Ecological Informatics 55 (2020) 101036

Contents lists available at ScienceDirect

Ecological Informatics

journal homepage: www.elsevier.com/locate/ecolinf

Active contour-based detection of estuarine dolphin whistles in spectrogram images

O.M. Serra*, F.P.R. Martins, L.R. Padovese

Universidade de São Paulo, Escola Politécnica, Department of Mechanical Engineering, Av. Prof. Mello Moraes, 2231, 05508-030 São Paulo, Brazil

ARTICLE INFO

Keywords:
(*Sotalia guianensis*
Spectrogram
Hough transform
Active contours
Random forest
Machine learning)

ABSTRACT

An algorithm for detecting tonal vocalizations from estuarine dolphin (*Sotalia guianensis*) specimens without interference of a human operator was developed. The raw audio data collected from a passive monitoring sensor in the Cananéia underwater soundscape was converted to spectrogram images. Detection is a four-step method: first, ridge maps are obtained from the spectrogram images; second, a probabilistic Hough transform algorithm is applied to detect ridges similar to thick line segments, referred to as line-like, which are adjusted to the geometry of the whistles in the images via an active contour algorithm; third, feature vectors are built from the geometry of each detected whistle, with 9 descriptive features; and fourth, the detections are fed to a random forest classifier to parse out mistakes by the detection process. We developed a system for classifying the characteristic patterns detected as *Sotalia guianensis* whistles or random empty detections. We obtained accuracy of 0.977 and F1-score of 0.981.

1. Introduction

The estuarine dolphin (*Sotalia guianensis*), also known as Guiana dolphin, is one of the smallest species of dolphin. It is native to the Atlantic coastal regions of Central and South America, ranging from Honduras to the state of Santa Catarina, Brazil (Flach et al., 2008).

It is a species that lives in cohesive groups, commonly ranging from 2 to 6 individuals, although groups of over 50 specimens have been observed (Monteiro Filho, 2008). Due to residing in coastal areas, the estuarine dolphin is particularly susceptible to human impacts. The greatest threats to the species are pollution, habitat degradation, accidental capture and increase in the frequency of maritime traffic (ICMBio, 2011).

Information on the populational and social dynamics of the estuarine dolphin is currently scarce. The data entry for this species in the International Union for Conservation of Nature (IUCN) Red List was classified as “data deficient” (DD) until 2017 due to lack of conclusive data. It has since been classified as “near threatened” (NT), with the rating being changed in 2018 (Secchi et al., 2018).

One of the vocalizations emitted by the estuarine dolphin is a tonal sound, or whistle. It may be useful in monitoring general activity of this species within a given region. Dolphin whistles and their shapes in spectrogram images were characterized in (Bazúa-Durán, 2004; Henderson et al., 2012; Kriesell et al., 2014). In these papers, the visual patterns corresponding to the whistles roughly resemble line segments, or combinations thereof.

As will be detailed in Section 2, the data used in this research contains several estuarine dolphin vocalizations as object of study, and is part of a soundscape of maritime protected regions in southeastern Brazil done in (Sánchez-Gendriz and Padovese, 2016). Our current research is then concerned with the automatic detection of the aforementioned estuarine dolphin vocalizations from raw audio data. This data was converted to spectrogram images. Those were used thereafter to identify the vocalizations based on their corresponding patterns in the images.

Features from the desired sounds can be extracted by audio analysis (Malfante et al., 2018), knowledge of the geography of the data collection sites (Henderson et al., 2012) or image recognition techniques in spectrogram images (Esfahanian et al., 2013; MacBride and Roth, 2016).

An algorithm for detection of sounds from fish choruses based only on audio analysis was implemented in (Malfante et al., 2018), relying solely on auditory patterns. This method had very high accuracy for well-defined conditions, but encountered problems when the selected time window contained multiple detections from different classes.

Erbs et al. (2017) and Henderson et al. (2012) propose methods based on the statistical distribution of features from the desired acoustic events for classifying whistle sounds from different species, and present

* Corresponding author.
E-mail address: otaviomserra@usp.br (O.M. Serra).

https://doi.org/10.1016/j.ecoinf.2019.101036
Received 28 August 2019; Received in revised form 9 November 2019; Accepted 15 November 2019
Available online 21 November 2019

high accuracy rates for inter- and intra-species classification.

Image segmentation algorithms (an umbrella term for image processing methods designed to identify patterns of interest in signals and images) are widely employed in the literature (Gillespie, 2004; Kershenbaum and Roch, 2013; Mellinger et al., 2011). Further classification of the identified sounds is often done with random forest classifiers (Henderson et al., 2012) or methods based on eigenfunction problems (Karnowski and Johnson, 2015; MacBride and Roth, 2016).

Acoustic events may also be classified by convolutional neural networks, as in (Kahl et al., 2017), though a random forest classifier is employed in our study, in keeping with most of the literature.

A similar algorithm to the one employed in our article is detailed in (Gillespie, 2004) for the detection of right whale calls. Edge maps were obtained from smoothened spectrogram images, from which outlines of the desired events were determined. These events were then run through a classifier to reliably detect whether they corresponded to the right whale calls or not.

Kershenbaum and Roch (2013) developed a method for the identification of cetacean tonal sounds via ridge maps on a spectrogram image, as is done in this study. Their method took the ridges detected in the spectrogram images as potential detections and tallied the number of false positives.

This method employs active contour algorithms (Kass et al., 1988) as a novelty if compared to the literature. The computational vision method employs an image segmentation method based on the Hough transform (Duda and Hart, 1972) and active contours. Further classification is done via a random forest classifier.

This article has the following structure: Section 2 details the process of data collection and the algorithms employed in this study; in Section 3, we present the most important patterns observed and the final performance of the classification algorithm; in Section 4, we present our conclusions over the study.

2. Materials and methods

The dataset used in this research was obtained following the article published in (Sánchez-Gendriz and Padovese, 2016). The acoustic signals were recorded at a depth of 8 m underwater via an autonomous passive monitoring system (OceanPod), developed at LACMAM (Laboratory of Acoustic and Environment of the Escola Politécnica da Universidade de São Paulo) (see Fig. 1).

The data acquisition campaign was conducted at a bay close to Cananeia (Fig. 2), a city in the south of São Paulo State, Brazil from February 07 to 10, 2017. The recording was carried out at a sampling frequency $f_s = 48kHz$, 16 bit resolution, and the grabbed data were continuously stored in an SD card, as a sequence of .wav files, each with size corresponding to 3 min of acoustic emission. The sensitivity of the system was set to −150 dB re 1 V/μPa and its frequency band from 10 Hz to $f_s/2$. Seeing as the desired acoustic events had a typical frequency range of 5 kHz to 15 kHz, these numbers were sufficient. During the campaign, 1460.wav files, totalizing 23,4 GB and roughly 72 h of usable audio, were recorded; from these, 110 were randomly selected to feed the algorithms presented in this article.

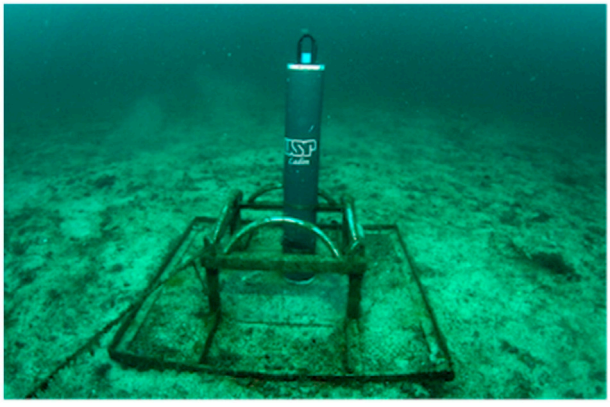

Fig. 1. OceanPod 1.0 used for recording.
Source: Sánchez-Gendriz and Padovese (2016).

Fig. 2. Location of the recording site.
Source: (C) Google Earth. [color].

This campaign represents the studies made possible by our economic resources available at the time of this research. Thus, an extensive monitoring of the species at different sites and periods of the year falls beyond the scope of this study. Detailed information on the behavior of the estuarine dolphin can be found in (Monteiro Filho, 2008).

As will be detailed in the following subsections, the methodology proposed for detecting vocalizations of estuarine dolphin specimens was based on the application of image processing and machine learning techniques to the recorded signal spectrograms. Python 3.6 (Python Software Foundation) was used as a programming language. Additional Python libraries and resources employed were: Numpy (Oliphant, 2006), Matplotlib (Hunter, 2007), Pandas (McKinney, 2010), scikit-learn (Pedregosa et al., 2011), scikit-image (van der Walt et al., 2014), PyDub (https://pypi.org/project/pydub/) and Jupyter (Kluyver et al., 2016).

2.1. Overview of the algorithm

The structure of the algorithm for recognizing estuarine dolphin vocalizations subdivides naturally into four steps: signal and image preprocessing, image segmentation, computation of feature vectors and classification. A simplified framework of the algorithm is shown in Fig. 3.

Each of the 110 audio files selected was partitioned into sixty 3-s

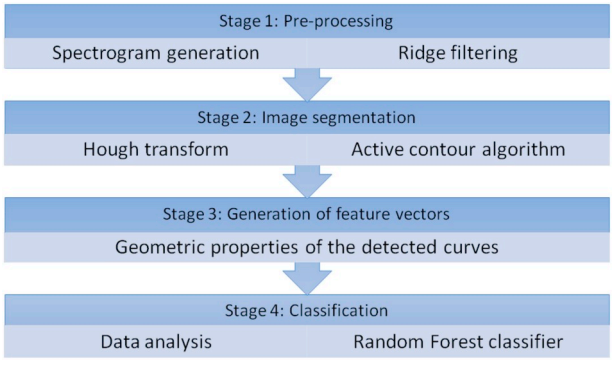

Fig. 3. Proposed algorithm for detecting estuarine dolphin vocalizations.

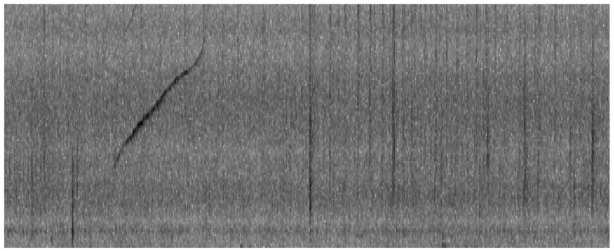

Fig. 4. Example of a roughly line-like pattern corresponding to the acoustic event of interest.

snippets, each of them being turned into a spectrogram image following the Matplotlib implementation. The 3-s window was a duration deemed appropriate for the visualization of the desired pattern. The dolphin vocalizations of interest form curves that roughly resemble line segments, observed to make angles with the horizontal axis within the interval from 15° to 75°. Fig. 4 shows an example of this typical pattern.

As such, our available database consisted of 6600 images and was, in total, 5 h and 30 min of audio.

A general summary of the algorithm is as follows. We detect possible instances of the desired pattern, via a Hough transform (Duda and Hart, 1972) on the intensity ridge map of the image. An active contour algorithm (Kass et al., 1988) is thereby employed to adjust the line segments to better fit the shape of the patterns, obtaining open-ended curves. Geometric features are calculated from each pattern, using manually written NumPy code, and fed to a machine learning algorithm, which finally determines whether the detection corresponded to an event of interest.

2.2. Image pre-processing

The first step in detecting the characteristic patterns of interest in a spectrogram image is to make such features stand out. The chosen method was the Frangi vesselness filter (Frangi et al., 1998). This filter is a ridge detection algorithm that singles out regions of the image with large dark patches.

As detailed in (Fu et al., 2017), the Frangi filter excels in detecting tubular shapes in structures, and is mostly used in medical research. The unique, roughly line-like structure of the estuarine dolphin whistles means that a filter capable of singling out tubular-shaped regions will produce good results in our spectrogram images. An example is shown in Fig. 5, where the intuitive similarity between the visual patterns in that study and in our research can be seen. This justifies the exceptional performance of the Frangi filter for detection of estuarine dolphin whistles.

If H is the Hessian matrix of the spectrogram image, we compute its eigenvalues $\lambda_{1,\,2}$, with numbering conventioned as $|\lambda_2| > |\lambda_1|$. The vesselness value is then calculated from the eigenvalues, and depends on the standard deviation σ used to compute the Hessian. Eq. 1 gives the output of the Frangi filter.

$$V_0(\sigma) = \begin{cases} 0, \; if \; \lambda_2 < 0; \\ e^{-\frac{R_B^2}{2\beta^2}}\left(1 - e^{-\frac{S^2}{2c^2}}\right), \; otherwise \end{cases} \tag{1}$$

In Eq. 1, $S = \sqrt{\lambda_1^2 + \lambda_2^2}$, $R_B = \|\lambda_1\|/\|\lambda_2\|$ and β, c are parameters. The scikit-image implementation used in this research follows the original Frangi filter.

Then, the resulting map was binarized through a manually chosen threshold for V_0, separating low- from high-intensity regions. An example of this process is shown in Fig. 6

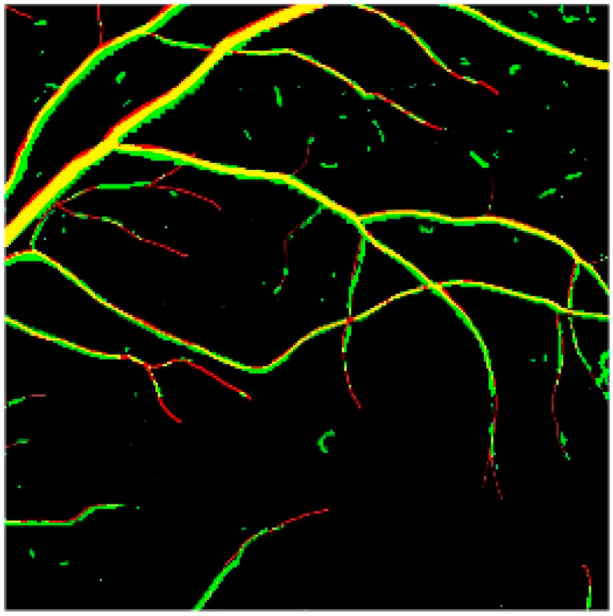

Fig. 5. Example of the Frangi filter.
Source: (Fu et al., 2017).

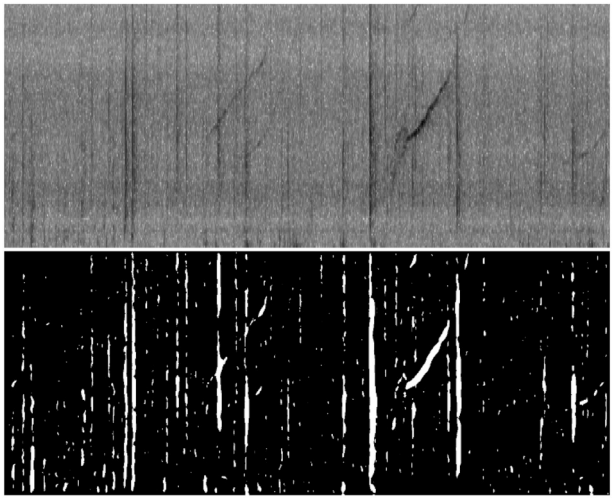

Fig. 6. (top) Original image; (bottom) Respective binarized Frangi map.

2.3. The Hough transform

In order to detect line-like patterns in the binarized Frangi maps, we applied a Hough transform algorithm. As described in (Duda and Hart, 1972), a straight line L (Fig. 7) of image space xy, described in polar coordinates by

$$r = x\cos\theta + y\sin\theta \tag{2}$$

is transformed into a point P with coordinates (r, θ) in Hough space.

Furthermore, as the number of image points incident on L corresponds to the "intensity" of the respective point P in the Hough map, these points of greatest intensity are the ones more likely to represent real straight lines in the image space.

The particular implementation employed in this research is from scikit-image, which is based on (Galambos et al., 2000). It applies a progressive probabilistic method to determine where a line starts and ends, in addition to verifying whether it exists or not. An example of this method, applied to the same source image as Fig. 6, is shown in Fig. 8.

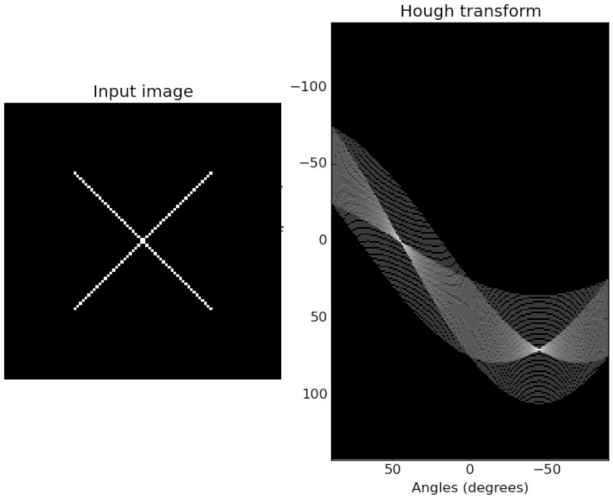

Fig. 7. The Hough parameter space for two straight lines.
Source: scikit-image.

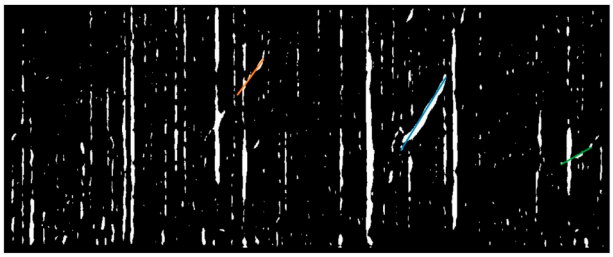

Fig. 8. Lines (in color) detected using the probabilistic Hough transform. [color].

2.4. The active contour (snake) algorithm

Since the locations of the vocalizations of interest in the spectrogram images are roughly located by the Hough Transform, an active contour algorithm is applied in order to accurately capture the shape of those patterns.

The resulting curves overlaying the referred objects will henceforth be referred to as "*snakes*," in agreement with the literature (Kass et al., 1988). A *snake* is defined to be a piecewise smooth spline

$$v(s) = (x(s), y(s)),\ s \in [0,1]$$

The scikit-image implementation is based on (Kass et al., 1988). The main points of this algorithm are presented in the following paragraphs.

The straight lines obtained previously, e.g. the ones in Fig. 8, are used as the first approximate solution for the variational problem implemented by the *snake* algorithm. The traditional algorithm searches iteratively for local minima of the energy functional associated to the *snake*, given in Eq. 3 in accordance with (Kass et al., 1988).

$$E^*_{snake} = \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))\, ds \tag{3}$$

Interpretations for the terms are as follows. The internal energy $E_{int}(v(s))$ of the *snake* is defined in Eq. 4.

$$E_{int}(v(s)) = \frac{1}{2}\left(\alpha(s)\left|\frac{\partial v}{\partial s}(s)\right|^2 + \beta(s)\left|\frac{\partial^2 v}{\partial s^2}(s)\right|^2\right) \tag{4}$$

It imposes a piecewise smoothness constraint. The second term $E_{image}(v(s))$, commonly referred to as image energy, consists of conditions that attract the *snake* to desired patterns in the image, e.g. edges or ridges. A formulation of this term for edge attraction is given in Eq. 5, assuming that detection of terminations of edges is not important, due to endpoints remaining fixed.

$$E_{image}(v(s)) = w_1 I(v(s)) + w_2\ |\nabla I(v(s))|^2 \tag{5}$$

In Eq. 5, $I(x,y)$ is the gray level image function at the point (x,y). The constraint energy $E_{con}(v(s))$ keeps the *snake* close to a desired local minimum of its energy functional. The definition of this term is situational and, in our research, it can be considered negligible.

In short, the algorithm attracts the *snake* to a desired feature of the image, while still maintaining its piecewise smoothness. In computational applications, a *snake* is a discretization of the analytic approach showed above, taking as a first estimate the straight line obtained via the Hough transform, with equally spaced points. This piecewise line segment then undergoes a numerical minimization of its energy functional, given by Eq. 3.

The scikit-image implementation takes the first- and second-order functional coefficients $\alpha(s)$ and (s) from Eq. 4 to be constants defined by the user. In some cases, it may be desirable to let one of the parameters equal zero at certain values of s. For instance, a value of $\beta = 0$ would allow for a second-order discontinuity (a sharp turn) at that point.

In our work, although α and β were defined as constants, little emphasis was given to second-order continuity, resulting in a low value for β and allowing for the formation of sharp edges.

As a boundary condition, the edge points of the original straight line were kept fixed, so as to avoid unnecessary strain on the *snake*. The *snakes* resulting from a full run of the algorithm starting from Fig. 8 are shown in Fig. 9, layered on top of the original image for ease of visualization.

Occasionally, a straight line found by the Hough transform algorithm will not cover a real pattern detection, resulting in a meaningless *snake*. Such mistakes can be seen in Fig. 10, and are the main object of the fourth stage of this study (machine learning classification).

2.5. Feature vectors

The occurrence of mistakes, as highlighted in Fig. 10, indicates that an algorithm based exclusively on the combination of the Hough transform and active contours is not robust enough to detect estuarine dolphin vocalizations in spectrograms of the landscape as intended. For this reason, a classification algorithm based on a supervised learning approach was added to those components and, as a consequence, it was necessary to extract data from the *snakes* obtained via the previous algorithms and organize them into an appropriate training set. We manually labeled a set of 3819 *snakes*, later split randomly into a training set – which contained 75% of the *snakes* – and a test set with the remaining ones.

Additionally, we can estimate from the data that 2.6% of the images contain whistles not picked up by the algorithm. Such missed detections were minimized by varying the parameters of the *snake* algorithm, although it should be noted that further investigation of these objects falls outside the scope of our study.

We now consider a *snake* a collection of points $v(i) = (x(i), y(i))$, where i is a discrete variable ranging from $i = 1$ to N, where N is the number of points in the *snake*. A value of $N = 50$ was chosen for this research, maintaining a balance between computational time and level

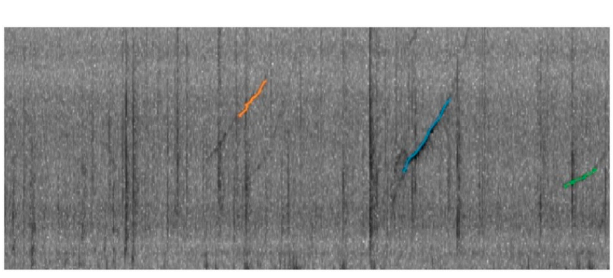

Fig. 9. *Snakes* overlaying the desired patterns. [color].

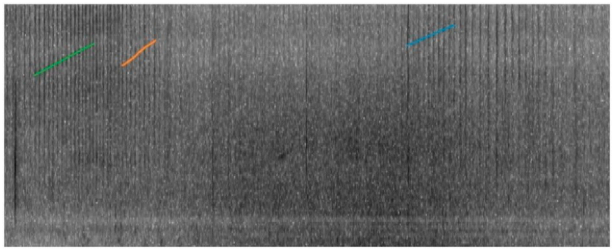

Fig. 10. A spectrogram image with three false positive events of interest. [color].

of detail in the *snake*.

The choice for this number of points is further justified by the average computational time. Being the longest section in our computations, the active contour algorithm took an average of 15 min/image for $N = 50$, but 34 min/image for $N = 100$. The decrease in performance was negligible, and so the lower number was deemed appropriate.

Six geometric features were obtained from each *snake* following manually written NumPy code: the two coordinates $(\bar{x}, \bar{y})$ of the centroid; the *snake*'s normalized length, taken as the distance between its endpoints divided by a known threshold; the moment of inertia relative to a horizontal axis containing the *snake's* centroid, and its average and relative masses. Each of these features is thereby described in this section.

Eq. 6 gives the position of the centroid, where $x(i)$ and $y(i)$ represent the x and coordinates, in pixels, of the -th point on the *snake*. It is important to note that the (horizontal) axis in the spectrogram image corresponds to a sequence of time intervals where the spectrogram images were computed, and has no influence on whether a detection by the Hough algorithm is a true positive or not. It was, however, useful for quick manual identification of the *snakes*. The (vertical) axis in the spectrogram image should be influential, given that a centroid measurement within this axis describes the average frequency of the corresponding sound.

$$(\bar{x}, \bar{y}) = \frac{1}{N}\sum_{i=1}^{N}(x(i), y(i)) \tag{6}$$

Length was simply defined as the distance between the endpoints, then normalized by a selected value equal to the length of the largest *snake* encountered in a given set of images, and is given in Eq. 7.

$$L_{snake} = \frac{1}{l}\sqrt{(x(N) - x(1))^2 + (y(N) - y(1))^2} \tag{7}$$

The next three features make use of the gray level image function, defined to return the intensity of the image at the point divided by 255, so that it ranges from 0 (black) to 1 (white). Here, an important (albeit counter-intuitive) distinction shows that darker areas in the spectrogram image correspond to higher-intensity frequency patches, but were assigned with low values in the Matplotlib color map. The gray levels of the image were employed to represent the "density" of each point, for purposes of calculating the moment of inertia and a concept of mass, so that the patterns associated with the events of interest correspond to areas of lower image density.

The moment of inertia was calculated with respect to a horizontal axis passing through the snake's centroid, and is therefore defined by Eq. 8.

$$J_{snake} = \sum_{i=1}^{N} I(x(i), y(i)) \times (y(i) - \bar{y})^2 \tag{8}$$

With an analogous notion to density as defined by the gray level image function, it becomes possible to calculate the mass of a *snake* with points, and to further compare it to the density of the spectrogram image as a whole, shaped (pixels), used as a discretization of the and axes. The motivation for this comparison comes from the assumption that true detections will be darker than the mean gray level of the image. This line of reasoning will be further detailed in Section 3.

From the aforementioned interpretations follow the definitions for the masses of the snake: average M_{snake} and relative RM_{snake}, respectively shown in (Eqs. 9–10), where are points of the snake, as considered earlier in this section, and the sum of refers to adding up the gray level of every pixel in the spectrogram image.

$$M_{snake} = \frac{1}{N}\sum_{i=1}^{N} I(v(i)) \tag{9}$$

$$RM_{snake} = \frac{M_{snake}}{\frac{1}{ab}\sum_{i=1}^{b}\sum_{j=1}^{a} I(i,j)} \tag{10}$$

2.6. The random forest classifier

A random forest classifier (Breiman, 2001) is a collection of decision tree-like classifiers (Breiman et al., 1984) averaged on random subsets of the original data. The original implementation of a single decision tree involves branching decisions such as, for instance, deciding to classify a detection as true if its relative density goes below a certain cutoff.

In decision trees (James et al., 2017), two of the main criteria for evaluating where a split is necessary (e.g. a branching decision at specific values of a discrete feature, or thresholds of a continuous one) are calculating the entropy and the Gini index of a split.

For a categorical feature Y with K values, the entropy of a set D of observations is defined in Eq. 11.

$$I(D) = -\sum_{k=1}^{K}\frac{|D_k|}{|D|}\log_2\frac{|D_k|}{|D|} \tag{11}$$

Here, D_k is the subset of D where $Y = k$ and $|D|$ is the number of elements of the set D. From this definition, we can calculate the gain ratio of a split into subsets $D^1, \ldots, D^m$ by Eq. 12.

$$I_{gain} = \frac{I(D) - \sum_{j=1}^{m}\frac{|D^j|}{|D|}I(D^j)}{-\sum_{j=1}^{m}\frac{|D^j|}{|D|}\log_2\frac{|D^j|}{|D|}} \tag{12}$$

Therefore, we compute the gain ratio of every possible split in the decision tree, and choose the one with largest ratio.

Alternatively, we can compute the Gini index of a region D through Eq. 13.

$$g(D) = \sum_{k=1}^{K}\frac{|D_k|}{|D|}\left(1 - \frac{|D_k|}{|D|}\right) \tag{13}$$

The parameter for evaluating a split into sets D^1 and D^2 is thereby defined in Eq. 14.

$$g_{gain} = g(D) - \frac{|D^1|}{|D|}g(D^1) - \frac{|D^2|}{|D|}g(D^2) \tag{14}$$

A random forest trains several decision trees on various subsets of the dataset and averages their decisions, an effective method to reduce over-fitting. The implementation used in this study is the Python version made by scikit-learn, which is itself based on (Breiman, 2001) and (Geurts et al., 2006).

The definition of a random forest is an ensemble of tree-structured classifiers $h(x, \Theta_k)$, where $\{\Theta_k\}_{k=1, \ldots}$ are random vectors that are independent and identically distributed. Then, the final decision made by the random forest is the most popular decision among the classifiers at the input x.

A classic implementation of a random forest, such as the one used for the classification of the estuarine dolphin vocalization detections, randomizes the features at each cutoff, resulting in slightly different

classifiers every time the algorithm is run. Therefore, the final accuracy value obtained may fluctuate by small amounts.

It can be shown that a random forest always converges, and so the randomization of the features is an effective way to prevent overfitting, thereby eliminating concerns of cross-correlation between features. In (Breiman, 2001), the features are selected via an "out-of-bag" method.

The random forest was deemed an adequate choice for this study based on the presence of various natural thresholds and cutoff points for the features, as will become clear during the discussion of the results of the proposed method.

The training of the classifier constitutes the only human interference needed for the method. Once it is completed, the algorithm should classify new detections automatically.

3. Results

The relevant patterns identified in the available dataset, as well as the optimal accuracy obtained with the previously discussed Random Forest classifier, are the focus of this section.

The target variable for the classifier is binary, indicating whether a *snake* detection truly corresponds to the characteristic pattern of interest. It will henceforth be referred to simply as "target." Examples of correct and incorrect detections are shown in Fig. 11. The incorrect *snake* is shown in red, near the upper left corner.

3.1. Noteworthy patterns in the dataset

Through manual classification of the available *snakes*, three main descriptive features were created and represented by unique variables, thus aiding the construction of the classifier. The validity of such patterns was subsequently tested, finishing the construction of the feature vectors with 9 descriptive features for each *snake*.

The first such feature is referred to as the relative mass of the *snake*, defined previously in Section 2.5 through Eqs. 9 and 10. This parameter is a comparison between the gray levels of the points covered by the *snake* and the average gray level of the whole spectrogram image. It is expected that, for a true detection, the *snake* covers darker points, which correspond to higher frequency intensities in the spectrogram, and therefore has a low relative mass. An arbitrary threshold was selected through careful observation of the dataset, separating "low-" from "high-density" snakes in a binary variable. Thus, it was defined that a "low-density" *snake* (that is, a *snake* in a low-density region) satisfies

$$RM_{snake} \leq 0.8$$

A graphical representation of the number of manually classified detections, seen in Fig. 12, confirms the expected behavior. The columns on the graph, as labeled, indicate the behavior of the target valuable when the new "low-density" feature has a value of 1 (left) or 0 (right).

A second feature of interest is the length of the *snake*. It arises from the observation that, since false detections tend to be a random collection of alligned darker points, most of them will not be among the longest in the dataset. A fixed threshold can then be defined from this

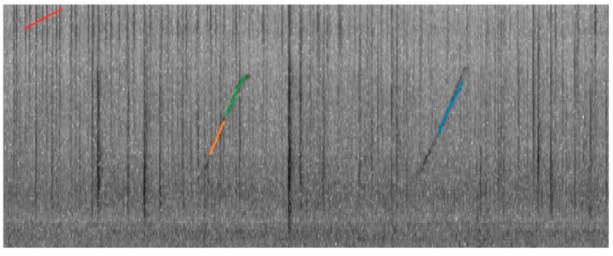

Fig. 11. Examples of three correct detections and a false positive one. [color].

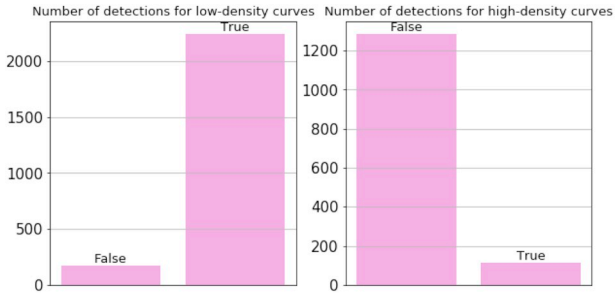

Fig. 12. Instances of the target valuable in relation to the binary density feature.

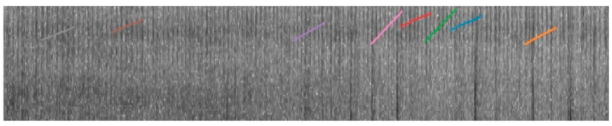

Fig. 13. Several false detections within a single image. [color].

intuitive notion. Meanwhile, many true detections may be relatively large in the image, such as the one in Fig. 4. Examples of this behavior in false detections are shown in Fig. 13.

With length defined by Eq. 7, a threshold was chosen for a desired length. It is expected that most long line-like patterns will be true detections, while short ones may not have any detectable bias. It was established that a "long" line-like pattern satisfies

$$L_{snake} \geq 0.52.$$

A graphical analysis of this new feature confirms the assumed bias, as shown in Fig. 14. The figure is a graphical representation of the behavior of the target variable when "long" is 1 (left) or 0 (right). Most "long" line-like patterns are, therefore, true detections, which may serve as a useful cutoff point for the random forest classifier.

A third and final new feature arises from a particular collection of images containing several counter-examples for the density feature. The presence of a constant low-frequency disturbance in the data, likely caused by passing ships, may generate darker patches in the image. Therefore, an algorithm set to detect dark line segments will output several false detections. *Snakes* that exhibit this behavior are shown in Fig. 15.

Hence, a calculation as to whether the *snake* corresponds to a range of low frequencies in the spectrogram is very important: in case it does, the detected event is likely to be a false positive. Other observations of the training data indicate that very few true detections lie between 0 Hz and 1 kHz, mostly due to the high-frequency nature of the pattern of interest.

This analysis leads to a third classification of line-like patterns as "low-" or "high-frequency". A "low-frequency" curve with N points is defined to be one such that its centroid satisfies

$$\bar{y} \leq 510$$

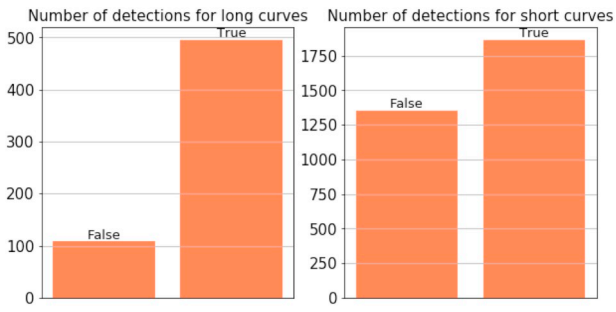

Fig. 14. Instances of the target valuable in relation to the binary length feature.

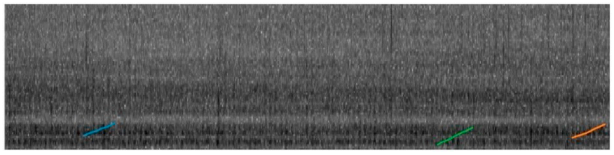

Fig. 15. Examples of false detections due to a low-density disturbance. [color].

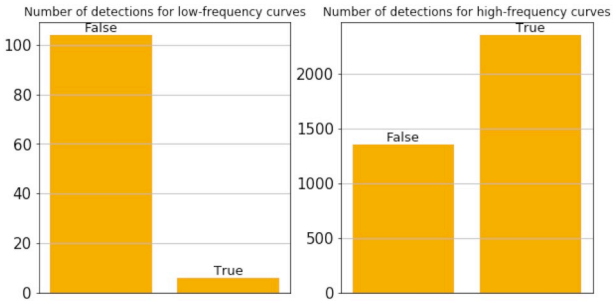

Fig. 16. Instances of the target valuable in relation to the binary frequency feature.

"High-frequency" line-like patterns are exempt from the considerations above, and thus expected to be free of any bias. This behavior is then visualized in Fig. 16.

A full correlation matrix for all 9 defined features plus the target variable, including both the new features and the ones previously discussed in Section 2.5, is shown in Fig. 17. It also confirms the previously predicted biases in the newly defined binary features.

The last row of the correlation matrix shows the correlations of the calculated features with the target variable. The patterns discussed above are thus confirmed: The binary feature "low-density" exhibits a very high correlation (0.84) with the target variable. The "low-frequency" feature, denoted simply as "low" in Fig. 17, has a negative correlation with the target variable, as expected.

Many of the features are cross-correlated. There are expected correlations between the binary features and their respective continuous ones. We also observe other correlations, e.g. between length and moment of inertia.

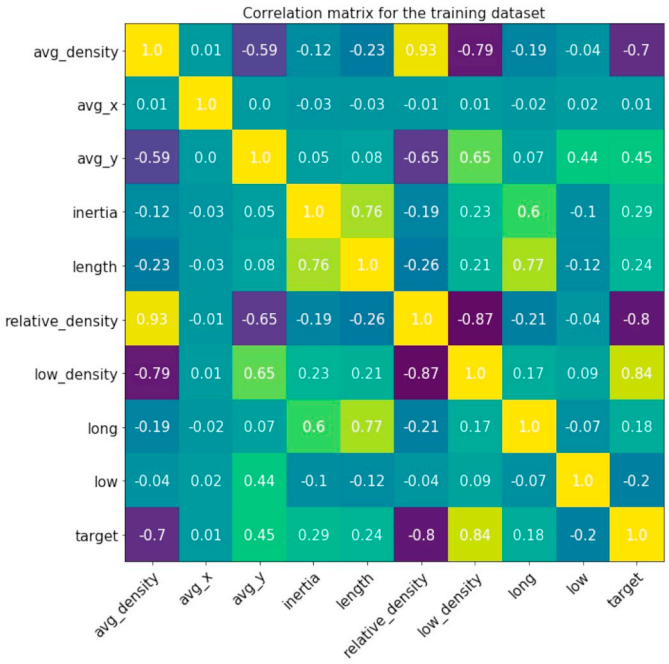

Fig. 17. Correlation matrix for the training dataset. [color].

3.2. Performance of the random forest classifier

We thus complete the feature vectors with three new binary features associated to the observed patterns. A Random Forest classifier was initialized and subsequently optimized by a grid search method. This optimization follows the scikit-learn implementation for Python, based on (Bergstra and Bengio, 2012).

A grid search consists of a brute force test of many combinations of the classifier's own hyperparameters within the training dataset. The hyperparameters chosen to be varied in the grid search process were two: the number of estimators within the Random Forest algorithm and the criterion (Gini or entropy) used in the creation of new branches.

Two different implementations of the Random Forest algorithm were attempted. The first made use of all available features, which can be seen in the correlation matrix of Fig. 17, except for *avg_x*, as previously discussed. The second one removed a continuous feature whenever a binary pattern had been defined from it. Thus, the three features defined in Section 3.1 were kept, and their corresponding features "avg_density," "length" and "avg_y" were removed.

Applications of those two classifiers after their optimization via the grid search method yielded predictions for the test dataset. The first implementation, with all the features, resulted in a total accuracy of 0.977. In the second implementation, the achieved accuracy was 0.966. This discrepancy held through 30 experimental re-runs of the algorithm, with the accuracies fluctuating between these values by negligible deviations, not large enough to cover the difference between them. The second implementation was thereby discarded.

The confusion matrix for the first implementation is shown in the following table, concerning classification among the test set. This set, as discussed on Section 2, is a random selection of 25% of the original *snakes*.

The confusion matrix shows a very small number of false positives and false negatives. Their numbers were small compared to the number of correct detections by the random forest classifier, obtained as $TN + TP = 933$.

For this classifier we calculate precision p and recall r:

$$p = \frac{TP}{TP + FP} = 0.978;\ r = \frac{TP}{TP + FN} = 0.984$$

The F1-score is then the harmonic mean:

$$F_1 = \frac{2pr}{p + r} = 0.981$$

We should note that the F1-score for the discarded classifier was 0.953, also lower than our final choice.

3.3. Computational performance of the algorithm

As discussed in Section 2, the raw audio data used for this research was, in total, 5 h and 30 min of audio.

Generation of the Frangi maps took 10 min in total. The Hough transform algorithm for straight line detection was faster, averaging at 1 min for all images. The *snake* algorithm took most of the runtime, averaging 15 min per image. The full implementation and testing of both Random Forest classifiers took, on average, 23 min per full run, after the training dataset had already been built.

All computations were run on a laptop operating Windows 10, with 8GB RAM available and an Intel(R) Core(TM) i7-5500U CPU @ 2.40 GHz processor, on the native Python 3.6 IDLE environment.

4. Conclusions

In this paper, a method for detecting estuarine dolphin vocalizations without intervention of a human operator was developed.

The proposed method is based on the application of a probabilistic Hough transform over the Frangi ridge map of a spectrogram image.

Table 1
Confusion matrix for the classifier.

		Predicted label	
		0	1
True label	0	True negatives: TN = 367	False positives: FP = 13
	1	False negatives: FN = 9	True positives: TP = 566

Furthermore, the line segments obtained are run through an active contour algorithm and geometric features are extracted. The resulting curves and their features form a dataset for the application of the random forest algorithm.

All data used in our research consists of feature vectors extracted from curves identified in spectrogram images from the underwater soundscape where the referred species is found.

The proposed algorithm offers the following advantages in relation to current methods in the literature: it requires no human interference aside from data collection and organization; it automatically classifies identified patterns as estuarine dolphin vocalizations or false detections; it is based only on the approximate linearity of the pattern, and therefore still performs well in spectrogram images containing high noise levels.

The best implementation of the random forest classifier obtained a raw accuracy of 0.977, as taken from Table 1, and F1-score of 0.981.

For future research, we intend to investigate the performance of the algorithm in spectrogram images that contain not only estuarine dolphin vocalizations, but include other characteristic acoustic events, both anthropogenic and natural.

We intend to test our algorithm in images with lower signal-to-noise ratio and extend the method implemented in this article to be able to recognize vocalizations from other species of cetaceans.

Declaration of Competing Interest

The authors declare that they have no competing interests.

Acknowledgements

This work was partly supported by CNPq grant 303992 / 2017-4 and FAPESP grants CEPID-Shell-RCGI 2014/50279-4.

References

Bazúa-Durán, C., 2004. Differences in the whistle characteristics and repertoire of bottlenose and spinner dolphins. An. Acad. Bras. Cienc. 76.
Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305.
Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.
Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth, Belmont, CA. .
Duda, R.O., Hart, P.E., 1972. Use of the hough transformation to detect lines and curves in pictures. Commun. ACM 15, 11–15 January.
Erbs, F., Elwen, S.H., Gridley, T., 2017. Automatic classification of whistles from coastal dolphins of the southern African subregion. J. Acoust. Soc. Am. 141, 2489–2500.
Esfahanian, M., Zhuang, H., Erdol, N., 2013. Using local binary patterns for classification of dolphin calls. J. Acoust. Soc. Am. 134, EL105.
Flach, L., Flach, P.A., Chiarello, A.G.D., 2008. Abundance and distribution of the guiana dolphin (Sotalia guianensis van Benéden, 1864) in Sepetiba Bay, Southeast Brazil. J. Cetacean Res. Manag. 10 (1), 31–36.
Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A., 1998. Multiscale vessel enhancement filtering. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Berlin Heidelberg, pp. 130–137.
Fu, W., Breininger, K., Würfl, T., Ravikumar, N., Schaffert, R., Maier, A., 2017. Frangi-Net: A Neural Network Approach to Vessel Segmentation. arXiv: 1711.03345v1 [cs.CV]. 9 Nov. .
Galambos, C., Matas, J., Kittler, J., Apr 2000. Progressive probabilistic Hough transform for line detection. IEEE Comp. Soci. Conf. Comp. Vis. Patt. Recog. 78 (1), 119–137.
Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63 (1), 3–42.
Gillespie, D., 2004. Detection and classification of right whale calls using an "edge" detector operating on a smooth spectrogram. Canad. Acoust. 32 (2), 39–47.
Henderson, E.E., Hildebrand, J.A., Smith, M.H., Falcone, E.A., Jul 2012. The behavioral context of common dolphin (*Delphinus* sp.) vocalizations. Marine. Mammal. Sci. 28 (3), 439–460.
Hunter, J.D., 2007. Matplotlib: a 2D graphics environment. Comput. Sci. Eng. 9 (3), 90–95.
ICMBio, 2011. Plano de Ação Nacional para a Conservação dos Mamíferos Aquáticos – Pequenos Cetáceos. Instituto Chico Mendes de Conservação da Biodiversidade. ICMBio, Brasília, Brazil.
James, G., Witten, D., Hastie, T., Tibshirani, R., 2017. An Introduction to Statistical Learning, 7th edition. .
Kahl, S., Hussein, H., Fabian, E., Schloßhauer, J., Thangaraju, E., Kowerko, D., Eibl, M., 2017. Acoustic Event Classification Using Convolutional Neural Networks. Gesellschaft für Informatik. .
Karnowski, J., Johnson, C., 2015. Dolphin detection and Tracking. 2015 IEEE Winter Conference on Applications of Computer Vision Workshops. pp. 51–56.
Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: active contour models. Int. J. Comput. Vis. 1 (4), 321.
Kershenbaum, A., Roch, M.A., 2013. An image processing based paradigm for the extraction of tonal sounds in cetacean communications. J. Acoust. Soc. Am. 134, 4435.
Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., 2016. Jupyter Development Team. Jupyter Notebooks – A Publishing Format for Reproducible Computational Workflows. IOS Press, pp. 87–90.
Kriesell, H.J., Elwen, S.H., Nastasi, A., Gridley, T., 2014. Identification and characteristics of signature whistles in wild bottlenose dolphins (*Tursiops truncatus*) from Namibia. PLoS One 9 (9), e106317.
MacBride, M., Roth, L., 2016. Bottlenose dolphin whistle characterization using Eigenwhistle based approach. IJCSET 6 (8), 306–310.
Malfante, M., Mars, J.I., Dalla Mura, M., Gervaise, C., 2018. Automatic fish sounds classification. J. Acoust. Soc. Am. 143, 2834–2846.
McKinney, W., 2010. Data structures for statistical computing in Python. Proce. 9th Python. Sci. Conf. 51–56.
Mellinger, D.K., Martin, S.W., Morrissey, R.P., Thomas, N., Yosco, J.J., 2011. A method for detecting whistles, moans, and other frequency contour sounds. J. Acoust. Soc. Am. 129, 4055–4061.
Monteiro Filho, E.L.A., 2008. Biologia, ecologia e conservação do boto-cinza.
Oliphant, T.E., 2006. A guide to NumPy (Vol. 1). In: Trelgol Publishing USA.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Sánchez-Gendriz, I., Padovese, L.R., 2016. Underwater soundscape of marine protected areas in the south Brazilian coast. Mar. Pollut. Bull. 105, 65–72.
Secchi, E., de Santos, M.C.O., Reeves, R., 2018. *Sotalia guianensis* (errata version published in 2019). In: *The IUCN Red List of Threatened Species* 2018: e.T181359A144232542, Downloaded on 22 June 2019.
van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., 2014. The scikit-image contributors. scikit-image: image processing in python. PeerJ 2, e453.