Universidad Autónoma de Madrid
Escuela politécnica superior

Doble Grado en Ingeniería Informática y Matemáticas

Trabajo Fin de Grado

APREDIZAJE AUTOMÁTICO A PARTIR DE FORMAS

Autor: Adrián Muñoz Perera
Tutor: Alberto Suárez González

Julio 2018

Universidad Autónoma de Madrid
Escuela politécnica superior

Double Degree in Computer Engineering and Mathematics

Final Degree Project

MACHINE LEARNING FROM SHAPES

Author: Adrián Muñoz Perera
Tutor: Alberto Suárez González

July 2018

MACHINE LEARNING FROM SHAPES

Author: Adrián Muñoz Perera
Tutor: Alberto Suárez González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
July 2018

Abstract

Resumen

En este trabajo se afronta el problema de aprender a partir de imagenes para realizar clasificación y agrupamiento de formas. La idea principal consiste en codificar los ejemplos de aprendizaje como datos direccionales que se utilizarán para identificar estas formas y realizar comparaciones entre ellas. Los objetos a estudiar serán formas de 2 y 3 dimensiones, que serán caracterizadas por la distribución de las direcciones de los vectores normales a los hiperplanos tangentes al borde de la misma. Estos datos direccionales manipulados como datos funcionales se pueden utilizar para codificar las formas en dos representaciones discretas: un histograma normalizado y una estimación kernel de la densidad que serviran como estimaciones para la distribucion de probabilidad de cada forma. Estas representaciones son utilizadas para extraer características relevantes basadas en métricas sobre el espacio de distribuciones circulares y para categorizar y comparar las formas codificadas. Estas técnicas de caracterización y comparación serán incluidas en algoritmos de clustering y de clasificación, aplicándose en un problema de reconocimiewnto de formas simples y en una aplicación al mundo real de clasificación de otolitos de peces.

Palabras Clave

Datos direccionales, distribución de Von Mises, agrupamiento de formas, clasificación de formas.

Abstract

In this work the problem of learning from images to perform grouping and classification of shapes is addressed. The key idea of the shape recognition approach is to encode the instances available for learning in the form of directional data, that will be used to characterize those instances and perform comparisons among them. The objects to study are thus 2 and 3 dimensional shapes, that will be characterized by the distribution of the direction of the normal vectors to the tangent hyperplanes at the boundary of the shape. In two dimensions, this boundary is a contour, and these directional data will in fact form a curve, that manipulated as functional data can be used to encode the shapes in two discrete representations: a normalized histogram and a kernel density estimation for the probability function. These representations are used to extract characteristics based on metrics defined in the space of circular distributions, categorize the encoded shapes and finally compare them. These characterization and comparison techniques will be later embedded in some clustering and classification algorithms, applying them in a simple shape recognition problem and a real world problem of clustering and classification with fish otolith shapes.

Key words

Directional data, Von Mises distribution, shape clustering, shape classification.

Acknowledgements

Gracias a mis padres, porque este es también su trabajo.

Gracias a mi familia y amigos, que por suerte son muchos.

Gracias a mi tutor, Dr. Alberto Suárez González por confiar en mí, por transmitirme sus conocimientos y darme la oportunidad de realizar una investigación tan bella y entretenida.

Gracias a todos los que nunca se han rendido, porque nunca se han rendido.

Contents

Figure Index

Table Index

1 Introduction

In this work, the problem of identifying shapes of object is addressed. Basically, the problem consists in given a set of figures (2-dimensional o 3-dimensional objects) with a recognizable shape, develop a method to identify different shapes among the set of figures and classify each figure with a particular shape. This problem of shape recognition has many real world applications in fields such as biology [1, 2, 3] or medicine [4]. In the approach proposed, a functional data point of view is adopted to characterize the shape of a figure and encode this information as a probability distribution [5, 6]. Different clustering and classification paradigms will later be used to test the robustness of the characterizations, in two different problems of shape recognition.

In this document, the methodology proposed is explained, including definitions and conventions adopted. Both shape recognition problems will serve to illustrate the basic lines of the approach and detail each subprocess. In Chapter 3, the representation method for planar figures will be explained, with an application to one of the shape recognition problems with simple planar figures. In this chapter the processes of retrieving, manipulating and evaluating data of these planar figures will be explained, proving the effectiveness with the results of the shape recognition problem.

A real world application in a otolith classification problem is performed in Chapter 4, applying similar methodology as in the simple figure problem in Chapter 3. In this chapter, the methodology is adapted to the otolith problem. The results are competitive with the state of art techniques [2, 7] proving the usefulness of the directional approach.

In Chapter 5, the software tool employed to simulate and perform the experiments is described, including the project plan and methodology followed during the development of the tool. This section will include requirements updated during the process and a description of frameworks employed, as well as a depiction of the system design. And finally, in Chapter 6 a conclusion is performed, pondering the process of development, the knowledge acquired during this process and the results obtained with the approach with a brief mention of future work proposals related with the issue, a few of them slightly explored during the development will be performed at the end of the chapter.

2

State of Art

Automatic induction from complex data that are characterized by functions, graphs, distributions or shapes is one of the important open problems in Machine Learning nowadays [5]. One of the difficulties of this task is to provide an appropriate characterization of shape that is tractable yet preserves sufficient amounts of information to allow grouping and discrimination while keeping the robustness in the characterization. The problem is that in general these objects usually can not be easily parameterized, and models can have a very complicated topology. Previous shape matching methods have employed characterization of figures based on 2D contour, 3D volumes and surfaces, structural models or statistics [8, 9, 10].

Among these approaches, a popular trend to shape analysis uses high-level representations of shapes, decomposing 2D and 2D objects into features and computing dissimilarity measures between objects according to these features. Examples of such representations are landmark methods [5], skeleton representations [11], medial representations [12, 4], generalized cylinders [13] or probability distributions over features of the figures [14, 15, 2, 7]. The main problem with many of these approaches is the well definition of these high-level representations in general, for instance identifying relevant and robust landmarks in objects for landmark representations, or when objects are not simple connected in the case of skeleton representations [15].

In this work a functional approach is adopted [6] and shapes are characterized by the distribution of the normal vectors to the curve (in 2 dimensions) or surface (in 3 dimensions) that delimits the figure [16]. This approach offers a statistical directional alternative to most methods in the current state of art based on distance measurements [2, 3]. These representations are based on encoding shapes using the distribution of distances between points within [2] or at the boundary of the object [3].

The representation employed in this work requires of the tools of directional statistics [17]. In particular, a very important phase in preprocessing is kernel density estimation with circular data [18] that extends the theory of this non-parametric estimation method. Tools for automatic kernel bandwidth selection [7, 19, 20] or parameter estimations [21, 17, 22] will be applied in this project to process and analyze data.

3

Figure Characterization

In this chapter, the approach proposed in this work is described. In Section 3.1, objects in the scope of the studies will be described. as well as some basic definitions to employ. In section 3.2 and 3.3 features to extract from these objects and strategies to manipulate them are developed. From these features, objects will be characterized, and finally in Section 3.4 various experiments of clustering and classification are performed, in order to test the characterization validity.

3.1 Figures and Shapes

According to Kendall [23], "Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object". In this definition are included "good deformations", this is deformations within some limits, so when they are applied, the information of the object is kept [2].

The objects to study in this paper are figures, that can be defined as compact manifolds in general, and in this particular case, 2-dimensional (planar) figures. It will define informally shape as a pattern that is shared among a set of figures that are obtained from a prototype figure applying these "good transformations", it is, transformations that are "good deformations" or isometries. This prototype must be understood as an ideal concept and difficult to define in general. These "good deformations" could be defined more precisely, but it is not clear the limits when a transformation is either good or bad, neither when the correspondence between the prototype and the figure is not clear and the figure looses the original shape, since for instance a square and a circle are even homotopic.

Shape also can be understood as a equivalence class in a set of figures, it is, given a set of figures, and a given number of shapes to group these figures into, the figures can be grouped into disjoint sets according to some geometrical or topological criteria. The subset in which a figure is labeled is the shape of the figure and defines an equivalence relation in the set of planar figures. If the division in shapes is not arbitrary, there must be a pattern among the figures inside each shape understood as a set. With this definition, a bad transformation could be understood as

one that makes a figure move to another shape, it is, change the subset in which is classified before applying the transformation. See Figure 3.1 for a visual example of the ambiguity of the concept of shape.

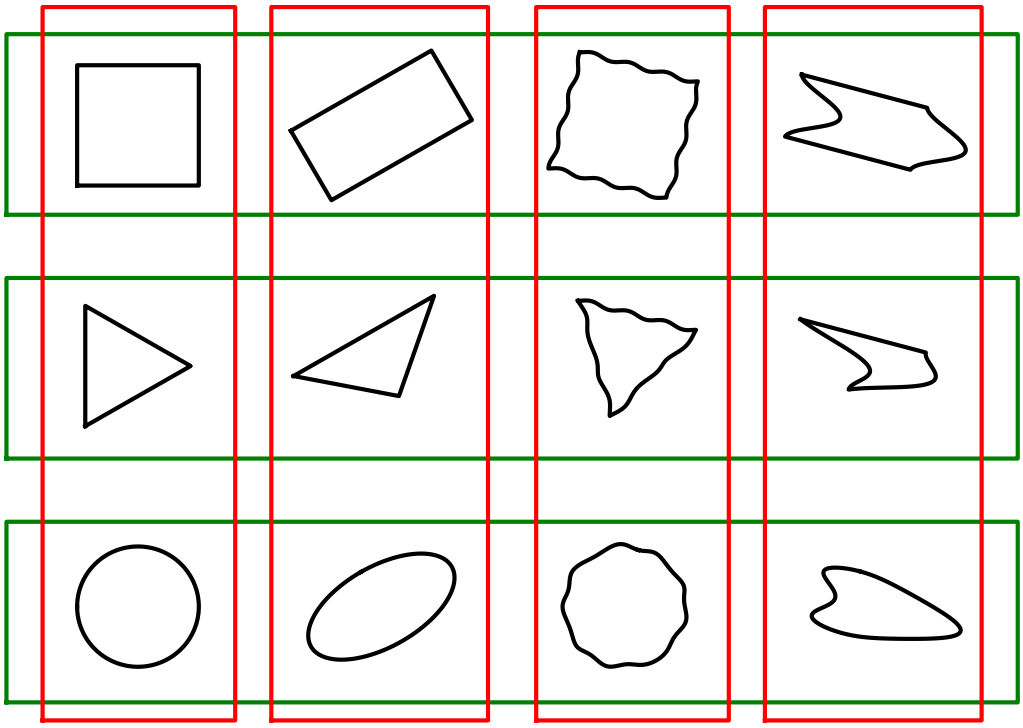
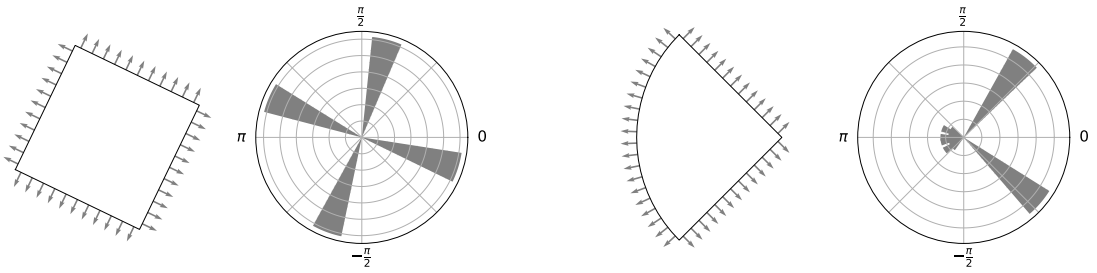


Figure 3.1: Different figures obtained applying isometries and deformations to basic templates. The last column could be an example of a "bad transformation". The information about the original template is very distorted. In green and red, two reasonable partitions according to two interpretations of the shape of these figures, one by the original template easily identified and other by the regularity of the boundary.

With this informal definition of shape, intuitively the goal is a method to obtain this information from a given set of figures, thus identifying different patterns among to create subsets of figures according to their shape. Figures must be encoded keeping that information of shape, and the idea is to reduce the process of comparing two figures to the process of comparing two simpler characterizations keeping the correspondence between this characterization and the original figure. In [5] a brief summary of different alternatives to encode two and three dimensional figures is developed.

3.2 Directional Variable

Figures will be characterized by their directional variable, an aleatory variable encoding the distribution of the normal unit vectors to the contour of the figure. These directional variables can be identified by the probability distribution of the direction of these vectors, which are periodic one dimensional probability distributions in the case of planar figures. These variables will take values in the interval $[-\pi, \pi]$ and in general are not continuous or discrete, these will be a mixture. Examples of such probability density estimates are depicted in Figure 3.2.



(a) Square (b) Circular sector

Figure 3.2: Empirical probability density for the directional variables in simple figures.

An estimation of the orientation distribution of a figure can be obtained by sampling N points at regular intervals along the contour of the figure. The number of intervals to consider is crucial for the characterizations, since bigger number of intervals will produce unstable estimations, while smaller number of intervals will cause a significant loss of information. At each of these points the direction of the normal to the contour is computed and store the corresponding angles $\{\theta_n\}_{n=1}^{N}$. An empirical estimate of the probability density is given by the histogram of the data using N_{bins} equally spaced bins in $[-\pi, \pi]$. The histogram is scaled so that the area under it is one. Alternatively, a kernel estimator is used to provide a smooth approximation of the density

$$f_{KDE}(\theta; \nu) = \frac{1}{N} \sum_{n=1}^{N} K(\theta - \theta_n; \nu), \tag{3.1}$$

where $K(\theta; \nu)$ is a periodic normalized kernel (i.e. its integral in $[-\pi, \pi]$ is 1), whose characteristic width is $h = 1/\nu$. In this work, the Von Mises kernel is used

$$K(\theta; \nu) = \frac{1}{2\pi I_0(\nu)} e^{\nu \cos(\theta)}, \tag{3.2}$$

where I_0 is the modified Bessel function of the first kind of order 0. For higher dimensions the von Mises-Fisher distribution can be used [24]. The quality of the kernel density estimate depends strongly on the value of this parameter [25, 19, 7, 20]: On the one hand, if the kernel is too narrow, the density estimate will lack stability and exhibit large variance. If, on the other hand, the width is too large, relevant features of the probability density will be smoothed out.

As it will be shown in Section 3.4, the kernel density characterization will provide better results in classification than histogram representation due to robustness of the data for this particular problem, and also in the real world application (See Section 4.3). In Figure 3.3 the function obtained for simple polyhedra using a kernel density estimator is depicted.

For the bandwidth selection, a crucial parameter for the estimation, an analogous for circular data to the rule of thumb adapted from [7] was studied, since this rule will provide robust estimations against deviations from von Mises-Fisher densities. The formula to obtain this optimal bandwidth is

$$h = \sqrt[5]{\frac{4\sqrt{\pi} I_0(\hat{\kappa})^2}{\hat{\kappa}[2I_1(2\hat{\kappa}) + 3\hat{\kappa} I_2(2\hat{\kappa})]n}}$$

where $\hat{\kappa}$ is the estimation of the concentration parameter κ of the variable, the equivalent to the inverse of the variance in normal distributions. The maximum likelihood estimate for this values is obtained as the solution to a equation involving Bessel functions, and a numerical approximation is proposed in [17]. See also [26] for even more information about estimation of the concentration parameter of von Mises-Fisher distributions.

In general, since figures are very heterogeneous and in most cases, the rules applied to select the bandwidth produce a significant loss of information with oversmoothed estimations, since these distributions of the orientations of shapes are in general far from von Mises-Fisher distributions , and the methods used suppose the sample to be originated from a von Mises-Fisher distribution mixture [7, 25]. In some cases, could be needed and easier to set by inspection the bandwidth parameter for the estimation, according to the problem and the shape of data. Good results in the problems have been obtained with bandwidths in the range between $2\pi/16$ and $2\pi/64$, being $2\pi/64$ a good value in most cases.

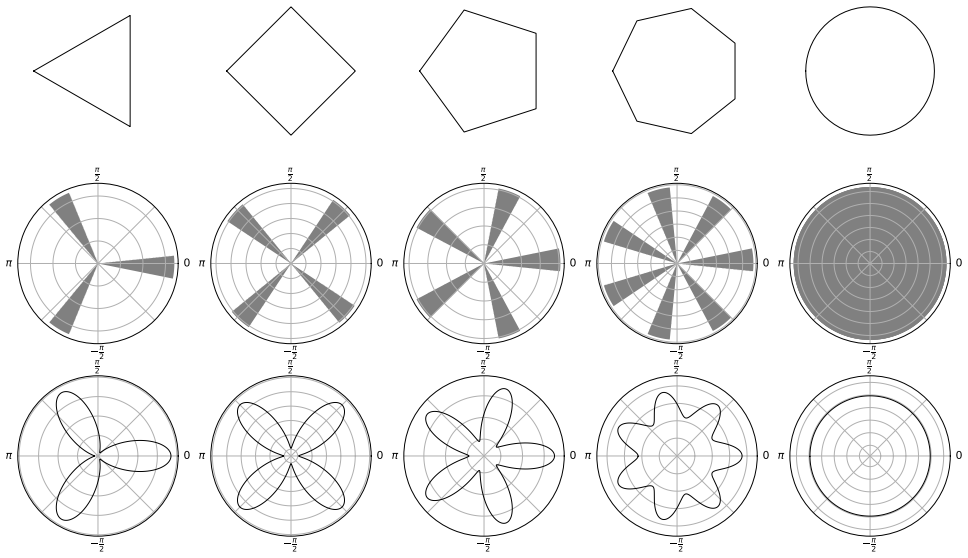


Figure 3.3: Characterization of simple planar geometrical figures (top row) using a scaled histogram (middle row) and kernel density (bottom row) empirical estimates of the probability density of the direction of the normal vectors along the contour of the figure.

3.3 Metrics

Once the object representations have been characterized by the corresponding probability density estimates, their shapes can be compared using different metrics. To define these metrics, discretized versions of the probability densities $f(\theta)$ and $g(\theta)$ in the circle ($\theta \in [-\pi, \pi]$), at the sampling points $\{\theta_n\}_{n=1}^N$ are considered; namely, $\mathbf{f} = \{f_n\}_{n=1}^N$, and $\mathbf{g} = \{g_n\}_{n=1}^N$, with $f_n \equiv f(\theta_n)$ and $g_n \equiv g(\theta_n)$, respectively. In terms of these discrete versions of the densities, the metrics in Table 3.1 have been considered.

Distance	**Expression**
Manhattan	$L_1(\mathbf{f}, \mathbf{g}) = \sum_{n=1}^N \lvert f_n - g_n \rvert$
Euclidean	$L_2(\mathbf{f}, \mathbf{g}) = \sum_{n=1}^N \lvert f_n - g_n \rvert^2$
Total Variation	$L_\infty(\mathbf{f}, \mathbf{g}) = \max_{n=1,2,\ldots,N} \lvert f_n - g_n \rvert$
χ^2	$\chi^2(\mathbf{f}, \mathbf{g}) = \sum_{n=1}^N \frac{\lvert f_n - g_n \rvert^2}{f_n + g_n}$
Hellinger	$H(\mathbf{f}, \mathbf{g}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{n=1}^N \left(\sqrt{f_n} - \sqrt{g_n}\right)^2}$
Earth Mover's	$EMD(\mathbf{f}, \mathbf{g}) = \inf \mathbb{E}\left[\lvert \theta - \theta' \rvert\right]$ where the infimum is taken over all possible joint distributions θ and θ', random variables whose marginals are $\mathbf{f}$ and $\mathbf{g}$, respectively.

Table 3.1: Distance functions between the discrete versions of the distributions $\mathbf{f} = \{f_n\}_{n=1}^N$, and $\mathbf{g} = \{g_n\}_{n=1}^N$.

Let F and G be two figures, characterized by $\mathbf{f}$ and $\mathbf{g}$, respectively. The relative orientations of F and G could be different, therefore, a distance between these figures can be defined as the minimum value of the metric between a rotation of the first density and the second density

$$\hat{D}(F, G) = \min_{n=1,2,\ldots,N} D\left(\mathbf{f}^{[n]}, \mathbf{g}\right) \tag{3.3}$$

where D is one of the metrics considered and $\mathbf{f}^{[n]} = \{f_n, f_{n+1}, \ldots, f_N, f_1, \ldots, f_{n-1}\}$ is a rotation of the density $\mathbf{f}$.

The distance function given by Eq. (3.3) requires the evaluation of the specified metric for all N sampling points, which is a costly computation. An alternative is to approximate those distances by a aligning solution, where before computing the distances, discrete versions are aligned according to the mode of their samples. They are rotated in order to situate this mode in the first position.

Even a more effective way to account for rotations is to measure distances with respect to C, the uniform circular distribution in two dimensions (or the uniform distribution on a n-sphere in higher dimensions), which is invariant by rotations:

$$\tilde{D}(F, G) = \lvert D(F, C) - D(G, C) \rvert . \tag{3.4}$$

As will be illustrated in the section on empirical evaluation this two alternatives retain

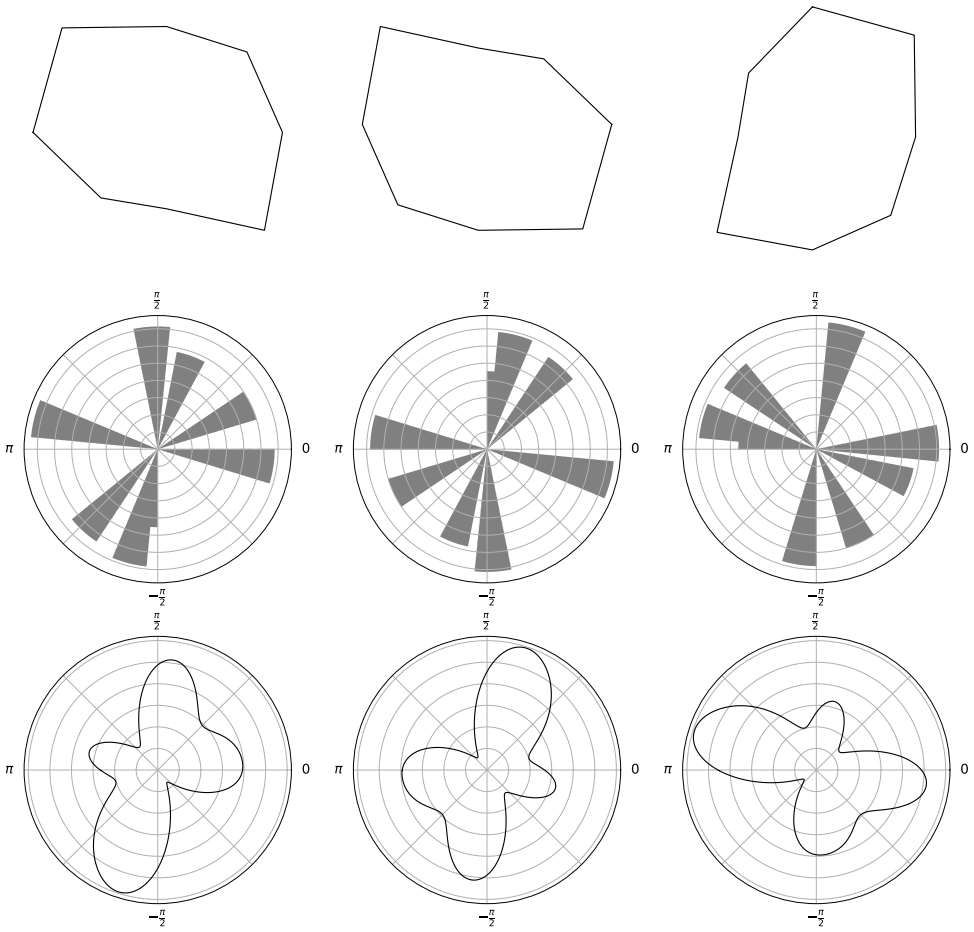


Figure 3.4: Rotated representations due to different orientations of the original figure (top row) using a scaled histogram (middle row) and kernel density (bottom row) empirical estimates of the probability density of the direction of the normal vectors along the contour of the figure.

sufficient information to provide reasonably good characterizations of shape at a reduced computational cost.

3.4 Clustering and Classification

To test the characterization, an easy problem of classification is faced creating a data set of 300 simple planar figures with recognizable shape. The shapes selected are triangular, rectangular and circular. 100 figures of each shape from a respective template are generated, by applying small deformations to the original template. Once the figures are generated, for each figure the histogram and the kernel density estimations are computed, including distances characterizations defined in Subsection 3.3. So in summary, the problem consists in classifying 300 simple figures, employing the characterizations previously defined.

To obtain the characterizations of these figures, the parameters mentioned of number of bins for the histogram representation and number of points of the discrete version of the kernel density are both 64 in all experiments performed in this section. Usually powers of two have been chosen for these parameters. In Figure 3.5 a caption of the data set is displayed, while in Figure 3.6 some examples of histogram and kernel density representations are displayed.

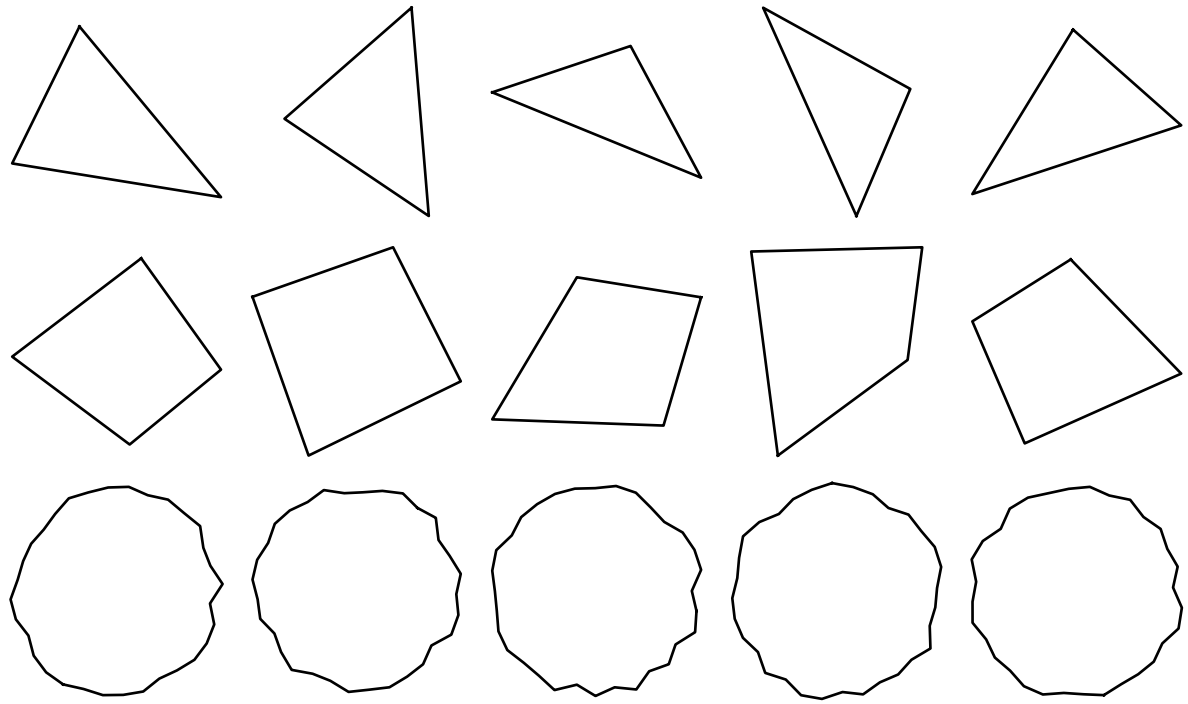


Figure 3.5: Caption of simple shapes used to test directional characterization

A K-Means model will be used to test the characterization as a unsupervised model, while a K Nearest Neighbors (K-NN) approach will be employed to test characterizations in supervised learning.

3.4.1 Clustering

For clustering, a K-means algorithm to generate three clusters is employed, corresponding to each shape in the data set (triangular, rectangular and circular). Two different models are proposed, with two alternatives obtained from the concepts developed in Section 3.3.

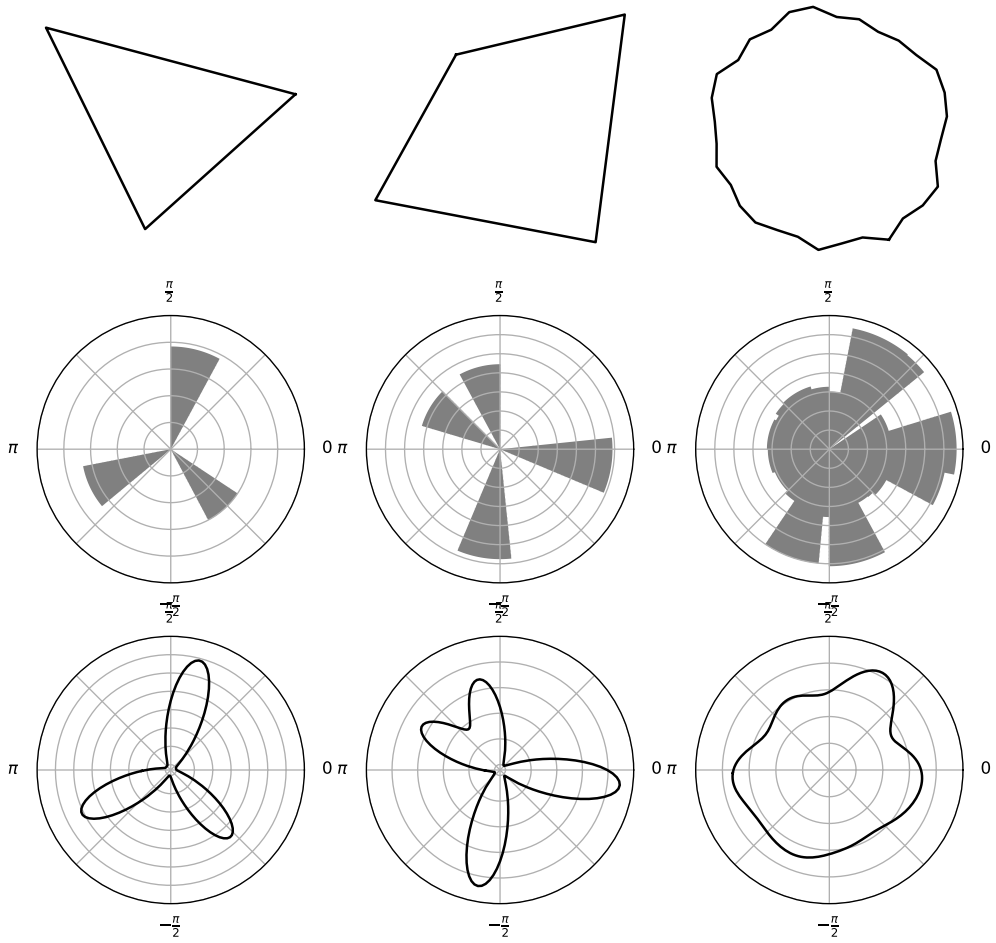


Figure 3.6: In the top row of this figure the contours of simple figures of different shapes are shown: *triangular* (left), *rectangular* (middle), and *circular* (right). The histogram and kernel density estimates of the direction variables for these figures are displayed in the middle and bottom rows, respectively.

Clustering with Aligning Approximation

In the first models, the attributes to employ are the mode aligned version defined in Section 3.3 of the histogram and the kernel density characterizations of figures in the data set. So the data set has 300 64-dimensional vectors for each characterization. The results of the clustering process are displayed in Table 3.2.

Clustering: Aligned						
	Histogram			Kernel Density		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
t	44	56	0	99	1	0
r	48	52	0	0	71	29
c	0	0	100	0	0	100
Err	0.35			0.10		

Table 3.2: Matrices of the result of the clustering process with histogram and kernel density representations. Labels in rows **t**, **r**, **c**, respectively correspond to labels triangle rectangle and circle while labels in columns correspond to different resulting clusters **C1**, **C2**, **C3** .

Clustering with Distances Approximation

In second models, data used consist on the different distance characterizations of figures defined in Section 3.3, so there are 300 instances of 6-dimensional vectors, each 6-dimensional vector encoding one figure in the data set, for the histogram and kernel density characterizations respectively. The result of the clustering experiment is shown in Table 3.3.

Clustering: Distances						
	Histogram			Kernel Density		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
t	0	100	0	100	0	0
r	0	100	0	0	100	0
c	45	0	55	0	0	100
Err	0.48			0.00		

Table 3.3: Matrices of the result of the clustering process for histogram and kernel density representations of simple figures. Labels in rows **t**, **r**, **c**, respectively correspond to labels triangle rectangle and circle while labels in columns correspond to different resulting clusters **C1**, **C2**, **C3** .

Note that figures are being characterized with 6-dimensional vectors when these distance characterizations are used as attributes, which is a significant reduction of the information, and still the figures are distinguishable. In this particular problem of clustering, the solution when aligned data is employed provides bad results in all cases, due to the regularity of the figures in the data set. This regularity causes the mode of the distribution to weaken.

3.4.2 Classification

The attributes employed in this classification experiment are exactly the same as used in the previous experiments on clustering. A K-NN approach is mainly used, while several models have been tested such as Gaussian Naive-Bayes or Random Forest classifiers, all of them providing good results, but not included in this paper. The number of neighbors is automatically selected during the training phase among values between 1 and 13, by 3-fold cross validation over the training set, with minimal variations in the results whit different values of this parameter. To obtain an estimation of the generalization error, 10-fold cross validation is performed over each

data set in all cases. Three different approaches employing different the three alternatives as distances explained in Section 3.3 will be explained in this three next subsections.

Classification with Periodic Metrics

In this first set of experiments, the K-NN classifiers will employ different periodic metrics defined in Section 3.3. The attributes are the discrete versions of the histogram and the kernel density representations. In Table 3.4 confusion matrices and generalization error estimates for each model are shown.

Classification: Periodic Distances													
		Histogram			Kernel Density				Histogram			Kernel Density		
		t	**r**	**c**	**t**	**r**	**c**		**t**	**r**	**c**	**t**	**r**	**c**
t	L_1	93	7	0	100	0	0	EMD	96	4	0	100	0	0
r		5	87	8	0	100	0		2	89	9	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.07 ± 0.04			0.00 ± 0.00				0.05 ± 0.02			0.00 ± 0.00		
t	L_2	92	8	0	100	0	0	H	91	8	1	100	0	0
r		2	81	17	0	100	0		7	77	16	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.09 ± 0.04			0.00 ± 0.00				0.11 ± 0.07			0.00 ± 0.00		
t	L_∞	94	6	0	100	0	0	χ^2	88	11	1	100	0	0
r		2	87	11	0	100	0		6	71	23	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.06 ± 0.06			0.00 ± 0.00				0.14 ± 0.06			0.00 ± 0.00		

Table 3.4: 10-fold cross-validation estimates of the confusion matrices for simple figures classification with periodic metrics. The instances, which are characterized by either the histogram or a kernel estimate of the probability density function of the normal vectors along the contour of the figure, are categorized as *triangular* (**t**), *rectangular* (**r**), and *circular* (**c**) using K-NN. The error indicates the proportion of incorrectly grouped instances.

Classification with Aligning Approximation

In this second set of experiments, the K-NN classifiers will employ the aligned approximation to the periodic distances defined in Section 3.3. The attributes are the discrete versions of the histogram and the kernel density representations. In Table 3.5 confusion matrices and generalization error estimates for each model are displayed.

Classification with Distances

In this last set of experiments, the K-NN classifiers will employ the distances defined in Section 3.3 based on the comparison against a uniform circular distribution. The characterization are the discrete versions of the histogram and the kernel density representations. In Table 3.6 confusion matrices and generalization error estimates for this approach are displayed.

Classification: Aligning Distances													
		Histogram			**Kernel Density**				**Histogram**			**Kernel Density**	
		t	**r**	**c**	**t**	**r**	**c**		**t**	**r**	**c**	**t**	**r**	**c**
t	L_1	90	8	2	100	0	0	EMD	94	6	0	100	0	0
r		12	40	48	0	100	0		3	84	13	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.23 ± 0.06			0.00 ± 0.00				0.07 ± 0.04			0.00 ± 0.00		
t	L_2	95	5	0	100	0	0	H	90	7	3	100	0	0
r		3	86	11	0	100	0		14	43	43	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.06 ± 0.03			0.00 ± 0.00				0.22 ± 0.05			0.00 ± 0.00		
t	L_∞	95	5	0	100	0	0	χ^2	90	7	3	100	0	0
r		3	85	12	0	100	0		13	44	43	0	100	0
c		0	0	100	0	0	100		0	0	100	0	0	100
Err		0.07 ± 0.05			0.00 ± 0.00				0.22 ± 0.06			0.00 ± 0.00		

Table 3.5: 10-fold cross-validation estimates of the confusion matrices for simple figures classification with the aligned representation method. The instances, which are characterized by either the histogram or a kernel estimate of the probability density function of the normal vectors along the contour of the figure, are categorized as *triangular* (**t**), *rectangular* (**r**), and *circular* (**c**) using K-NN. The error indicates the proportion of incorrectly grouped instances.

Classification: Distances						
	Histogram			**Kernel Density**		
	t	**r**	**c**	**t**	**r**	**c**
t	100	0	0	100	0	0
r	0	100	0	0	100	0
c	0	0	100	0	0	100
Err	0.00 ± 0.00			0.00 ± 0.00		

Table 3.6: 10-fold cross-validation estimates of the confusion matrices for simple figures classification with the distances representation. The instances, which are characterized by either the histogram or a kernel estimate of the probability density function of the normal vectors along the contour of the figure, are categorized as *triangular* (**t**), *rectangular* (**r**), and *circular* (**c**) using K-NN. The error indicates the proportion of incorrectly grouped instances.

Best results are achieved when the kernel density representation is used, in fact, in all cases for all metrics the classification is perfect. The histogram representation offers weaker results due to variability. These results clarify the importance of the smoothing during the preprocessing of the inputs. See Annex A for a deeper exploration of these distance representations of simple figures.

4

Real world application: Otolith Classification

In this section, a real world problem is faced to test the usefulness of the methodology: the problem of classification of fish otoliths. In Section 4.1, the otolith data set will be described and in Section 4.2 also the preprocessing performed to each instance. Metrics and distance characterizations defined in Section 3.3 will be employed to encode the fish otoliths available in Section 4.2. Finally, in Section 4.3 various experiments of clustering and classification are performed, to measure the accuracy of the characterizations.

4.1 Otolith Data Set

In this section the categorization of otoliths for grouping and identification of fish species is considered. Otoliths are concretions of calcium carbonate and other inorganic salts that are formed by aggregation on a protein matrix in the inner ear of vertebrates [27]. The data set studied consists of 240 high-contrast images of otoliths for three different families of fish: *labridae* (125 images), *soleidae* (70 images), and *scombridae* (45 images). The images are centered and oriented so the to the frontal part of the otolith appears to the right of the image. This set has been retrieved from the AFORO database (http://www.icm.csic.es/aforo/), which is a an extensive open online repository of data for different fish species. Otoliths in *labridae* family are cuneiform, oval, bullet-shaped, or rectangular. They present a cleavage in the frontal zone, which, in general, is more prominent than in the other fish families. *Soleidae* otoliths are mainly discoidal and elliptic. Their shapes are in general more regular and smooth than in other two families. Finally, otoliths of the *scombridae* family have serrate contours and generally are more elongated [28]. Examples of these otoliths are displayed in Figure 4.1. *Labridae* and *soleidae* otolithds present higher shape variability than *scombridae* otoliths, which are typically more regular.

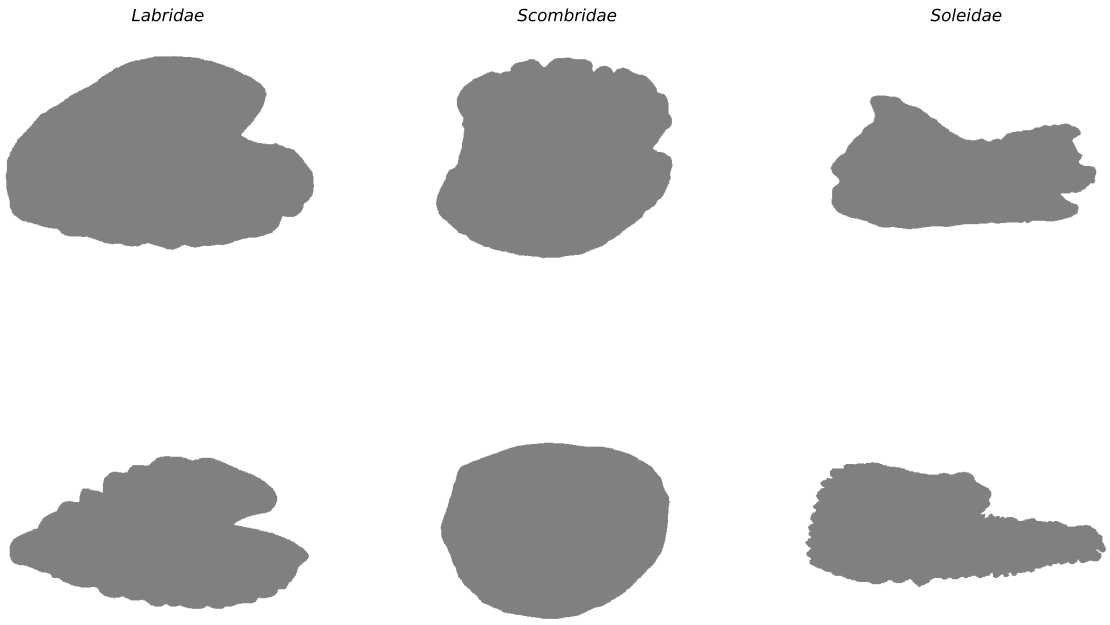


Figure 4.1: Examples of *labridae*, *soleidae*, and *scombridae* otoliths. The frontal part of the otolith, which corresponds to the head of the fish, appears on the right of the image.

4.2 Otolith Data Preprocessing

In each of the images, the contour of the otolith is retrieved using the *marching squares* algorithm [29]. The contour is rectified so that all figures have the same number of vertices. In the experiments, a maximum of 64 vertices is considered. This quantization of the contour reduces the variability in the representation and allows to preserve a sufficient amount of detail for an accurate characterization of the shape of the object [10]. An illustration of this preprocessing step is given in figure 4.2.

The figure is then characterized by sampling a total of $N_{sample} = 1000$ points at regularly spaced intervals along the contour. For each of these sampling points the directional variable is computed. As discussed earlier, this variable is the direction of the normal vector at the point considered. For planar figures, it can be represented as the angle that specifies the direction of the normal vector. The probability density of these direction values is then approximated using either a scaled histogram with 64 bins or a KDE estimate that utilizes Von Mises kernels of width $h = \frac{2\pi}{64}$. In both cases, the probability density estimates are discretized at $N = 64$ points located at the center of the histogram bins.

From the probability density estimates, two different characterizations will be used. In a first characterization, the figures are aligned at the maximum of the corresponding density estimates. The vector of attributes in this *aligned* representation consists in the $N = 64$ values of the corresponding histogram or KDE. A second characterization (labeled *distances*) is by a vector composed of the 6 distances between the corresponding density estimation and the uniform distribution, according to Eq. (3.4). See Annex B for a deeper exploration of this distance

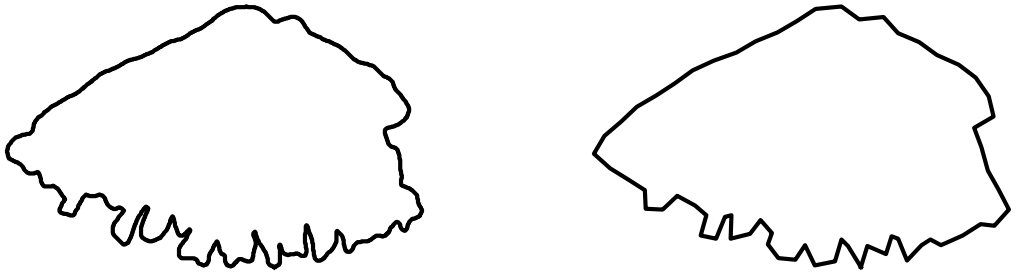


Figure 4.2: Visualization of the smoothing process applied to the contour of a otolith by reducing the number of vertices to 64.

representations for the otolith data set.

4.3 Otolith Clustering and Classification

Both unsupervised (clustering) and supervised (classification) learning tasks are considered.

4.3.1 Otolith Clustering

In this section the K-means algorithm is used to group the otoliths into $K = 3$ clusters. Two alternatives will be tested, one where the aligned orientation data will be used as attributes and a second one where the distances defined in Section 3.3 are employed.

Otolith Clustering with Aligned Data

In this first alternative, the data used as input for the algorithm are the 64 different values of the histogram and the kernel density discrete versions of the figure. In Table 4.1 the results of the clustering process is shown.

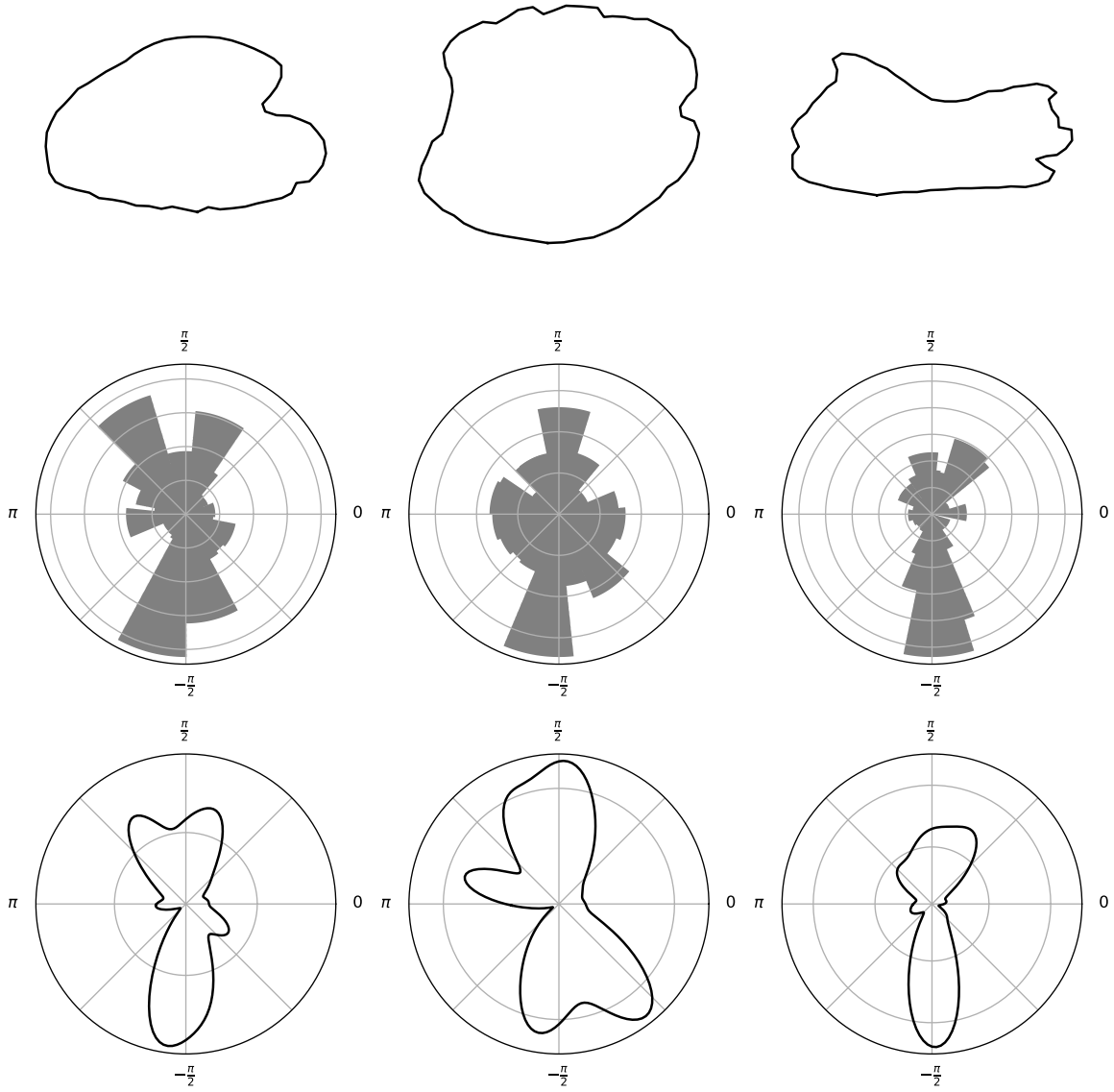


Figure 4.3: In the top row of this figure the contours of otoliths of different fish families are shown: *labridae* (left), *soleidae* (middle), and *scombridae* (right). The histogram and kernel density estimates of the direction variables for these figures are displayed in the middle and bottom rows, respectively.

Otolith Clustering with distances

In this second alternative, the input for the clustering process are different distances defined in Section 3.3. The results of this unsupervised learning task are displayed in Table 4.2.

Several conclusions can be drawn from these results. The first one is that the clusters identified when the characterization based on distances to the uniform circular distribution are rather impure. The results are significantly better when the representation based on alignment is used, especially when the kernel density estimates (KDE) are employed. The reason why alignment is useful in these data is because otoliths typically have an oblong shape, with clearly defined axis. As illustrated by results of the experiments with simple geometrical shapes (Section 3.4), if the data do not exhibit a clear direction, alignment can actually have a detrimental effect.

Otolith Clustering: Aligned						
	Histogram			Kernel Density		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
lab	113	7	5	113	6	6
sol	3	67	0	0	70	0
sco	4	0	41	3	0	42
Err	0.08			0.06		

Table 4.1: Number of otoliths from the *labridae* (lab), *soleidae* (sol), and *scombridae* (sco) assigned to each of the 3 clusters identified using K-means. The results on the left-hand side correspond to the 6-dimensional *distances* representations. The results obtained when the 64-dimensional *aligned* representations are used are displayed on the right-hand side. The error indicates the proportion of incorrectly grouped instances.

Otolith Clustering: Distances						
	Histogram			Kernel Density		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**
lab	86	20	19	93	11	21
sol	7	63	0	4	66	0
sco	6	0	39	4	0	41
Err	0.22			0.17		

Table 4.2: Number of otoliths from the *labridae* (lab), *soleidae* (sol), and *scombridae* (sco) assigned to each of the 3 clusters identified using K-means. The results on the left-hand side correspond to the 6-dimensional *distances* representations. The results obtained when the 64-dimensional *aligned* representations are used are displayed on the right-hand side. The error indicates the proportion of incorrectly grouped instances.

4.3.2 Otolith Classification

In this section, is explained the k-nearest neighbors (K-NN) used to predict the shape of the figure based on various characterizations proposed. The number of neighbors is determined using 3-fold cross-validation within the training data. The range of values explored is $k = 3, 5, \ldots, 13$. In most cases the values selected are either $k = 3$ or $k = 5$. The results are not particularly sensitive to the choice of this parameter. The generalization error is estimated using 10-fold cross-validation over the complete data set.

Otolith Classification with Periodic Distances

In this first alternative, the attributes to input are the discrete versions of the histogram and the kernel density estimations. The number of points in each representation is 64. Several models will be tested, each of them using one of the periodic metrics defined in Section 3.3. In Table 4.3 the results of the generalization error estimation is displayed.

Otolith Classification: Periodic Distances														
		Histogram			**KDE**				**Histogram**			**KDE**		
		lab	**sol**	**sco**	**lab**	**sol**	**sco**		**lab**	**sol**	**sco**	**lab**	**sol**	**sco**
lab	L_1	119	0	6	123	0	2	EMD	120	1	4	124	0	1
sol		2	68	0	0	70	0		3	67	0	0	70	0
sco		8	0	37	5	0	40		7	0	38	4	0	41
Err		0.05 ± 0.04			0.05 ± 0.04				0.11 ± 0.06			0.10 ± 0.06		
lab	L_2	121	0	4	124	0	1	H	119	0	6	123	1	1
sol		3	67	0	0	70	0		2	68	0	0	70	0
sco		9	0	36	5	0	40		7	0	38	5	0	40
Err		0.05 ± 0.04			0.04 ± 0.04				0.06 ± 0.05			0.06 ± 0.05		
lab	L_∞	119	0	6	123	0	2	χ^2	118	1	6	121	2	2
sol		2	68	0	0	70	0		2	68	0	0	70	0
sco		6	0	39	5	0	40		7	0	38	6	0	39
Err		0.05 ± 0.04			0.05 ± 0.04				0.06 ± 0.04			0.06 ± 0.04		

Table 4.3: 10-fold cross-validation estimates of the confusion matrices for otolith classification. The instances, which are characterized by either the histogram or a kernel estimate (KDE) of the probability density function of the normal vectors along the contour of the figure, are categorized as *labridae* (**lab**), *soleidae* (**sol**), and *scombridae* (**sco**) using K-NN. The error indicates the proportion of incorrectly grouped instances.

Otolith Classification with Aligning Approximation

In this second alternative, the attributes to input are again the discrete versions of the histogram and the kernel density estimations, but models will employ the aligned approximation to the metrics defined in Section 3.3. In Table 4.4 the results of the generalization error estimation is displayed.

Otolith Classification with Distances Approximation

In this last experiment, the figures are characterized by the distances defined in Section 3.3. Confusion matrices for each representation are displayed in Table 4.5.

A final set of experiments is carried out using the rotationally invariant distance between shapes given by Eq. (3.3) in the nearest-neighbors algorithm. The distances are computed using one of the six different metrics described in Section 3.3. The results of these experiments are summarized in Table 4.3. As in the previous set of experiments, the number of neighbors is estimated on the training data using 3-fold cross-validation. The generalization error is estimated using 10-fold cross-validation on the complete data set. In all cases, the predictions are very accurate for all metrics, specially when the smoother kernel density estimation is used. However, because of the minimization step required in computing the distances Eq. (3.3), the computational cost of this algorithm is high. As in the previous experiments, the *scombridae* and *soleidae* otoliths are well separated. For some of the metrics the *soleidae* otoliths, which tend to present smoother shapes than the other classes, are correctly identified in all the cases considered.

As in the clustering experiment, the accuracy is better when the aligned histograms are used to characterize the instances. The best overall accuracy is obtained when kernel density

Otolith Classification: Aligned Approximation													
		Histogram			**KDE**				**Histogram**			**KDE**	
		lab	**sol**	**sco**	**lab**	**sol**	**sco**		**lab**	**sol**	**sco**	**lab**	**sol**	**sco**
lab	L_1	116	7	2	121	4	0	EMD	113	10	2	122	3	0
sol		7	62	1	2	68	0		9	61	0	3	67	0
sco		7	0	38	8	0	37		6	0	39	8	0	37
Err		0.10 ± 0.04			0.06 ± 0.05				0.11 ± 0.06			0.06 ± 0.03		
lab	L_2	117	6	2	122	3	0	H	116	6	3	120	5	0
sol		10	59	1	4	66	0		10	60	0	2	68	0
sco		10	0	35	7	0	38		9	0	36	7	0	38
Err		0.12 ± 0.06			0.06 ± 0.04				0.12 ± 0.06			0.06 ± 0.07		
lab	L_∞	116	7	2	121	3	1	χ^2	114	8	3	120	5	0
sol		8	62	0	3	67	0		10	59	1	3	67	0
sco		11	0	34	7	0	38		10	0	35	8	0	37
Err		0.12 ± 0.05			0.06 ± 0.05				0.13 ± 0.06			0.07 ± 0.04		

Table 4.4: 10-fold cross-validation estimates of the confusion matrices for otolith classification. The instances, which are characterized by either the histogram or a kernel estimate (KDE) of the probability density function of the normal vectors along the contour of the figure, are categorized as *labridae* (**lab**), *soleidae* (**sol**), and *scombridae* (**sco**) using K-NN. The error indicates the proportion of incorrectly grouped instances.

Otolith Classification: Distances						
	Histogram			**KDE**		
	lab	**sol**	**sco**	**lab**	**sol**	**sco**
lab	98	5	22	110	7	8
sol	18	52	0	10	60	0
sco	11	0	34	15	0	30
Err	0.27 ± 0.06			0.15 ± 0.06		

Table 4.5: Confusion matrices and 10-fold cross-validation errors for the classification of *labridae* (lab), *soleidae* (sol), and *scombridae* (sco) otoliths using a characterization based on distances to the uniform circular distribution.

estimates are used, also when instances are characterized by the distances of the corresponding density estimates to the uniform circular distribution.

It is apparent from Figure 4.3, that the shapes of *soleidae* and *scombridae* otoliths are markedly different from each other. As a result, the discrimination between items from these classes is fairly easy. If fact, they are well separated in all the representations considered. This is not the case for the *labridae* otoliths. Some otoliths from this class present elongated shapes, which are more characteristic of *scombridae*. Others present a circular, more regular shapes, and can be mistaken for instances from the *soleidae* class.

5

Project Development

In this next Chapter the complete project and its development process will be detailed. Main objectives and features as well as scope of the project will be first depicted in Sections 5.1 while in Section 5.2, a complete set of functional and non-functional requirements captured for this system to implement during the development will be listed. In Section 5.3 the main software tool implemented and used in the project *MLS* will be briefly described. Finally, in Section 5.4 the development process for the complete project is pictured.

5.1 Project objectives

The main objective of the project is to provide a directional high-level characterization method for figures as an alternative to other state of art characterization techniques. One important objective that will prove the usefulness of the methodology is its application to shape matching problems with positive results. In particular, results in the real world problem of fish otoliths shape classification and matching could offer relevant information to compare with different techniques applied in the same problem in the past [2, 7]. To perform all simulations and experiments needed for this objectives, a software tool has been developed, and will be described in Section 5.3. Among the project objectives, the quality of both this software tool and the process of its development are included.

The software tool developed will need functionality to load a data set of images and apply the methods described in Chapter 3 to obtain kernel density and histogram representations of these images, and the rest of computations defined in Sections 3.2 and 3.3 to obtain the data set of different representations proposed. The system should allow tunning hyperparameters such as sampling size, bin anchor or kernel bandwidth, among others specified in Sections 3.2 and 3.3 to adjust these representation acquirement process. The system should also allow storing these computations and perform the experiments defined in Section 3.4 to the data set obtained, displaying relevant information about these results. The scope of the project is restricted to 2d figures, and 3D figures methods will not be treated.

To sum up, the system described will provide the tools to run the simulations required to evaluate the results of the proposal described in this project, while also allow performing the experiments required to develop the methodology described.

5.2 Requirements

In this section, requirements captured for the project are explained. In Sections 5.2.1 and 5.2.2 functional and non-functional requirements are listed. These sets of requirements have been updated during the entire project to add or correct specifications.

5.2.1 Functional Requirements

Next list includes all functional requirements captured for the *MLS* system.

FR-1. Loading a figure from an image.

The system should be able to load an image and compute a contour from it. This contour will be stored as a list of coordinates, each element corresponding to a point in this contour. The system should allow applying a reduction on the number of vertices of the contour, so contours are computed to obtain a result with a fixed number of vertices.

Input: The path to the image and optionally a given number of vertices. The image can be in `png`,`jpg`, `jpeg` or in `tif` format.
Process: The system stores the contour obtained with the bigger number of vertices, among the contours detected in the image. The contour is processed so the final results has the number of vertices specified.
Output: The system stores the contour as a figure, using as id for the figure its file name.

FR-2. Generating a figure from a deformation of a regular template.

The system should allow creating a figure from a deformation of a regular template.

Input: The number of vertices, the radius of the regular template and a noise value.
Process: The system generates a random deformation, from a regular polygon with the given number of vertices. To apply this deformation, the angles of the regular template are expressed in their polar coordinates and the angle and radius of each vertex is randomly altered.
Output: The system stores the resulting set of vertices as a figure.

FR-3. Loading a complete data set of figures from a data set of images.

The system should allow loading a data set of images, to obtain a data set of figures. For each image, the process described in **FR-1.** will be applied. The resulting set of figures should be stored as a data set of figures.

Input: A list with the folders where each population of images is stored and a given number of vertices for the contour reduction computation.
Process: The system tries to obtain sequentially the figures as in **FR-1** for each file in the path provided. The process will fail if it fails at processing at least one of the files.
Output: A data set with the loaded images is stored as a data set of figures.

FR-4. Generating a complete dataset of figures from regular templates.

The system should allow generating a data set of figures from deformation of simple templates. For each template given, the process described in **FR-2.** should be applied to generate a population of deformations. The resulting set of figures should be stored as a data set of figures.

Input: The process takes as input a population size,a list of number of vertices and a radius and noise parameters.
Process: Ths system generates one population per number of vertices provided in the list, whit a number of instances in each population equal to the population size provided. These figures are generated each of them as in **FR-2**, all of them in the data set with the equal noise parameter and radius. Each figure generated is labeled with a unique identifier inside the data set.
Output: The system stores the generated data set of figures.

FR-6. Obtaining the directional variables of a data set of figures.

The system should be able to compute a data set of samples of the directional variables for a given data set of figures.

Input: This process will take as input a data set of figures, and a sample size, that will be the size of the sample to obtain for each figure in the data set.
Process: First a set of points in the contour of the images are selected, where the number of points in the set is equal to this sample size given. Then, for each of these points, the angles of the normal unitary vectors to the contour of the figures are computed and stored.
Output: A data set with one sample for the directional variable of each figure in the data set provided is stored.

FR-7. Computing histogram characterizations of a data set of directional samples.

The system should be able to compute normalized histogram characterizations from a given data set of directional samples.

Input: A number of bins for the histogram representation and a data set of directional samples must be provided.
Process: From each of the directional samples in the data set provided, the system computes a normalized histogram.
Output: The computed normalized histograms are stored as a data set.

FR-8. Computing kernel density characterizations of a data set of directional samples.

The system should be able to compute kernel density characterizations from a given data set of directional samples.

Input: A number of points for the discrete representation of the kernel density estimation to compute and a data set of directional samples must be provided. Also, a kernel anchor can be provided for the process, or this kernel anchor can be automatically selected following the "rule of thumb" for circular kernel density estimation [7].
Process: From each of the directional samples in the data set provided, the system computes a kernel density estimation, following the methodology explained in Section 3.2.
Output: Resulting kernel densities estimations are stored as a data set.

FR-9. Computing distances characterizations

The system should be able to compute distance characterizations from a given data set of histogram or kernel density representations, and a given list of metrics to employ.

Input: The process will need as input a data set of discrete versions of histogram or kernel density representations and a list of metrics to employ.
Process: For each representation in the data set provided, the distances between the representation and a uniform distribution is computed using different metrics provided. For each representation, the result is a list of distances, one per metric provided, that will be stored as the distance characterization of the discrete representation.
Output: Ths system stores the data set of distances characterizations computed.

FR-10. Perform classification of shapes with directional characterizations.

The system should be able to perform classification with the characterizations described in Chapter 3, using a K-NN algorithm. This will include performing cross validation to obtain an estimation of the generalization error and cross validation during the train phase, in order to tune the K parameter of the algorithm.

Input: It should be provided a data set of representations (normalized histogram, kernel density or distance representations) with the corresponding data set of labels, a metric to employ in the K-NN algorithm, a number of folds for the generalization error, a number of folds for K hyperparameter selection and a feasible list of orders of neighborhoods.
Process: The system will perform cross validation with the given number of folds to estimate the generalization error of a K-NN algorithm using the metric provided and the data provided as attributes. During each train phase, cross validation with the specified number of folds will be performed to select the K hyperparameter that obtained best results. The folds are stratified in the process.
Output: The system will display information of the data set (data dimensions), the cross validation processes (fold count) and the results (execution time, error estimated and confusion matrix)

FR-11. Perform clustering of shapes with directional characterizations.

The system should be able to perform clustering with aligned or distances representations, using a K-Means algorithm.

Input: It should be provided a data set of representations (normalized histogram, kernel density or distance representations) with the corresponding data set of labels, and a number of clusters to compute.
Process: The system executes the K-NN algorithm for data provided.
Output: The information of the execution time, the instances distribution among computed clusters and a estimation of a error measure for the result is displayed.

5.2.2 Non-functional Requirements

In this list other non-functional requirements defined for the system are displayed.

Performance

The system does not have strong constraints in this performance aspect. Even though some computations such as performing classification with periodic distances are complex, there are not real time constraints and data dimensions are small.

Scalability

This requirement is crucial, since there are already several future work proposals, and several techniques to be added to the methodology, and therefore potential functionalities to be added to the system. The system should be clear and well documented, to ease applying changes and corrections, since future implementations could recycle functionality from this system.

Documentation and Software Quality

The code must be well documented. Each class and method must be well described in the documentation with each attribute or parameter. Closely related with the scalability requirement, it is crucial that functionalities are clear and well documented, since there are some complex concepts and processes are developed. Also, the code must follow naming conventions to ease readability adn usability.

5.3 System characteristics

The *MLS* system is compounded of multiple tools to provide the material needed during the investigation. The requirements and the design of *MLS* have been redefined during the development, adding or correcting functionality when needed. The final design, described in this Chapter, includes functionality to extract contours from labeled images and store these contours

as a polygonal figure. Also, includes functionality to compute from these figures histograms and kernel density estimations of their directional variable (See section 3). Finally *MLS* includes the functionality needed to perform from these representations the validation methods defined in Section 3.4 of clustering and classification.

The *MLS* system is composed of multiple Python modules that provide the functionality required. The implementation is written in Python version 3.6, and several Jupyter notebooks have been developed to perform the unitary validation test of functionalities and the validation of the methodology proposed in this work. This has dependecies on `numpy`, `scipy` and `sklearn` Python libraries.

This system allows processing a data set of images to obtain a data set of figures form their contours. No preprocessing is applied to these images, since it was not needed in the problem applied (See Section 4.2). Also, a data set of figures can be generated from a set of regular templates, with the application of small deformations. These data set of figures are stored as a list of vectors with the coordinates of each point obtained in the contour.

From these data sets of figures, the system *MLS* can be used to generate data sets of orientation variables, where each orientation variable corresponds to a unique figure. These orientation variables will mainly be characterized by three attributes: the directional variable sample, the discrete representations of the normalized histogram and the kernel density estimations. The directional sample, from which these discrete representations are computed, is generated sampling the directions of the normal vectors across the contour of these figures. The normalized histogram and kernel density estimations are obtained applying directional statistics tools [17, 17].

The metrics defined in Section 3.3 are implemented in the system *MLS*. Both periodic distances and each aligned approximation can be used to quantitatively compare discrete representations of figures. These metrics can also be used as metrics in the classification models used in the project. Mention that the complexity of these periodic distances is quadratic, while the complexity of the aligned approximations is linear.

The system also contains functionality required to perform clustering and classification experiments as defined in Sections 3.4 and 4.3, using the implementations of the `sklearn` library. Used classes are in particular `sklearn.cluster.KMeans` and `sklearn.neighbors.KNeighborsClassifier` as implementations of each algorithm.

In Figure 5.1 a class diagram for the design of the project is displayed, and in Figure 5.2 a diagram for the python packages compounding the project is depicted.

5.4 Project Management

In this section the development process of the project is presented, with the description of the methodology and the project plan performed. In Subsection 5.4.1 this project plan followed will be described exhaustively. These subsections will focus on the development of the software tool *MLS*, while also the process to generate all documentation will be described.

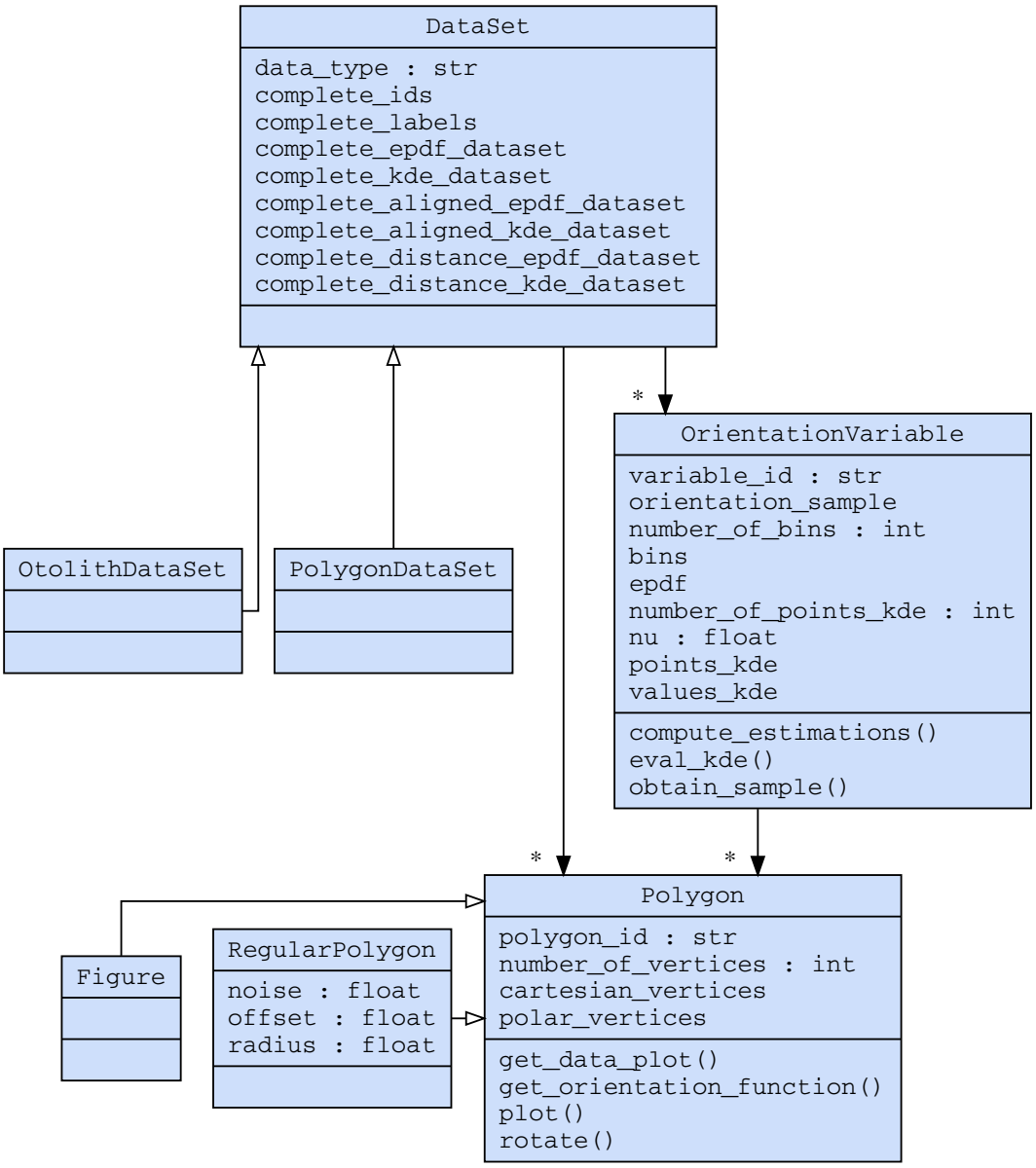


Figure 5.1: Class diagram for *MLS* design

Initially the scope of the project was undefined, so the methodology used during the development it has been agile, with increments of one week usually. At the end of each increment, a tutorial with the tutor took place to evaluate progress performed, and increments for next increment were defined. During the development, it has been necessary to parallel research and development, usually being necessary to perform corrections after the increment review. To maintain a robust version control, the software tool `git` was used during the develop to create and maintain a git repository to keep track of code and documentation. Time devoted to this project is 3 hours per week in average, so about a total of 120 hours have been invested in the development of the project.

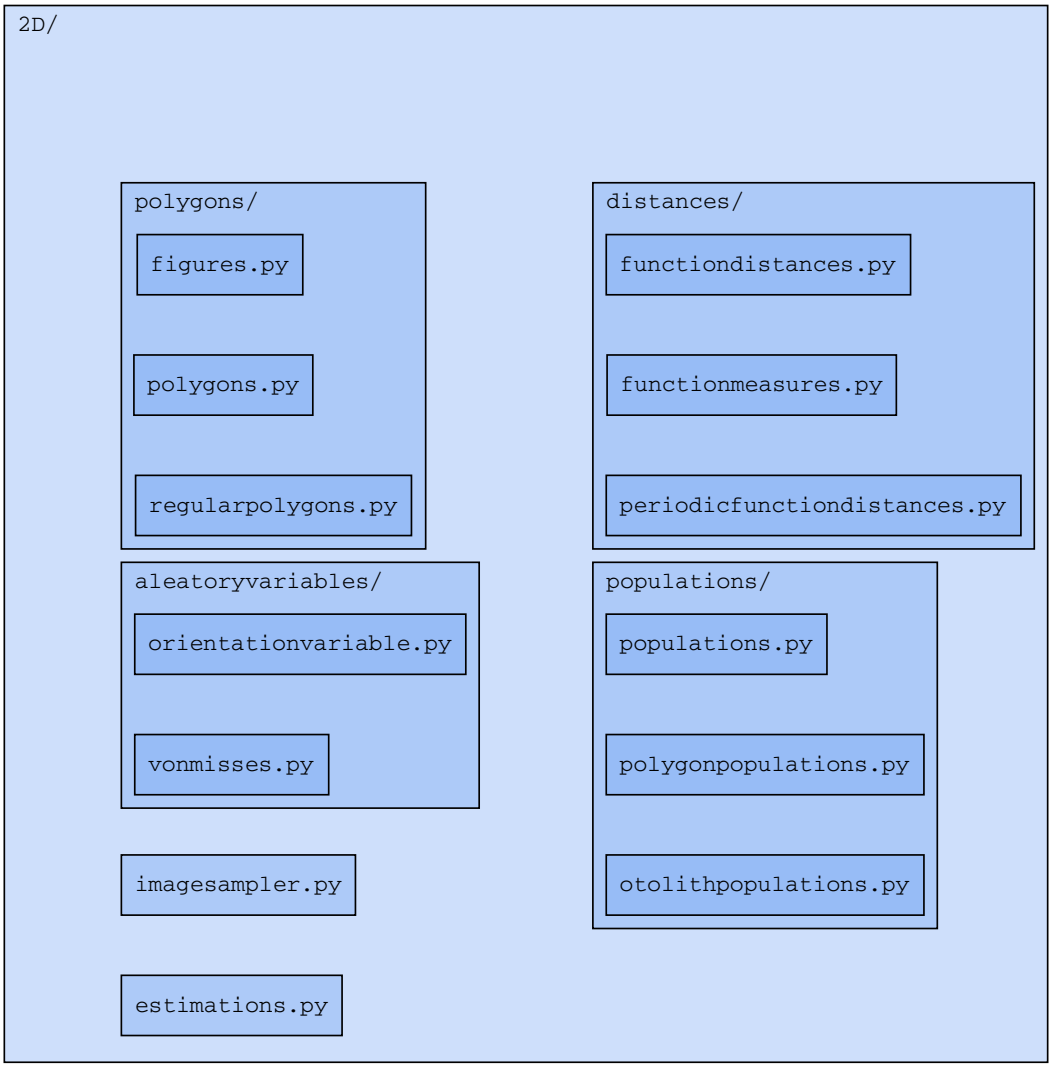


Figure 5.2: Package diagram for *MLS* implementation

5.4.1 Project Plan

During the development four phases can be identified: during first three parts the main software tool was developed, and in final part all the methodology documentation, an article describing the proposal and this document were written. This three first phases can be matched with each one of the problems studied, where different increments were necessary to complete and different solutions were achieved. Next list summarizes different solutions and problems, grouping close related milestones by increments. In Figure 5.3, a Gantt diagram for the development of the software system *MLS* is displayed.

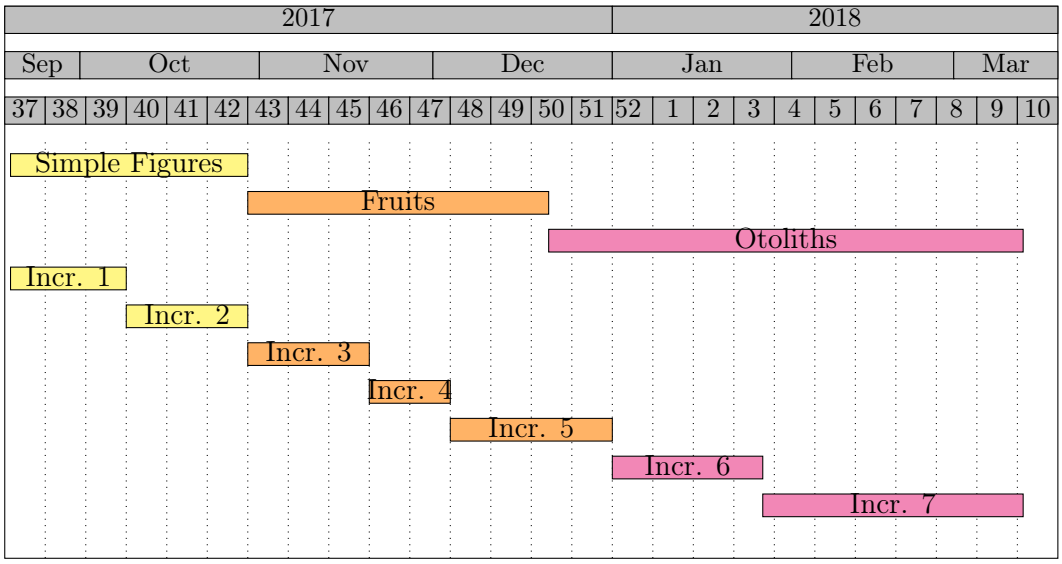


Figure 5.3: Gantt diagram for the software project *MLS* development. Increments are coloured as the studied problem their compounding tasks are motivated by

Increment 1: Obtaining histogram representations of simple figures

In this first increment, the functionality needed to generate a set of polygons from deformations of regular templates was implemented. Also, was developed the functionality to compute histogram representations of the directional variables of these figures. Initially the directional variable sample would be obtained computing the direction of points sampled taking uniformly distributed directions from a central point and checking the collision with the contour of the figure. The problem in the definition of a robust landmark for this central point, and problems in this strategy with non convex figures fostered the redefinition of this sampling process.

Increment 2: Classification with simple figures

During this increment, functionality to transform these sets of histogram representations into attributes usable for classification was developed, and the functionality required to test these representations in simple classification models (K-NN, Random Forest and Gaussian Naive-Bayes). K-NN will be the only model used in classification experiments after this phase.

Increment 3: Obtaining histogram representations of fruits

A real world problem is faced during this increment of fruit classification. Images of apples, bananas and pears were retrieved to test the methodology. As a result, functionality to load a contour from an image was develop using the `numpy` implementation of matching squares algorithm [29]. Also, preprocessing and cleaning these fruit images was mandatory to obtain representative contours. Functionality was develop to achieve this goal, but the images could not be processed in a satisfactory way, due to lack of homogeneity in the images. Therefore, many instances where not usable in the methodology and the data set did not have enough instances to obtain relevant results.

Increment 4: Clustering and Classification with fruit images

In this increment, the processed images of fruits were employed to test the methodology, using them in some experiments of classification with several models. In this phase functionality adde covered new features such as the aligned histogram representation, as well as the k-Means clustering algorithm experiment described in Section 3.4. Results obtained from this phase are weak, due to the quality of the data set obtained.

Increment 5: Smoothing: Vertices reduction, splines, kernel density representations

Several techniques to pre-process data were developed during this phase. First, while the fruit problem was still being explored, two techniques were proposed to be tested: using cubic splines to interpolate of the contour of figures [30] and computing kernel density representation of the directional variable [31]. The lack of a easy fitting implementation in Python, and the good results obtained while testing kernel density representations in the simple figure problem caused the development of this splines process to be postponed as future work. While exploring the fish otolith images data set, functionality was develop to uniform the number of vertices of the contours retrieved from images [29].

Increment 6: Obtaining histogram and kernel density representations of fish otoliths

During this phase the methodology is adapted and tested in the real world application of otolith classification. The available functionality is adopted in order to load the data set of fish otolith images. Functionality to automatically select the bandwidth of kernel density estimators is added [2, 16, 25, 7].

Increment 7: Clustering and classification of fish otoliths

The real world problem of fish otolith shape classification is faced in this final phase of development. Functionality to perform cross validation for generalization error estimation and for order of neighborhood selection during train phase is added to the system.

6

Conclussions and Future Work

In this work a characterization of shapes of objects based on directional data is proposed. The ultimate goal is to categorize these objects according to their shapes. Intuitively, shape can be defined as the geometrical property shared by different objects that is invariant to a loosely-defined family of transformations, which includes translations, scaling, rotations, and small deformations. In the method proposed, shape is encoded by the distribution of unit vectors along the normal direction at the boundaries of the object. In this manner, information on scale and absolute distances is eliminated while preserving directional information, which is expected to encode shape. This representation is obtained by first locating these boundaries with standard image processing algorithms [29]. Then, normal unit vectors are computed at a set of points located on this boundary. These vectors can be seen as realizations of a directional random variable. This random variable can be characterized by its distribution, and in two dimensions, the boundary is a curve, so the distribution of normal vectors is defined on the circle. Empirical estimates of the probability density function can be computed using a scaled histogram, or a smoother kernel density estimation using, for instance, Von Mises kernels [19, 17, 22]. In this work both options are explored. For three-dimensional representations, the boundary of the object is a surface. The distribution of three-dimensional unit normal vectors are defined on a sphere. In this case Von Mises-Fisher kernels can be used [24]. Since the representation is functional (and therefore, infinite dimensional), further reductions of information are needed so that it can be used in practice. In particular, for two-dimensional data, representations are discretized at regularly $N = 64$ spaced points along the circle. Since shape should be invariant with respect to rotations, computationally expensive shape-alignment operations are needed when this representations are used for clustering or classification, but low complexity operations can be used as approximations to these expensive alignments with the with an acceptable error. A further dimensionality reduction, which is rotationally invariant, can be made using as features the distances of the discretized densities and the uniform circular distribution. The usefulness of these representations for clustering and classification of images of objects according to their shapes is illustrated with otolith data. From these experiments it can be concluded that smoothing is crucial when the features considered are distances to the uniform circular distribution. This representation is computationally efficient and is reasonably accurate. Techniques based on alignment, while being computationally costly, are very accurate and provide the best overall results, especially when kernel estimates of the density of normal unit vectors are used. This empirical investigation illustrates the effectiveness of the directional data representation

proposed to encode shapes.

As a product of this project, a software tool *MLS* has been developed to perform experiments and test the approach. This software tool can be used to generate and store histogram and kernel density representations of figures, allowing the configuration of hyperparameters so the process can be adapted to each data set, while also computing distance characterizations of figures 3.3. This tool also includes functionality to perform clustering and classification of figures using the methodology described in Chapter 3.

Some procedures were proposed during the development and remain as future work proposal. Most of them are related with the data preprocessing, since there exist many alternatives to apply in different phases of the process. For instance, a more sophisticated process to smooth the contour of figures, with periodic cubic spline interpolation [32], or different techniques to realize the directional variables, applying smoothing processes when sampling the contour of the figures, such as a simple moving average strategy. Also, in the future is intended to test the methodology in the MNIST database (http://yann.lecun.com/exdb/mnist/), a set of 70.000 instances of images of handwritten digits. This database has a well defined format and needs minimum effort to preprocess. Finally, different models such as a vote based classifier, combining different metrics and distances defined in Section 3.3, or a clustering algorithm applying the same metrics are proposed to explore in the future.

Glossary

- **Fish Otolith:**: Calcium carbonate structure in the the inner ear of fishes sensitive to accelerations.
- **Concentration parameter:**: Measure of dispersion analogous to the variance for circular aleatory variables.
- **Directional Variable:**: Aleatory variable identified by the directions of the normal unit vectors to the tangent hyperplanes at the boundary of a figure.
- **Histogram representation**: Representation of a figure in which the figure is encoded in a discrete version of a normalized histogram as a estimation of the probability distribution of the directional variable.
- **Kernel density representation**: Representation of a figure in which the figure is encoded in a discrete version of a kernel density estimation of the probability distribution of the directional variable.
- **Aligned representation**: Representation of a figure in which the figure is encoded with a discrete representation, and this representation is rotated according to a landmark.
- **Distance representation**: Representation of a figure in which the figure is encoded with the distance from its histogram representation or kernel density representation to a uniform circular distribution.
- **EPD**: Empirical Probability Distribution. Estimation of the probability density function of an aleatory variable obtained from a normalized histogram.
- **KDE**: Kernel Density Estimator. Non-parametric method to estimate the probability density function of a random variable.
- **K-NN**: K Nearest Neighbors. Non-parametric method used for classification and regression.
- **K-Means**: Non-parametric method used for clustering and data mining.
- **MLS**: Software project developed to test the methodology proposed.
- **Jupyter notebook**: Document produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc).
- **Jupyter notebook App**: server-client application that allows editing and running notebook documents via a web browser
- **von Mises distribution**: Continuous probability distribution on the circle, that is a close approximation to the wrapped normal distribution, which is the circular analogue of the normal distribution.

Bibliography

[1] Ji-Xiang Du, Xiao-Feng Wang, and Guo-Jun Zhang. Leaf shape based plant species recognition. *Applied Mathematics and Computation*, 185(2):883 – 893, 2007. Special Issue on Intelligent Computing Theory and Methodology.

[2] José R. Berrendero, Antonio Cuevas, and Beatriz Pateiro-López. Shape classification based on interpoint distance distributions. *Journal of Multivariate Analysis*, 146:237 – 247, 2016. Special Issue on Statistical Models and Methods for High or Infinite Dimensional Spaces.

[3] P. Montero-Manso and J.A. Vilar. Shape classification through functional data reparametrization and distribution-based comparison. *IWFOS: International Workshop on Functional and Operatorial Statistics A Coruña (Spain) - Fundación Barrié*, June 2017.

[4] Paul Yushkevich, Stephen M. Pizer, Sarang Joshi, and J.S. Marron. Intuitive, localized analysis of shape variability. In Michael F. Insana and Richard M. Leahy, editors, *Information Processing in Medical Imaging*, pages 402–408, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[5] Haonan Wang and J. S. Marron. Object oriented data analysis: Sets of trees. *Ann. Statist.*, 35(5):1849–1873, 2007.

[6] J. Ramsay and B. Silverman. *Functional Data Analysis*. Springer-Verlag, New York, 1997.

[7] Eduardo García-Portugués. Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electron. J. Statist.*, 7:1655–1685, 2013.

[8] Paul Besl and Ramesh Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17:75–145, 03 1985.

[9] Farshid Arman and J. K. Aggarwal. Model-based object recognition in dense-range images – a review. *ACM Comput. Surv.*, 25(1):5–43, March 1993.

[10] Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey. *Handbook of Computational Geometry*, 1996.

[11] Duane W. Storti, George Turkiyyah, Mark A. Ganter, Chek T. Lim, and Derek M. Stal. Skeleton-based modeling operations on solids. *SMA '97: Proceedings of the Fourth Symposium on Solid Modeling and Applications*, pages 141–154, 01 1997.

[12] Stephen M. Pizer, Andrew L. Thall, and David T. Chen. M-reps: A new object representation for graphics. *ACM Transactions on Graphics*, 2000.

[13] Thomas O. Binford. Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control (Miami, FL)*, 1971.

[14] Mark Moyou, Koffi Eddy Ihou, and Adrian M. Peter. LBO-Shape densities: A unified framework for 2d and 3d shape classification on the hypersphere of wavelet densities. *Computer Vision and Image Understanding*, 152:142 – 154, 2016.

[15] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, October 2002.

[16] Mairéad Grogan and Rozenn Dahyot. Shape registration with directional data. *Pattern Recognition*, 79:452 – 466, 2018.

[17] K. V. Mardia and P.E. Jupp. *Directional Statistics*. Wiley Series in Probability and Statistics. Wiley, New York, 2009.

[18] Isaías Salgado-Ugarte and Marco Aurelio Pérez-Hernández. Estimación de densidad por núcleo (kernel) para datos circulares, 06 2017.

[19] Marco Di Marzio, Agnese Panzera, and Charles Taylor. A note on density estimation for circular data. *Advanced Statistical Methods for the Analysis of Large Data-Sets*, 2012.

[20] Yogendra Chaubey. Smooth kernel estimation of a circular density function: A connection to orthogonal polynomials on the unit circle. *Journal of Probability and Statistics*, 2018, 2016.

[21] Daijin Ko. Robust estimation of the concentration parameter of the von mises-fisher distribution. *Ann. Statist.*, 20(2):917–928, 1992.

[22] K. V. Mardia. *Statistics of directional data*. London, Academic Press, 1972.

[23] David G. Kendall. A Survey of the Statistical theory of Shape. *Statist. Sci.*, 4(2):87–99, 1989.

[24] Ronald Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 217(1130):295–305, 1953.

[25] Charles C. Taylor. Automatic bandwidth selection for circular density estimation. *Computational Statistics & Data Analysis*, 52(7):3493 – 3500, 2008.

[26] Geert Schou. Estimation of the concentration parameter in von Mises-Fisher distributions. *Biometrika*, 65(2):369–377, 1978.

[27] J. Giménez, A. Manjabacas, V. M. Tuset, and A. Lombarte. Relationships between otolith and fish size from Mediterranean and North-eastern Atlantic species to be used in predator-prey studies. *Journal of Fish Biology*, 89(4):2195–2202, 2016.

[28] V.M. Tuset, A. Lombarte, and C.A. Assis. Otolith atlas for the Western Mediterranean, North and Central Eastern Atlantic. *Scientia Marina*, 72(SUPPL. 1):7–198, 2008.

[29] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[30] George Wolberg. Cubic spline interpolation: A review. *Columbia University Computer Science Technical Reports*, 1988.

[31] Bernard Silverman. *Density Estimation For Statistics And Data Analysis*, volume Vol. 26. Chapman & Hall, London - New York, 01 1986.

[32] Graham N. Y. Smoothing with periodic cubic splines. *Bell System Technical Journal*, 62(1):101–110, 1983.

A

Exploratory Analysis: Polygon Metrics

In this annex a brief study of the metrics selected 3.3 applied to the categorization of polygons problem is performed. A complete data set of histogram and kernel density representations are obtained from three basic templates of triangular, rectangular and circular shapes. In total 300 instances are generated, with noisy variations. In Figure 3.5 a caption of the data set is displayed. Different distances as defined in Section 3.3 are studied when applied to figures in the data set. In Figures A.1 and A.2 the results are shown for the histogram and the kernel density representation.

In these Figures is clear that labels can easily be differentiated using this characterization whit multiple distances. From the image it is clear that in most cases each cases labels lay in well distinguishable regions. In Section 3.4.2 using a K-NN model this feature is exploded. In Figure A.3 the relation between the evaluation of the distances in both characterizations, histogram and kernel density is displayed.

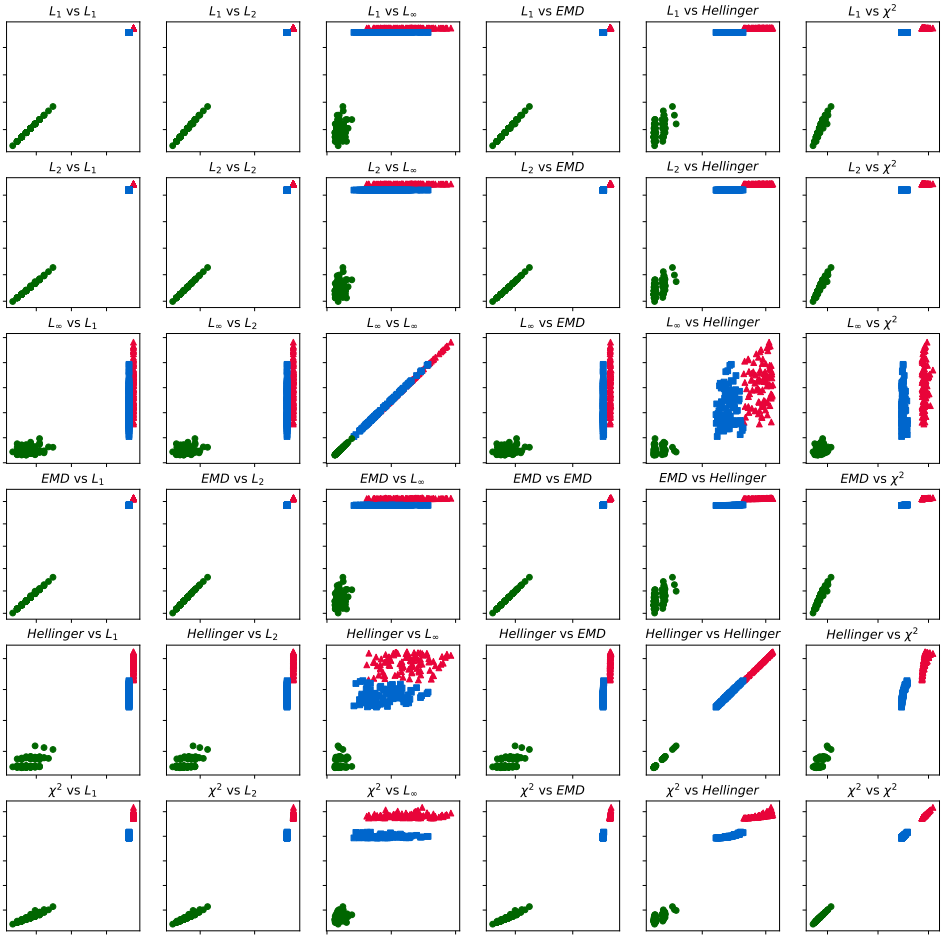


Figure A.1: Visualization of the correlation between different metrics applied to the histogram representation of the polygons data set. Shapes *circular*, *rectangular*, and *triangular* in green dots, blue squares and red triangles respectively.

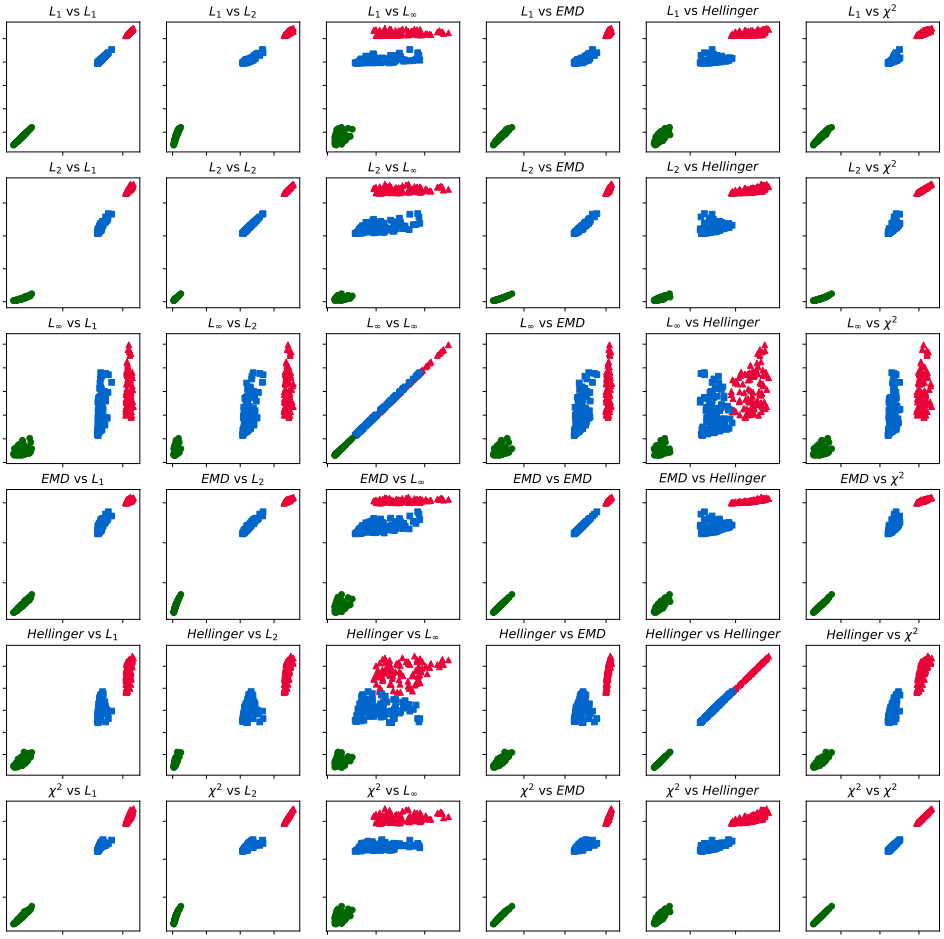


Figure A.2: Visualization of the correlation between different metrics applied to the kernel density representation of the polygons data set. Shapes *circular*, *rectangular*, and *triangular* in green dots, blue squares and red triangles respectively.

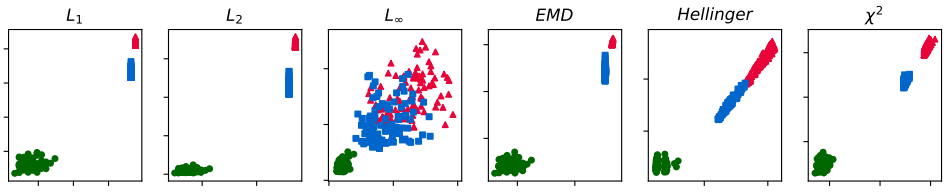


Figure A.3: Visualization of the correlation between different metrics when applied to the histogram or the kernel density representation of the polygons in the data set. Shapes *circular*, *rectangular*, and *triangular* in green dots, blue squares and red triangles respectively.

B

Exploratory Analysis: Otolith Metrics

In this annex a brief study of the metrics selected 3.3 applied to the otolith classification problem is performed. A complete data set of histogram and kernel density representations are obtained from the figures of the otoliths. Both representations will be encoded in 64 dimensional vectors, and obtained from a sample of 1000 points sampled across the contour of the otoliths. In Figure B.1 the correlation between distances when applied to the otolith data set is displayed. In Figure B.2 the same information is visualized, but when distances are applied to the kernel density representation.

In Figures B.1 and B.2, it can be observed that there seem to exist in most cases a region occupied mainly by a family. Also, note that *soleidae* family, the most regular family, obtains smaller values, while scombridae, the most elongated family, tends to obtain biggest values.

In Figures B.3 and B.4, the most irregular family, *labridae* is removed from the visualization. When this label is removed from the data sat, the problem eases significantly, turning into a linear separable problem for many representations.

In this last Figure B.5 the relation between two characterizations, histogram and kernel density is displayed.

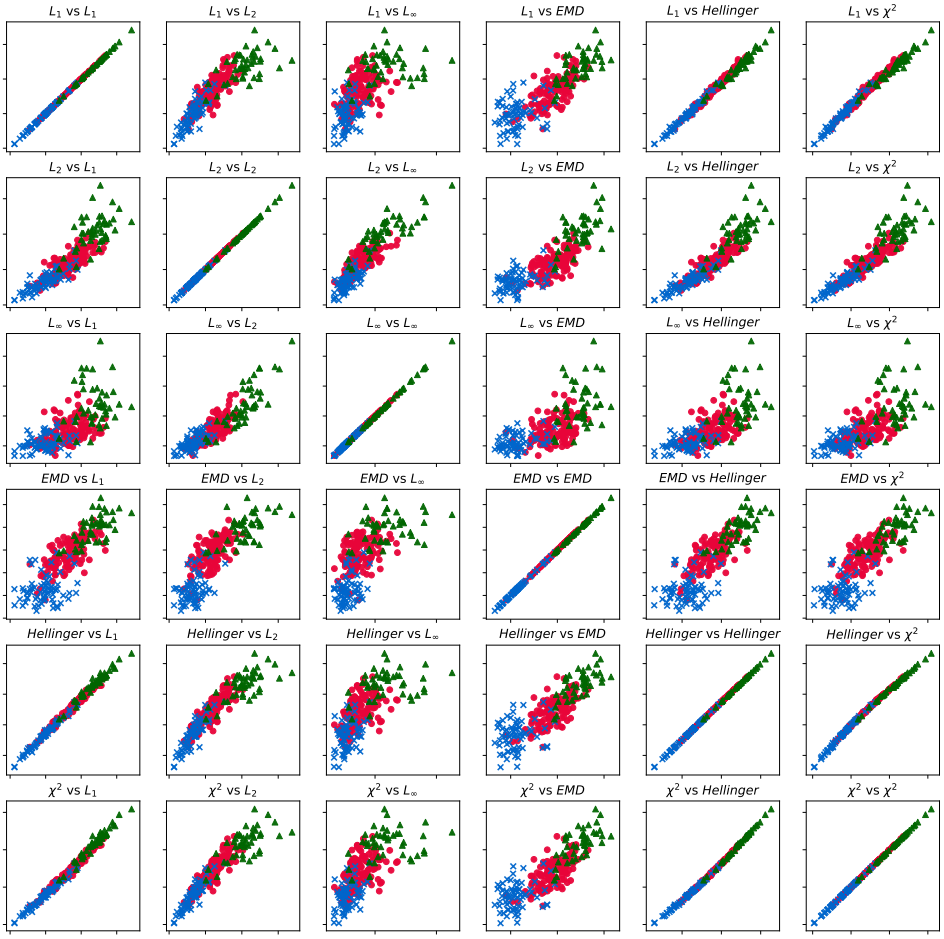


Figure B.1: Visualization of the correlation between different metrics applied to the histogram representation of the otolith data set. Families *labridae*, *soleidae*, and *scombridae* in red dots, blue crosses and green triangles respectively.

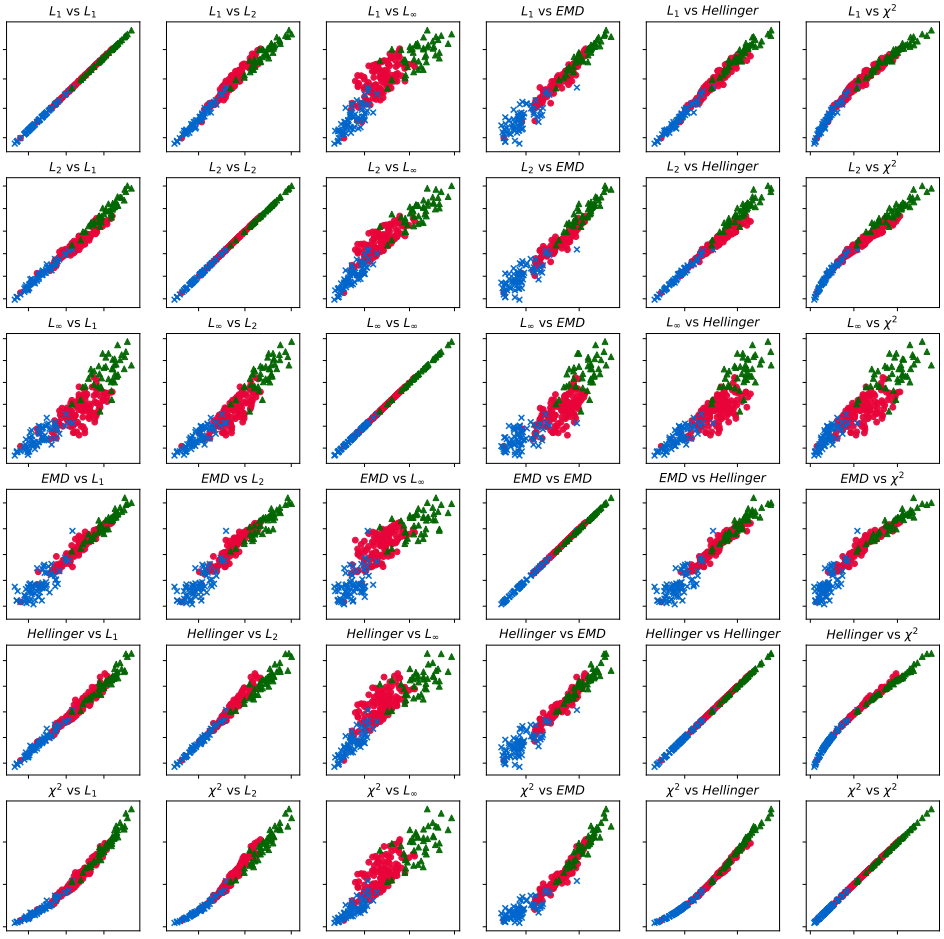


Figure B.2: Visualization of the correlation between different metrics applied to the kernel density representation of the otolith data set. Families *labridae*, *soleidae*, and *scombridae* in red dots, blue crosses and green triangles respectively.

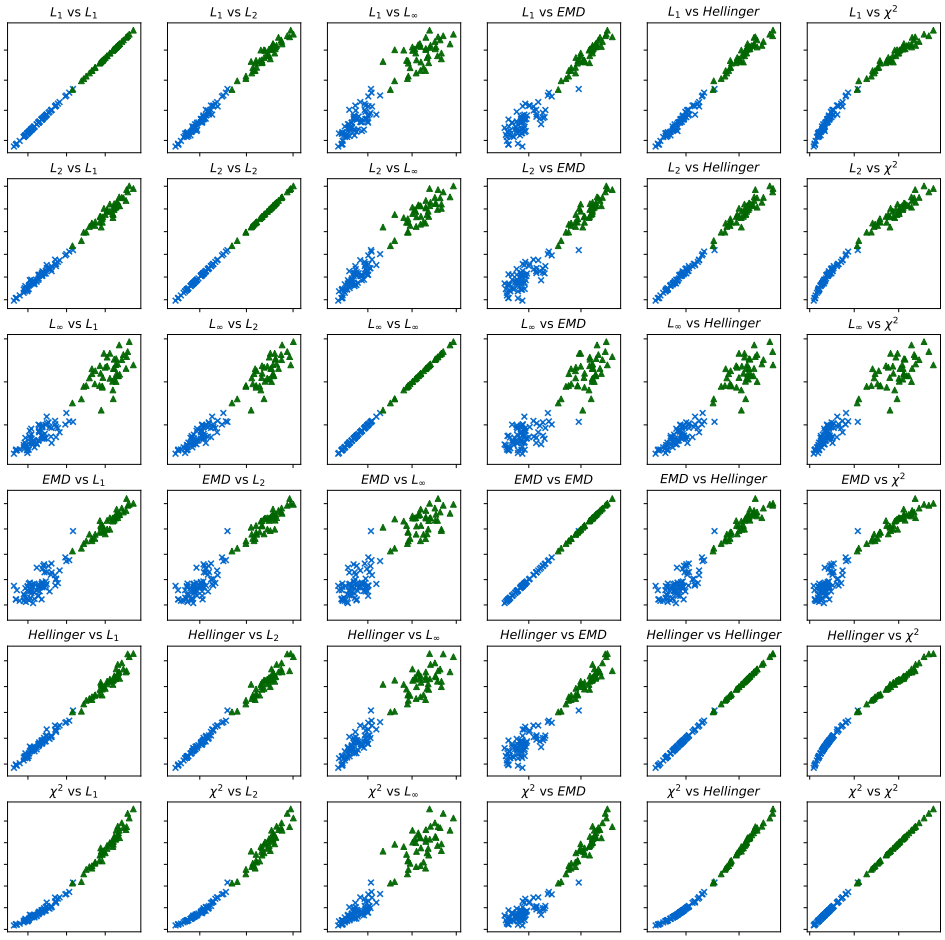


Figure B.3: Visualization of the correlation between different metrics applied to the histogram representation of the otolith data set. Families *soleidae*, and *scombridae* in blue crosses and green triangles respectively.

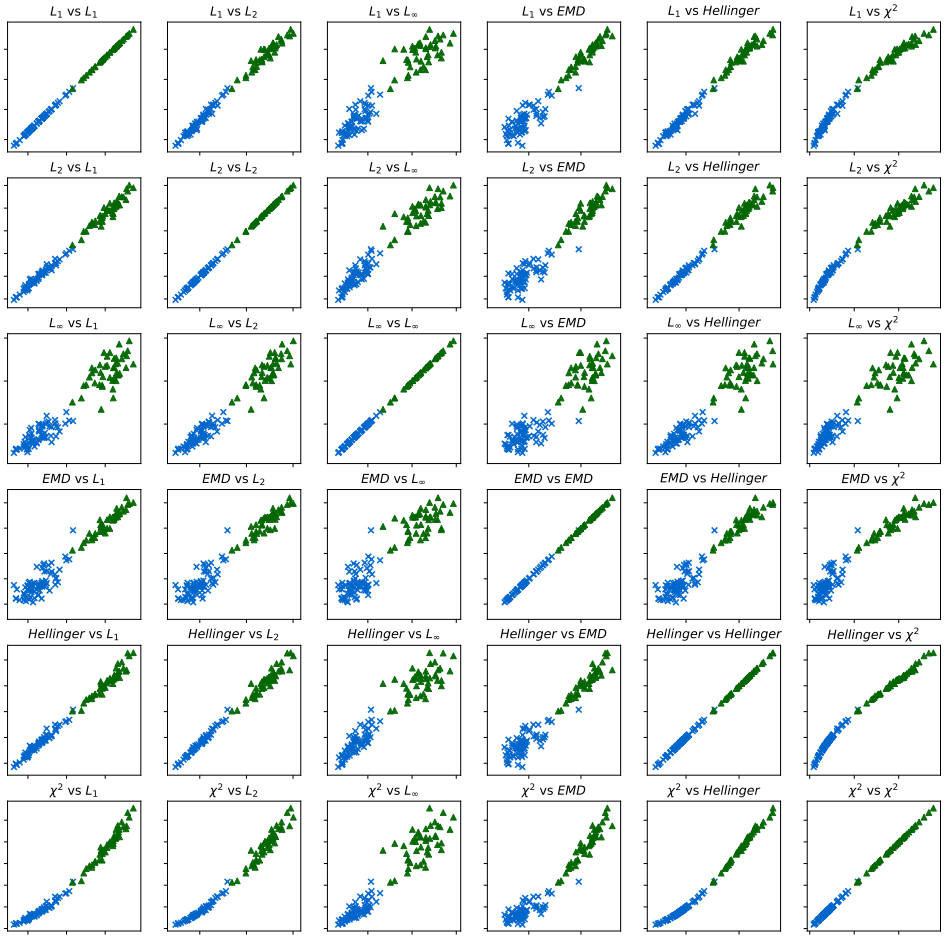


Figure B.4: Visualization of the correlation between different metrics applied to the kernel density representation of the otolith data set. Families *soleidae*, and *scombridae* in blue crosses and green triangles respectively.

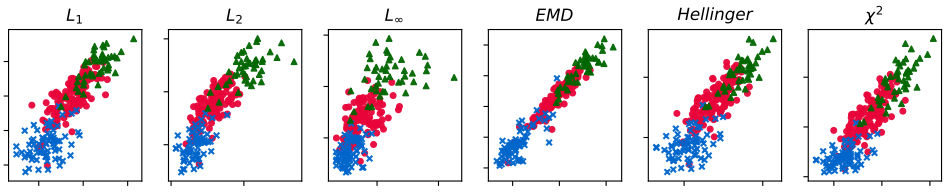


Figure B.5: Visualization of the correlation between different metrics when applied to the histogram or the kernel density representation of the otolith data set. Families *labridae*, *soleidae*, and *scombridae* in red dots, blue crosses and green triangles respectively.

C

Examples of Jupyter notebooks

In this chapter some examples of Jupyter notebooks developed to test the Python implementation are displayed. In these notebooks, examples of executions of experiments described in Sections 3.4 and 4.3, will be displayed in Sections C.2 and C.1 respectively.

C.1 Example of Jupyter notebook for Simple Figures Clustering and Classification

In this notebook experiments related to the simple figures classification and clustering problem are executed. Relevant information about the procedure such as results and execution time will be displayed. The parameters are diferent to the ones used in Section 3.4 to save time and space in the document.

Modules involved

```
In [1]: %load_ext autoreload
        %autoreload 2
        %reload_ext autoreload
        import populations.polygonpopulations as pp
        import distances.functionmeasures as fn
        import distances.functiondistances as fd
        import distances.periodicfunctiondistances as pfd
        import estimations
        import time
```

Data set generation

```
In [2]: number_of_vertices = 16
        number_of_bins = 16
        sampling_size=1000
```

```
        number_of_points= 16
        h = 1/16
        # Metrics for distance characterization
        distances = [fn.distance_L_2, fn.distance_emd]
        # Metrics to test in the classification algorithm
        metrics =  [fd.distance_L_2, fd.distance_emd]
        # Periodic Metrics to test in the classification algorithm
        pmetrics =  [pfd.distance_L_2, pfd.distance_emd]
        # Polygon population configuration
        numbers_of_vertices=[3,4,32]
        shape_names=["Triangle","Rectangle","Circle"]
        max_noise=0.7
        population_size = 100
        # Metric for the sklearn KNN call below
        metric = 'minkowski'
        # Cross validation configuration
        # Number of folds for generalization score estimates
        N_folds = 10
        # Number of folds for selecting K in training
        n_folds = 3
        # Feasible neighbor orders
        l_neigh = [3,5,7]
        start = time.clock()
        dataset = pp.PolygonDataSet(shape_names,max_noise, population_size,
                              numbers_of_vertices, sampling_size, number_of_bins,
                              distances=distances, h=h)
        print("Exec time:",time.clock() - start)
```

```
Exec time: 6.031575
```

Classification with periodic metrics

Histogram representation

```
In [3]: for pmetric in pmetrics:
            estimations.cross_validate(dataset.complete_histogram_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, pmetric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                             distance_L_2
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
```

```
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 8, 5: 2, 7: 0}
Top K: 3 (8 times selected)
Time elapsed: 508.84
Error Estimate: 0.1600 ∓ 0.0904
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       12.0          74.0             14.0
Triangle        1.0           21.0             78.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                             distance_emd
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 6, 5: 4, 7: 0}
Top K: 3 (6 times selected)
Time elapsed: 1109.70
Error Estimate: 0.1567 ∓ 0.0517
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       13.0          74.0             13.0
Triangle        1.0           20.0             79.0
********************************************************************************
```

Kernel density representation

```
In [4]: for pmetric in pmetrics:
            estimations.cross_validate(dataset.complete_kde_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, pmetric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                             distance_L_2
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
```

```
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 9, 5: 1, 7: 0}
Top K: 3 (9 times selected)
Time elapsed: 560.02
Error Estimate: 0.0333 ∓ 0.0333
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       4.0           91.0             5.0
Triangle        0.0           1.0              99.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                             distance_emd
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 9, 5: 1, 7: 0}
Top K: 3 (9 times selected)
Time elapsed: 1176.41
Error Estimate: 0.0233 ∓ 0.0335
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       1.0           94.0             5.0
Triangle        0.0           1.0              99.0
********************************************************************************
```

Classification with aligned representation

Histogram representation

```
In [5]: for metric in metrics:
            estimations.cross_validate(dataset.complete_histogram_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, metric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                             distance_L_2
Instances:                          300
```

```
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 9, 5: 1, 7: 0}
Top K: 3 (9 times selected)
Time elapsed: 5.86
Error Estimate: 0.1533 ∓ 0.0521
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       14.0          73.0             13.0
Triangle        2.0           17.0             81.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                             distance_emd
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 10, 5: 0, 7: 0}
Top K: 3 (10 times selected)
Time elapsed: 36.17
Error Estimate: 0.1633 ∓ 0.0526
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       10.0          73.0             17.0
Triangle        2.0           20.0             78.0
********************************************************************************
```

Kernel density representation

```
In [6]: for metric in metrics:
            estimations.cross_validate(dataset.complete_kde_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, metric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                             distance_L_2
```

```
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 8, 5: 2, 7: 0}
Top K: 3 (8 times selected)
Time elapsed: 5.89
Error Estimate: 0.0233 ∓ 0.0335
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       1.0           94.0             5.0
Triangle        0.0           1.0              99.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                             distance_emd
Instances:                          300
Attributes dimensions:              16
Error estimation number of folds:   10
Error estimation train size:        270
    K estimation number of folds:   3
    K estimation train size:        180
    K estimation test size:         90
Error estimation test size:         30
K's Selected: {3: 10, 5: 0, 7: 0}
Top K: 3 (10 times selected)
Time elapsed: 36.22
Error Estimate: 0.0267 ∓ 0.0200
Confusion Matrix:
                Circle        Rectangle        Triangle
Circle          100.0         0.0              0.0
Rectangle       3.0           93.0             4.0
Triangle        0.0           1.0              99.0
********************************************************************************
```

Classification with distances representation

Histogram representation

```
In [7]: estimations.cross_validate(dataset.complete_distance_histogram_dataset,
                                   dataset.complete_labels, l_neigh,
                                   N_folds,n_folds,'minkowski',
                                   dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                              minkowski
Instances:                           300
Attributes dimensions:               16
Error estimation number of folds:    10
Error estimation train size:         270
    K estimation number of folds:    3
    K estimation train size:         180
    K estimation test size:          90
Error estimation test size:          30
K's Selected: {3: 10, 5: 0, 7: 0}
Top K: 3 (10 times selected)
Time elapsed: 0.08
Error Estimate: 0.0000 ∓ 0.0000
Confusion Matrix:
                 Circle        Rectangle        Triangle
Circle           100.0         0.0              0.0
Rectangle        0.0           100.0            0.0
Triangle         0.0           0.0              100.0
********************************************************************************
```

Kernel density representation

```
In [8]: estimations.cross_validate(dataset.complete_distance_kde_dataset,
                                   dataset.complete_labels, l_neigh,
                                   N_folds,n_folds, 'minkowski',
                                   dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                              minkowski
Instances:                           300
Attributes dimensions:               16
Error estimation number of folds:    10
Error estimation train size:         270
    K estimation number of folds:    3
    K estimation train size:         180
    K estimation test size:          90
Error estimation test size:          30
K's Selected: {3: 10, 5: 0, 7: 0}
Top K: 3 (10 times selected)
Time elapsed: 0.08
Error Estimate: 0.0033 ∓ 0.0100
Confusion Matrix:
                 Circle        Rectangle        Triangle
Circle           100.0         0.0              0.0
Rectangle        0.0           99.0             1.0
Triangle         0.0           0.0              100.0
```

```
**************************************************************************
```

Clustering with aligned representation

Histogram representation

```
In [9]: estimations.cluster(attributes=dataset.complete_aligned_histogram_dataset,
                            k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
Circle      0   100 0
Rectangle   24  65  11
Triangle    38  5   57
Error:      0.5933333333333334
```

Kernel density representation

```
In [10]: estimations.cluster(attributes=dataset.complete_aligned_kde_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
Circle      100 0   0
Rectangle   16  79  5
Triangle    0   5   95
Error:      0.08666666666666667
```

Clustering with distances representation

Histogram representation

```
In [11]: estimations.cluster(attributes=dataset.complete_distance_histogram_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
Circle      100 0   0
Rectangle   0   100 0
Triangle    0   0   100
Error:      0.0
```

Kernel density representation

```
In [12]: estimations.cluster(attributes=dataset.complete_distance_kde_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
Circle      100 0   0
Rectangle   0   100 0
Triangle    0   0   100
Error:      0.0
```

C.2 Example of Jupyter notebook for Otolith Clustering and Classification

In this notebook experiments related to the fish otoliths classification and clustering problem are executed. Relevant information about the procedure such as results and execution time will be displayed. The parameters are different to the ones used in Section 4.3 to save time and space in the document.

Modules involved

```
In [1]: %load_ext autoreload
        %autoreload 2
        %reload_ext autoreload
        import compileall
        compileall.compile_dir(".",force=True,quiet=True)
        import populations.otolithpopulations as op
        import distances.functionmeasures as fn
        import distances.functiondistances as fd
        import distances.periodicfunctiondistances as pfd
        import estimations
        import time
```

Data set generation

```
In [2]: number_of_vertices = 16
        number_of_bins = 16
        sampling_size=1000
        number_of_points=16
        # Metrics for distance characterization
        distances = [fn.distance_L_2, fn.distance_emd]
        # Metrics to test in the classification algorithm
        metrics =  [fd.distance_L_2, fd.distance_emd]
        # Periodic Metrics to test in the classification algorithm
        pmetrics =  [pfd.distance_L_2, pfd.distance_emd]
        # Folders of data sets of images
        folder_names=["./data/families/labridaeFILLED/",
                      "./data/families/soleidaeFILLED/",
                      "./data/families/scombridaeFILLED/"]
        # Cross validation configuration
        # Number of folds for generalization score estimates
        N_folds = 5
        # Number of folds for selecting K in training
```

```
        n_folds = 3
        # Feasible neighbor orders
        l_neigh = [3,5,7]
        start = time.clock()
        dataset = op.OtolithDataSet(folder_names, number_of_vertices, sampling_size,
                                    number_of_bins, distances)
        print("Exec time:",time.clock() - start)
```

```
Exec time: 5.352024
```

Classification with periodic metrics

Histogram representation

```
In [3]: for pmetric in pmetrics:
            estimations.cross_validate(dataset.complete_histogram_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, pmetric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                  distance_L_2
Instances:                               240
Attributes dimensions:                   16
Error estimation number of folds:        5
Error estimation train size:             192
        K estimation number of folds:    3
        K estimation train size:         128
        K estimation test size:          64
Error estimation test size:              48
K's Selected: {3: 0, 5: 3, 7: 2}
Top K: 5 (3 times selected)
Time elapsed: 144.08
Error Estimate: 0.0500 ∓ 0.0283
Confusion Matrix:
                 labridae        soleidae        scombridae
labridae         121.0           2.0             2.0
soleidae         0.0             70.0            0.0
scombridae       8.0             0.0             37.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                                  distance_emd
Instances:                               240
Attributes dimensions:                   16
Error estimation number of folds:        5
Error estimation train size:             192
        K estimation number of folds:    3
```

```
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 2, 5: 1, 7: 2}
Top K: 3 (2 times selected)
Time elapsed: 304.13
Error Estimate: 0.0833 ∓ 0.0295
Confusion Matrix:
                labridae        soleidae        scombridae
labridae        118.0           5.0             2.0
soleidae        2.0             68.0            0.0
scombridae      11.0            0.0             34.0
********************************************************************************
```

Kernel density representation

```
In [4]: for pmetric in pmetrics:
            estimations.cross_validate(dataset.complete_kde_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, pmetric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                     distance_L_2
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 1, 5: 3, 7: 1}
Top K: 5 (3 times selected)
Time elapsed: 142.67
Error Estimate: 0.0375 ∓ 0.0243
Confusion Matrix:
                labridae        soleidae        scombridae
labridae        122.0           1.0             2.0
soleidae        0.0             70.0            0.0
scombridae      6.0             0.0             39.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                                     distance_emd
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
```

```
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 2, 5: 2, 7: 1}
Top K: 3 (2 times selected)
Time elapsed: 298.89
Error Estimate: 0.0250 ∓ 0.0243
Confusion Matrix:
                  labridae        soleidae        scombridae
labridae          124.0           0.0             1.0
soleidae          0.0             70.0            0.0
scombridae        5.0             0.0             40.0
********************************************************************************
```

Classification with aligned representation

Histogram representation

```
In [5]: for metric in metrics:
            estimations.cross_validate(dataset.complete_histogram_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, metric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                     distance_L_2
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 0, 5: 1, 7: 4}
Top K: 7 (4 times selected)
Time elapsed: 1.53
Error Estimate: 0.0667 ∓ 0.0306
Confusion Matrix:
                  labridae        soleidae        scombridae
labridae          119.0           5.0             1.0
soleidae          0.0             70.0            0.0
scombridae        10.0            0.0             35.0
********************************************************************************
################################################################################
------Cross Validation------
Metric:                                     distance_emd
```

```
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 1, 5: 2, 7: 2}
Top K: 5 (2 times selected)
Time elapsed: 9.02
Error Estimate: 0.0708 ∓ 0.0250
Confusion Matrix:
                 labridae        soleidae        scombridae
labridae         118.0           4.0             3.0
soleidae         1.0             69.0            0.0
scombridae       9.0             0.0             36.0
********************************************************************************
```

Kernel density representation

```
In [6]: for metric in metrics:
            estimations.cross_validate(dataset.complete_kde_dataset,
                                       dataset.complete_labels, l_neigh,
                                       N_folds,n_folds, metric,
                                       dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                     distance_L_2
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 4, 5: 1, 7: 0}
Top K: 3 (4 times selected)
Time elapsed: 1.49
Error Estimate: 0.0208 ∓ 0.0132
Confusion Matrix:
                 labridae        soleidae        scombridae
labridae         123.0           0.0             2.0
soleidae         0.0             70.0            0.0
scombridae       3.0             0.0             42.0
********************************************************************************
################################################################################
------Cross Validation------
```

```
Metric:                                     distance_emd
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 4, 5: 1, 7: 0}
Top K: 3 (4 times selected)
Time elapsed: 9.03
Error Estimate: 0.0292 ∓ 0.0167
Confusion Matrix:
                  labridae        soleidae        scombridae
labridae          123.0           1.0             1.0
soleidae          0.0             70.0            0.0
scombridae        5.0             0.0             40.0
********************************************************************************
```

Classification with distances representation

Histogram representation

```
In [7]: estimations.cross_validate(dataset.complete_distance_histogram_dataset,
                                   dataset.complete_labels, l_neigh,
                                   N_folds,n_folds,'minkowski',
                                   dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                     minkowski
Instances:                                  240
Attributes dimensions:                      16
Error estimation number of folds:           5
Error estimation train size:                192
        K estimation number of folds:       3
        K estimation train size:            128
        K estimation test size:             64
Error estimation test size:                 48
K's Selected: {3: 2, 5: 2, 7: 1}
Top K: 3 (2 times selected)
Time elapsed: 0.04
Error Estimate: 0.2125 ∓ 0.0306
Confusion Matrix:
                  labridae        soleidae        scombridae
labridae          105.0           10.0            10.0
soleidae          17.0            53.0            0.0
scombridae        14.0            0.0             31.0
```

```
********************************************************************************
```

Kernel density representation

```
In [8]: estimations.cross_validate(dataset.complete_distance_kde_dataset,
                                   dataset.complete_labels, l_neigh,
                                   N_folds,n_folds, 'minkowski',
                                   dataset.different_classes)
```

```
################################################################################
------Cross Validation------
Metric:                                  minkowski
Instances:                               240
Attributes dimensions:                   16
Error estimation number of folds:        5
Error estimation train size:             192
        K estimation number of folds:    3
        K estimation train size:         128
        K estimation test size:          64
Error estimation test size:              48
K's Selected: {3: 1, 5: 2, 7: 2}
Top K: 5 (2 times selected)
Time elapsed: 0.04
Error Estimate: 0.1125 ∓ 0.0468
Confusion Matrix:
                 labridae        soleidae        scombridae
labridae         120.0           0.0             5.0
soleidae         10.0            60.0            0.0
scombridae       12.0            0.0             33.0
********************************************************************************
```

Clustering with aligned representation

Histogram representation

```
In [9]: estimations.cluster(attributes=dataset.complete_aligned_histogram_dataset,
                            k=3, real_labels=dataset.complete_labels)
```

```
           C1  C2  C3
labridae   99  20  6
scombridae 2   68  0
soleidae   7   6   32
Error:     0.17
```

Kernel density representation

```
In [10]: estimations.cluster(attributes=dataset.complete_aligned_kde_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
labridae    109 9   7
scombridae  1   69  0
soleidae    3   0   42
Error:      0.08
```

Clustering with distances representation

Histogram representation

```
In [11]: estimations.cluster(attributes=dataset.complete_distance_histogram_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
labridae    92  21  12
scombridae  6   64  0
soleidae    11  0   34
Error:      0.21
```

Kernel density representation

```
In [12]: estimations.cluster(attributes=dataset.complete_distance_kde_dataset,
                             k=3, real_labels=dataset.complete_labels)
```

```
            C1  C2  C3
labridae    95  8   22
scombridae  7   63  0
soleidae    4   0   41
Error:      0.17
```