves lowest results. Despite reaching 88% accuracy for concentration and 74% for thickness when 50% of white noise has been applied, this value is lower than its homologous A, C and D types. We think that the cause for such phenomenon is the fact that B-type algae show lower K_d values for the same values of γ than A,C or D types. Therefore, because the multiplication factor of the linear regression on B-type algae

is greater, the errors derived from smoothing are also magnified leading to poorer accuracy on θ.

5. Analysis using partial and dynamic data

The second part of the project tackles the problems associated with the ascent stage of the probe's cycle. In the previous block, we have seen how thin layers have been detected and characterised when the probe reaches the bottom of its trajectory. This knowledge is used to build a set of hypotheses that states the expected thin layers that are likely to be found during the probe's ascension to the surface. However, these hypotheses might be wrong or partially incorrect. This fact gives rise to the need for an online decision component able to adapt or even change those hypotheses on the fly in order to provide the probe with the proper positions where to close its recollection bottles and gather the samples.

5.1 Approach

5.1.1 Preliminaries

Identifying thin layers in the sinking stage of the probe's cycle allows us to take wise decisions about the best depths where the samples should be retrieved. The output of the first module consists of the depth (δ), concentration (γ), thickness (θ) and algal group (α) for any found thin layer. Moreover, we can also pick a single wavelength to be of reference for monitorizing the vertical profile.

Such information is provided to the biologists who decide the specific thin layers of interest for them. For those thin layers we can select those depths (δ) that correspond to the maximum level of K_d (κ) in the reference wavelength (usually $435.5nm$). Keeping in mind the bell shape of the vertical profile for any given thin layer we also select the depth of the FWHM under the peak as well as those places where the K_d value is 1/4 and 3/4 of κ (aka cutting points).

The purpose of selecting these four depths is that we want to be able to determine what kind of thin layer we are facing in the probe's ascent before reaching δ (the earlier we find out, the more time we have to act accordingly).

Then, the spectral signature (values along the spectrum for a given depth) for every of the above mentioned depth points are retrieved and transformed using the already explained DFT in order to reduce dimensionality (Section 4.1.2).

The theory behind this technique states that the spectral signature at δ uniquely identifies the thin layer and implicitly provides information regarding its properties. Hence, whenever approaching δ, the spectral signature at the current depth becomes more alike to that at δ.

5.1.2 Case definition

The system achieves its purpose based on a set of hypotheses that are built at the end of the probe's descent and are later being almost continuously checked along the ascent path.

Every thin layer found during the descent or in any of the previously seen scenarios (either in descent or ascent) ends up creating a case.

The depth, concentration, thickness and algal group of any thin layer, as defined in 4.1.3 corresponding to δ, γ, θ and α, contribute to build a case. Besides, we also need to keep the information regarding the four cutting points for every thin layer, namely C_1, C_2, C_3 and C_4 as well as their corresponding spectral signatures after suffering the DFT transformation, being S_i the firsts DFT coefficients. Therefore S_1, S_2, S_3 and S_4 correspond to the DFT coefficients for the C_1, C_2, C_3 and C_4 cutting points respectively, being C_1 the closest cutting point to δ and C_4 the furthest one (Figure 14).

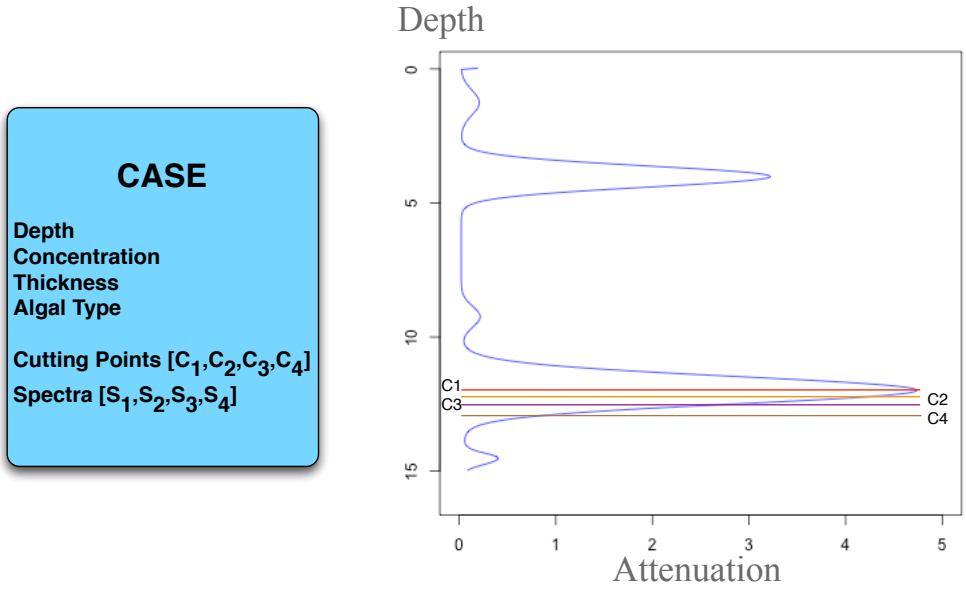


Figure 14: Case definition

Later in this report, the cases' dynamics will be explained. However, in order to understand the flow of cases among different sources, a similarity measure between them has to be defined. In this work, when comparing two cases we assess their similarity by means of calculating the Euclidean distance between the 'current' S_i of one case and the 'current' S_j of the other.

5.1.3 Cases sources

All the cases built along the system's lifetime representing the knowledge gathered throughout runs will be stored in the *Case Base* (Figure 15). This source of information will be readily available whenever the execution-specific cases are not valid anymore. Thus, this case base would be rarely accessed taking into account that the ascent should be somehow similar to the descent and the hypotheses built during descent will hold even if some adaptation is required. For those scenarios where the ascent is too dissimilar with respect to the descent, though, the Case Base will be used.

There also exists another source of cases which is built specifically at each run. Such a container is the *A Priori Source* which stores the cases found during the probe's descent in

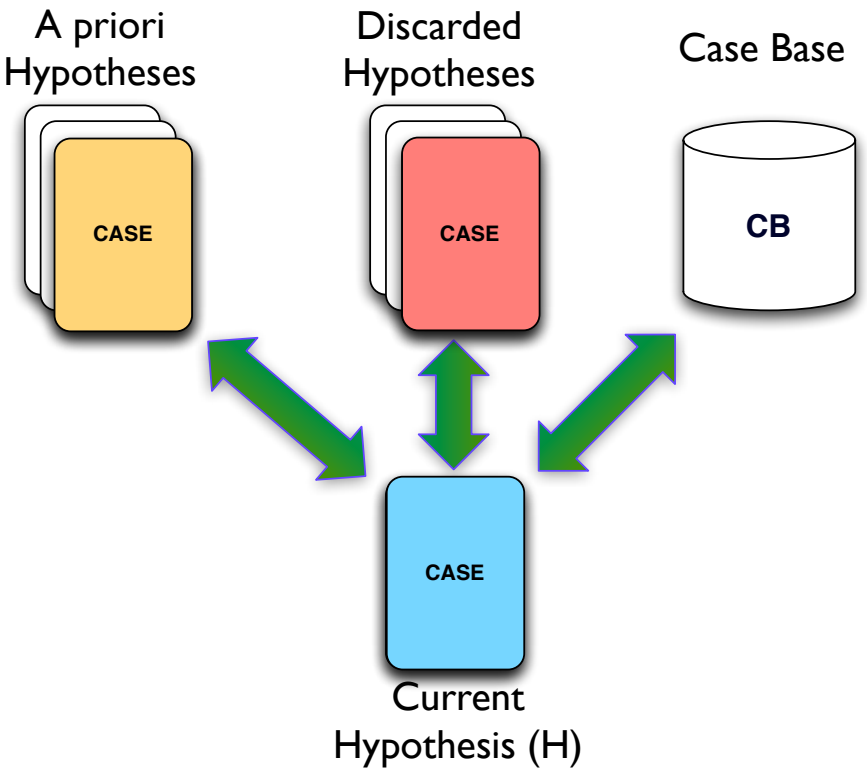


Figure 15: Cases' dynamics

a stack-like fashion (the last cases added to the container at descent will be the ones that will be retrieved first when ascent). Notice the local nature of the cases in this container (compared to the non-local nature in the Case Base). For instance, if during descent there is a hypothesis stating that a specific thin layer is present at depth 12 and during ascent such thin layer is not present, but there is a similar one at depth 4, this might be a completely different thin layer than the one expected at 12, meaning that it could be more appropriate to refute this hypothesis and search for a different case in the Case Base.

Similarly, there is the *Discarded Hypotheses Source*. This container is also scenario-specific and tied to local scope. The purpose of it is to store those hypotheses that have been refused during the probe's cycle in order to be reused if necessary. It does make sense when thinking, for example, on those situations where a thin layer was expected at a certain depth (let's say 4 m.) and it has slightly been 'delayed' (found at 3 m. in the ascent). The system might temporarily throw away the hypothesis of finding a thin layer at 4 m., but it should consider it again when a very similar thin layer is found at 3 m.

5.1.4 Ascent algorithm

The system relies on the hypotheses made at descent to make informed decisions on the best places where samples should be gathered. Therefore, at each time step during the

probe's ascent, the current hypothesis is compared to the real current data retrieved from the sensor (Algorithm 1).

```
Data: APriory from descent, CaseBase
//Initialisation;
Discarded ← ∅;
repeat
    if currentHypothesis = NULL then
        currentHypothesis ← pop(APriory);
    end
    if sensor_data <> currentHypothesis then
        adapt_hypothesis(currentHypothesis, sensor_data);
        if not_valid(currentHypothesis) then
            add(currentHypothesis, Discarded);
            currentHypothesis ← pop(APriory);
            if not_valid(currentHypothesis) then
                currentHypothesis ← most_similar(sensor_data, Discarded);
                if not_valid(currentHypothesis) then
                    currentHypothesis ← most_similar_case(sensor_data, CaseBase);
                end
            end
        end
        adapt_offset(currentHypothesis, sensor_data);
    end
    if depth_eq(depth, currentHypothesis) then
        close_probe_bottle();
        currentHypothesis ← NULL;
    else
        if valid_next_cut_point(currentHypothesis) then
            next_cut_point(currentHypothesis);
        else
            currentHypothesis ← NULL;
        end
    end
    obtain_next(sensor_data, sensor);
until surface_is_reached() = true ;
```

Algorithm 1: Probe's ascent algorithm

The algorithm starts by taking the last hypothesis introduced in the A Priory Source as the *current hypothesis* (the case found at bottom depth level at descent since the probe is moving from the bottom to the surface at ascent).

Next step is to compare if what is being measured by the sensor is similar to what is expected according to the current hypothesis. For assessing the similarity, we first check the K_d level at the current depth in the reference wavelength. If the level is high enough, meaning that there might be a thin layer in that location, we perform a second checking. This time, we retrieve the whole spectrum from the sensor data (regardless of the reference

wavelength) at the current depth in order to infer the algal type for the current thin layer. This information is used to validate that the current K_d level in the reference wavelength is above its algal-specific threshold (a predefined different threshold for each one of the known algal types).

Finally, if this two-step comparison evaluates true, we can actually assess the similarity between sensed data and the current hypothesis by retrieving the measured K_d values along the whole spectrum at the current depth and perform the DFT. The first n DFT coefficients are then compared with S_4 from the hypothesis (we start by comparing the spectrum at C_4 because it is the cutting point that is located furthest from the surface and, therefore, should be found first since the probe is ascending) by calculating the Euclidean distance between them. If the distance is below a given threshold, we consider that the hypothesis still holds. In case that this distance is above the mentioned threshold, the hypothesis is not valid anymore leading the system to act accordingly. Notice the three-step nature of the approach used to validate the current hypothesis with respect to the sensed data.

If everything goes as expected, meaning that the hypothesis still prevails, the probe keeps ascending until next cut point (C_3) is reached. Then, a comparison between the sensed data and the current hypothesis is performed as explained in the previous paragraph. This process is repeated similarly for C_2 and C_1 as far as the hypothesis holds. However, when C_1 is found and properly evaluated (the core of the thin layer has been reached, according to the hypothesis), a bottle closes in order to retrieve the water sample. Immediately, the current hypothesis is rejected and a new case from the A Priori Source becomes the current hypothesis, starting again the whole process from C_4 to C_1 in the upcoming depths, leaded by the new current hypothesis.

So far, the algorithm is pretty straightforward when sensor data correspond to the expected according to the hypotheses, but what happens if they do not? That is, what if the thin layer is located slightly before it was expected? And what if it is located slightly after the expected depth? What if they change in concentration or thickness? Not mentioning if an expected thin layer is not present or if there is one in a place that should not be.

In order to tackle all these issues, there is the need to introduce different modules to the system. They will be crucial for improving the system's accuracy but also its performance.

In case the data measured by the sensor do not match with the expected current hypothesis we perform a cascade of procedures until a valid case able to become the current hypothesis is found. Next, such cascade of tests is explained.

After realising about the missmatch between sensor data and current hypothesis, we try to adapt the current hypothesis to match the sensed data by using other C_i, performing the three-step comparison already mentioned.

Still if there is no match, we reject and store the current hypothesis into the Discarded Hypotheses Source and a new current hypothesis is obtained by retrieving the last hypothesis introduced into the A Priory Source. This hypothesis is validated by applying the three-step comparison, but also taking into account the difference between current depth and current C_i (if the difference between them is too high, the new hypothesis will not be valid even if the three-step comparison matches). Such constraint is necessary due to the local neighbouring character of the A Priory and Discarded Hypothesis Sources (A thin layer can be found slightly before or after the expected, but not too far. If a similar thin layer

is found out of a predefined depth interval, we consider we are actually facing a new thin layer, not the expected).

At this point, if the hypothesis retrieved from the A Priori Source does not match the sensed data, we search for a case in the Discarded Hypotheses Source. Specifically, we retrieve any case that contains any S_i that matches the current sensed data, but forcing the distance between current depth and C_i to remain close enough.

Finally, if we could not find any case that can become the current hypothesis, our last attempt is to search into the Case Base searching for the most similar case ever seen during the system's lifetime. This means that the system will retrieve the case that best matches the sensed spectrum at current depth (the Euclidean distance between any S_i and current depth's DFT coefficients is the closest among all the cases in the Case Base and below a certain threshold) independently of the depth that such case was seen. If such a case is found, it becomes the new current hypothesis.

After all this cascade procedure is performed, whenever a new current hypothesis is found, it is then adapted by applying an offset to the depth and all the cutting points in the case in order to set the case to the current depth. Finally, the new hypothesis is ready.

Now it is time to check whether the current depth is approximately the same as the one stated by the current hypothesis as being the thin layer's core. If so, a bottle in the probe is closed, sampling the water presently at current depth. At that point, the current hypothesis is rejected and the cycle starts again searching for next thin layer.

However, if the current depth does not approximate enough to the current hypothesis depth, we search for the next valid cutting point within the current hypothesis and change to it if valid (it can either be another cut point or the actual one). The new cut point will be valid if it is closer to the surface than the current depth. Nevertheless, if there are no valid cut points in the current hypothesis, we reject the current hypothesis, leaving the current hypothesis sought for next loop iteration.

As an example, focus on the special case where a thin layer is expected, but nothing is being found according to the data retrieved by the sensors. A primal intuition would be to reject this hypothesis and retrieve the next case from the A Priory Source aiming at the next thin layer, considering that the current thin layer is not to appear. The problem will arise when the expected thin layer is not 'removed' from the place but it has been 'delayed'. In order to overcome this situation, the rejected hypothesis is not actually rejected, but it is stored in the Discarded Hypotheses Source. This container maintains the rejected hypotheses during the probe's ascent in order to be used in future, if needed.

The presence of cases like this and their counterparts (when a thin layer is slightly 'advanced' to what was expected) are the situations that make the most of the system under development.

5.2 Experiments

The previous section explains the approach that has been taken to build the online decision component. Here, the implementation of the techniques is empirically assessed in order to validate their suitability and performance.

5.2.1 Set up

The system evaluation has been conducted by using the same scenarios as the ones introduced in Section 4.2.1 as input data. Furthermore, new scenarios have been created by manipulating the existing ones in order to enlarge the scenarios set. Specifically, we have used those scenarios where thin layers' thickness was 40 and $50cm$ respectively, their concentration was $20mg/cm^3$ and were located at 4 and $12m$ deep (the *reference scenarios* where all the other scenarios were derived from) to create new scenarios.

Each one of the reference scenarios suffered from a data shift consisting in relocating each datum 0.1, 0.3 or 0.5 meters from its real location towards the surface and towards the ocean depths. Therefore, 6 new scenarios were built from each reference scenario, resulting in a total amount of 96 new scenarios.

The reason for the creation of these 96 new scenarios is the need to test the system in the specific cases where the thin layer found at descent is almost the same as what is being found at ascent but it is located in a slightly different depth. It is expected that the A priory and Discarded sources make the most of their capabilities in such situations.

Benchmarking is performed by executing the system for every pairwise of scenarios in the data set containing thin layers composed of the same algal type. At each evaluation the whole probe's cycle is simulated by using one scenario as the data retrieved during the probe's descent and the other scenario corresponding to the data sensed while the probe is ascending. These combinations of scenarios allow to test a multiple set of possibilities (differences between descent and ascent data) that the system is expected to face in real situations.

Although the decision component is guided by the information gathered at descent, the real location of thin layers found during ascent is not known a priori. Hence, in order to evaluate the correct performance of this module, we need to first assess whether the thin layers have been found and finally, considering those thin layers properly found, calculate the error existing between the real depth of the core of the thin layer and what the system considers to be the thin layer's core.

Every time the system decides there exists a thin layer in a certain location, its decision is checked with the real data. An interval of $0.5m$ (the probe's length) has been defined as the maximum allowed error. That is, due to the probe's dimensions, if a real thin layer is present within $0.5m$ from the decided location, we consider it as a *hit* for the system. However, if the closest real thin layer is further than $0.5m$ from the system's decided location, the decision must be count as a *miss*.

According to the explanation in the paragraph above, the typical statistical rates for counting hits and misses are here redefined for the current domain: *true positives* are those cases where the system decides that there exists a thin layer and, indeed there exists one. *False positives* occur when the system decides there exists a thin layer in a certain location, but in fact, there is none. A *false negative* appears when the system misses to find a thin layer. Finally, the concept of *true negatives* rate is not valid due to the nature of the problem to solve (the whole remaining time; when the system does not detect any thin layer and, indeed, there is none).

The absence of the true negative rate does not allow us to perform any test based on accuracy. Thus, a new score is introduced here in order to assess the system performance. The selected measure is the *F-score* which is defined as follows:

$$F\text{-}score = 2 * \frac{precision * recall}{precision + recall}$$

As it can be appreciated, this score relies on the *precision* and the *recall*, which, as a reminder, are formulated next:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

where,

TP is the true positives rate;
FP if the false positives rate; and
FN is the false negatives rate.

Finally, for those thin layers properly found (true positives), an error measure is defined, consisting in calculating the difference on depth between the real thin layer's core and the one determined by the system. The measure will be expressed in *cm* and its value will be the absolute value of the difference.

5.2.2 New technique suitability

A main contribution in terms of Artificial Intelligence regarding this second system's block is the enhancement of the classic Case-Based Reasoning technique. Due to an in depth analysis on our domain, we realised that only using past experience to tackle the current problem of locating thin layers during ascent might not be enough to succeed. Instead, we can profit from local knowledge acquired at descent to improve our chances. This part of the report shows the experimental procedures we performed to validate this theory.

We forced our system to keep a particular set of cases in the Case Base. They are very different from those that will be found by the probe during its simulation. Specifically, we stored thin layers with a very low level of concentration ($\gamma = 4mg/cm^3$) into the Case Base.

Regarding the scenarios used for descent and ascent, we used scenarios whose thin layers were located in different positions (from 1.3 to $4m$ away one to another) in order to ensure that the system fails on finding the thin layer when only using the information gathered from descent.

Because of the strict constraints used to perform this test, when no external sources were used (neither A priori nor Discarded nor Case Based), the system was unable to find a single thin layer. However, this situation changed when the Case Base was incorporated to the system, functioning in a classical Case-Based Reasoning fashion. In this test, the F-score raised to nearly 0.62 (Fig. 16).

On the other hand, we tried the same test but using the A priori and Discarded sources instead of the Case Base. In this situation, the F-score raised to over 0.66, reaching a Precision of 1.0.

Finally, we tested the same scenarios using a combination of A priori and Discarded sources with Case Base. Now, despite the decrease on Precision, the overall score was the highest of the 3 situations, reaching an F-score near 0.72, thanks to the notable Recall growth.

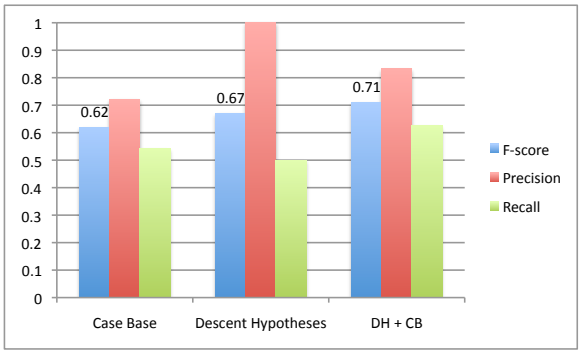


Figure 16: Comparison between using Descent Hypotheses and a Case Base

This test set has been used to proof the motivation of the new technique developed in this project to increase the performance for this specific domain compared to the classical CBR. It is true that, in this test, the system behaves only slightly better under our approach that with respect to CBR. Nevertheless, we must take into account that the scenarios that we will find in nature are likely to be more dissimilar from one immersion to another, since many factors are in play. Not mentioning the differences between data retrieved at distinct days or geographical locations.

Moreover, it is also worth mentioning the high cost of accessing to the Case Base. Every access to CB is much more costly than accessing to the local A Priori or Discarded sources. Even more, contrarily to the cost of accessing to the local sources, the cost of accessing to CB increases whenever more knowledge is stored in it.

The discussion of the last paragraphs clearly sustain the suitability of the new developed technique for specific domains such as the one at hand.

5.2.3 System performance

Finally, we assessed the system performance on the probe's ascent using the new technique developed in this project. As explained, different dimensions of the problem must be taken into consideration when designing the proper tests for the system to overcome. On the one hand, we need to evaluate the correctness of the system on determining the presence of a thin layer in a certain depth, while, on the other hand, for those thin layers properly found, a measure of the error for determining the exact depth of the thin layer core must be calculated.

Based on these premises, two different sets of tests were carried out to evaluate the system. One was called the *overall* since it is in charge of evaluating the overall performance of

the system. The other one has been called *variations*, because it tests the system behaviour in front of variations in a specific thin layer's property.

Overall

The overall test was conducted by applying a 10-fold cross validation over the whole bunch of scenarios (400 items). More precisely, given the 400 scenarios, composed of two thin layers each with different properties, they were split into 10 different folds, keeping one fold as testing scenario and the thin layers of the remaining 9 were stored as cases into the Case Base. This resulted in a Case Base containing 720 random thin layers.

The testing fold consisted in 40 different scenarios, containing 2 thin layers each. Every scenario was used once for descent and tested against the rest of the scenarios at a time, which were used as ascent scenarios. The combination of all these scenarios in a pairwise-like strategy resulted in a total number of 15,210 thin layers evaluated at ascent.

The results of this test show the good performance of the system obtaining an average F-score over 0.89 (Fig. 17). Precision and Recall were also evaluated obtaining barely the same score (over 0.89 each).

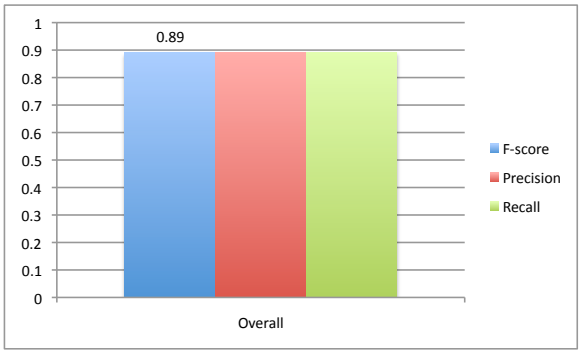


Figure 17: Overall detection of thin layers at ascent

For every thin layer that was properly detected, an error measure was calculated as explained in the previous section. The mean error on locating the exact position of the thin layer's core is $0.19m$ (Fig. 18), which is acceptable taking into account that the prototypical probe's length is $0.5m$, meaning that the sample will be properly retrieved even having this small error.

Notice the high F-score obtained by the system even using a random selection of thin layers as Case Base. Indeed, using wiser selection criteria to feed the Case Base will end up improving the system performance. Nevertheless, the heterogeneity of thin layers that are to be found in nature will, a priori, reduce this score. Therefore, the task of the deployment team will be to find a subset of real thin layers representative enough to keep a high system performance.

Variations

Despite the good performance obtained in the previous tests, a more specific evaluation was carried out in order to identify the strengths and weaknesses of the system with regards to variations in thin layer's properties. In that sense, we have identified four different categories of variations between descent and ascent scenarios:

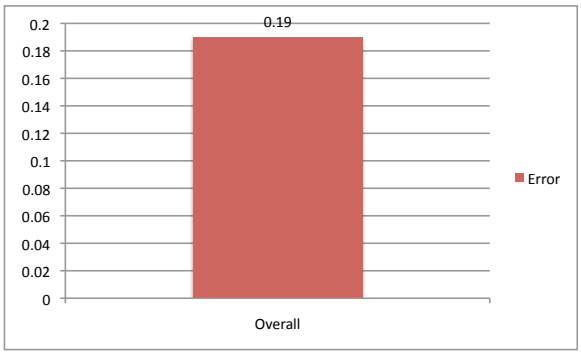


Figure 18: Overall fine error on detected thin layer depth (in meters)

The first category is composed by those executions where the descent and ascent scenarios only differ in the depth of the thin layer, being such difference lower than $1m$.

Similarly, the second category is formed by executions whose descent and ascent scenarios have thin layers located at different depths varying from 1 to $4m$.

The third category is made of scenarios whose thin layers' difference consists in different concentration levels (from 4 to $20mg/cm^3$).

Finally, the fourth category is composed of scenarios varying in thin layer thickness (from 0.4 to $1.4m$) when comparing descent and ascent scenarios.

For each one of these categories, a test was performed consisting in feeding the Case Base with the reference scenarios ($\delta = 4$ and $12m$, $\gamma = 20mg/cm^3$ and $\theta = 0.4$ and $0.5m$) and launching a simulation for each one of the pairwise scenarios within the category.

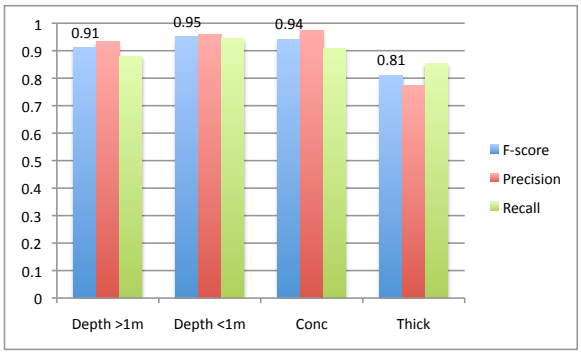


Figure 19: Detection of thin layers at ascent for variations test

The results obtained are the average of 4,416 thin layers per category assessed at ascent. Fig. 19 shows the good performance obtained for small variations in depth (F-score = 0.91), larger depth variations (F-score = 0.95) and concentration variations (F-score = 0.94). The weakest category, not surprisingly, corresponds to scenarios varying in thickness. In such situations, the F-score reaches an average of 0.81.

For the thin layers properly found, the error between the real thin layer's core and the one determined by the system was calculated (Fig. 20). The results state the better

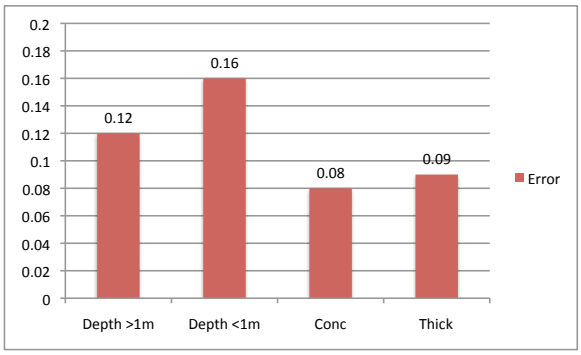


Figure 20: Fine error on detected thin layer depth (in meters) for variations test

resolution of depth location for concentration and thickness categories (0.08m and 0.09m respectively) rather than for depth variation; either low (0.12m) or high (0.16).

In any case, even though this error does not affect the sample recollection due to the probe's size, further research to improve this error can lead to better results or even an increase on the thin layer's detection.

6. Conclusions and future work

In this project we have introduced an ongoing research topic consisting in detecting and characterising phytoplanktonic thin layers from data supplied by hyperspectral optical sensors in order to predict the location of planktonic patches where water samples should be gathered. The task has been to determine not only the location, but also concentration, thickness and type membership of thin layers.

The final purpose of this study is to embed the developed techniques into an oceanographic probe able to retrieve biologically interesting water samples in real marine scenarios.

The probe works in a two clearly differentiated phases. The first one corresponds to the descent, where the probe is introduced into the water and it sinks until reaching a certain depth level. At this point, data about the environment sensed during descent is analysed.

The second phase corresponds to the ascent, where the probe moves from the bottom depth level towards the surface. In this phase, the data retrieved during descent is used to predict the existence of biologic matter that is to be found during its way up.

In the project's first phase, corresponding to the probe's descent, two powerful (and yet very different) Artificial Intelligence techniques have been reviewed and applied to meet the project goals. In particular, Particle Swarm Optimization has been run for locating thin layers whilst Case-Based Reasoning has been used to characterise them.

Despite using synthetically generated data for implementing and evaluating the methods, they featured great dimensionality and high noise perturbation in an attempt to build a robust system able to deal with the unpredictable real data. These two facts have been crucial in selecting the two AI techniques we have employed.

Furthermore, Discrete Fourier Transform has been applied to drastically reduce the data dimensionality and a smoothing technique has been required to soften the consequences of noise.

Concentration and thickness have been predicted by means of a linear regression model.

The methods' suitability and robustness have been assessed by performing different tests focusing on the thin layer detection performance, algal group classification and concentration and thickness prediction.

In the second phase, the information collected during descent has been used to make informed decisions about the best locations where thin layers are present. Specifically, a Case-Based Reasoning technique has been enhanced by adding a module that wisely manages the data found at descent.

Some more specific conclusions are:

1. The resulting system is able to predict the location of the thin layer to be found during ascent with high accuracy and acceptable error.

2. Variability on thin layer's properties has been used to test the system's strengths and weaknesses, being the thickness variations the ones that affect the most.

3. Finally, the need for a real large Case Base has been identified, which would lead to remedy the effect of variations

The methodology presented in this work should be applied whenever real data from sensors is feasibly available. However, further studies are needed to face this new scenario. Another relevant research line, which might be found in nature, is a mixture of several algal types in a single thin layer. Hence, this could be an interesting approach that remains future work. / FInally, adequate Case Base maintenance policies should be defined to improve the system's performance. This fact will be specially relevant if real data is more heterogenous than our synthetic data.

Acknowledgments

This work was partially funded by projects ANERIS (CSIC-PIF08-15-2), Next-CBR (TIN2009-13692-C03-01), and by the Generalitat de Catalunya under the grant 2009-SGR-1434.

References

A. Aamodt. A knowledge representation system for integration of general and case-specific knowledge. In *Proceedings from IEEE Tools with Artificial Intelligence (TAI)*, pages 836–839, 1994.

R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, volume 730 of *Lecture Notes in Computer Science*, pages 69–84. Springer Verlag, 1993.

P. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII*, pages 601–610. Springer, 1998.

T. Bartz-Beielstein, D. Blum, and J. Branke. Particle swarm optimization and sequential sampling in noisy environments. In *Metaheuristics. Progress in Complex Systems Optimization*. Springer, 2007.

R. Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-Based Applications (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence)*. Springer, October 2002.

R. Bergmann, M.M. Richter, S. Schmitt, A. Stahl, and I. Vollrath. Utility-oriented matching: A new research direction for case-based reasoning. In *Proceedings of the Ninth German Workshop on Case-Based Reasoning*, 2001.

R. Bergmann, J. Kolodner, and E. Plaza. Representation in case-based reasoning. *The Knowledge Engineering Review*, 20:209–213, 2006.

P.K. Bjørnsen and T.G. Nielsen. Decimeter scale heterogeneity in plankton during a pycnocline bloom of gyrodinium aureolum. *Marine Ecology Progress Series*, 73:263–267, 1991.

T. Blackwell. Particle swarm optimization in dynamic environments. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 29–49. Springer, 2007.

T. Blackwell and J. Branke. Multiswarm, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006.

K. Börner. Structural similarity as guidance in case-based design. In *Proceedings of the First European Workshop on Case-Based Reasoning*, pages 197–208, 1983.

M.G. Brown. An underlying memory model to support case retrieval. In S. Wess, K.D. Althoff, and M. Richter, editors, *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 1994.

H. Bunke and B.T. Messmer. Similarity measures for structured representations. In S. Wess, K.D. Althoff, and M. Richter, editors, *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Computer Science*, pages 106–118. Springer, 1994.

E.J. Carpenter, S. Janson, R. Boje, F. Pollehne, and J. Chang. The dinoflagellate dinophysis norvegica: biological and ecological observations in the baltic sea. *European Journal of Phycology*, 30:1–9, 1995.

M. Clerc. *Particle swarm optimization*. ISTE, 2006.

M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6 (1):58–73, 2002.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

T.J. Cowles and R.A. Desiderio. Resolution of biological micro-structure through in situ fluorescence emission spectra. *Oceanography*, 6(3):105–111, 1993.

T.J. Cowles, R.A. Desiderio, and M.E. Carr. Small-scale planktonic structure; persistence and trophic consequences. *Oceanography*, 11(1):4–9, 1998.

S. Craw, J. Jarmulak, and R. Rowe. Maintaining retrieval knowledge in a case-based reasoning system. *Computational Intelligence*, 17(2):346–363, 2001.

M.M. Dekshenieks, P.L. Donaghay, J.M. Sullivan, J.E.B. Rines, T.R. Osborn, and M. Twardowski. Temporal and spatial occurrence of thin phytoplankton layers in relation to physical processes. *Marine Ecology Progress Series*, 223:61–71, 2001.

B. Díaz-Agudo and P. González-Calero. A declarative similarity framework for knowledge intensive cbr. In L. Monostori, J. Váncza, and M.Ali, editors, *Proceedings of ICCBR 2001*, volume 2080 of *Lecture Notes in Artificial Intelligence*, pages 158–172, 2001.

P.L. Donaghay and T.R. Osborn. Toward a theory of biological-physical control of harmful algal bloom dynamics and impacts. *Limnol Oceanogr*, 42(5):1283–1296, 1997.

P.L. Donaghay, H.M. Rines, and J.M. Sieburth. Simultaneous sampling of fine scale biological, chemical and physical structure in stratified waters. *Arch Hydrobiol*, 36:97–108, 1992.

M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italie, 1992.

R.C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, pages 39–43. IEEE, 1995.

B. Falkenhainer, K.D. Forbus, and D. Gentner. The structure mapping engine: algorithm and examples. *Artificial Intelligence*, 41:1–63, 1990.

J.L. Fernandez-Marquez and J.L. Arcos. An evaporation mechanism for dynamic and noisy multimodal optimization. In *Proceedings of the 10th Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 17–24, 2009.

F. Gebhardt, A. Voß, W. Gräther, and B. Schmidt-Belz. *Reasoning with Complex Cases*. Kluer Academic Publishers, 1997.

A. Goel, J. Kolodner, M. Pearce, R. Billington, and C. Zimring. Towards a case-based tool for aiding conceptual design problem solving. In R. Bareiss, editor, *Proceedings of DARPA Workshop on Case-Based Reasoning*, pages 109–120. Morgan Kaufmann, 1991.

A. Gómez de Silva Garza and M.L. Maher. A process model for evolutionary design case adaptation. In *Proceedings of the Artificial Intelligence in Design Conference*, pages 393–412. Kluwer Academic Publishers, 2000.

H.R. Gordon, O.B. Brown, and M.M. Jacobs. Computed relationships between the inherent and apparent optical properties of a flat, homogeneous ocean. *Applied Optics*, 14:417–427, 1975.

K.J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, 1989.

K.J. Hammond. Explaining and repairing plans that fail. *Artificial Intelligence*, 45:173–228, 1990.

A.K. Hanson and P.L. Donaghay. Micro- to fine-scale chemical gradients and layers in stratified coastal waters. *Oceanography*, 11(1):10–17, 1998.

T. Hendtlass. A combined swarm differential evolution algorithm for optimization problems. In L. Monostori, J. Váncza, and M.Ali, editors, *Proceedings of the 14th international conference on industrial and engineering applications of artificial intelligence and expert systems (IEA/AIE)*, volume 2070 of *Lecture Notes in Computer Science*, pages 11–18. Springer, 2001.

F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In E. Krasner, editor, *The ubiquity of chaos*, pages 233–238. AAAS Publications, 1990.

N. Holden and A.A. Freitas. A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In *Proceedings of the IEEE swarm intelligence symposium (SIS)*, pages 100–107. IEEE, 2005.

D.V. Holliday, R.E. Pieper, C.F. Greenlaw, and J.K. Dawson. Acoustical sensing of small scale vertical structures in zooplankton assemblages. *Oceanography*, 11(1):18–23, 1998.

S. Janson and M. Middendorf. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on System Man and Cybernetics B*, 35(6):1272–1282, 2005.

S. Janson and M. Middendorf. A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):329–354, 2006.

P.W. Johnson, P.L. Donaghay, E.B. Small, and J.M. Sieburth. Ultrastructure and ecology of perispira ovum (ciliiophora: Litostomatea): an aerobic, planktonic ciliate that sequesters the chloroplasts, mitochondria, and paramylon of euglena proxima in a micro-oxic habitat. *Journal of Eukaryotic Microbiology*, 42(3):323–335, 1995.

J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks IV*, pages 1942–1948, 1995.

M.K. Kibler and D. Aha. Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 24–30. Morgan Kaufmann, 1987.

J. Kolodner. *Case-based Reasoning*. Morgan Kaufmann, 1993.

P. Koton. Reasoning about evidence in causal explanations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 256–261. AAAI Press, 1988.

D.B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons and Future Directions*. MIT Press, 1996.

D.B. Leake, A. Kinley, and D. Wilson. Learning to integrate multiple knowledge sources for case-based reasoning. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 674–679. Morgan Kaufmann, 1997.

M. Lenz and H.D. Burkhard. Case retrieval nets: Basic ideas and extensions. In *Proceedings of the 20th Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence*, volume 1137 of *Lecture Notes In Computer Science*, pages 227–239. Springer, 1996.

J.J. Liang and P.N. Suganthan. Dynamic multiswarm particle swarm optimizer (dms-pso). In *Proceedings of the IEEE swarm intelligence symposium (SIS)*, pages 124–129. IEEE, 2005.

R. Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. Maher, M. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson. Retrieval, reuse, revision, and retention in CBR. *Knowledge Engineering Review*, 20(3):215–240, 2005.

F.J. Martin and E. Plaza. Ceaseless case-based reasoning. In P.A. Gonzalez and P. Funk, editors, *Advances in Case-Based Reasoning. Proceedings of the 7th European Conference on Case Based Reasoning (ECCBR 2004)*, volume 3155 of *Lecture Notes in Artificial Intelligence*, pages 287–301. Springer-Verlag, 2004.

R. Mendes. *Population topologies and their influence in particle swarm performance*. PhD thesis, Departamento de Informatica, Escola de Engenharia, Universidade do Minho, 2004.

S. Minton. Qualitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391, 1990.

C. D. Mobley and L. K. Sundman. Hydrolight-ecolight 5.0 user's guide. Technical report, Sequoia Scientific, Inc., Bellevue, WA, 2008.

H. Muñoz-Avila. Case-base maintenance by integrating case index revision and case retention policies in a derivational replay framework. *Computational Intelligence*, 17(2): 280–294, 2001.

T.G. Nielsen, T. Kiørboe, and P.K. Bjørnsen. Effect of a chrysochromulina polylepis subsurface bloom on the planktonic community. *Marine Ecology Progress Series*, 62:21–35, 1990.

K.E. Parsopoulos and M.N. Vrahatis. Particle swarm optimizer in noisy and continuously changing environments. *Artificial Intelligence and soft computing*, pages 289–294, 2001.

T. Peram, K. Veeramachaneni, and C. Mohan. Fitness-distance ratio based particle swarm optimization. In *Proceedings of the IEEE swarm intelligence symposium (SIS)*, pages 174–181. IEEE, 2003.

M. J. Perry. Optical sensors. Prepared for the ALPS Workshop on Autonomous and Lagrangian Platforms and Sensors, La Jolla, California, March - April 2003.

E. Plaza. Cases as terms: A fetaure term approach to the structured representation of cases. In M. Veloso and A. Aamodt, editors, *Proceedings of the First International Conference on Case-Based Reasoning 1995*, volume 1010 of *Lecture Notes in Artificial Intelligence*, pages 265–276. Springer, 1995.

R. Poli. An analysis of publications on particle swarm optimization applications. Technical Report CSM-469, Department of Computer Science, University of Essex, 2007.

R. Poli and C.R. Stephens. Constrained molecular dynamics as a search and optimization tool. In *Proceedings of the 7th European conference on genetic programming (EuroGP)*, volume 3003 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2004.

R. Poli, C. Di Chio, and W.B. Langdon. Exploring extended particle swarms: a genetic programming approach. In *GECCO 2005: Proceedings of the 2005 conference on genetic and evolutionary computation*, pages 169–176. ACM, 2005a.

R. Poli, W.B. Langdon, and O. Holland. Extending particle swarm optimization via genetic programming. In *Proceedings of the 8th European conference on genetic programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 291–300. Springer, 2005b.

R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. an overview. *Swarm Intelligence*, 1:33–57, 2007.

B.W. Porter, R. Bareiss, and R.C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.

J. Pugh, A. Martinoli, and Y. Zhang. Particle swarm optimization for unsupervised robotic learning. In *Proceedings of the IEEE swarm intelligence symposium (SIS)*, pages 92–99, 2005.

A. Ram and J.C. Santamaría. Continuous case-based reasoning. *Artificial Intelligence*, 90: 86–93, 1993.

T. Reinartz, I. Iglezakis, and T. Roth-Berghofer. Review and restore for case-based maintenance. *Computational Intelligence*, 17(2):214–234, 2001.

J.E.B. Rines, P.L. Donaghay, M.M. Dekshenieks, J.M. Sullivan, and M.S. Twardowski. Thin layers and camouflage: hidden pseudo-nitzschia populations in a fjord in the san juan islands, washington, usa. *Marine Ecology Progress Series*, 225:123–137, 2002.

J. Robinson, S. Sinton, and Y. Rahmat-Samii. Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. In *Proceedings of the IEEE international symposium on antennas and propagation*, pages 314–317. IEEE, 2002.

R.C. Schank. *Dynamic Memory*. Cambridge University Press, 1982.

Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE international conference on evolutionary computation*, pages 69–73. IEEE, 1998.

J.M. Sieburth and P.L. Donaghay. Planktonic methane production and oxidation within the algal maximum of the pycnocline: seasonal fine-scale observations in an anoxic estuarine basin. *Marine Ecology Progress Series*, 100:3–15, 1993.

E. Smith and D. Medin. *Categories and concepts*. Harvard University Press, 1981.

B. Smyth and M.T. Keane. Retrieving adaptable cases. In S. Wess, K.D. Althoff, and M. Richter, editors, *Topics in Case-Based Reasoning*, Lecture Notes in Computer Science, pages 209–220. Springer, 1994.

B. Smyth and M.T. Keane. Experiments on adaptation-guided retrieval in case-based design. In *Proceedings of the First International Conference on Case-Based Reasoning*, pages 313–324. Springer, 1995a.

B. Smyth and M.T. Keane. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 377–383. Morgan Kaufmann, 1995b.

B. Smyth and M.T. Keane. Using adaptation knowledge to retrieve reusable designes. *Knowledge-Based Systems*, 9(2):127–135, 1996.

B. Smyth and M.T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293, 1998.

B. Smyth and E. McKenna. A portrait of case competence: Modelling the competence of case-based reasoning systems. In *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 208–220. Springer, 1998.

B. Smyth and E. McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.

M. Stramska and D. Stramski. Effects of a nonuniform vertical profile of chlorophyll concentration on remote-sensing reflectance of the ocean. *Applied Optics*, 44:1735–1747, 2005.

P.N. Suganthan. Particle swarm optimizer with neighbourhood operator. In *Proceedings of the IEEE congress on evolutionary computation (CEC)*, pages 1958–1962. IEEE, 1999.

E. Tulving. Episodic and semantic memory. In E. Tulving and W. Donaldson, editors, *Organization of memory*. Academic Press, 1972.

M. Veloso and J.G. Carbonell. Case-based reasoning in prodigy. In R. S. Michalski and G. Teccuci, editors, *Machine Learning: A Multistrategy Approach Volume IV*, pages 523–548. Morgan Kaufmann, 1994.

C. Wei, Z. He, Y.Zhang, and W. Pei. Swarm directions embedded in fast evolutionary programming. In *Proceedings of the IEEE congress on evolutionary computation (CEC)*, pages 1278–1283. IEEE, 2002.

J.R. Zaneveld and W.S. Pegau. A model for the reflectance of thin layers, fronts, and internal waves and its inversion. *Oceanography*, 11(1):44–47, 1998.