A proposal to compare the similarity between musical products. One more step for automated plagiarism detection?

Aarón López-García[1][0000−0001−8332−0381], Brian Martínez-Rodríguez[2][0000−0002−1109−0078], and Vicente Liern[3][0000−0001−5883−9640]

[1] Dep. of Computer Science,
Universitat de València, Spain.
logara8@alumni.uv.es
[2] Universidad Internacional de la Rioja (UNIR), Logroño, La Rioja, Spain
brian.martinez@unir.net
[3] Dep. Matemáticas para la Economía y la Empresa,
Universitat de València, Spain.
vicente.liern@uv.es

Abstract. In previous works, the authors presented a measure of similarity between melodies by identifying them with sequences of ordered vectors and using a clustering process based on fuzzy logics. Along the same line, we propose a measure of musical similarity between fragments of digital audio. We present the SpectroMap algorithm that allows us to detect the local maxima of the audio spectrogram representation (also known as constellation map) and we compared the similarity between different maps belonging to different audio excerpts. As a result, it is obtained a value that represents the resemblance between two musical products. This procedure could be used as a non-subjective tool in automatic plagiarism detection. To illustrate this method, three experiments have been carried out comparing different versions famous pop songs. The results point to the usefulness of the method, although this should be contrasted with an analysis of the human perception of this similarity.

Keywords: Fuzzy Clustering · Similarity · Plagiarism.

1 Introduction

In past editions of Mathematics and Computation in Music (MCM) we have presented a method to estimate the similarity between different characteristics of symbolic music (melody, rhythm, harmony, tunning) [9], [10]. In 2019 we presented *Mercury*, a computer framework in which techniques from fuzzy clustering were implemented to Computer-Assisted Musical Composition. This saved, to a certain extent, the uncertainty/inaccuracy inherent in any kind of music [7]. Despite the use of software, the approach has always been from the point of view of symbolic music, never from the pure treatment of sound. In this paper we propose to extend the applicability of the techniques showed in [9] to comparison of digital audio, based on some attributes of the spectrogram representation.

To achieve our goal, it is necessary to previously process the audio. For this, we have designed an algorithm, which we have called *SpectroMap*, for filtering local maxima (peaks) of the spectrograms. Once this filtering process has been carried out, we obtain the constellation map or fingerprint [12] of the audio fragment. Constellation map can be easily incorporated into the similarity calculations implemented in Mercury, thus obtaining a non-subjective numerical value of similarity between digital audio fragments.

The assessment of the similarity in the conditions described above can be considered as an important element to take into account for the detection of plagiarism. We do not mean to say that the subjective and perceptual part is not important, but if the calculation of similarity between two musical productions is automated, a high value of similarity between them should alert us. In this case, we could also conduct the traditional and pertinent tests [4] to evaluate the existence or not of plagiarism.

We present some examples of similarity estimation between different versions of the same song, using both the spectrogram filtering methods and the similarity calculation methods implemented in Mercury over three different corpora, one for each reference song. The results obtained, beyond giving a ranking of which version is most similar to the original, also provide information about possible compositional interrelationships between the different versions.

2 Theoretical Background

2.1 Clustering methods

Clustering methods are aimed to create groups of elements within an initial data set, so that the elements included in each group can be considered similar to each other. Unlike the classification methods, in which the elements are assigned with a pre-existing class, in the clustering methods the different classes or subgroups in which the data set is going to be divided have to be defined beforehand the execution of the analysis phase. The clustering procedure will consist of finding a partition of a data set $\mathbf{X}$ that satisfies certain grouping criteria. Following the criteria exposed in [6], given a data set, we will call *element* to each minimum unit of information belonging to it. Each element will have associated a total of q scalar magnitudes called *characteristics* or *attributes*. The term *cluster* (also *group* or *class*) will be used to designate each of the c groupings made from the data set. In *hard* clustering, it is understood that the elements that belong to a certain cluster share properties or characteristics with each other and are differentiable from the elements belonging to another cluster. In *fuzzy* clustering this distinction is no longer so clear. The term *centroid* denotes the central point of each of the clusters. The set of n data is

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^q. \tag{1}$$

where each $\mathbf{x}_i \in \mathbb{R}^q$, will be a point of q characteristics belonging to a metric space q-dimensional $\mathbb{R}^q$. The index i will designate the i-th element $\mathbf{x}_i$; the number x_{ik} will designate the value of the k-th characteristics of $\mathbf{x}_i$. The total amount of characteristics q is known as the dimensionality of the data set X, and it will have to be a finite and integer number greater than zero.

2.2 Hard and soft partitions

In [1] and [2] we find the necessary theoretical fundaments to define the different types of partitioning of a data set. Suppose that $\mathbf{X}$ is a finite set of n elements such that $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ and we want to distribute the elements of the set $\mathbf{X}$ in a number c of subsets $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_c\}$ with $2 \leq c \leq n$. This family of subsets $\{\mathbf{C}_j \colon 1 \leq j \leq c\} \subset \mathbf{X}$ will be a partition of type *hard* if:

$$\bigcup_{j=1}^{c} \mathbf{C}_j = \mathbf{X}, \quad \mathbf{C}_j \cap \mathbf{C}_k = \varnothing, \quad 1 \leq j \neq k \leq c. \tag{2}$$

The matrix $\mathbf{U} = [u_{ij}]$ will represent the membership coefficients of each element $\mathbf{x}_i$ to each subset $\mathbf{C}_j$.

2.3 k-means clustering

The *k-means* algorithm, first described by [8], is one of the most widely used clustering methods. It can be classified as a non-hierarchical partitioning method of clustering, in which the data set is divided into a number k of groups, each with a *centroid* called *mean*. This algorithm requires setting the number of clusters k in advance, as well as perform a previous initialization of the groups. The grouping results obtained will depend deterministically both on the number of clusters and on the initialization performed, so to trust the results it will be convenient to repeat the procedure with different initializations.

As we have seen, the operation of the algorithm has two main phases: the initialization phase and the iteration phase. In the first phase, each of the n elements will be randomly assigned to one of the k clusters. Is it possible to formulate the k-means algorithm as an optimization problem of an objective function that will be minimized under given convergence conditions [3].

Definition 1. *Let* $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^q$ *be a data set of* n *elements. The* k-means *objective function* $J_w \colon \boldsymbol{M}_c \times \mathbb{R}^{cq} \to \mathbb{R}^+$ *is defined as*

$$J_W(U, \mathbf{v}) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} (d_{ij})^2. \tag{3}$$

where $d_{ij} = d(\mathbf{x}_i, \mathbf{v}_j)$ is a distance function calculated between the element i and the centroid j; $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \mathbb{R}^{cq}, \mathbf{v}_j \in \mathbb{R}^q \forall j$ is the set of centroids from the clusters; $\mathbf{v}_j$ is the centroid of the cluster $u_j \in U, 1 \leq j \leq c$; and the matrix $\mathbf{U} = [u_{ij}] \in \mathbf{M}_{cp}$ is the belonging matrix to a *hard* partition, accomplishing

$$u_{ij} \in [0,1], \quad \sum_{j=1}^{c} u_{ij} = 1, \quad 1 \leq i \leq n, \quad 1 \leq j \leq c. \tag{4}$$

2.4 Fuzzy C-Means clustering (FCM)

The *fuzzy c-means* algorithm supposes a generalization of the functions described in 3, transforming them into an infinite family of functions. The first of these generalizations was made in [5], later formulated by [1] as an extension of the well known k-means algorithm.

Definition 2. *Let* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^q$ *be a data set of* n *items. The fuzzy objective function* c-means $J_w \colon \boldsymbol{M}_{fc} \times \mathbb{R}^{cq} \to \mathbb{R}^+$ *is defined as*

$$J_\lambda(U, \boldsymbol{V}) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{\lambda} (d_{ij})^2. \tag{5}$$

where $U \in \mathbf{M}_{fc}$ is a fuzzy partition of $\mathbf{X}$, and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \mathbb{R}^{cq}, \quad \mathbf{v}_j \in \mathbb{R}^q$ is the set of centroids associated to the clusters $u_j, 1 \leq j \leq c$; and $d_{ij} = d(\mathbf{x}_i, \mathbf{v}_j)$ is any distance function in $\mathbb{R}^q$; u_{ij} is the membership coefficient of the element $\mathbf{x}_i$ to the cluster j; and finally $\lambda \in [1, \infty)$ is the weight exponent, or *fuzziness degree* of the process.

The function originally proposed by [5] is obtained by setting $\lambda = 2$ and selecting the Euclidean distance $d(ij) = d_{euc}(ij)$. It was later generalized by [1] into the following family of functions $\{J_\lambda | 1 \leq \lambda < \infty\}$. We can now see that the objective functions have the distance weighted by the membership coefficients u_{ij}. Since $\mathbf{M}_{fc}$ is a fuzzy partition, the coefficients $u_{ij} \in [0,1]$.

The fuzzy clustering process will be achieved through an iterative optimization of the objective function J_λ, updating in each iteration both the membership coefficients u_{ij} and the centroids $\mathbf{v}_j$ by following the expressions (see [1]):

$$u_{ij} = \left(\sum_{k=1}^{c} \left[\frac{d(\mathbf{x}_i, \mathbf{v}_j)}{d(\mathbf{x}_i, \mathbf{v}_k)} \right]^{\frac{2}{\lambda - 1}} \right)^{-1}, \qquad \mathbf{v}_j = \Big(\sum_{i=1}^{n} u_{ij}^{\lambda} \mathbf{x}_i \Big) / \sum_{i=1}^{n} u_{ij}^{\lambda}. \tag{6}$$

The matrix $U = (u_{ij}), 1 \leq i \leq n, 1 \leq j \leq c$ is now a fuzzy partition of $\mathbf{X}$, built by the membership coefficients u_{ij}. The fuzzy partition verifies

$$\sum_{j=1}^{c} u_{ij} = 1, \quad 1 \leq i \leq n. \tag{7}$$

In what follows we will show the implementation of the fuzzy c-means clustering algorithm proposed by Bezdek in [1]:

STEP 1. Fix a number of clusters m, $2 \leq m < n$. Choose any inner product norm metric for $\mathbb{R}^q$; fix λ, $1 \leq \lambda < \infty$. Initialize $U^{(0)}$.

STEP 2. Calculate the fuzzy centroids $\{v_j^{(k)}\}$ with $U^{(k)}$ and expression (6).

STEP 3. Update $U^{(k)}$ using expression (6) and $\{v_j^{(k)}\}$.

STEP 4. Compare $U^{(k)}$ to $U^{(k+1)}$ using a convenient matrix norm, being $\epsilon \in (0,1)$ and arbitrary termination criterion. If $\| U^{(k+1)} - U^{(k)} \| \leq \epsilon$ then stop, otherwise set $k = k+1$ and return to STEP 2.

A dissimilarity based on FCM algorithm

Definition 3. *Let $\mathscr{T}^A = \{\mathbf{x_1}, \dots, \mathbf{x_n}\} \subset \mathbb{R}^q$ and $\mathscr{T}^B = \{\mathbf{y_1}, \dots, \mathbf{y_m}\} \subset \mathbb{R}^q$ be two data sets, where $n > m$. Let $d : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ be a distance function. Let u_{ij} be the final membership coefficients calculated with FCM algorithm, using data set $\mathscr{T}^A$ as data to be partitioned and $\mathscr{T}^B$ as initial set of centroids. The average dissimilarity $\mathscr{D}$ from the data set $\mathscr{T}^A$ to the data set $\mathscr{T}^B$ is defined by*

$$\mathscr{D}(\mathscr{T}^A, \mathscr{T}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij} d(\mathbf{x_i}, \mathbf{y_j}) \ . \tag{8}$$

It is noteworthy that dissimilarity $\mathscr{D}$ does not consider the possible natural order that could exist in both data sets, achieving a partition of $\mathscr{T}^A$ without any special weight to the elements whose degree of neighbourhood is stronger.

2.5 Fuzzy Ordered C-Means clustering (FOCM)

In [9] and [11] we presented FOCM, an improvement of the FCM algorithm in which the order of both data set and centroids sequences were taken into account during the partition process. Instead of partitioning a data set $\mathbf{X}$ with a given set of c centroids belonging to C categories, let us consider the possibility to implement the partition process introducing the order of the elements in the fuzzy partition process. For that purpose, let us consider two sequences $\mathscr{S}^A$ and $\mathscr{S}^B$ with different number of elements. Sequence $\mathscr{S}^A$ will be the ordered data set to be partitioned, and sequence $\mathscr{S}^B$ will represent the initial set of centroids.

In FOCM, the Neighbourhood Functions will provide higher weights of comparison to the pair of elements of the sequences that share closer positions in the order of each sequence. At the same time, they will decrease the contribution to the global dissimilarity to those pair of elements that are ordinally distant.

The purpose of FOCM is to modify the algorithm FCM so the natural order of both data set sequence $\mathscr{S}^A = \{\mathbf{x_1}, \dots, \mathbf{x_n}\}$ and centroids sequence $\mathscr{S}^B = \{\mathbf{y_1}, \dots, \mathbf{y_m}\}$, with, $n < m$, is considered during the partition process.

FOCM algorithm works as follows: for every step in which the fuzzy partition U has been calculated, the coefficients u_{ij} will be multiplied by a weight by means of a specific neighbourhood function $f(i,j)$. For accomplishing with de convergence criterion, the matrix should be normalized as follows into $\widetilde{U}$

$$\tilde{u}_{ij} = \frac{u_{ij} f(i,j)}{\sum\limits_{k=1}^{m} u_{ik} f(i,k)} . \tag{9}$$

STEP 1. Set $\{v_j^{(0)}\} = \{y_j\}$. Let m, n be the number of notes of $\mathscr{S}^B$ and $\mathscr{S}^A$, respectively. Choose any convenient neighbourhood function.

STEP 2. Choose any inner product norm metric for $\mathbb{R}^q$, and fix $\lambda \geq 1$. Calculate the initial $\widetilde{U}^{(0)}$ using (6), (9) and $\{v_j^{(0)}\}$.

STEP 3. Calculate the fuzzy cluster centers $\{v_j^{(k)}\}$ with $\widetilde{U}^{(k)}$ and the equation (6).

STEP 4. Update $\widetilde{U}^{(k)}$ using the equations (6), (9) and $\{v_j^{(k)}\}$.

STEP 5. Compare $\widetilde{U}^{(k)}$ to $\widetilde{U}^{(k+1)}$ using a convenient matrix norm; being $\epsilon \in (0,1)$ and arbitrary termination criterion. If $\| \widetilde{U}^{(k+1)} - \widetilde{U}^{(k)} \| \leq \epsilon$ then stop; otherwise set $k = k + 1$ and return to STEP 3.

There is a big number of possible neighbourhood functions (Gaussian, Triangular, Exponential, Sigmoidal, etc.) [9]. In this paper we have chosen the Gaussian neighbourhood function, i.e.

$$f_G(i,j) = Ae^{-\frac{1}{2\sigma^2}\left[i+1-\frac{(n-1)\cdot(j-1)}{(m-1)}\right]^2}. \tag{10}$$

2.6 Definition of a dissimilarity based on FOCM clustering

Using the FOCM algorithm, in [9] was defined a dissimilarity between any pair of sequences with different number of elements.

Definition 4. *Let $\mathscr{S}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^q$ and $\mathscr{S}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\} \subset \mathbb{R}^q$ be two sequences, where $n > m$. Let $d : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ be a distance function. Let u_{ij} be the final membership coefficients calculated with FOCM algorithm, using sequence $\mathscr{S}^A$ as data to be partitioned and sequence $\mathscr{S}^B$ as initial set of centroids. The average dissimilarity $\widetilde{\mathscr{D}}$ from the sequence $\mathscr{S}^A$ to the sequence $\mathscr{S}^B$ is defined by*

$$\widetilde{\mathscr{D}}(\mathscr{S}^A, \mathscr{S}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{u}_{ij} d(\mathbf{x_i}, \mathbf{y_j}). \tag{11}$$

In what follows we show the utility of expression (11) for evaluating de dissimilarity between songs or music compositions.

3 A comparison of musical products based on FOCM

Establishing an objective measurement for calculating the dissimilarity between musical products like pop, rock songs or classical music compositions, can be very useful as a tool for automatic plagiarism detection. Our approach for comparing two musical products will consist of: selecting the digital audio excerpts to be compared; extracting the constellation map (proposed in [12]) from the spectrogram of each excerpt; calculating the FOCM dissimilarity between constellation maps of both excerpts, with equation 11, taking into account that a constellation map is a sequence of points formed by time and frequency.

3.1 FFT process

With the aim of implementing a fingerprint extraction for a given musical signal X_t, we have designed an algorithm that computes a global peak detection over the spectrogram associated to give us its constellation map. Let N_{FFT} and N_O be the length of the Fast Fourier Transform (FFT) window and the number of elements to overlap between segments respectively, we first compute the spectrogram of the signal (S_{tfa}), by using the Hamming window, in order to get the (time, frequency, amplitude) vectors by considering these two parameters. Such representation contains the amplitude spatial information to analyze. Our engine search determines whether a time-frequency point can be considered locally relevant according to its neighbourhood. Then, the detection is processed regarding a required band. Let $\{T_i\}_{i=1}^{n}$ and $\{F_j\}_{j=1}^{m}$ be the time and frequency bands of the spectrogram with the amplitude of the event, we can reformulate the spectrogram $S_{tfa} = (T_i)_{i=1}^{n} = (F_j)_{j=1}^{m}$ as its rows and columns representations.

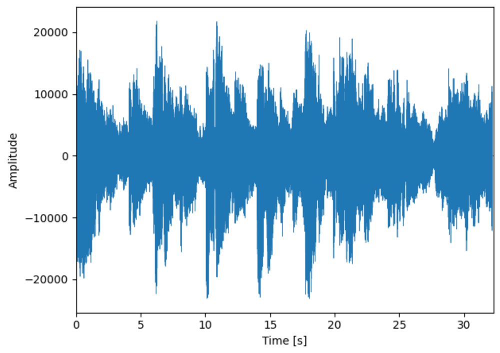

Fig. 1. Example of waveform from an excerpt of a pop song.

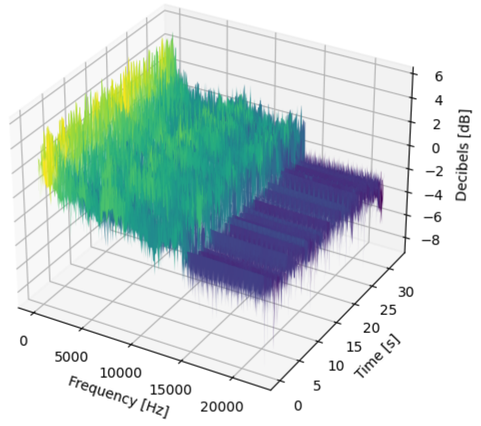

Fig. 2. Spectrogram visualization of the previous excerpt.

3.2 Peak detection Algorithm

As part of the engine search, we define two windows $\phi_T^{d_T}$ and $\phi_F^{d_F}$ to process the local pairwise comparisons with a respective length of d_T and d_F, whose functionality is to extract a number of elements of the band and return the local maximum. We can describe the time-band window mechanism with length of $0 < d_T \leq n$ and structure $T_i = (T_i^1, ..., T_i^n)$ as

$$\phi_T^{d_T}(T_i) = \left(\max\{T_i^k, ..., T_i^{k+d_T}\}\right)_{1\leq k\leq n-d_T-1}, 1 \leq i \leq n. \tag{12}$$

When we group all the values we drop those elements that have equal index to avoid duplicates. We can group the window of each band to create the set:

$$\Phi_T^{d_T} = \{\phi_T^{d_T}(T_i)\}_{i=1}^n. \tag{13}$$

This way, we get the topologically prominent elements per each feature vector. With 12, it is easy to note that even though there are $n - d_T - 1$ matches, the window $\phi_T^{d_T}(T_i)$ may contain a smaller number of elements whenever $d_T > 2$. Depending of how restrictive we need to be, we can proceed with just one of the bands or combine them to create a more stringent search since it is returned only if the peaks that are prominent in both directions. Finally, the algorithm merges all the band-dependent peaks 13, to give us the total number of spatial points that determines the audio fingerprint. Our engine search, *SpectroMap*, processes audio signals in order to return an output file with the (time, frequency, amplitude) peaks detected. The algorithm works in these steps:

STEP 1. Decide the window to use and set the parameters N_{FFT} and N_O.
STEP 2. Read the audio file to get its amplitude vector and its sample rate.
STEP 3. Compute the spectrogram through the associated Fourier transformations.
STEP 4. Set a fixed window length (d_T, d_F or both) for the pairwise comparisons.
STEP 5. Choose the settings to proceed with the peak detection over a selected band or a combination of both.
STEP 6. Create an identification matrix which consists in a binary matrix with the same shape as the spectrogram with the position of the highlighted prominences.
STEP 7. Extract such elements and create a file with the (time, frequency, amplitude) vectors.

Regarding the step 5, authors highly recommend to select both bands to perform the peak detection since the output is more filtered and spatially consistent. For the remainder steps, its choice is a personal decision that depends on the scope of the research.

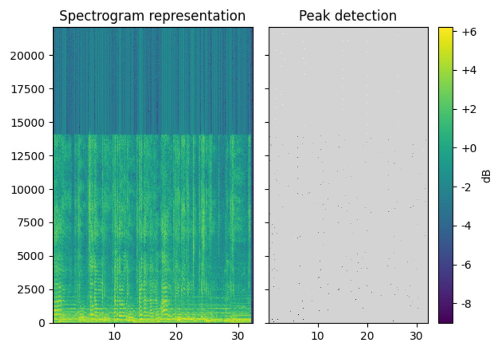

Fig. 3. Spectrogram and result of peak detection algorithm.

3.3 Constellation Map

As it was previously explained, the constellation map is obtained by means of the filtering of local maximum (peak detection) using the algorithm SpectroMap. The sequence of peaks is created by sorting each peak by its appearance time.

Definition 5. *A Constellation Map is the sequence $\mathscr{M}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^2$ where each $\boldsymbol{x}_i \in \mathbb{R}^2$ is an observable defined by 2 features: time and frequency. Each element has been sorted by its appearance time.*

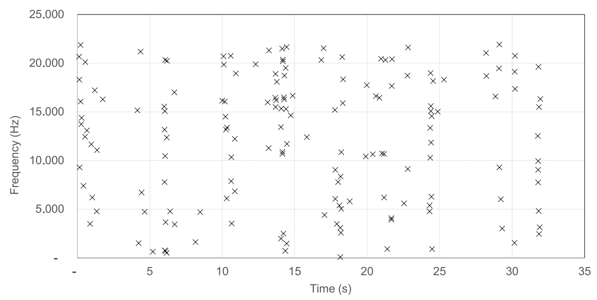

Fig. 4. Example of constellation map generated from an audio excerpt.

3.4 Calculation of the dissimilarity based on FOCM

Using the FOCM algorithm, we can define a dissimilarity between any pair of constellation maps.

Definition 6. *Let $\mathscr{M}^A = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\} \subset \mathbb{R}^2$ and $\mathscr{M}^B = \{\mathbf{y_1}, \ldots, \mathbf{y_m}\} \subset \mathbb{R}^2$ be two constellation maps, where $n > m$. Let $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ be a distance function.*

Let u_{ij} be the final membership coefficients calculated with FOCM algorithm, using constellation map $\mathscr{M}^A$ as data to be partitioned and constellation map $\mathscr{M}^B$ as initial set of centroids. The average dissimilarity $\widetilde{\mathscr{D}}$ between this two constellation is defined by

$$\widetilde{\mathscr{D}}(\mathscr{M}^A, \mathscr{M}^B) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{u}_{ij} \cdot d(\mathbf{x_i}, \mathbf{y_j}). \tag{14}$$

The expression (14) allows us to evaluate the musical plagiarism between any two given excerpts of digital audio.

4 Experiments

To illustrate the applicability of this method, we have designed three experiments to estimate the similarity between different versions of three different pop songs. We have chosen the songs: *Someone Like You*, by Adele; *When I was your man*, by Bruno Mars; *All of me*, by John Legend. This selection is convenient for creating the three different corpora, since there are numerous and different covers available on YouTube. The videos have been downloaded, and the digital audio has been extracted in wav format at 44.100Hz and 16 bits, selecting the same fragment of the song. With this excerpts we have created three experimental corpora. For each corpora we will calculate the similarities using the method explained in the previous section: applying the SpectroMap algorithm and equation 14. In Tables 1, 3 and 5 are shown the audio sources used in each experiment. For each corpora, we will compare the different versions with each other and with the original, in order to sort them from greater to lesser similarity. The results obtained are shown in Tables 2, 4 and 6.

For experiment No.1, according to the data shown in the table 2, the closest resemblance to the official one is the version by the artist jordan (8.21). However, the *leo*, *imy2* and *masha* versions are more similar to each other than to the official version. This fact could indicate a notable influence between these three artists. The version farthest from the official one is that of *leo* (13.62). Once this result is obtained, we listen to the version and verify that the artist has made a version in *rock* style of Adele's theme, effectively far removed in perceptual terms from the original version. The furthest versions are those of *nursera* and *leo* (22.85). Again, if we listen to both versions, the auditory difference is evident, since both versions represent antagonistic musical styles.

The results for experiment No.2 are shown in the table 4. The closest resemblance to the official one is the live version by the same artist John Legend (1.311), due to the big similarity of tempi between two versions. The versions of artist *smith* and *imy2* are more similar to the live version. The artist closest to the oficial version is *scaccia* (1.326). The version farthest from the official one is of *stewart* (13.62).

In experiment No.3 (table 6). The closest resemblance to the official song is the cover by artist *scaccia* (2.099), that is also the closest to the live version (1.935). The farthest version from the official one is of *leroy* (2.927).

Table 1. Audio sources used in the first experiment for song *Someone like you*

Artist	YouTube url
Adele Oficial Video (oficial)	`https://www.youtube.com/watch?v=hLQl3WQQoQ0`
Adele Live Performance (britawards)	`https://www.youtube.com/watch?v=qemWRToNYJY`
Angelina Jordan (jordan)	`https://www.youtube.com/watch?v=nU9TA7OfXro`
Nursera Yener (nursera)	`https://www.youtube.com/watch?v=Z9iylN-IiUA`
Masha (masha)	`https://www.youtube.com/watch?v=0EwSEsSvxGY`
imy2 (imy2)	`https://www.youtube.com/watch?v=qIuPgPyTNKE`
Leo Moracchioli (leo)	`https://www.youtube.com/watch?v=pkbbd3fhcMw`

Table 2. Dissimilarities calculated between different covers from experiment No.1

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
leo	imy2	5.405	jordan	brit	9.693	oficial	brit	10.270	leo	oficial	13.620
leo	masha	5.416	oficial	masha	9.698	imy2	brit	10.455	leo	jordan	14.392
imy2	masha	5.603	brit	jordan	9.778	oficial	imy2	12.759	nursera	brit	14.440
masha	imy2	5.658	brit	masha	9.788	jordan	imy2	13.568	nursera	masha	17.487
oficial	jordan	8.212	jordan	nursera	9.831	oficial	leo	13.586	nursera	imy2	21.603
oficial	nursera	9.680	leo	brit	10.998	imy2	jordan	13.608	nursera	leo	22.847

Table 3. Audio sources used in the second experiment for song *When I was your man*

Artist	YouTube url
Bruno Mars Oficial Version (bmo)	`https://www.youtube.com/watch?v=ekzHIouo8Q4`
Bruno Mars Live performing (bml)	`https://www.youtube.com/watch?v=gY4GZgZK9H0`
Alexander Stewart (stewart)	`https://www.youtube.com/watch?v=j_d3gq5JCAc`
Sam Smith (smith)	`https://www.youtube.com/watch?v=_ZaLIiV7c7Y`
imy2 (imy2)	`https://www.youtube.com/watch?v=uBh_7PBy8cg`
Stephen Scaccia (scaccia)	`https://www.youtube.com/watch?v=NhmOMHQKYDY`

Table 4. Dissimilarities calculated between different covers from experiment No.2

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
smith	bml	1.018	bmo	scaccia	1.326	smith	imy2	1.427	smith	scaccia	1.571
stewart	smith	1.262	scaccia	bml	1.335	bml	imy2	1.430	imy2	stewart	1.615
imy2	bml	1.264	imy2	bmo	1.340	bml	bmo	1.466	bmo	smith	1.651
bmo	bml	1.311	imy2	smith	1.398	bml	scaccia	1.495	scaccia	stewart	1.766
imy2	scaccia	1.324	stewart	bml	1.425	scaccia	smith	1.554	stewart	bmo	2.042

Table 5. Audio sources used in the third experiment for song *All of me*

Artist	YouTube url
John Legend Oficial Video (jlo)	`https://www.youtube.com/watch?v=450p7goxZqg`
John Legend Live Performance (jll)	`https://www.youtube.com/watch?v=s18cJqrBI0k`
Leroy Sanchez (leroy)	`https://www.youtube.com/watch?v=Im6_k-UMJeo`
Luciana Zogbi (zogbi)	`https://www.youtube.com/watch?v=39_0mBO9jVg`
Stephen Scaccia (scaccia)	`https://www.youtube.com/watch?v=07McLNDuffo`
Scott Hoying (hoying)	`https://www.youtube.com/watch?v=d0GR60bul4M`

Table 6. Dissimilarities calculated between different covers from experiment No.3

Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.	Cover	Cover	Dissim.
scaccia	jll	1.935	jlo	jll	2.191	jll	hoying	2.309	hoying	jlo	2.832
jll	scaccia	1.989	leroy	hoying	2.210	jll	leroy	2.316	leroy	jlo	2.917
scaccia	jlo	2.099	scaccia	hoying	2.211	scaccia	zogbi	2.662	zogbi	hoying	2.969
jlo	scaccia	2.137	hoying	scaccia	2.270	zogbi	scaccia	2.686	zogbi	leroy	3.196
scaccia	leroy	2.142	leroy	jll	2.273	jlo	zogbi	2.722	zogbi	leroy	3.196

5 Conclusions

The fuzzy logic-based procedures that were implemented for computer-assisted music composition in Mercury software can be used for automatic assessment of music plagiarism from digital audio files.

The Internet and social networks offer an excellent platform for the dissemination of musical content. However, plagiarism detection requires automatic tools that allow quickly and effectively discriminate those versions that may be suspicious in terms of their resemblance to others. Beyond the legal and ethical aspects, this resemblance can be useful for the performers or authors themselves, who can discover their own and other influences in other artists.

References

1. Bezdek, J. C.: Pattern Recognition with Fuzzy Objective Function Algoritms. Plenum Press, New York (1981)
2. Bezdek, J. C. and Keller, J. and Krisnapuram, R. and Pal, N.: Fuzzy models and algorithms for pattern recognition and image processing. Kluwer Academic Publishers, Boston, London, Dordrecht (1999)
3. Gan, G. and Ma, C. and Wu, J.: Data clustering: theory, algorithms, and applications. SIAM. Philadelphia (2007)
4. R. De Prisco et al.: Music Plagiarism at a Glance: Metrics of Similarity and Visualizations. In: 21st International Conference Information Visualisation (IV), pp. 410–415. IEEE, London (2017).https://doi.org/10.1109/iV.2017.49
5. Dunn, J. C.: A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics, 3, 32–57 (1973)
6. Jain, A. K. and Dubes, R. C.: Algorithms for clustering data. Prentice-Hall, Inc. (1988)
7. Liern, V.: Fuzzy tuning systems: the mathematics of musicians. Fuzzy sets and systems, 150(1), 35–52 (2005)
8. MacQueen, J.: Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 3, 281–297.Oakland, USA (1967)
9. Martínez–Rodríguez B., Liern V.: A Fuzzy-Clustering Based Approach for Measuring Similarity Between Melodies. In: Agustín-Aquino O., Lluis-Puebla E., Montiel M. (eds) Mathematics and Computation in Music. MCM 2017. Lecture Notes in Computer Science, vol 10527. Springer, Cham (2017).https://doi.org/10.1007/978-3-319-71827-9_21
10. Martínez–Rodríguez B., Liern V.: Mercury: A Software Based on Fuzzy Clustering for Computer-Assisted Composition. In: Montiel M., Gomez-Martin F., Agustín-Aquino O. (eds) Mathematics and Computation in Music. MCM 2019. Lecture Notes in Computer Science, vol 11502. Springer, Cham (2019).https://doi.org/10.1007/978-3-030-21392-3_19
11. Martínez–Rodríguez B.: El fuzzy clustering y la similitud musical: aplicación a la composición asistida por ordenador. (PhD Thesis). Universidad Politécnica de Valencia, Valencia, Spain (2019). https://doi.org/https://doi.org/10.4995/Thesis/10251/134056
12. Wang, A.: An industrial strength audio search algorithm. Ismir **2003**, 7–13 (2003).