Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

Mesoscale Events Classifier: an algorithm for the detection and classification of upwelling events using Sea Surface Temperature satellite data

Oscar Papini, CNR-ISTI, Pisa, Italy

ISTI-TR-2023/011

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

Mesoscale Events Classifier: An algorithm for the detection and classification of upwelling events using Sea Surface Temperature satellite data

Papini O.

ISTI-TR-2023/011

Abstract

In two previous technical reports we described a tool that produces a so-called spaghetti plot, i.e. a plot that is able to capture the trends of the sea surface temperature (SST) in a chosen time interval and within a target area; and the formalization of spaghetti plots through the definition of two custom Python 3 classes. In this report we outline an algorithm that uses SST data to detect and classify mesoscale upwelling events. In particular, the algorithm (called Mesoscale Events Classifier, MEC) takes as input the SST data organized as a SpaghettiData dictionary and returns a map of the area of interest where the zones in which the algorithm detects an event are highlighted and labelled with an event type.

Keywords

Citation

Papini O., *Mesoscale Events Classifier: An algorithm for the detection and classification of upwelling events using Sea Surface Temperature satellite data* ISTI Technical Reports 2023/011. DOI: 10.32079/ISTI-TR-2023/011

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

Mesoscale Events Classifier: An algorithm for the detection and classification of upwelling events using Sea Surface Temperature satellite data

Oscar Papini

12th December 2023

Contents

In two previous technical reports we described a tool that produces a so-called *spaghetti plot*, i.e. a plot that is able to capture the trends of the sea surface temperature (SST) in a chosen time interval and within a target area [3]; and the formalization of spaghetti plots through the definition of two custom Python 3 classes [4].

In this report we outline an algorithm that uses SST data to detect and classify mesoscale upwelling events. In particular, the algorithm (called *Mesoscale Events Classifier*, MEC) takes as input the SST data organized as a `SpaghettiData` dictionary and returns a map of the area of interest where the zones in which the algorithm detects an event are highlighted and labelled with an event type.

As a case study, we focus on the problem of detecting and classifying mesoscale events in the southwestern part of the Iberian peninsula, which is part of the Iberian/Canary Current System (ICCS). More precisely, our area of interest is

defined as the rectangle of the points with latitude between 35° and 40° N, and longitude between 12° and 6° W.

1 Context: Upwelling events

The upwelling phenomenon is the upward vertical transport of cold, nutrient-rich water from the depths to the ocean surface, mainly caused by the wind action and the rotation of Earth. Upwelling zones can be identified by cool SST and high concentrations of chlorophyll *a*.

The geomorphological peculiarities of the ICCS with respect to other upwelling ecosystems (including the discontinuity given by the Gulf of Cádiz and the nearby entrance of the Mediterranean Sea, as well as numerous topographical features of the continental shelf, such as prominent capes and submarine canyons; see Figure 1) affect the upwelling jet that runs southwards along the western coast of Portugal, eventually reaching Cape St. Vincent (37°1′30″ N, 8°59′40″ W). The

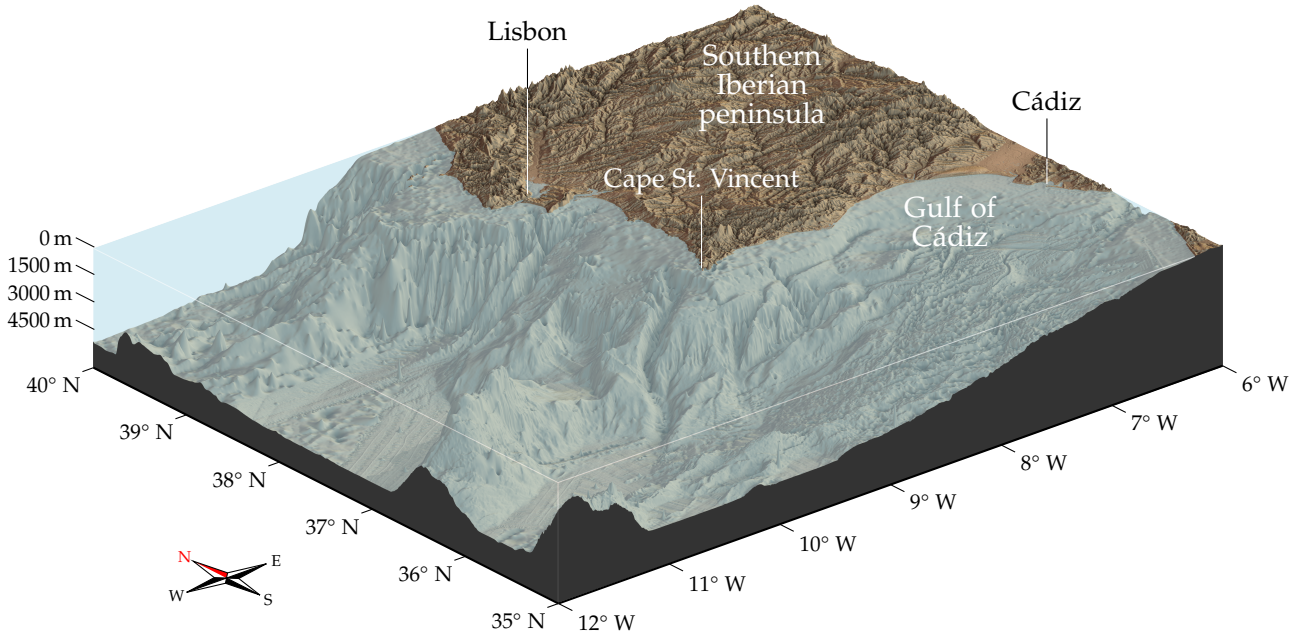


Figure 1: 3D rendering of the ocean floor in our region of interest. The depth scale has been exaggerated to highlight the topographical features.

different behaviours of the upwelled water give rise to a variety of patterns that can be seen in the SST maps obtained from satellite data. In particular, a visual inspection of a large quantity of SST maps by expert oceanographers resulted in the identification of four main recurring patterns in our area of interest, which

will be called E1, E2, E3 and E4. Table 1 contains a brief description of these events, and Figure 2 shows some examples of their appearance in the SST maps.

Table 1: Description of the four main types of upwelling events in the southwestern part of the Iberian peninsula.

Type	Description
E1	a filament of cold water originating from the upwelling jet, going westwards
E2	a filament of cold water going southwards, extending the upwelling jet beyond Cape St. Vincent
E3	a stream of cool water that bends eastwards from the upwelling jet, overtaking Cape St. Vincent and running along the southern coast of the Iberian peninsula
E4	a warm countercurrent originating in the Gulf of Cádiz and running westwards along the southern Iberian coast, eventually reaching Cape St. Vincent and turning northwards

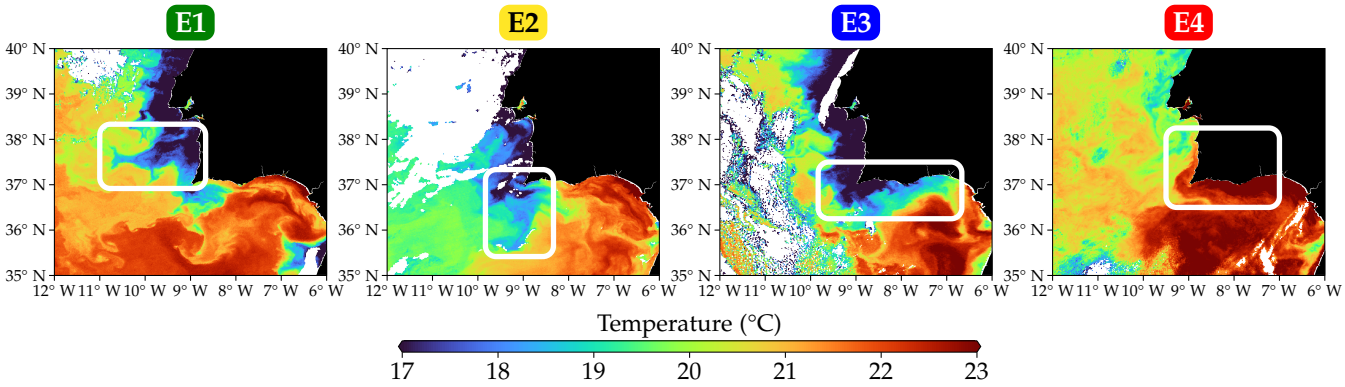


Figure 2: The different types of patterns recognisable in SST maps (highlighted with a white rectangle).

We say that a (mesoscale upwelling) event of type En occurs at a certain date if the SST map of that date shows a pattern of type En. Notice that multiple types of events may occur at the same date. The objective of MEC is to analyse SST data in order to detect the occurrence of an upwelling event and classify it into one of the four types.

2 The algorithm

The workflow of the algorithm is sketched in Figure 3. A description of the MEC core is already available in the literature (see for example [7; 9] for a general presentation and [6] for a more in-depth explanation); in this document we recall each part of the algorithm, focusing in particular on their technical aspects. For all our experiments we implemented MEC in a Python environment—the source code is available at `https://github.com/ospapini/nautilos-T8.5-sst`.

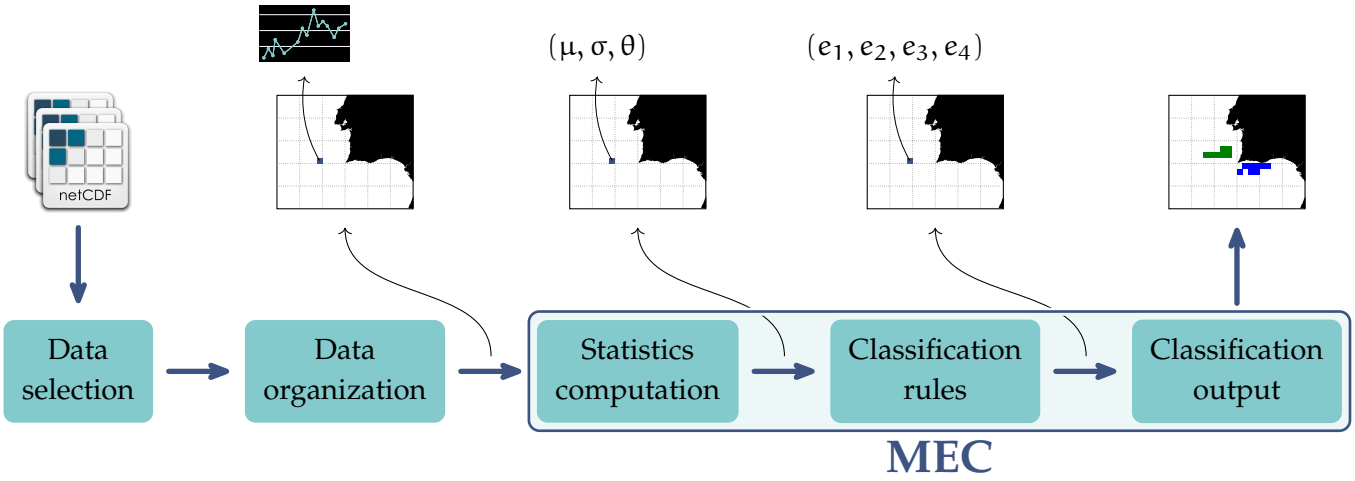


Figure 3: Brief schematics of the MEC algorithm.

2.1 Data selection

As already mentioned in [3], two main sources of SST data have been taken into consideration: the satellites of the *Metop* programme of EUMETSAT and the *Aqua* satellite of NASA. More specifically, for our analysis we used data from the years between 2009 and 2017. Table 2 outlines the main characteristics of these two sources.

Both products cover the entire surface of Earth; we downloaded only the files containing SST data within our area of interest, resulting in a dataset comprising an average of 2–3 images per day. However, not all of them can be used in the SST analysis: as shown in Figure 4, sometimes the radiometers on the satellites fail to measure the radiation signal, for example due to an excessive cloud coverage, resulting in large portions of the area with data either unreliable or completely missing. Therefore, we selected and discarded some images depending on the quantity of data available in them—in particular, the following criterion has been chosen:

Table 2: Main features of the satellite sources.

Institution	Satellite	Sensor	Resolution (at nadir)	Temperature accuracy	Binning rate	Processing level[▸a]	File format	Source
EUMETSAT	Metop-A/B[▸b]	AVHRR	1 km	0.01 °C	3 min	L2P	NetCDF-4	[2]
NASA	Aqua	MODIS	1 km	0.005 °C	5 min	L2P	NetCDF-4	[1]

[▸a] In accordance with the Group for High-Resolution Sea Surface Temperature (GHRSST) data processing specification (see `https://www.ghrsst.org/ghrsst-data-services/products/`, accessed on 29 June 2023).

[▸b] The product uses data from Metop-A before 19 November 2016 and Metop-B after that date.

1. considering the declared spatial resolution of the products, we estimate to have one data point every 0.01° in both latitude and longitude, thus about 300 000 points in the area;
2. among these, 91 346 correspond to coordinates on land (this number has been computed with the help of Python's `global-land-mask` module), so we obtain an expected number of valid SST data points of 208 654;
3. we discard an image if the number of SST data points in it is less than 15% of the expected number.

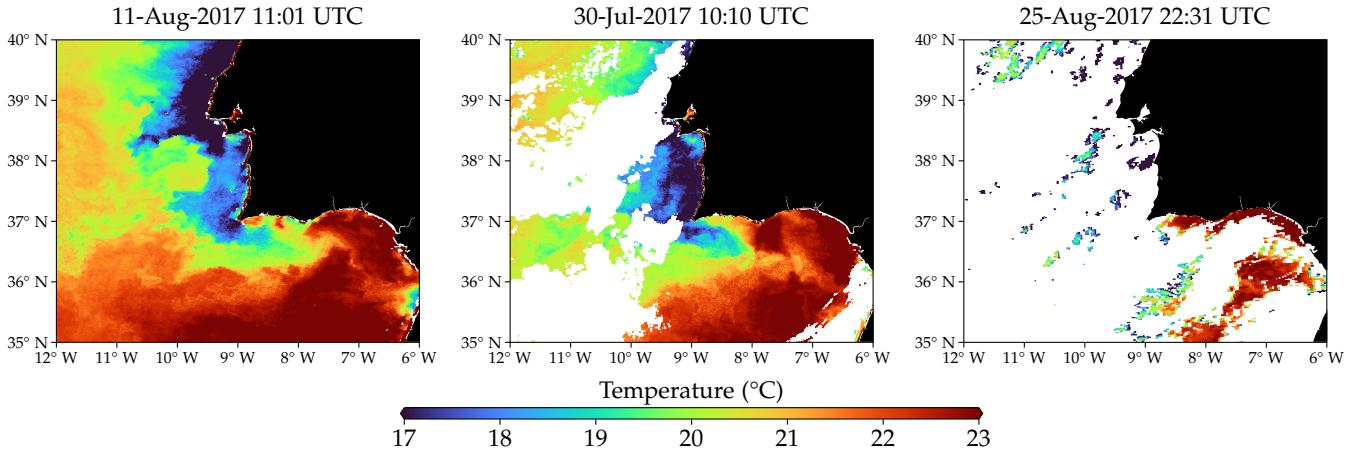


Figure 4: Examples of varying quality of SST images from the Metop dataset.

2.2 Data organization

Once the NetCDF files have been acquired and selected, in the next step we extract the SST data and organize them in a format suitable for the subsequent blocks of the algorithm. MEC exploits both the spatial distribution and the temporal

trend of the SST in the area of interest, therefore we will use the SST data in the NetCDF files to create a dictionary of `SpaghettiData` objects (as described in [4, Section 2]) with appropriate parameters. Recall that a `SpaghettiData` object represents a georeferenced set of pairs

$$\{(t_1, \mathrm{SST}_1), \ldots, (t_n, \mathrm{SST}_n)\}$$

where t_k is a timestamp and SST_k is a temperature value for each $k = 1, \ldots, n$.

The process of generation of these sets has been described in detail in [6, Section 4.1]; for the sake of completeness we report it again here. In the following, we denote with $T(t, \varphi, \lambda)$ the SST recorded at time t in the point with latitude φ and longitude λ. Note that $T(t, \varphi, \lambda)$ may not be defined for every t, φ and λ—this depends on the data availability in the NetCDF files. Moreover we consider the target area $A = [\varphi_{\min}, \varphi_{\max}] \times [\lambda_{\min}, \lambda_{\max}]$ and choose a time interval $\tau = [t_{\mathrm{start}}, t_{\mathrm{end}}]$.

1. Choose a resolution, i.e. a real number $r > 0$ such that the numbers $\ell = (\varphi_{\max} - \varphi_{\min})/r$ and $m = (\lambda_{\max} - \lambda_{\min})/r$ are integers. Divide the area A in $\ell \times m$ squares

   $$a_{i,j} = [\varphi_{\min} + ir, \varphi_{\min} + (i+1)r] \times [\lambda_{\min} + jr, \lambda_{\min} + (j+1)r]$$

   with $i = 0, \ldots, \ell - 1$ and $j = 0, \ldots, m - 1$.

2. For each satellite image with timestamp $t \in \tau$, compute the spatial average of the SST at time t in each square $a_{i,j}$ that contains a sufficient amount of data. Formally, choose a data abundance threshold $\alpha > 0$ and let

   $$X_{i,j}(t) = \{(\varphi, \lambda) \in a_{i,j} \mid T(t, \varphi, \lambda) \text{ is defined}\}$$

   i.e. the set of geographical points in $a_{i,j}$ that have a recorded SST in the image; if $\#\big(X_{i,j}(t)\big) \geqslant \alpha$, compute

   $$\overline{T}_{i,j}(t) = \frac{1}{\#\big(X_{i,j}(t)\big)} \sum_{(\varphi,\lambda) \in X_{i,j}(t)} T(t, \varphi, \lambda),$$

   otherwise leave $\overline{T}_{i,j}(t)$ undefined.

3. Let $t_1, \ldots, t_N$ be the timestamps of the images belonging to the time interval τ and, for each square $a_{i,j}$, compute the time series

   $$p_{i,j} = \{(t_k, \overline{T}_{i,j}(t_k)) \mid k = 1, \ldots, N \text{ s.t. } \overline{T}_{i,j}(t_k) \text{ is defined}\}.$$

   Notice that $n_{i,j} = \#\big(p_{i,j}\big)$ may not be the same for all squares: once again it depends on the quantity and the quality of SST data in the satellite images with timestamps in the chosen time period.

In our case the target area has $\varphi_{\min} = 35$, $\varphi_{\max} = 40$, $\lambda_{\min} = -12$ and $\lambda_{\max} = -6$ (western longitudes are considered negative). After a careful analysis of different choices for the parameters involved in the generation of the time series, the following values have been selected:

- resolution $r = 0.25°$ (so that our grid has $\ell = 20$ and $m = 24$); in addition to this being the typical spatial scale at which mesoscale patterns become recognizable, a too small value of r requires too much computational effort for the algorithm;
- time interval $\tau = 15$ days, i.e. if we want to detect events occurring at time t_0 we select images with timestamps in the interval $[t_0 - 15 \text{ days}, t_0]$; once again this represents the typical temporal scale at which the events appear and fade;
- data abundance threshold $\alpha = 100$, which at the chosen resolution means that the average SST in a square is not computed if there is less than about 16% of the expected amount of data in that square.

Despite all the quality filters applied in the previous steps, the time series may still show an erratic behaviour, with unrealistic peaks and valleys between consecutive images. Therefore an additional regularization step is applied to all the time series before proceeding to the next part of the algorithm.

We focused on two possible regularization methods for a time series. In the following, let $p = \{(t_1, SST_1), \dots, (t_n, SST_n)\}$.

1. The *discard* method consists of removing a pair (t_k, SST_k) from p if SST_k is too different from the SST values in a temporal neighbourhood of t_k. More precisely, let σ be the standard deviation of all the SST values in p, i.e.

$$\sigma = \sqrt{\frac{1}{n} \sum_{k=1}^{n} (SST_k - \mu)^2}$$

where μ is the average of the SST values

$$\mu = \frac{1}{n} \sum_{k=1}^{n} SST_k,$$

and choose a real number $\beta > 0$; for each $k = 1, \dots, n$ compute the median m_k of the five values SST_{k-2}, SST_{k-1}, SST_k, SST_{k+1} and SST_{k+2} (for $k = 0$, use only SST_0, SST_1 and SST_2; for $k = 1$, use SST_0, SST_1, SST_2 and SST_3; analogously for $k = n - 1$ and for $k = n$), then remove the pair (t_k, SST_k) from p if $SST_k \notin [m_k - \beta\sigma, m_k + \beta\sigma]$.

2. The *replace* method consists of "smoothing" p by substituting SST_k with the median of the SST in a temporal neighbourhood of t_k. More precisely, for each $k = 1, \ldots, n$ compute the median m_k of the three values SST_{k-1}, SST_k and SST_{k+1} (for $k = 0$, use SST_0, SST_1 and SST_2; analogously for $k = n$), then replace the series p with $p' = \{(t_1, m_1), \ldots, (t_n, m_n)\}$.

For our analysis we chose the discard method with $\beta = 1.5$.

2.3 Statistics computation

By plotting all the time series $p_{i,j}$ relative to a target area in a common time-temperature reference system, we obtain a *spaghetti plot* (see [3]). These plots can be visually analysed to recognise patterns in temperature trends associated with mesoscale upwelling events, with positive results (see for example [5; 8]).

The SST time series also provide the data upon which the automatic analysis of MEC is performed. The next step of the algorithm consists of extracting some statistical features from the series $p_{i,j}$ that describe in a concise but meaningful way the SST trend inside the squares $a_{i,j}$ in the considered time period. In particular, for each series $p_{i,j} = \{(t_k, SST_k) \mid k = 1, \ldots, n_{i,j}\}$ the following three statistics are computed:

1. the *temporal mean* of $p_{i,j}$,

$$\mu_{i,j} = \frac{1}{n_{i,j}} \sum_{k=1}^{n_{i,j}} SST_k;$$

2. the *standard deviation* of $p_{i,j}$,

$$\sigma_{i,j} = \sqrt{\frac{1}{n_{i,j}} \sum_{k=1}^{n_{i,j}} (SST_k - \mu_{i,j})^2};$$

3. the *linear regression coefficient* of $p_{i,j}$, denoted with $\theta_{i,j}$ and defined as the slope of the straight line that better interpolates the points in $p_{i,j}$, as computed by NumPy's `polyfit` method with `deg` equal to 1.

The units of measure are °C for $\mu_{i,j}$ and $\sigma_{i,j}$, and °C/day for $\theta_{i,j}$. By a slight abuse of notation, we will use the symbols μ, σ and θ as functions of the grid squares, meaning that, for example, if a is a square, then $\mu(a)$ is the temporal mean of the SST series associated with a, and analogously for σ and θ.

As we said before, the number of pairs in each series $p_{i,j}$, denoted by $n_{i,j}$ in the above formulas, is not the same for each square $a_{i,j}$ but varies depending

on the quality and quantity of data in the NetCDF files within $a_{i,j}$. Therefore we consider $n_{i,j}$ as a fourth relevant feature and interpret it as a reliability index for the final classification of the square $a_{i,j}$, under the assumption that a larger quantity of data present in the square implies a more faithful computation of the values of the statistics relative to it, and in the end a more accurate classification.

Moreover, an additional quality check is preformed at this stage: an integer threshold $\gamma > 0$ is chosen such that if $n_{i,j} \leqslant \gamma$ then no statistic is computed for $a_{i,j}$. Given the characteristics of our datasets and the choice of the parameter τ in the previous step, we expect to have $n_{i,j} \approx 30$ for each square, so for our analysis we chose $\gamma = 4$.

At code level, this step receives as input a dictionary of `SpaghettiData` and returns four $\ell \times m$ NumPy's masked arrays containing the values of $\mu_{i,j}$, $\sigma_{i,j}$, $\theta_{i,j}$ and $n_{i,j}$ (an entry is masked in an array if the statistic is not computed for the square corresponding to that entry).

2.4 Classification rules

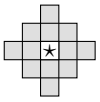
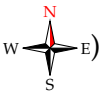
The set of classification rules is the core of the MEC algorithm: in this step the computed statistical features are automatically analysed in order to identify the squares in which an event is more likely to have occurred at the end of the considered time period. In particular, in this step a vector $\boldsymbol{e}_{i,j} = (e_1, e_2, e_3, e_4)$ (for a better readability we don't write the indices i and j of the components) is assigned to each square $a_{i,j}$, where for each $k \in \{1, 2, 3, 4\}$ the number $e_k \in [0, 1]$ represents a belief index for an event of type Ek to have occurred inside $a_{i,j}$. Each e_k is computed through a series of conditional rules applied to the statistics μ, σ and θ obtained in the previous step.

Remark. The rules have been handcrafted taking into consideration both the chosen values of the parameters involved (e.g. resolution of the grid) and the peculiarities of the ICCS upwelling ecosystem. It is possible to adapt the workflow of MEC to perform detection and classification of mesoscale events in other ecosystems, but a preliminary knowledge of the environment and the phenomena occurring inside it is required in order to properly define the classification rules.

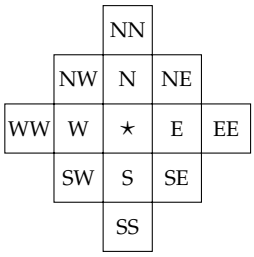
The rules used to compute $\boldsymbol{e}_{i,j}$ for a square $a_{i,j}$ involve not only the values $\mu_{i,j}$, $\sigma_{i,j}$ and $\theta_{i,j}$ of the statistics in $a_{i,j}$, but also the values of μ, σ and θ for other squares in the proximity of $a_{i,j}$. Therefore, before showing the actual rules we have to introduce some notation.

Let a be a square; its *full neighbourhood* $\mathcal{N}_{\text{full}}(a)$ is the set of grey squares in the following picture, where a is denoted with the $\star$ symbol (some of the squares

may be missing in case a is located near the edges of the grid):

$\mathcal{N}_{\text{full}}(a) =$  (Reference compass: )

In the description of the rules, the squares of $\mathcal{N}_{\text{full}}(a)$ have special subscripts that identify them. The reference is the following diagram:

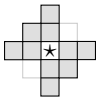


so, for example, a_S denotes the square immediately south of a. This notation extends also to the values of the statistics in those squares (e.g. the value $\sigma(a_{SE})$ will be denoted by σ_{SE}).

The *square neighbourhood* $\mathcal{N}_{\text{sq}}(a)$ is the set of squares adjacent to a:

$\mathcal{N}_{\text{sq}}(a) =$ 

The *basic neighbourhood* $\mathcal{N}_{\text{bsc}}(a)$ is obtained from $\mathcal{N}_{\text{full}}(a)$ by removing the two squares a_{NE} and a_{SW}:

$\mathcal{N}_{\text{bsc}}(a) =$ 

The *land mask* $\text{Land}(a)$ is a boolean function that returns `True` if and only if a is considered a "land" square (as opposed to "sea" square); the *coast distance* $\text{Coast}(a)$ is an integer-valued function that roughly estimates the distance of a from the coastline in terms of grid units; if a is a land square then $\text{Coast}(a) = 0$, otherwise

$$\text{Coast}(a) = 1 + \min\{\text{Coast}(b) \mid b \in \mathcal{N}_{\text{sq}}(a)\}.$$

Remark. The function Coast can be computed iteratively: first assign the value 0 to each land square and a special value "undefined" to each sea square, and define a variable $c = 1$; then, for each sea square a with an undefined Coast value, set $\text{Coast}(a) = c$ if and only if there is at least a square $b \in \mathcal{N}_{\text{sq}}(a)$ such that $\text{Coast}(b)$ is non-undefined; finally increase c by 1 and repeat.

The values of $\text{Land}(a)$ and $\text{Coast}(a)$ for all squares a in our region of interest are shown in Figure 5. With these two functions we can define two more

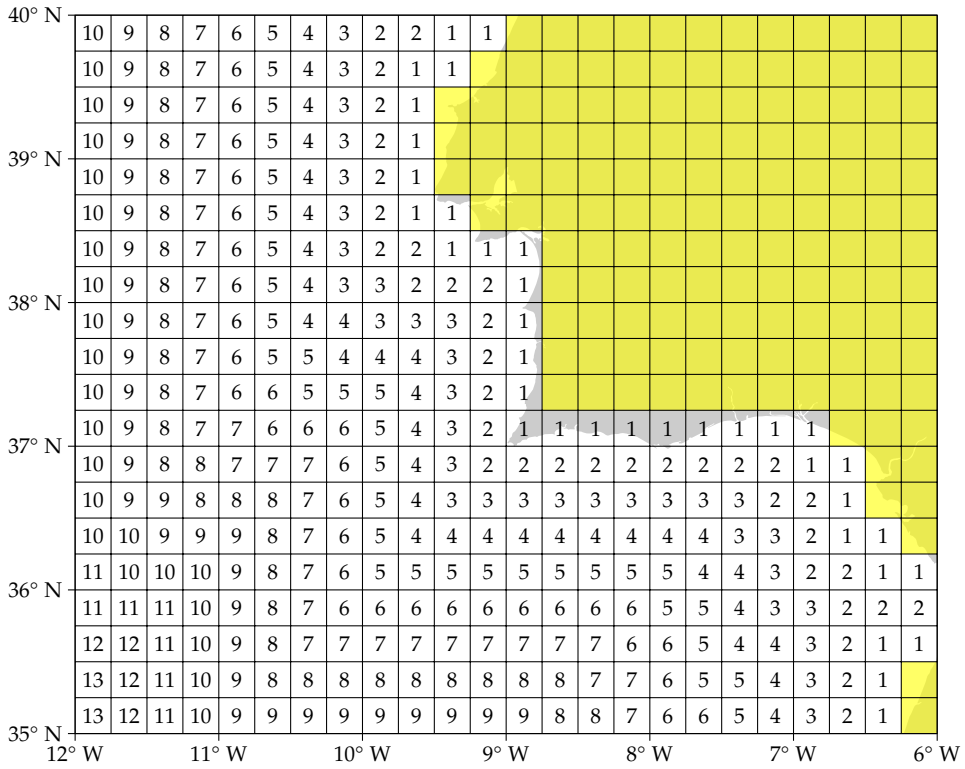


Figure 5: Values of the Land and Coast functions for all the squares in our area of interest. In particular, yellow squares ▢ have Land = `true` (and Coast = 0), and the number in a square corresponds to the value of Coast for that square.

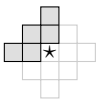
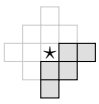
neighbourhoods of a square a: the *sea neighbourhood* $\mathcal{N}_{\text{sea}}(a)$, consisting of the non-land squares in $\mathcal{N}_{\text{bsc}}(a)$:

$$\mathcal{N}_{\text{sea}}(a) = \{b \in \mathcal{N}_{\text{bsc}}(a) \mid \neg\,\text{Land}(b)\};$$

and the *coastal neighbourhood* $\mathcal{N}_{\text{cst}}(a)$ that contains the squares of $\mathcal{N}_{\text{sea}}(a)$ with coast distance similar to $\text{Coast}(a)$:

$$\mathcal{N}_{\text{cst}}(a) = \big\{b \in \mathcal{N}_{\text{sea}}(a) \;\big|\; \big|\text{Coast}(b) - \text{Coast}(a)\big| \leqslant 1\big\}.$$

The last two types of neighbourhood that we define are the *geographical neighbourhoods* $\mathcal{N}_{\text{NW}}(a)$ and $\mathcal{N}_{\text{SE}}(a)$ as the parts of $\mathcal{N}_{\text{cst}}(a)$ that are respectively at NW and SE of a:

$\mathcal{N}_{\text{NW}}(a) = \mathcal{N}_{\text{cst}}(a) \cap$  $\qquad\qquad \mathcal{N}_{\text{SE}}(a) = \mathcal{N}_{\text{cst}}(a) \cap$ 

Finally, in the rules boxes only, if X is any (multi)set of numbers we will use the symbol $\overline{\mu}(X)$ to denote the arithmetic mean of its elements, to avoid a possible confusion with the statistic μ introduced before.

We are now ready to describe the procedure that defines the vectors $\boldsymbol{e}_{i,j}$. In the following, all the notation refers to a fixed square a of the grid (the procedure is repeated for all squares).

For each type of event $k \in \{1, 2, 3, 4\}$, define a score S_k initially set to 0; then this score can modified by the rules listed in the boxes below. In particular, each condition described in the "Trigger" columns is checked and, if it is verified, the points listed in the "Effect" columns are added to the score relative to the corresponding type of event. Notice that each box contains a preliminary condition than can prevent the rules form being tested if there is no relevant SST variation in a.

After all the conditions are checked, the four indices e_k are computed from the scores as $e_k = S_k/M_k$, where M_k is the maximum score obtainable from the square a for the type of event Ek. In particular, from the definition of the rules, we have that $M_1 = M_2 = 22$ for all squares, whereas M_3 and M_4 depend on the square a as follows:

$$M_3 = 16 + \sum_{b \in \mathcal{N}_{sea}(a)} n_b \qquad \text{where } n_b = \begin{cases} 3 & \text{if } b = a_S \text{ or } b = a_E \\ 2 & \text{otherwise,} \end{cases}$$

$$M_4 = 16 + \sum_{b \in \mathcal{N}_{sea}(a)} n_b \qquad \text{where } n_b = \begin{cases} 3 & \text{if } b = a_N \text{ or } b = a_W \\ 2 & \text{otherwise.} \end{cases}$$

At code level, this step receives as input the three arrays with the values of μ, σ and θ for all the squares of the grid, and returns four more $\ell \times m$ NumPy's masked arrays, one for each type of event, with the values of e_k.

Rules for E1

Apply only if there is a relevant variation of the SST in a, i.e.

$$\theta(a) < -0.05\,°C/\text{day} \quad \wedge \quad \sigma(a) > 0.2\,°C.$$

Rule	Trigger	Math. condition	Effect
(1.1)	SST of a_E also decreases, before the SST of a	$\theta_E < \theta(a)$	+6 pts
(1.2)	SST of a_{EE} also decreases, before the SST of a	$\theta_{EE} < \theta(a)$	+3 pts
(1.3)	SSTs of a, a_E and a_{EE} decrease sequentially	(1.1) $\wedge$ (1.2) $\wedge$ $\theta_{EE} < \theta_E$	+2 pts
(2.1)	a is colder, on average, than the squares to its N	$\mu(a) < \overline{\mu}(\mu_{NW}, \mu_N, \mu_{NE})$	+4 pts
(2.2)	a is colder, on average, than the squares to its S	$\mu(a) < \overline{\mu}(\mu_{SW}, \mu_S, \mu_{SE})$	+4 pts
(2.3)	both previous rules are applied	(2.1) $\wedge$ (2.2)	+2 pts
(HV)	*High Variation* — there is a significant variation of the SST in a	$\theta(a) < -0.1 \wedge \sigma(a) > 1$	+1 pt

Rules for E2

Apply only if there is a relevant variation of the SST in a, i.e.

$$\theta(a) < -0.05\,°C/\text{day} \quad \wedge \quad \sigma(a) > 0.2\,°C.$$

Rule	Trigger	Math. condition	Effect
(1.1)	SST of a_N also decreases, before the SST of a	$\theta_N < \theta(a)$	+6 pts
(1.2)	SST of a_{NN} also decreases, before the SST of a	$\theta_{NN} < \theta(a)$	+3 pts
(1.3)	SSTs of a, a_N and a_{NN} decrease sequentially	(1.1) $\wedge$ (1.2) $\wedge$ $\theta_{NN} < \theta_N$	+2 pts
(2.1)	a is colder, on average, than the squares to its E	$\mu(a) < \overline{\mu}(\mu_{NE}, \mu_E, \mu_{SE})$	+4 pts
(2.2)	a is colder, on average, than the squares to its W	$\mu(a) < \overline{\mu}(\mu_{NW}, \mu_W, \mu_{SW})$	+4 pts
(2.3)	both previous rules are applied	(2.1) $\wedge$ (2.2)	+2 pts
(HV)	*High Variation* — there is a significant variation of the SST in a	$\theta(a) < -0.1 \wedge \sigma(a) > 1$	+1 pt

Rules for E3

Apply only if there is a relevant variation of the SST in a, i.e.

$$\theta(a) < -0.05\,°C/\text{day} \quad \wedge \quad \sigma(a) > 0.2\,°C.$$

Rule	Trigger	Math. condition	Effect
(1)	all SSTs of squares in $\mathcal{N}_{cst}(a)$ also decrease	$\max\{\theta(b) \mid b \in \mathcal{N}_{cst}(a)\} < 0$	+5 pts
(2.1)	SSTs of NW squares decrease before the SST of a	$\forall\, b \in \mathcal{N}_{NW}(a)$ s.t. $\theta(b) < \theta(a)$	+2 pts
(2.2)	SSTs of SE squares decrease after the SST of a or increase	$\forall\, b \in \mathcal{N}_{SE}(a)$ s.t. $\theta(a) < \theta(b)$	$+n$ pts[▸a]
(3.1)	a is warmer, on average, than the squares to its NW	$\mu(a) > \overline{\mu}(\mathcal{N}_{NW}(a))$	+4 pts
(3.2)	a is colder, on average, than the squares to its SE	$\mu(a) < \overline{\mu}(\mathcal{N}_{SE}(a))$	+4 pts
(3.3)	both previous rules are applied	(3.1) $\wedge$ (3.2)	+2 pts
(HV)	*High Variation* — there is a significant variation of the SST in a	$\theta(a) < -0.1 \wedge \sigma(a) > 1$	+1 pt

[▸a] Here $n = 3$ for $b = a_S$ and $b = a_E$, and $n = 2$ otherwise.

Rules for E4

Apply only if there is a relevant variation of the SST in a, i.e.

$$\theta(a) > 0.05\,°C/day \quad \wedge \quad \sigma(a) > 0.2\,°C.$$

Rule	Trigger	Math. condition	Effect
(1)	all SSTs of squares in $\mathcal{N}_{cst}(a)$ also increase	$\min\{\theta(b) \mid b \in \mathcal{N}_{cst}(a)\} > 0$	+5 pts
(2.1)	SSTs of NW squares increase after the SST of a or decrease	$\forall\, b \in \mathcal{N}_{NW}(a)$ s.t. $\theta(b) < \theta(a)$	$+n$ pts[a]
(2.2)	SSTs of SE squares increase before the SST of a	$\forall\, b \in \mathcal{N}_{SE}(a)$ s.t. $\theta(a) < \theta(b)$	+2 pts
(3.1)	a is warmer, on average, than the squares to its NW	$\mu(a) > \overline{\mu}(\mathcal{N}_{NW}(a))$	+4 pts
(3.2)	a is colder, on average, than the squares to its SE	$\mu(a) < \overline{\mu}(\mathcal{N}_{SE}(a))$	+4 pts
(3.3)	both previous rules are applied	(3.1) $\wedge$ (3.2)	+2 pts
(HV)	*High Variation* — there is a significant variation of the SST in a	$\theta(a) > 0.1 \wedge \sigma(a) > 1$	+1 pt

[a] Here $n = 3$ for $b = a_N$ and $b = a_W$, and $n = 2$ otherwise.

2.5 Classification output

The final block of the MEC algorithm uses the data from the vectors $e_{i,j}$ to assign one or more labels "Ek" to each square $a_{i,j}$ of the grid. The procedure is simple: we define *a priori* a real number $\delta \in [0, 1]$ to be a detection threshold; for each square $a_{i,j}$, the algorithm computes $E = \max\{e_1, e_2, e_3, e_4\}$, then

- if $E \geqslant \delta$, $a_{i,j}$ gets the labels "Ek" for each $k \in \{1, 2, 3, 4\}$ such that $e_k = E$;
- otherwise, no label is assigned to $a_{i,j}$.

In order to choose an appropriate value for δ (and in general to have a basis for a first assessment of the MEC performance), we were given a ground truth dataset by expert oceanographers, containing their classification of some mesoscale events obtained directly from a visual examination of the SST maps. After some experiments, we chose the value $\delta = 0.6$ which seems to produce classifications that better agree with the results of the visual inspections.

The last step of the classification process is a refinement of the labels through the application of a geographical filter. In fact, a mesoscale event of a certain type may only occur within a specific zone of our area of interest. Therefore we defined four geographical masks (pictured in Figure 6), one for each type of event, such that only the squares belonging to the Ek event zone may have the "Ek" label. In particular, for each square $a_{i,j}$ and for each label "Ek" assigned to

it, the algorithm checks if $a_{i,j}$ belongs to the Ek zone and removes the label if this condition is not true.

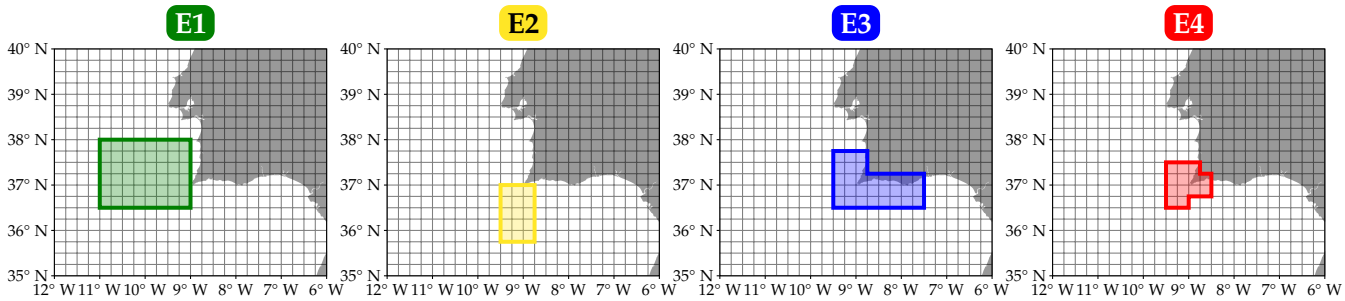


Figure 6: The four zones where a mesoscale event may occur.

At code level, this step of MEC receives as input the four arrays representing the scores $e_{i,j}$ for each square $a_{i,j}$ and returns a single $\ell \times m$ NumPy's masked array whose entries are integer numbers between 0 and 15 included. The (i,j)-th entry of this array encodes the set of labels assigned to $a_{i,j}$ as a binary number; in particular, if $L_{i,j} \subseteq \{1,2,3,4\}$ is the set defined as

$$L_{i,j} = \{k \mid \text{“Ek” is a label assigned to } a_{i,j}\},$$

then the (i,j)-th entry is

$$\sum_{k \in L_{i,j}} 2^{k-1}.$$

The array is then plotted using a custom script that involves `matplotlib.pyplot`'s `imshow` method in combination with the geographical plot capabilities of the `cartopy` module, resulting in a map with the labelled squares of the grid coloured according to their label(s). In order to have an immediate feedback of the reliability of the classification, each classified square displays also the percentage of available data in the time series associated with it: the script receives as an additional input the array with the numbers $n_{i,j}$ obtained in the "Statistics computation" step and for each classified square computes the percentage as

$$\left\lfloor \frac{n_{i,j}}{30} \cdot 100 \right\rfloor$$

(once again assuming 30 images per day in a time interval of 15 days—notice that this way it is possible to get percentages over 100%, since this is only a rough estimate).

3 An example

To better understand how MEC works, in this section we present an example of a detection and classification performed by the algorithm. The chosen date t_0 is the 7th October 2017 at about 21:00 UTC.

The SST map of that date, coming from the Metop dataset, is shown in Figure 7. In the picture we can see:

1. a cold filament at about 37.5° N extending for about one degree of longitude between 9.5° and 10.5° W;

2. a warm current along the coast between 8° and 9° W.

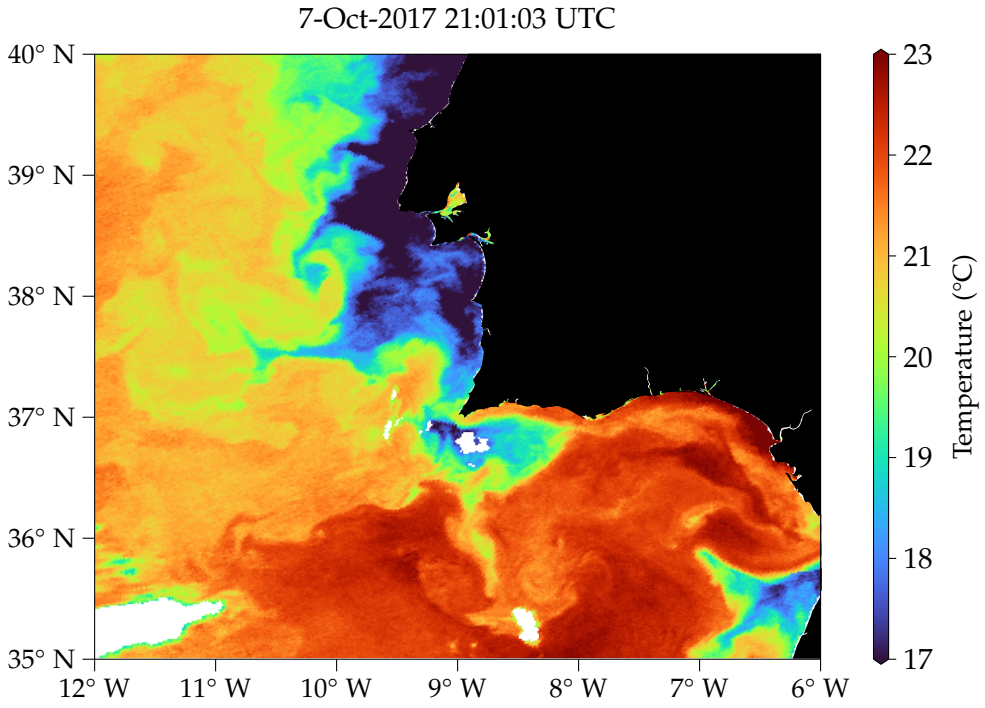


Figure 7: SST map of the area of interest at time t_0.

If we analyse the SST maps of the days before t_0 (Figure 8), we notice that the SST behaviour in the two areas is consistent with two upwelling events of type E1 and E4 respectively. Let's see if MEC is capable of detecting them.

First of all, we retrieve the SST information from the NetCDF files. In this example we will use data from the Metop dataset. For the time period of 15 days before t_0 there are 70 NetCDF files in the dataset, of which 32 are discarded during the "Data selection" step. The remaining files are elaborated and their SST data are organised in a dictionary of `SpaghettiData` objects. At this step we

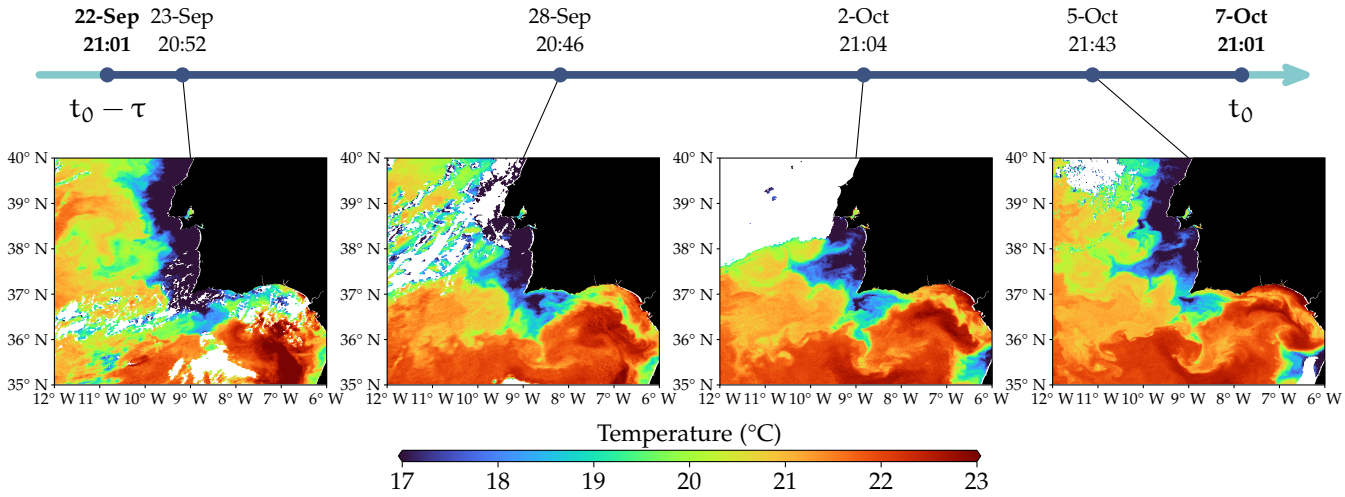


Figure 8: SST maps of the area of interest in some dates within 15 days prior to t_0. We can notice the process of formation of the cold water filament going westwards between 37° and 38° of latitude, as well as the warm current slowly replacing the cold water along the southern coast. All times are UTC.

can produce some spaghetti plots of the areas where the events are supposedly occurring, and indeed we find that the behaviour of the plots agrees with the one expected (see Figure 9).

In the following steps MEC computes the values of the statistics μ, σ and θ for all squares in the grid (Figure 10) and then the four events scores (Figure 11). Finally the classification array is computed and plotted (Figure 12).

Let us analyse briefly the output of MEC in this case. First of all, the percentages reported in the squares are quite high, so we may consider this classification reliable; moreover the two types of events that appear in the final map correspond to the types that we conjectured to have occurred in the considered time period. A more careful look reveals that actually the location of the classified squares does not match the one that we can infer by examining the SST maps—however, by going a step back and looking at Figure 11, we notice that along the southern coast of the Iberian peninsula there are many squares with high e_4-score that just barely miss the $\delta = 0.6$ threshold.

Overall, we can consider MEC a tool that supports oceanographers by pointing out possible occurrences of upwelling events. This automatic process reduces the time and effort spent on the visual analysis of the SST maps; additionally, it provides a way to remove the subjectivity intrinsic in the researchers' interpretation of the maps.

A thorough study regarding the MEC performances, that also takes into account the peculiarities of the algorithm, can be found in [10].

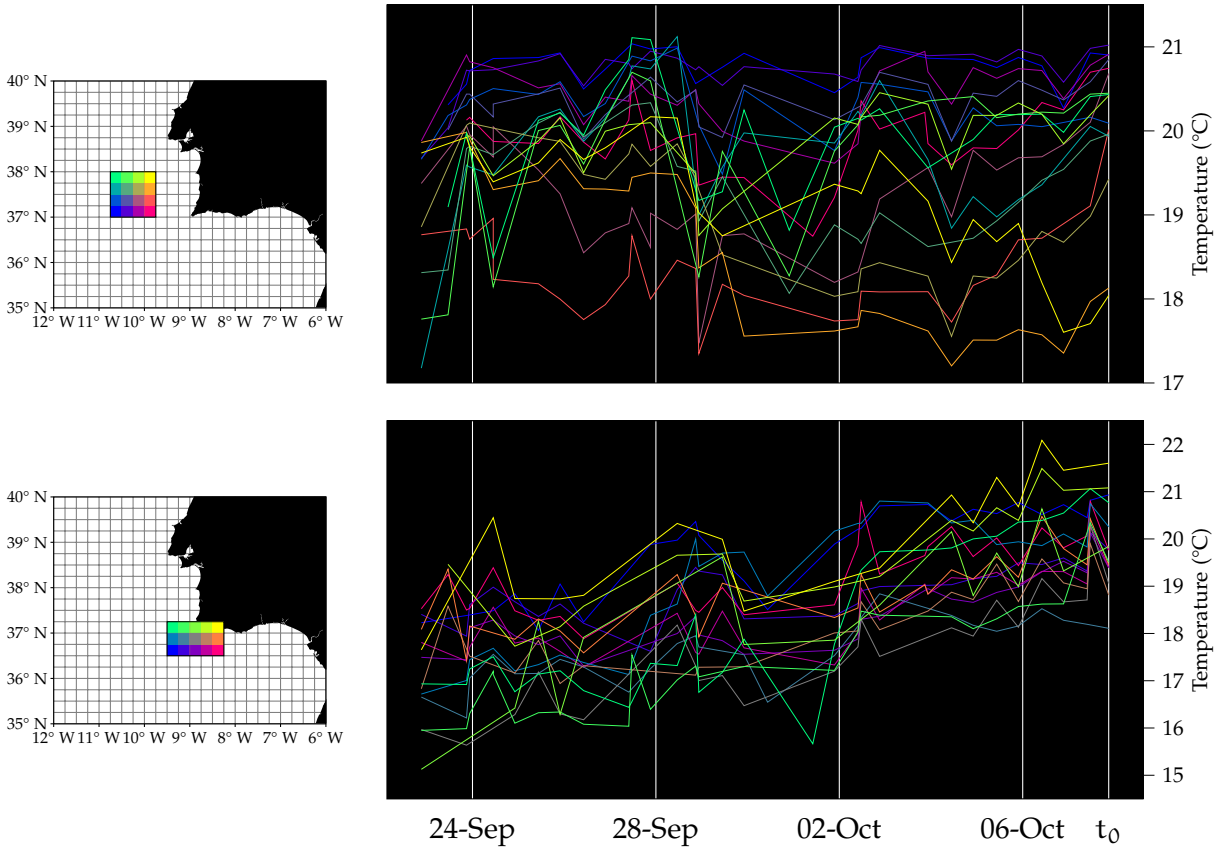


Figure 9: Spaghetti plots for two selected areas where two upwelling events seem to occur. Each line in the plots on the right represents the SST trend in the square marked with the same colour in the reference grid on the left.

References

[1] NASA/JPL. *GHRSST Level 2P Global Sea Surface Skin Temperature from the Moderate Resolution Imaging Spectroradiometer (MODIS) on the NASA Aqua satellite (GDS2)*. 2020. DOI: 10.5067/GHMDA-2PJ19.

[2] OSI SAF. *Full resolution L2P AVHRR Sea Surface Temperature MetaGRanules (GHRSST) - Metop*. 2011. DOI: 10.15770/EUM_SAF_OSI_NRT_2013.

[3] Oscar Papini. *A tool for the temporal analysis of sea surface temperature maps*. ISTI Technical Reports 2021/011. ISTI-CNR, 2021. DOI: 10.32079/ISTI-TR-2021/011.

[4] Oscar Papini. `SpaghettiData` *and* `SpaghettiPlot`*: two Python classes for analysing and visualising SST trends*. ISTI Technical Reports 2022/001. ISTI-CNR, 2022. DOI: 10.32079/ISTI-TR-2022/001.

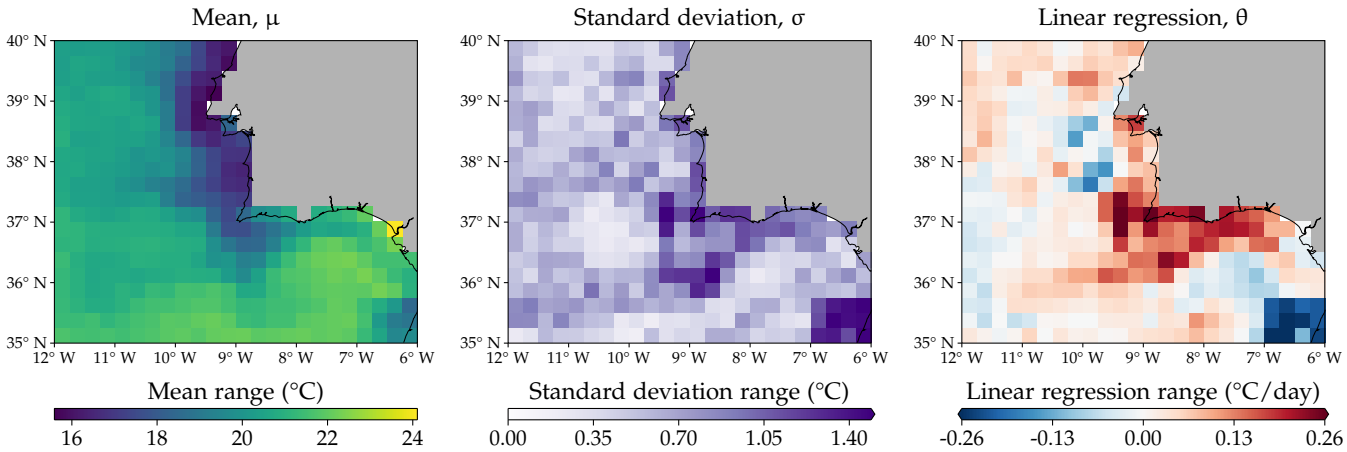


Figure 10: Visual representation of the arrays containing the values of the statistics for each square of the grid.

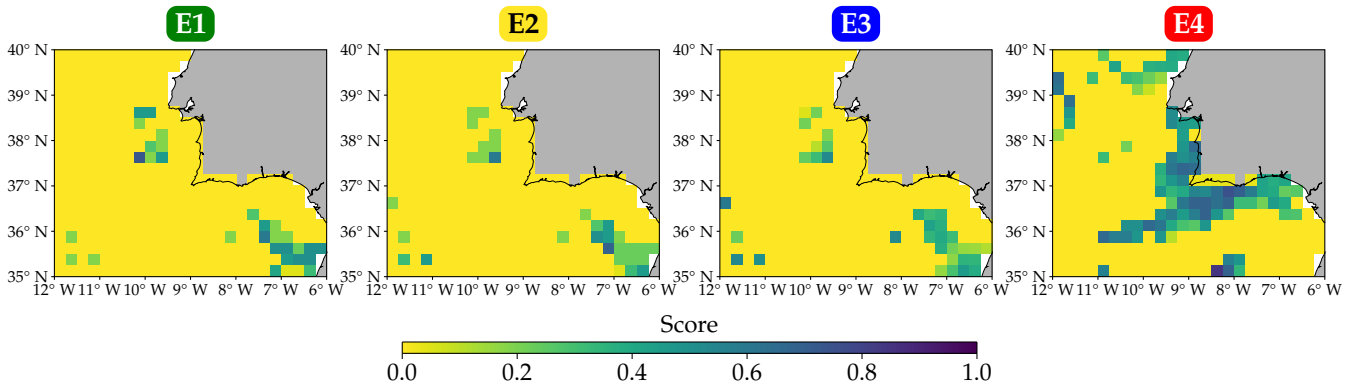


Figure 11: Visual representation of the arrays containing the values of the scores e_k for $k \in \{1, 2, 3, 4\}$ for each square of the grid.

[5] Oscar Papini, Marco Reggiannini and Gabriele Pieri. "SST Image Processing for Mesoscale Patterns Identification". In: *Engineering Proceedings* 8, 5 (2021). Presented at the 16th International Workshop on Advanced Infrared Technology and Applications — AITA. DOI: 10.3390/engproc2021008005.

[6] Gabriele Pieri, João Janeiro, Flávio Martins, Oscar Papini and Marco Reggiannini. "MEC: A Mesoscale Events Classifier for Oceanographic Imagery". In: *Applied Sciences* 13.3, 1565 (2023). DOI: 10.3390/app13031565.

[7] Marco Reggiannini, João Janeiro, Flávio Martins, Oscar Papini and Gabriele Pieri. "Mesoscale Events Classification in Sea Surface Temperature Imagery". In: *Machine Learning, Optimization, and Data Science. Proceedings of the 8th International Workshop LOD 2022, Revised Selected Papers, Part I*. Ed. by

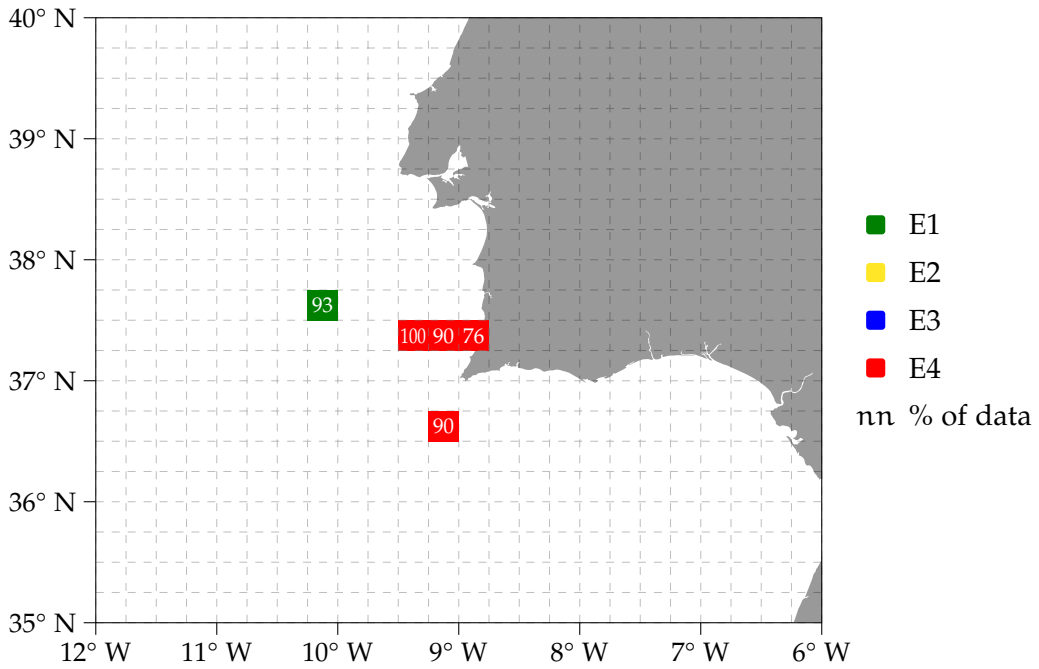


Figure 12: Map representing the output of the MEC process of detection and classification.

Giuseppe Nicosia et al. Vol. 13810. Lecture Notes in Computer Science. Springer, 2023, pp. 516–527. DOI: 10.1007/978-3-031-25599-1_38.

[8] Marco Reggiannini, João Janeiro, Flávio Martins, Oscar Papini and Gabriele Pieri. "Mesoscale Patterns Identification Through SST Image Processing". In: *Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems — ROBOVIS*. Available also at openportal.isti.cnr.it. SciTePress, 2021, pp. 165–172. DOI: 10.5220/0010714600003061.

[9] Marco Reggiannini, Oscar Papini and Gabriele Pieri. "Automated Image Processing for Remote Sensing Data Classification". In: *Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges. Proceedings, Part II*. Ed. by Jean-Jacques Rousseau and Bill Kapralos. Vol. 13644. Lecture Notes in Computer Science. Springer, 2023, pp. 553–560. DOI: 10.1007/978-3-031-37742-6_43.

[10] Marco Reggiannini, Oscar Papini and Gabriele Pieri. "Evaluation of a Mesoscale Event Classifier". In: *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. Presented at the 11th International Workshop on Computational Intelligence for Multimedia Understanding — IWCIM. 2023. DOI: 10.1109/ICASSPW59220.2023.10193234.