Evolutionary Computing for Detection of Retentive Structures in Coastal Waters

Marc Segond, Denis Robilliard, Virginie Marion, Cyril Fonlupt
Laboratoire d'Informatique du Littoral,
Maison de la Recherche Blaise Pascal,
50 rue Ferdinand Buisson - BP 719,
62228 CALAIS Cedex, France
Email: {segond,robillia,poty,fonlupt}@lil.univ-littoral.fr,

***Abstract*— The demography of anchovy fishes in the Gulf of Biscay seems to be related to the presence of so-called "retentive" hydrodynamical structures, that keep fish eggs and larvae in a favorable environment. To verify this hypothesis, an automatic detection tool is needed to process a database of thousands of hydrodynamical maps to be compared with biologists observations and fisheries statistics. The results could be used to decide fishing quotas or bans for the sake of preserving the natural resource. We propose two evolutionary schemes, one based on ant systems, the other on induction of genetic programming filters, in order to automatically detect these retentive structures. Our heuristics are shown to be competitive with human experts and are compared to other detection schemes.**

***Index Terms*— Evolutionary computation, genetic programming, iterative filters, automatic structure detection**

I. Introduction

Problem overview

Presence and density of animal species in the ocean and coastal waters are strongly influenced by the location of physical structures, such as upwellings, temperature fronts or vortices. In the case of the anchovy in the Gulf of Biscay, oceanographers and biologists from the Ifremer[1] institute are studying the correlation between so-called "retentive" structures and the demography of these fishes. Retentive structures are meso-scale vortices, whose size ranges from 10 km to 200 km, that keep fish eggs and larvae in favourable environmental conditions, different from those encountered outside the structure, and that last long enough to allow fish growth. The presence of these retentive structures during Spring is dependent upon the weather, and it is supposed to be a good predictor for future fish demography, and this, in turn, could allow to decide adequate fishing quotas or bans to preserve this natural resource.

Verifying this hypothesis is not obvious: a first major problem lies in efficiently identifying such hydrodynamical patterns in massive data-sets, in order to match their presence against biologists observations and fisheries statistics for several years.

[1]French Research Institute for Exploitation of the Sea

Characterizing the detection task

At this stage of the study, there is no formal model for these meso-scales retentive structures, but rather their detection is made by experts on stream vector maps. During this process some structures that could be retained by a naive observer are rejected, e.g. because the stream aspect is chaotic in the neighborhood, suggesting these are only transient patterns or artifacts due to the model digitization.

When specialists highlight retentive structures by hand on the stream maps, they certainly use hidden expertise about plausible structures, and knowledge about the typical stream dynamic in the Gulf of Biscay and of the characteristics of the simulation model. Thus the physics-based vortices detection problem is probably mixed with a hidden criteria learning task. An efficient detection scheme for this problem may therefore build over these two aspects: using hydrodynamics and being able to learn part of the expert's knowledge.

Maps and data set

The data comes from a large set of vector fields issued from more than ten years of daily hydrodynamic simulations of the Gulf of Biscay. The simulations are based on two 3D hydro-dynamical models: Mars3D [1] developed by Pascal Lazure at Ifremer laboratory and the Mercator model[2]. A typical stream map is a 2 dimensional matrix containing the orthogonal u and v components of the stream vector at 10 meters depth for every intersection (x, y) of a discrete grid with 10km by 10km cells, as illustrated on Figure 1. Maps are collected at regular time steps, usually every 24 hours and are stored in the NetCDF[3] format. One needs to record the location of interesting vortices over many years in order to conduct further statistical analyses. This yields a very large amount of maps to be processed, thus an automatic and efficient detection tool is needed to conduct the study.

Vorticity-based methods

As a starting point, two "classical" state-of-the-art vector fields analysis methods have been tested, one based

[2]http://www.mercator-ocean.fr
[3]http://my.unidata.ucar.edu/content/software/netcdf/index.html

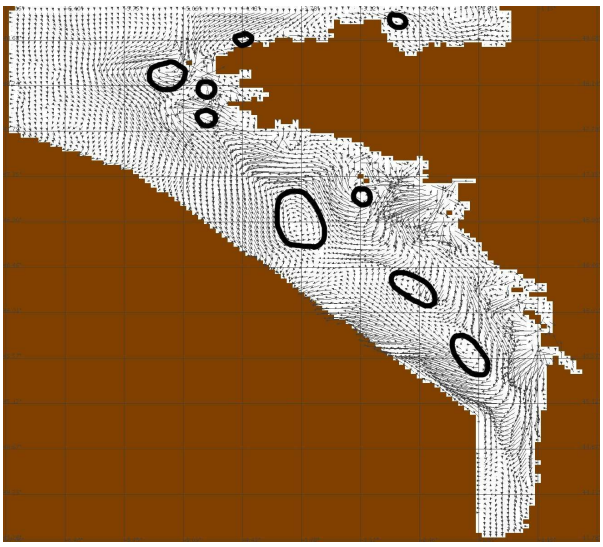

Figure 1. An example of detection performed by an expert: interesting vortices are circled in black. Every vortex will be digitized as a set of map cells. The coastline (at 10 meters depth) of South Western France can be recognized on the picture right, while the grey part in the lower left corner was not output by the hydrodynamical simulator in this example.

on the vorticity of the vector field, the other based on the Helmholtz decomposition of this vector field. In the first one, the vorticity (or *curl*) in each cell is computed, which can be done on a discrete field by a convolution, and then a threshold is applied on this scalar field to extract significant features (see [2]).

The second method relies on the Helmholtz decomposition of the vector field $\omega(x, y) = (u(x, y), v(x, y))$ into two fields, one solenoidal field ω_{so} and one irrotational field ω_{ir} such that $\omega = \omega_{so} + \omega_{ir}$ (see [3]).

From the knowledge of $\omega_{so} = (u_{so}(x, y), v_{so}(x, y))$, one can build the orthogonal field $\omega_{so}^{\perp} = (-v_{so}(x, y), u_{so}(x, y))$, and then integrate a scalar potential function Ψ such that:

$$w_{so}^{\perp} = \nabla\Psi$$

The extrema of Ψ are characteristic points of the solenoidal field, and vortices centers can be extracted. The vortices spatial extensions can then be modeled by Rankine functions. The interested reader may also refer to [4] for a good review of the state of the art.

Although theoretically sound, these classical vortices detection schemes do not comply well with the problem at hand. Their main weakness is they are based on the local vorticity (or curl), while we are searching for areas where the stream draws some kind of loop which does not need to be circular, and which dimension is up to 200 km. It is obviously difficult to assess the presence of such structures by using only local information. In open ocean, where stream perturbations are low, classical schemes perform quite well since retentive structures are mainly large, almost circular, well-defined vortices, but when it comes to coastal waters, the shallow depths and

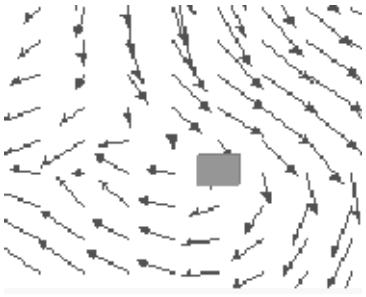

Figure 2. A typical example of a hard case for automatic detection: a strong local extremum of curl that triggers classical methods but does not match with a retentive structure. This map detail is taken from the surroundings of South Brittany.

coastal river plumes induce many deviations of the stream circulation. This results in the detection of inaccurate structure, as shown in Figure 2.

Thus we think the identification of retentive structures certainly need to take into account information coming from larger areas of the stream vector map. The relative failure of these classical methods also hints at the possibility to try to learn part of the experts knowledge in order to perform a better detection, as said above.

Streamlines method

An alternative choice to vorticity-based schemes consists in choosing a "Lagrangian point of view": retentive structures could be retrieved by dropping virtual buoys and following their trajectories. Neglecting buoys mass and water viscosity, this be approximated by integrating the streamlines of the vector field, e.g. as illustrated in [5]. The detection of looping streamlines can be done by tracking trajectories that come close enough to any given point at repeated time intervals.

Although simple in concept, applying this scheme needs to determine places on the map where to seed streamlines (usually random, or regular grid-wise places), and to set a threshold radius to detect looping trajectories. These are two tricky points, since small variations in these parameters can yield very different detection results. As it is reported in the result section, results were a lag behind those from evolutionary methods (see also [24]).

Evolutionary methods - Ant algorithms and genetic programming

In the following, we first present a detection heuristic based on a new and very active paradigm, ant algorithms. Although mostly used for combinatorial optimisation tasks, we show that ant algorithms can be efficient pattern detection schemes. Another original and important characteristic of our ant algorithm is its use of several teams of ants, distinct teams focusing on the detection of distinct structures. Through an individual behaviour that remains easy to program, artificial ants are able to collaborate in order to make retentive structures emerge, taking into account both local and global information on the map.

This gives an original and competitive heuristic, but the ability to learn specific traits of a given area is somewhat crude: it is limited to tuning the number of ants, teams, iterations and a bias parameter favouring exploration versus exploitation of the so-called pheromone matrix that is characteristic of ant systems. Anyway this tuning can be done on a set of example maps, providing a kind of limited parametric learning.

Then, in a second part of the paper, we tackle this problem as a plain non-parametric learning task, that we solve with a refinement of genetic programming (GP) filters trained on example maps. It has been shown [6] that evolution of GP filters can provide efficient pattern detection heuristics, that are able to take into account the knowledge an expert has of the land features he is studying. A problem we encountered again is accessing more global information on the map than is usual through the restricted "window" that is read by standard GP filters. We propose an iterative wrapper that allows them to gather information across the map without need for a large increase of the number of GP terminals, that would otherwise render the search fruitless. This scheme has been first introduced in [7], we provide here a more formal description of the iterative code wrapper, and a detailed comparison with other techniques, notably assessing the advantages and drawbacks of the ant algorithm versus iterative GP filters.

The remainder of this paper is organized in the following way. Section II gives a brief overview of ants in nature and how the ants foraging mechanism has given birth to artificial ant algorithms. Section III describes the main principles used by our ant algorithm and how we adapt the ant paradigm to automatic retentive structure detection. Section IV introduces the genetic programming filters concept and Section V shows how this scheme has been improved to incorporate global information. Section VI describes the experimental results, before conclusion.

II. An overview of ant algorithms

Behavior of social insects like ants, bees, termites and some wasps is characterized by a kind of self-organization called *stigmergy*. This means that interactions between insects are mostly based on local information without explicit references to any global goal. Individuals typically communicate by changing local properties of their environment, e.g. chemical deposits dropped by ants, and through this limited medium of communication some kind of distributed and collective intelligence emerges [8], [9].

Real ants have inspired a family of "ant algorithms", that were introduced with Marco Dorigo's PhD. The basic idea is to mimic the cooperative behavior of an ant colony, and this has been used mainly in order to solve large combinatorial optimization problems [10], [11], [12], [13]. Primary works have been based on a simple model of the ant foraging mechanism, that allows an ant group to find the shortest path between its nest and some food source. During their trip, ants leave a chemical trail of *pheromone* on the ground, whose role is to guide the other ants towards the target point. This chemical substance evaporates and thus has a decreasing action over time, and the quantity left by one ant depends on the amount of food it found. Every ant chooses its path partly at random and partly according to the quantity of pheromone it smells in its neighborhood.

This model of communicating ants has been used as a framework for solving combinatorial optimization problems like the TSP [11], [14], [15], routing problems, load balancing in communication networks [16], numerical optimization [17], graph coloring problem [18],... New trends based on the behavior of social insects have also appeared these last years (modeling of ants chemical recognition system, ...) and have led to new topics of research: classification [19], automatic music generation and automatic painting[4] for instance.

III. Marsouin: an ant algorithm for retentive structure detection

In ant algorithms, new solutions are usually created by virtual ants using some quality function of the local environment, e.g. length of edges for a graph shortest path problem, and some simulation of pheromone communication, e.g. a matrix of pheromone concentrations. A typical basic ant algorithm can be summed up as a loop over three main steps:

- generation of solutions by ants according to local and pheromone information
- application of a local search scheme to improve the solutions found (although not inspired from the biological model, this phase is necessary to obtain competitive results on many combinatorial search problems);
- update of the pheromone information to simulate new deposits and evaporation

A. Ants behavior

Real ants live in colonies and are able to distinguish both their own trail and that of their colony fellows. In our case, we consider a population of ants divided into several colonies, called *teams*, sharing a specific pheromone that does not attract ants from other teams. This is done in order to allow the teams to focus on different vortices. These virtual ants are randomly spawned on the grid points (i.e. intersections) of the 2-dimensional map. Each point is characterized by the speed and direction of the stream, and it also has two arrays of pheromone concentrations, one indexed by ants and the other indexed by teams. This allows an ant to detect its own trail and the composite trail of its own team. Pheromone concentrations are initialized to the null value.

Ants iteratively move from a map point to one of its 8 closest neighbors. These moves are synchronized and thus all ants move at the same rate. The choice of an ant

[4]see `http://www.antsearch.univ-tours.fr/WebRTIC/default.asp?FCT=DP&ID_PAGE=52`

next position is determined according to a stochastic rule modulated by four parameters:

1) the direction of the stream is the main parameter in the choice of the next move: ants are allowed a maximum 45 degrees deviation from the direction of the stream
2) the velocity of the stream: the higher its speed, the higher the chance to follow its direction
3) the level of pheromone dropped by all ants of the same team on neighboring map points: the higher the level, the more attractive the point
4) an individual random directional bias, that is drawn with a flat distribution at the creation of every ant: as stream vectors are always tangential to any rotating circulation of the stream, this directional bias is useful to allow ants to follow curved trajectories.

The general course of the algorithm is changed on three occurrences:

- when an ant reaches the coast, it is destroyed and re-spawned at random elsewhere
- when an ant loops over its own individual trail, the looping part of the trail is recorded to be examined latter as a candidate retentive vortex (see next subsections). The ant continues its trip.
- when a given number of moves (i.e. iterations of the algorithms) has been performed, the algorithm stops.

Notice that this simple set of rules takes into account both vorticity and divergence: in the presence of a flow that is both rotating and diverging, ants may be efficiently prevented from achieving a loop around a vorticity extremum. Notice also that the directional bias is attributed once for all to a given ant: some of them will always favour left turns, others right turns, some others straight moves. As the detection of a vortex may need all types of moves, a run of the algorithm is a collaborative process through both the pheromone communication and the directional bias.

Experiments have been performed with a number of teams ranging from 1 to 10, and population sizes as small as 10 ants up to 500, adapting the number of iterations to keep the same computational effort. Bigger population, and secondarily a greater number of teams, provide less variance in the detection as can be expected due notably to the greater number of directional biases that are sampled.

B. Local search

Almost all ant algorithms are hybridized with a local search method in order to enhance their performance. In our case, there is no true local search scheme, but a simple filter is applied when a candidate vortex has been found: looping paths that are too short to be compatible with that of a meso-scale structure (less than 40 kilometers) are discarded, as are "forward-backward" moves, i.e. flat loops, that may happen in places where the stream is chaotic.

C. Pheromone update

Many different pheromone update schemes have been proposed in the literature [13], [11], [20], [21]. They usually consist in two parts: evaporation and amplification. During the evaporation phase, all pheromone trails are decreased according to a given value (percentage, probability...) while during the amplification phase the best pheromone trails are increased, in analogy with real ants increasing their deposits after finding a food source.

As a candidate vortex can be found at any time step, evaporation in Marsouin is done after every move phase, and not as a dedicated step performed after having built complete solutions as in many other ant algorithms. The evaporation factor ρ is an input parameter, and experiments show that values between 0.1 and 0.5 provide interesting results for the set of maps we used. For every map point (i, j) the following evaporation formula is applied on all pheromone values τ:

$$\tau_{i,j}^{\text{new}} = (1 - \rho)\tau_{i,j}^{\text{old}}$$

Standard ways of performing amplification can be roughly split up into two sets: either only the "best" ant can improve its trail; or all ants improve their trails in proportion to the quality of the solution they just built. The intuitive idea is that amplification should take place when a given task has been fulfilled, e.g. completing a circuit through all cities in the classic Traveling Salesperson Problem.

Here, our implementation is very close to the natural model: each time an ant moves on a map point, it drops some pheromone. The reinforcement effect is computed according to a classical update scheme:

$$\tau_{ij}^{\text{new}} = \tau_{ij}^{\text{old}} + Cst$$

We also tried an extra amplification of pheromone dropped on looping trails, in order to model increased deposits after finding a goal. However it did not bring significant improvement, since a looping ant already increases the amount of pheromone dropped on its solution.

D. Final step

Once the ant algorithm is over, we perform a final analysis of the candidate structures:

- structure barycenters are computed
- when the intersection of several structures contains all their barycenters, these structures are merged into a larger one, and a new barycenter is computed

Thus concentric or almost concentric cyclic paths are merged together to obtain a unique largest "envelope" of the retentive vortex structure. Then the coordinates of the center, the direction of rotation and the area are computed and recorded.

As we said above, although this scheme was simple to design and effective, it does not allow significant learning of an expert knowledge. To assess the possibilities of evolutionary learning, a genetic programming scheme has

also been developed, and is detailed in the next two sections.

IV. Genetic programming filters

Our basic GP scheme is inspired by the work of Daida [6] on detecting pressure ridges in the arctic ice cover. We evolve filters (i.e. classifier programs) in a supervised learning framework. These are selected on their ability to correctly classify cells of a stream map whether they belong or not to a structure of interest. Each filter classifies one map cell at a time, and it is successively applied to every cell of the map. Evaluation is done against a set of reference maps tagged by the expert. In contrast to the previous ant algorithm, where ants are seeking to follow a vortex frontier moving from a grid intersection to another, here we try to find not the vortex frontier but the grid cells belonging to the vortex area.

GP filters have been implemented with the ECJ[4] Java evolutionary library, using the standard Lisp-like tree representation. Inputs available to a filter are floating point physical data such as stream strength and vorticity. We enforce the so-called closure property and use only GP nodes that return a floating point value.

The conversion between the floating point returned by the GP tree and the boolean value expected for a classification problem can be done in the standard way, i.e. by determining a threshold level that divide results between lower values interpreted as negative answers and higher values that are considered positive. Continuously increasing the threshold from the lowest returned value to the highest one, we obtain a monotonous increase of the true positive and false positive rates, from 0% to 100%, thus we can draw a Receiver Operating Characteristics (ROC) curve. This is a standard technique (see e.g. [22]) that will be used later when evaluating and comparing heuristics. The end-user will have the choice of the threshold level that corresponds to his preferred trade-off between sensibility and specificity.

A. Basic GP filters presentation

The set of function and terminal nodes is shown in table I, and it has been chosen to allow computations on the physical characteristics of the stream.

For example, it seems relevant to use information from the 8 neighbors of the cell we are working on: the "strength3x3" terminal returns the mean value of the stream strength in the neighboring cells, and the "angle3x3" terminal gives the mean value of the angle of the vector stream in those same cells. The "min" and "max" function nodes have been introduced to allow simple comparisons. The "curl" and "divergence" are standard operators used in vortices detection. Notice that in order to speed up the evaluation phase, most terminal nodes (curl, divergence, strength, strength3x3, angle3x3) are pre-computed for the maps in the learning set. The evolution parameters are shown in table II, and are quite standard.

Name	Value
Number of generations	80
Size of the population	600
Max depth for a tree	15
Mutation rate	5%
Crossover rate	85%
Reproduction rate (with elitism)	5%

TABLE II.
General parameters used in the GP algorithm.

B. Fitness function choice

One of the difficulty in Genetic Programming is to find the adequate fitness function to optimize. Basically, we evaluated the fitness of individuals by measuring their performance on a learning set of 10 maps tagged by an expert. However the actual performance of a filter would depend on the choice of the threshold level as explained above. A different approach consists in maximizing the area under the ROC curve, denoted as AUC — Area Under Curve —, so the need for a threshold is relieved. This is now a standard way of doing, e.g. see Sebag et al. [23] for a discussion about efficient computation of the AUC.

Optimizing the AUC is efficient, but, as we will see later, the ant algorithm dominates when the threshold is set to obtain very low false-positive rates. We therefore propose another fitness function that focuses on having a steeper slope in the left part of the ROC curve (corresponding to the low false-positive rates). This can be achieved by choosing a set of 10 values on the ROC x-axis, 5 in the range $[0.25, 0.35]$ that we want to focus on, the others equally spaced on the range $]0, 1] \setminus [0.25, 0.35]$. Then we get the corresponding points (x_i, y_i) on the ROC curve and we maximize the following fitness function:

$$f = \frac{\sum_{i=1}^{n} \frac{y_i}{x_i}}{n}$$

C. Basic GP filters results and discussion

Although GP filters have been previously successful on classification and pattern detection cases, they did not give conclusive results on our problem. GP produced rough and noisy classification especially near the coast, that reminds of results obtained by vorticity analysis.

We conjecture that these filters have a too reduced "sight range" to recognize global vortices shapes that can be spread over 20 grid cells or more. We previously saw in the introduction that whether or not a cell is a member of a retentive structure certainly depends on the stream characteristics of this cell, but also on the relation it shares with surrounding cells that may or not be members of the same structure. In this regard, the "strength3x3" and "angle3x3" nodes probably give a too local information, and we need to add more problem specific knowledge to allow GP to cross the gap.

[4] http://cs.gmu.edu/eclab/projects/ecj/

Name	Meaning	Input	Output
add	addition	2 reals	1 real
sub	subtraction	2 reals	1 real
mul	multiplication	2 reals	1 real
div	protected division	2 reals	1 real
min	minimum of 2 arguments	2 reals	1 real
max	maximum of 2 arguments	2 reals	1 real
cos	cosine	1 real	1 real
sin	sine	1 real	1 real
strength	normalized stream strength	null	1 real $\in [0, 1]$
strength3x3	stream strength averaged over a 3x3 cells matrix	null	1 real $\in [0, 1]$
angles3x3	stream vector angle averaged over a 3x3 cells matrix	null	1 real
curl	cell vorticity	null	1 real
divergence	cell divergence	null	1 real
erc	ephemeral random constant	null	1 real

TABLE I.
SUMMARY OF NON-TERMINAL AND TERMINAL NODES USED IN THE BASIC GP FILTERS.

Experiments have been conducted to let the evolution process determine the size of these matrix-shaped terminals, but these were not successful probably due to the increased dimension of the search space. This has lead us to propose a solution based on the propagation of classification results across the map, as explained in the next section.

V. ITERATIVE GENETIC PROGRAMMING FILTERS

To remedy the failure of the previous scheme, we need to provide some means of communicating information over the map, while keeping a manageable search space: a large increase in the number of terminals to access a variety of distant cells would prevent successful learning by GP.

Our proposition is iterative filters, i.e. filters that are applied in several successive classification steps on a map, that can access a memory of neighbors previous decisions at each step to compute their new classification value, and that returns the decision computed in the last iteration. Indeed, if a filter incorporates a node that accesses the memory of previous classifications from neighbor cells, then step by step it will gain some information from distant cells as far as the number of previous iterations.

Iterative GP filters presentation

From a technical point of view, two terminal nodes are added to the node set: `lastValue` and `meanLastValue`.

- `lastValue`: returns a value that aggregates the filter results at previous iterations. This value is initialized at a given value (e.g. 0.5) for all cells during the first iteration (no previous classification results), and it is later updated using the following equation:

$$\texttt{lastValue}_{i+1} = \frac{2 * \texttt{lastValue}_i + F}{3}$$

  were $\texttt{lastValue}_i$ is the value returned by this terminal at iteration i, and F is the classification value computed by the filter at iteration i.

- `meanLastValue`: returns the mean of `lastValue` for the 8 neighboring cells.

In order to take into account this extra information, the fitness function is wrapped into an iterative scheme detailed in Table III. Notice that the pseudo code can be very easily modified to accommodate for a set of maps rather than a single one.

Thanks to the `meanLastValue` node, a filter is now able to take into account classification results from its immediate neighbors, aggregated over the previous iterations. Within successive iterations, it can grasp some classification information about cells distant from two, three or more grid cells, depending on the maximum number of iterations we allow. The classification values produced by the GP filter on every cell during the last iteration will serve to compute its ROC-based fitness.

Experiments also showed that it is very difficult for a filter to avoid false positives near the coast line, almost setting a higher bound to performances. To tackle this problem, a `distCoast` node has also been introduced that returns 1 if the cell is farther than 2 grid steps from the coast, else 0.

VI. EXPERIMENTS

In this section we present some illustrations of structure detections performed on a set of maps generated by the Ifremer laboratory. This set consists of 10 "snapshots" of the Gulf of Biscay during the year 1995. It sums up to 153 retentive structures reported by the expert, 1778 positive cells and 38152 negative cells.

A. Ant algorithm results

Figure 3 illustrates the results of the ant algorithm with two different settings, to be compared with the same map tagged by the expert in Figure 1. The running time is less than a minute on a 1.6 GHz PC, all structures are detected, two false positive patterns are found on the left, one false positive on the right. Notice that some of the vortices area are only roughly approximated. It is typical of what can be expected from this algorithm: most individual structures

```
declare lastValue as an array of floats indexed by cells of the map
declare F as an array of floats indexed by cells of the map
foreach cell of the map // initialize
   lastValue[cell] = 0.5;
endfor
for(int iter=0; iter < maxIter; iter++) // iterate filter
   foreach cell of the map // classify map cells
      F[cell] = evaluateGPTree(cell, lastValue); // compute classification for given cell
   endfor
   foreach cell of the map // aggregate previous classifications
      lastValue[cell] = (2 * lastValue[cell] + F[cell]) / 3;
   endfor
endfor
return F; // keep last classification
```

TABLE III.
PSEUDO CODE FOR ITERATIVE GP FILTERS CLASSIFICATION.

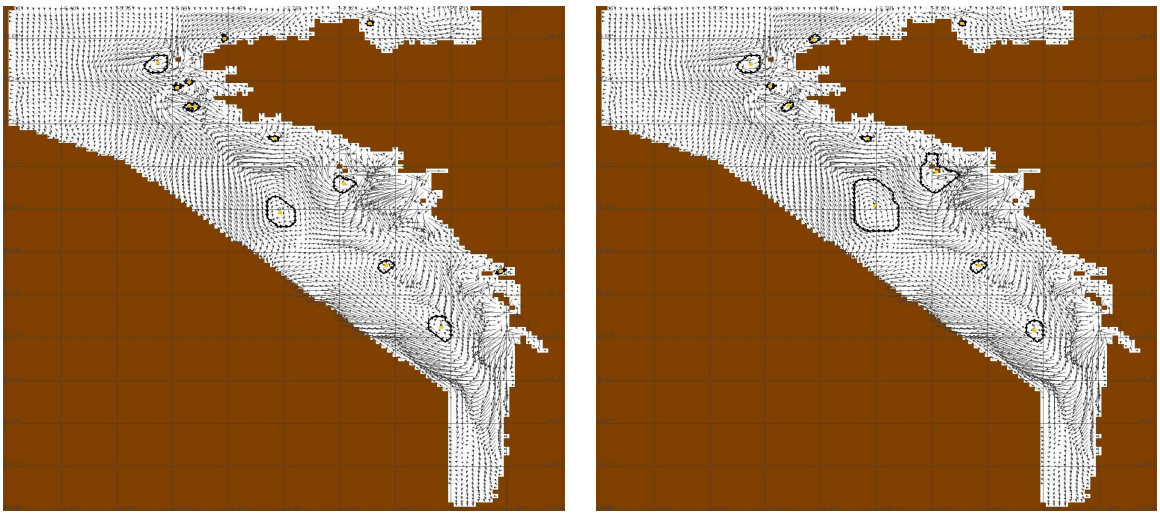

Figure 3. Structures detected by the ant algorithm, left with 1 team of 100 ants, right with 5 teams of 20 ants, both within a limit of 400 ant moves, to be compared with expert prediction on Figure 1.

are found but their size is not very precise, with some false positives.

It is to be noted that the expert worked first in a span of time limited to 2 hours on this set of 10 maps (so a little less than one minute per vortex), then the ant algorithm was launched and the expert had the opportunity to correct his first judgement on the basis of the algorithm results, to settle the reference maps. This was done in order to evaluate the level of human mistakes due to eye strain, and this led the human expert to add about 20% new structures that were forgotten in the first, time-limited, phase. Misses were mostly small structures on the most encumbered maps that contain up to 26 vortices, but sometimes even large structures have gone un-noticed. This stresses the benefits brought by our automatic detection scheme.

As this method performs quite well, it has been embedded within an automatic tracking scheme. This allows to automatically tag and follow retentive structures over several days/months. It allows biologists to gather interesting statistics, such as the number of retentive structures over a year, the average surface of these structures, the average lifetime and speed of displacement.

Iterative GP filters results

To illustrate the influence of the number of iterations on GP filters, Figure 4 plots the value $(1 - \mathrm{AUC})$ versus generations (the lower the curve, the better) for several parameter values. We can see that iterative filters have an increased efficiency, with a maximum at 6 iterations, which may be a parameter dependent on the typical size of vortices in our area. The `distCoast` node also boosts the performance, reducing the number of false positives within the neighborhood of the coast line, as is illustrated in Fig. 5. Notice that vortices are not output as individual structures, contrary to the ant algorithm.

B. Comparisons between heuristics

A comparison between iterative GP filters, the ant algorithm and streamlines is given in Figure 6 using ROC curves when possible and simple points elsewhere. For GP, depending on the trade-off desired either the "steeper slope" or the AUC fitness function may be preferred, since the "steeper slope" results are slightly better than the ant algorithm for very low false positives rates, but globally worse.

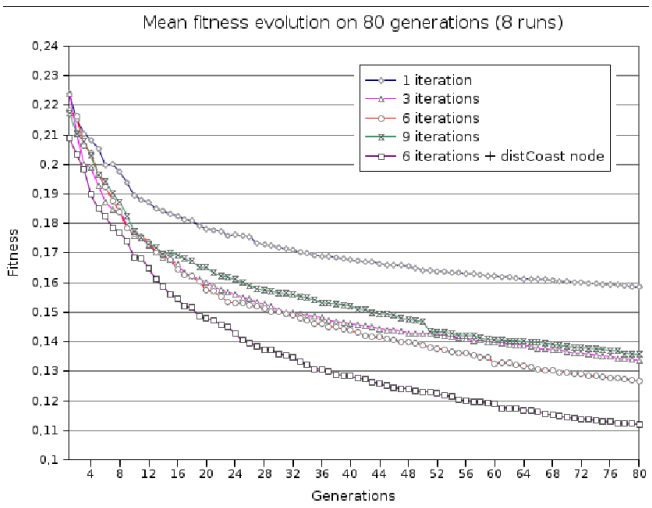

Figure 4. Comparison of fitness evolution for different iteration parameters.

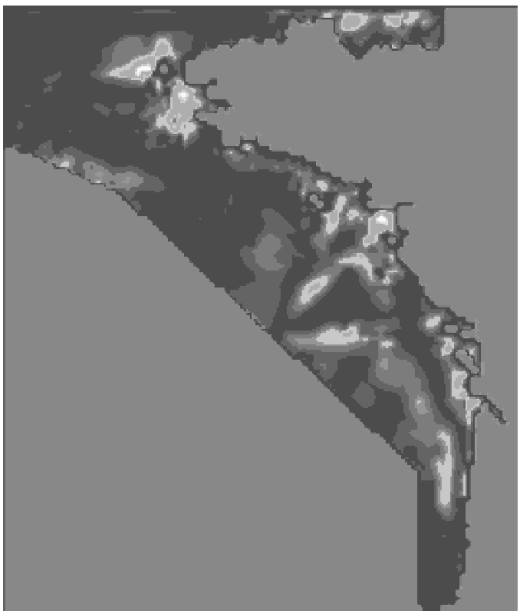

Figure 5. Example map filtered by GP using 6 iterations and the "distCoast" node.

We have led a cross-validation experiment at maximizing the AUC, with 6 iterations and `distCoast` node, using 9 maps as the learning set and evaluating the best GP filter on the last map. This has been done 10 times, varying the validation map, and results have been averaged. The averaged $(1-\text{AUC})$ value on the validation maps is 0.1179 with a standard deviation of 0.0344, thus the validation error remains close to the averaged learning error shown on Figure 4.

We have also performed a similar experiment with a non-recurrent back-propagation artificial neural network (see e.g. [25]), taking 54 inputs i.e. the same 6 terminal inputs as GP computed for 9 cells evenly spaced in a 70km x 70km area around the classification focus. Limiting the learning time to 15 min as for GP, we obtained an averaged (1-AUC) value of 0.2485 with a standard deviation of 0.0178. We do not claim to have spent as much time in tuning the artificial neural network (ANN) as we have spent for the GP algorithm, nonetheless it gives some hints about GP being competitive with ANN for this problem.

VII. Conclusion

In this article, we presented two innovative schemes for detection of retentive meso-scale vortices on simulated stream vector fields. New artificial intelligence techniques may be of help especially when such tasks have no well defined models in mathematical or physical terms. As the detection relies on information that is spread over space, and possibly on expert hidden criteria, classical methods for vector field analysis do not bring satisfying solutions.

The two approaches introduced are part of the active evolutionary computation field but with two different paradigms. We saw that the ant based algorithm gave good results due to its ability to overcome the difficulties that classical methods had. Its main advantage is its capacity to directly output individual structures, which allow to easily implement a movement tracking feature. The main drawback of this scheme is its small range of true over false positives ratios.

We presented iterative GP filters for detection of retentive meso-scale vortices on simulated stream vector fields. The GP-based method delivers a fully functional and fast program that can be executed on every map. This scheme has needed considerable insights into the problem in order to develop not only suitable GP functions and terminals, but also an original iterated scheme for GP classification.

With our GP based filtering method, we are able to learn some part of the expert knowledge, while also performing meaningful computations in term of vector field analysis, as can be judged by the results. It does not provide individual structures, but it is possible to adapt the true to false positives ratio to what the user need.

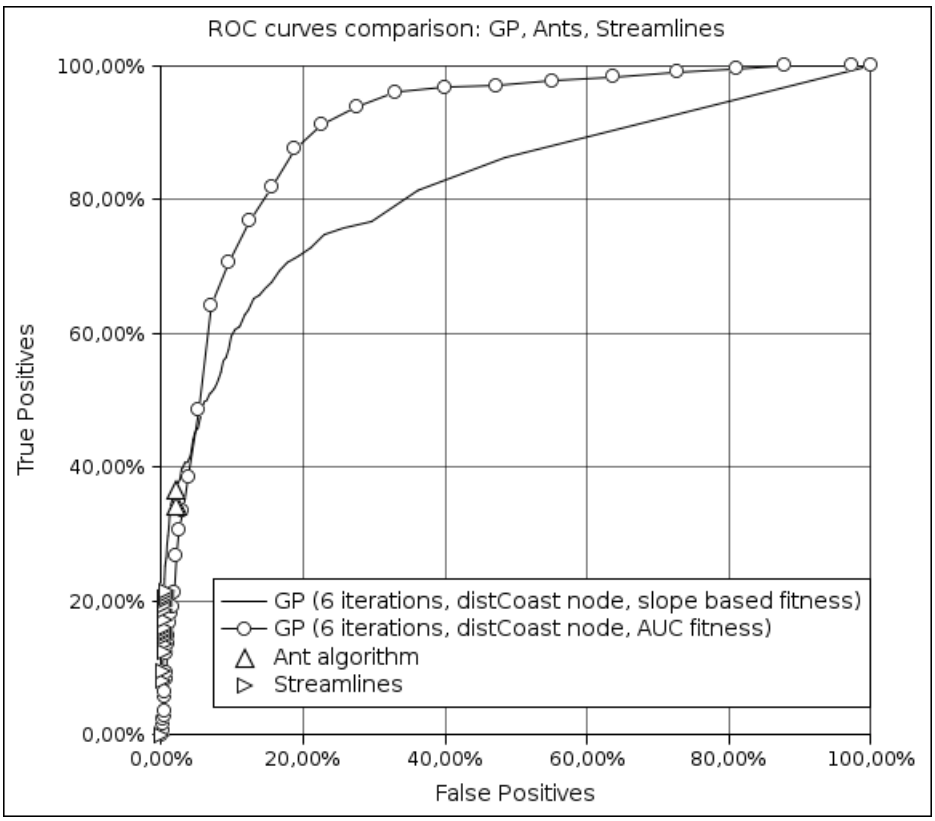

Figure 6. ROC curve based comparison between GP filters and other methods.

We think that the iterating scheme for GP classification may well be of interest in the image analysis domain and possibly for some general classification tasks.

REFERENCES

[1] Pascal Lazure and Anne-Marie Jégou. 3D modelling of seasonal evolution of loire and gironde plumes on biscay bay continental shelf. *Oceanologica Acta*, 21(2):165, 1998.

[2] T. Corpetti, E. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Transactions on pattern analysis and machine intelligence*, 24(3):365–380, 2002.

[3] T. Corpetti, E. Mémin, and P. Pérez. Extraction of singular points from dense motion fields: an analytic approach. *Journal of Mathematical Imaging and Vision*, 2003.

[4] Thomas Corpetti. *Estimation de l'analyse de champs denses de vitesses d'écoulements fluides*. PhD thesis, Université de Rennes 1, France, 2002. in French.

[5] Jobard B. and Lefer W. Creating evenly-spaced streamlines of arbitrary density. In *Eurographics Workshop*, pages 43–56. Springer Verlag, 1997.

[6] J Daida, T.F. Bersano-Begey, S.J. Ross, and J.F Vesecky. Computer-assisted design of image classification algorithms: dynamic and static fitness evaluations in a scaffolded genetic programming environment. In *[30]*, pages 279–284, 1996.

[7] Marc Segond, Denis Robilliard and Cyril Fonlupt. Iterative Filter Generation Using Genetic Programming. In *Proceedings of the 9th European Conference on Genetic Programming EUROGP 2006*, volume 3905 of *LNCS*, Springer, pages 145–153, 2006.

[8] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[9] David Corne, Marco Dorigo, and Fred Glover, editors. *New Ideas in Optimisation*. Mc Graw-Hill, 1999.

[10] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Milano, Italy, 1991.

[11] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Mans, and Cybernetics*, 1(26):29–41, 1996.

[12] M. Dorigo and L. Gambardella. Ant colonies for the traveling salesman problem. Technical Report TR/IRIDIA/1996-3, IRIDIA, Université Libre de Bruxelles, Belgium, 1996.

[13] M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[14] L. Gambardella and M. Dorigo. ANT-Q: A reinforcement learning approach to the traveling salesman problem. In *[26]*, pages 252–260, 1995.

[15] M. Dorigo and L. Gambardella. A study of some properties of ant-q. In *[27]*, pages 656–665, 1996.

[16] G. Di Caro and M. Dorigo. Antnet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997.

[17] Nicolas Monmarché, Mohand Slimane, and Gilels Venturini. on how *Pachycondyla apicalis* ans suggest a new search algorith. *Future Generation Computer Systems*, 16(8):937–946, 2001.

[18] D. Costa and A. Hertz. Ants can colour graphs. *Journal of Operation Research Socoety*, (48):295–305, 1997.

[19] Nicolas Monmarché, Mohand Slimane, and Gilles Venturini. L'algorithme antclass : classification non supervisée par une colonie de fourmis artificielles. *Extraction des Connaissances et Apprentissage : Apprentisage et*

évolution, 1(3):131–166, 2001.
[20] T. Stutzle and H. Hoos. the MAX-MIN ant system and local search for the traveling salesman problem. In *[28]*, pages 308–313, 1997.
[21] T. Stutzle and H. Hoos. Improvements on the ant system: Introducing the MAX-MIN ant system. In *[29]*, 1997.
[22] W.B. Langdon and B.F. Buxton. Evolving receiver operating characteristics for data fusion. In *[31]*, pages 87–96, 2001.
[23] M. Sebag, J. Azé, and N. Lucas. ROC-based evolutionary learning: Application to medical data mining. In *[32]*, pages 384–396, 2003.
[24] Marc Segond, Cyril Fonlupt, and Denis Robilliard. Ant algorithms for detection in coastal waters. In *[32]*, pages 166–175, 2003.
[25] Tom Michael Mitchell. *Machine Learning*. Mc Graw-Hill, 1997.
[26] A. Prieditis and S. Russell, editors. *Proceedings of the twelth International Conference on Machine Learning*, Tahoe City, CA, USA, 1995. Morgan Kaufmann.
[27] Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors. *International Conference on Evolutionary Computation. The 4th Conference on Parallel Problem Solving from Nature*, number 1141, Berlin, Germany, September 1996. Springer-Verlag.
[28] *International Conference on Evolutionary Computation*, Anchorage, USA, 1997.
[29] *Proceedings of the third International Conference on Artificial Neural Networks and Genetic Algorithms*, Univerity of East Anglia, Norwich, UK, 1997.
[30] Koza, Goldberg, Fogel, and Riolo, editors. *Proceedings of the First Annual Conference on Genetic Programming*, Stanford University, CA, USA, 1996. MIT Press.
[31] Julian Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea Tettamanzi, and William Langdon, editors. *4th European Conference, EuroGP 2001*, volume 2038 of *LNCS*. Springer, apr 2001.
[32] Pierre Liardet, Pierre Collet, Cyril Fonlupt, Evelyne Lutton, and Marc Schoenauer, editors. *Proceedings of Evolution Artificielle EA'03*, volume 2396 of *LNCS*, Marseille, France, oct 2003. Springer.

Marc Segond is currently working toward the Ph.D. degree in computer science at the University of Littoral, France. His research interests include ant colony optimization and genetic programming.

Denis Robilliard received the Ph.D. degree in computer science from the University of Lille, France. He is currently an Associate Professor of computer science at the University of Littoral, Calais, France. His current research interests include evolutionary computing, genetic programming and machine learning.

Virginie Marion received the Ph.D. degree in computer science from the University of Lyon, France. She is currently an Associate Professor of computer science at the University of Littoral, Calais, France. His current research interests include genetic programming and scientific vizualization.

Cyril Fonlupt received the Ph.D. degree in computer science from the University of Lille, France. He is currently a Professor of computer science at the University of Littoral, Calais, France, where he is also the director of the evolutionary computing group. His current research interests include evolutionary computing, genetic programming and automatic programming.