Detection and localization of partial audio matches in various application scenarios

Milica Maksimović¹ · Patrick Aichroth¹ · Luca Cuccovillo¹

Received: 21 March 2019 / Revised: 17 July 2020 / Accepted: 16 September 2020 /

Published online: 19 June 2021

© The Author(s) 2020

Abstract

In this paper, we describe various application scenarios for archive management, broadcast/stream analysis, media search and media forensics which require the detection and accurate localization of unknown partial audio matches within items and datasets. We explain why they cannot be addressed with state-of-the-art matching approaches based on fingerprinting, and propose a new partial matching algorithm which can satisfy the relevant requirements. We propose two distinct requirement sets and hence two variants / settings for our proposed approach: One focusing on lower time granularity and hence lower computational complexity, to be able to deal with large datasets, and one focusing on fine-grain analysis for small datasets and individual items. Both variants are tested using distinct evaluation sets and methodologies and compared with a popular audio matching algorithm, thereby demonstrating that the proposed algorithm achieves convincing performance for the relevant application scenarios beyond the current state-of-the-art.

Keywords Audio matching · Partial audio matching · Duplicate detection · Provenance analysis · Metadata tracking · Copy-move-forgery

1 Introduction

Audio matching is a topic which was thoroughly investigated over the last decades: Being able to match a query file against a reference dataset is important for many application domains, including broadcast monitoring, music identification, copyright management, etc.

Typical requirements for such identification use cases include robustness against distortion, the ability to deal with very large reference datasets, low computational cost and efficient search. The respective implementations have matured over the years and have become an integral part of commercial products and systems.

More recently, however, several application scenarios have emerged which do not, or do not only require query-based, robust matching (see Fig. 1). Instead, they require partial

✉ Milica Maksimović
milica.maksimovic@idmt.fraunhofer.de

¹ Fraunhofer IDMT, Ehrenbergstr. 31, 98693, Ilmenau, Germany

Fig. 1 Classic, query-based matching

matching without a given query, i.e. *detection and accurate localization of arbitrary partial matches*, the existence and amount of which is unknown (see Fig. 2).

The paper is organised as follows: In Section 2 we are going to outline in details the relevant application scenarios for partial audio matching, and their corresponding requirements; Section 3 describes the existing state-of-the-art approaches for classic query-based audio matching; Section 4 presents our approach for partial audio matching, which is then evaluated in Section 5 using two distinct requirement sets. Section 6 closes with conclusions and an outlook.

2 Application scenarios

Partial duplicates can result from different types of editing actions. In Fig. 3, such actions are summarized: Item 1 is an original item within a dataset, and items 2-5 are derivatives created by:

- *cutting (removal)*, as represented with item 2, where the segments A-B and C-D of item 1 are spliced together and B-C is removed
- *pasting (insertion)*, as represented with item 3, where new material is inserted between two consecutive segments of item 1, namely A-B and B-C
- *cutting and pasting (replacement)*, as represented with item 5, where from segment A-D of item 1, subsegment B-C is removed and replaced with other content. This case is especially difficult to detect if the segment removed has the same length as the segment inserted, which happens frequently in practice
- *splicing*, as represented in items 2-5, where segment(s) of item 1 are concatenated with each other and with new content (i.e., not present in item 1)

Fig. 2 Partial matching without query

Fig. 3 Actions / cases resulting in partial duplicates

Various application scenarios include such actions and hence require detection and localization of partial audio matches (as opposed to classic, query-based matching). They can be grouped into four application domains:

- **archive / asset management**, which addresses issues related to content and metadata storage and tracking
- **broadcast and stream analysis**, which aims at supporting the analysis of broadcasted / streamed content and programs
- **media search**, which addresses topics related to querying and retrieval of media content
- **media forensics**, which aims at detecting content tampering

In the following, we will present these application scenarios in more detail.

2.1 Archive management

As for A/V archive and asset management, especially if used for production purposes, it is common that partial audio duplicates - sometimes transcoded, including gain changes etc. - are created. However, such duplicates often cannot be automatically tracked, because production applications and systems used do not provide such functionality, the existing workflows include legacy tools or external workers, or final edits right before the broadcasting are not tracked.

Within this context, partial audio matching can be used for **(partial) duplicate detection and repository clean-up**: By detecting and localizing partial duplicates, it can inform systems and tools and hence trigger further actions regarding cleanup, which can decrease storage costs.

The detection and localization of partial matches within an archive can also be applied for **automatic metadata propagation and tracking** for duplicates throughout an archive, and for **metadata validation and cleanup** in case of erroneous and conflicting information.

Similarly, it can also be used for **provenance and rights tracking**, in order to simplify or automate the task of tracking and reporting provenance and copyright information for productions by “linking” them back to the original source material.

For all the described functionalities, it is possible to **exchange extracted fingerprints between several archives**, to detect partial duplicates in order to share and curate respective metadata or propagate rights information across different locations, however without the need to exchange content. This can be useful to build up cross-archive metadata stores and

to support collaboration, e.g. joint metadata curation, while avoiding copyright and data protection issues and traffic costs.

One technology that can perfectly complement partial matching in such scenarios is audio phylogeny analysis [18, 22], which can automatically detect parent-child relations between different files. For instance, audio phylogeny can be applied to identify the root item within a set of near-duplicate audio items (see Fig. 4). For duplicate removal, for instance, it can be used to decide about which item should be kept among a set of duplicates. If combined with audio partial matching, on the other hand, audio phylogeny can further enhance the aforementioned rights and provenance tracking, by providing information about the chronological order of processing steps (see Fig. 5).

2.2 Broadcast and stream analysis

Partial audio matching can also be used to **analyze broadcast / streamed content reuse and program structures**: By self-comparing audio segments of broadcast or streamed material e.g. over the course of time for an individual channel, it is possible to gain interesting information such as general repetitiveness, and repetitiveness over time. As depicted in Fig. 6, where groups of partial duplicates are marked with the same color, such analysis is also a good starting point for an automatic or semi-automatic analysis of the overall structure of a program, including detection of specific content types.

The same technique can also be used to compare content from different stations and sources (see Fig. 7)), to investigate whether, how and when content is reused across different stations. This is especially interesting for the case of **news tracking** that includes the same original material e.g. from public speeches or events, as it can reveal time delays and difference in reporting across various sources. This technique can also be used to **measure outreach and PR (public relations) effectiveness** for audio material by tracking its reuse across broadcasts and social media channels.

← Phylogeny Analysis for Content De-Duplication

Fig. 4 Phylogeny analysis results for a set of near-duplicates, as visualized within the respective software tool: Nodes represent audio files, connections represent parent-child relations between nodes, and node color and size depend on the generation, i.e. amount of past transformation that an item has undergone. For instance, the big dark green node represents the root node (generation zero) and hence the file that was least processed, while the light green node placed furthest on right belongs to generation one and was created by encoding the root node with the *mp3* codec using a bitrate of *128kBit/s*

Fig. 5 Tracking of partial audio duplicates within a dataset, as visualized within the respective software tool: The colored rectangles present audio files and the light gray connections represent (full or partial) matches. This represents the results of a joined phylogeny and partial matching analysis, where detected near or partial duplicates were subsequently analyzed using the audio phylogeny software. This allowed for the tracking of content transformations in chronological order from unprocessed original audio files on the left to the derived production on the right. The dataset for this analysis was created manually

2.3 Media search and synchronization

Partial audio matching can also be used for content-based search, e.g. to **find video material that stems from the same event** and shares the same audio material, but was recorded from different camera perspectives (see Fig. 8). It can also be used to **synchronize and**

Fig. 6 Audio reuse detection for one radio channel, as visualized within the respective software tool. The horizontal axis represents time (seconds) for one-hour-blocks of the broadcast, while the vertical axis covers chronologically ordered blocks over the course of several days for *one* channel. Rectangles with the same color represent partial matches (please note that an arbitrary selection of matches was selected and highlighted in this example for better visibility). The result were obtained by analyzing real-world radio material from one channel that was recorded over the course of 3 days, using partial audio matching

Fig. 7 Audio reuse detection for different local radio channels from the same broadcaster, as visualized within the respective software tool. The horizontal axis represents time (seconds) for one-hour-blocks of the broadcast, while the vertical axis covers blocks from *different* channels that were all recorded starting at the same time. The result was obtained by analyzing real-world broadcasted radio streams from 29 different channels that were recorded over the course of 1 day, using partial audio matching

Fig. 8 Detecting A/V material from the same event, as visualized within the respective software tool: Two different video files are presented, one being the original recording from an event, and the other being a weekly summary. The pink rectangle on the respective progress bars of the videos indicates a partial audio match, which is caused by both videos covering parts of the same speech. For this analysis, we analyzed real-world data from the White House video gallery <https://obamawhitehouse.archives.gov/> using partial audio matching

reconstruct A/V material that contains a common audio stream, in that it can localize overlaps and reconstruct the full version.

2.4 Media forensics

Within the domain of media forensics and tampering detection, partial audio matching is useful for several purposes: Firstly, by allowing **detection of relevant items** that are distributed e.g. via social networks and have partial overlaps, it can provide important clues to assess the credibility of an item under inspection and its source, and to understand how content was distributed.

More importantly, it can also be used for **the detection and localization of copy-move forgeries**. Copy-move forgeries are attacks in which parts of the content are deliberately copied and moved within the same audio file in order to modify a message, typically using short segments with single words, e.g. moving the word “yes” from one location to another. Such attacks are difficult to detect manually, but partial matching can be applied to efficiently and reliably detect such cases (Fig. 9).

Finally, partial matching can also be used for **perceptual integrity** checks (Fig. 10): By validating that all parts of a given item appear in the same order and without any perceptual modification, partial matching can provide a means to check the integrity of a file, while being robust to content modifications such as transcoding.

2.5 Requirements

All the aforementioned application scenarios share the need for accurate detection and localization of partial matches. They differ significantly, however, for what concerns their requirements in terms of computational cost, and time granularity of the matches:

- Partial matching requires a comparison of all item pairs within a dataset, which is why computational cost grows exponentially with the data volume to be analyzed. Depending on the scenarios, that volume can range from a few minutes to thousands of hours of material to be analyzed. And the more it is, the stronger the need to reduce computational cost.
- Computational cost for matching can be reduced, in principle, by imposing a low (coarse grain) **time granularity** of the detected matches. This is a tradeoff which is acceptable for all the described scenarios *except the ones within the media forensics*

Fig. 9 Copy-move tampering detection: A segment containing the word “YES” is used to replace a segment containing the word “NO”, resulting in a forged content version

Fig. 10 Perceptual integrity check, as visualized within the respective software tool: The horizontal axis depicts the length of audio files (seconds), while the vertical axis depicts two files. Parts of the original file *b2* were reused and reordered for the creation of a longer, tampered audio file *b1*. Both files stem from a real-world forensics analysis to assess the integrity of a recording

domain, which require a granularity down to the length of individual short words, i.e. a few hundred milliseconds. Audio forensics, however, typically requires the analysis of only small amounts of data, often including only very few or even single items with a duration of one hour or less. As a consequence, computational cost is typically not an issue within that domain.

Both requirements influence each other: Increased time granularity results in increased computational cost and hence represents an issue especially when dealing with larger datasets. Hence, it makes sense to define two different sets of requirements: One set that aims at lowering computational cost (e.g. via decreased time granularity) to deal with larger datasets, which is useful for most domains except for media forensics. And one set which aims at high time granularity (which comes with computational cost, but that does not matter), which is needed for the forensics domain.

3 Related work

In the scientific community, classic audio matching (sometimes also referred to as *Content-Based Copy Detection* (CBCD)) has been addressed and solved mainly by the use and development of suitable audio fingerprints.

An audio fingerprint, as clarified by Bisio et al. [3], can be considered a “digital summary, deterministically generated from an audio signal, which can be used to identify an audio sample or quickly locate similar items in an audio database”. By extracting an audio fingerprint, one obtains a compact signal representation, and this process needs to be computationally simple in order to ensure time efficiency. Beyond that, a key characteristic of such fingerprints is that they need to be sensitive with respect to perceptual content modifications, while being robust against perceptually irrelevant processing.

Cano et al. in [4] performed a thorough review of audio fingerprinting, and proposed a corresponding general framework: After defining audio fingerprinting and clarifying its relationship to watermarking, the paper introduces the properties of audio fingerprinting through several requirements¹, emphasizing the related application-specific trade-offs between them. Moreover, the specific audio fingerprinting approaches of [1, 9, 10, 20] and several other state-of-the-art algorithms are compared in order to show the diversity of solutions within the aforementioned general framework.

¹i.e. accuracy, reliability, robustness, granularity, security, versatility, scalability, complexity and fragility being the main ones

The amount of publications related to audio matching is impressive, and still growing. Much of the related work aims at increasing the quality of CBCD by constantly improving the robustness of audio fingerprinting against additive noise and distortions, speed, tempo and frequency changes. Many of these algorithms [2, 11, 14, 19, 23, 24] have participated in TRECVID [21] workshops and achieved good results in the CBCD task. In one of the latest publications, Sonnleitner and Widmer [26] demonstrated that their algorithm can identify heavily distorted audio content with changes in speed, tempo and frequency $\pm 30\%$ with an SNR between 10 dB and 50 dB , while still supporting very fast queries. An interesting similarity among all aforementioned papers, is that the algorithms described by Haitsma [10] and Wang [27] are often used as baseline for benchmarks and evaluations, the latter being the approach that we will also use to compare our algorithm with.

Regarding the actual matching, the different implementations, just as fingerprint extraction mentioned earlier, strongly depend on the target application scenarios and related requirements. The matching approaches mentioned above are designed for “query by example” systems, where the fingerprint of a short excerpt of a content query is searched within a fingerprint database, as depicted at the beginning of our paper in Fig. 1. These algorithms typically retrieve the best matching database entry, or flag the input query excerpt as deriving by an unknown audio file. This paradigm has been widely adopted, and is often used, e.g., for retrieving music content. A key requirement of such algorithms is that they have to cope with queries being heavily distorted or corrupted by noise. Moreover, they need to ensure a high performance by minimizing false positives, with some of the state-of-the-art algorithms applying temporal alignment approaches [6, 11, 23, 24, 27].

A common strategy consists of comparing the query fingerprint with a reference one by building a so called “matching matrix” describing the frame-by-frame distance/similarity between the two inputs: A correct match between query and reference file can be visualized in this matrix as a unique diagonal line, which is identified by applying techniques derived from image processing such as Hough transform or RANSAC [8].

In order to temporally align matches, as stated in [5], “techniques like Expectation Maximization [15], RANSAC [8] or Dynamic Time Warping [6] are used”. Besides these computationally expensive approaches, there are also some more simple temporal alignment techniques. In the one presented by Wang et al. [27] the matching positions are peaks in the histogram of all possible offsets between query and reference files. The score of the match is equal to the number of matching points in the histogram peak. Two similar retrieval techniques are described in [23], where the number of matches is counted for every possible offset of a reference frame and its closest query fingerprint frame.

Even though these techniques represented a significant step forward, all of them rely on the assumption that a query consists of a *unique* excerpt from a *single* audio file. However, this may or may not be part of the dataset. To the best of our knowledge, no publications so far investigated the case of a query file composed by several *segments* of unknown length. In this scenario, the assumption of having, for instance, a *unique* relevant diagonal in the matching matrix does not hold. In a similar way, the presence of a *unique* optimal offset is not granted.

The only algorithm that came closer to addressing the requirements of audio segment matching was Dan Ellis’ Python implementation [7] of Wang’s algorithm [27]: In addition to the standard functionalities required by the classic CBCD, [7] provides the possibility of returning a pre-determined amount of best matches, each one also describing the *location* of the matched query/reference file portion in the files compared.

Although this does not address all of the mentioned requirements of partial matching (e.g. auto-detection of the amount of segments), this implementation could be used for a comparison of our approach with Wang’s algorithm [27].

4 Proposed method

Considering the relevant applications outlined in Section 2, our goal is to develop an approach that supports detection and accurate localization of partial matches, with the key requirements being:

- detect an arbitrary amount of partial matches (the existence and amount of which is unknown beforehand)
- accurately localize partial matches (detect exact starting time of every match and its duration, up to 2 seconds tolerance)
- allow for a tradeoff between scalability and time granularity

We propose a fingerprinting-based approach that can address these requirements, consisting of a fingerprint extraction algorithm and a retrieval algorithm. The distinctive feature of our approach lies within the retrieval algorithm, which is designed to detect and locate previously unknown matching segments of arbitrary size.

The retrieval algorithm could be used, in principle, also with other kind of fingerprints. By devising our own, however, we are able to be in full control of both the robustness to transformations, and the tradeoff between scalability and time granularity.

This approach, with its first configuration (Section 5.1), we have originally proposed in [17]. However, in the following we will show that the same approach with some parameters variations can be successfully used in different application scenarios.

4.1 Fingerprint extraction method

In order to extract an audio fingerprint F for every analyzed audio file a , we apply the process depicted in Fig. 11 that consist of the following steps:

1. Reduce the sampling frequency to 8kHz
2. Compute the short-time Fourier transform (STFT) $S(t, f)$ of a , using a Hamming window of length d_{stft} , with h_{stft} hop size
3. From the spectrogram $S(t, f)$ select M non-overlapping frequency bands in range 300 – 3000Hz with logarithmic spacing deriving $S_{log}(t, m)$, with $m \in \{1, \dots, M\}$
4. Split $S_{log}(t, m)$ in several sub-matrices, corresponding to d_{sm} ms of content (no overlap)
5. For every sub-matrix, select the 32 spectrogram peaks with the highest energy and map their locations in a binary matrix $S_{max}(t, m)$
6. Frame the binary matrix $S_{max}(t, m)$ into binary sub-matrices $S_l(t, m)$ corresponding to d_{smb} ms of content, using a hop size of h_{smb} ms. Where, $l \in \{1, \dots, L\}$ represents the index of the l -th binary sub-matrix, and L the total amount of binary sub-matrices
7. For every binary sub-matrix $S_l(t, m)$ from the previous step, compute its sum over time $V_l(m) = \sum_t S_l(t, m)$
8. The feature vector $V_l(m)$ describes d_{smb} ms of content and corresponds to one of the sub-matrices from step 7

Due to the h_{smb} ms hop, we have a final feature matrix $F(m, l)$ with $m \in \{1, \dots, M\}$ values every h_{smb} ms and $l \in \{1, \dots, L\}$ representing the index of the l -th feature vector where L is the amount of feature vectors.

We will further refer to F as the fingerprint of audio file a and to $V_l(m)$ as one of its sub-fingerprints.

Fig. 11 Flow chart of proposed fingerprint extraction algorithm

4.2 Retrieval algorithm

4.2.1 Comparison of fingerprints

The matching routine is based on a pairwise comparison of fingerprints: Two fingerprints F_a and F_b , extracted from audio files a and b , are compared sub-fingerprint-wise (sub-fingerprints being the final feature vectors $V_{la}(m)$ and $V_{lb}(m)$) which results in a distance matrix D_{ab} . This comparison is done via Euclidian distance.

We further apply threshold to the distance matrix D_{ab} in order to obtain the binary matching matrix D_{Bab} that represents the matching between two fingerprints F_a and F_b .

Two fingerprints F_a and F_b do not share the same dimension. They have different number of feature vectors (L_a and L_b) that depend on the length of audio file. Hence, the binary matching matrix D_{Bab} can have different number of rows and columns.

4.2.2 Retrieval of the matches

In the binary matching matrix D_{Bab} , we search for diagonals of 1's, and based on the diagonal position within this matching matrix, we retrieve start and end positions of the respective matching segments within F_a and F_b . We propose the following approach that takes as *input*: binary matrix D_{Bab} and gives as *output*: matching result sets:

1. **Propose diagonals:** Search for diagonal line patterns by keeping a track of consecutive matching frames in every possible (left to right) diagonal in a matching matrix. Parameter: *maxGap1* gap defines the maximum allowed number of non-matching frames on

tracked diagonal between two matches while parameter *minLength1* defines the shortest consecutive track that could be marked as diagonal proposal and sent for further processing. Figure 12 shows binary matching matrix D_{Bab} with five proposed diagonals (between points AB, CD, EF, GH and IJ). Length of all diagonals have to exceed *minLength1* and in case that consecutive track of non-matching frames exists, like the one on the AB diagonal, it has to be shorter than *maxGap1*. Number of non-matching frames between points F and G was longer than *maxGap1*, hence (even if they share the same offset) matches between EF and GH are proposed as two separate diagonals.

2. **Join diagonals:** Based on parameters *minLength2* and *maxGap2*, decide whether two short diagonals with the same offset will be joined or represented as two separated matches. In the Fig. 12 two separately proposed diagonals that have the same offset (lying on the same matrix diagonal) EF and GH and will be presented as one consecutive match, eg. EH, since:
 - the gap in between (amount of non-matching frames) is lower than defined parameter *maxGap2*
 - and the length of the merged diagonal EH exceeds *minLength2*
3. **Remove dominated diagonals:** Select the longest diagonal from the ones marked as a match in a previous step, and discard all the other diagonals that match the same content from the perspective of both matching files (dominated diagonals). In the Fig. 12 red colored diagonals CD and IJ are dominated by the longest detected diagonal in that area EH and will therefore be removed in this step.
4. **Noise and silence flagging:** Detect noise and silence matching patterns and, depending on the configuration, exclude them from further processing. This detection can be performed on the matching matrix by identifying the squared patterns appearing whenever

Fig. 12 Search for diagonals in the binary matching matrix

silence or noise are present, as well as by using an external tool. The specific choice is not relevant, but for the sake of efficiency we selected the first option.

5. **Determine the matching result sets** based on the remaining diagonal coordinates: $\{start_a, start_b, duration, confidence, a, b\}$ Where $start_a$ is the start of matching excerpt in audio a , $start_b$ start of the match in audio b and $confidence$ is a share of matching sub-fingerprints over the duration ($duration$) of detected match. In the Fig. 12, we finally select two remaining diagonals AB and EH and form our matching result set that will in this case contain two partial matches.

5 Evaluation

In Section 2 we have described various application scenarios that demand detection and localization of partial audio matches. As summarized in 2.5, in order to address all described application scenarios we define two different sets of requirements, aiming at either lowering computational cost (via decreased time granularity) or high time granularity.

Consequently, the two different requirement sets lead to two different settings of our general fingerprinting method from Section 4.1, which we present in next two sub-sections. Within each sub-section, our partial matching system with corresponding fingerprinting setting is going to be evaluated using a proper test set, designed to resemble as closely as possible the real-life environments.

5.1 Algorithm settings for high amount of data

5.1.1 Algorithm set up

The compactness of audio fingerprint used with our retrieval algorithm is directly influencing the time complexity of the matching system. Hence, we designed the audio fingerprint to be as compact as possible and reduce the computational burden of matching, as would happen within all the application domains mentioned in Sections 2.1, 2.2 and 2.3;

- length of STFT window: $d_{stft} = 40ms$
- hop size for the STFT $h_{stft} = 10ms$
- Number of frequency bands $M = 15$
- Amount of content withing sub-matrices (step 4.): $d_{sm} = 60ms$
- Amount of audio content per binary sub-matrix (step 6.): $d_{smb} = 630ms$
- Hop size for binary sub-matrices (step 6.) $h_{smb} = 120ms$

A big hop size (120 ms) between final features assures a low frame rate, which speeds up our matching process. This is important, since all sub-fingerprints of the first audio file a need to be matched with all sub-fingerprints of the second audio file b . That's how the number of sub-fingerprints directly influences the size of the matching matrix that will be processed with our retrieval algorithm, and processing time grows with the matrix size.

5.1.2 Test set up

The test set for the evaluation of our approach contains 6000 test files created by applying one of the *editing actions* on the audio content taken from *reference* and *non-reference* datasets. The *reference* dataset contains content from radio and TV programs including speech and music (≈ 50 hours of material in total), while the *non-reference* dataset contains

speech and music files that do not appear in the reference dataset. In order to recreate all possible editing cases as depicted in Fig. 3, we create every test file by applying *one* of the following editing actions:

- **cutting (removal)**: one continuous excerpt from a reference file where 2 seconds are cut from the middle
- **pasting (insertion)**: one continuous excerpt from a reference file where 2 seconds of non-reference content are pasted in the middle
- **cut and paste (replacement)**: one continuous excerpt from a reference file where 2 seconds are cut from the middle and 2 seconds of non-reference content are pasted instead
- **splicing**: a random number (2 to 5) of audio excerpts from the same or different reference files are spliced together

After the operations of **removal**, **insertion** and **replacement** each test file contains two audio excerpts to be matched against the reference content, where each excerpt can have a variable length of 10, 5 or 3 seconds. For the operation of splicing, audio excerpts of 10, 5 or 3 seconds have been extracted from the reference files, and then concatenated to create one test file. The total number of test files is 500 for every length of every editing type, resulting in $500 \times 4 \times 3 = 6000$ files that include 14250 partial matches to be detected and localized.

For the *cutting*, *pasting* and *cut and paste* editing types, the content excerpt length of 2 seconds is selected arbitrarily. Changing this parameter is not influencing the performance of the algorithm, as long as it is $\geq \text{maxGap2}$.

After the test set has been created, every test file underwent up to two randomly selected transformations of encoding with $MP3\{128, 192, 320\text{ kbit/s}\}$ or $AAC\{128, 192, 320\text{ kbit/s}\}$ and volume change between 0.5 up to the maximum possible before clipping occurred. This is done in order to simulate usual transformations of content that happen during production process (considering application scenarios that do not require robustness against heavy degradation of content quality) and also to carry out a study on the impact of this transformations to detection process.

5.1.3 Evaluation metrics

Every test file was matched against all the other files in the reference dataset, and all the reported partial matches were then evaluated and classified as: True Positives (TP) or False Positives (FP). If no match was retrieved for the segment of a test file with a match within the reference dataset, the number of False Negatives (FN) was increased by one. TP, FP and FN are defined as follows:

- True Positives (TP) are retrieved matching segments that have correct start and end location within the test file (with defined tolerance τ) and are found in the correct reference audio file also with the correct start and end location
- False Positives (FP) are the retrieved matching segments that do not fulfill all mentioned conditions for it to be a True Positive, namely: correct location within a test file, correct reference file and correct location within a reference file
- False Negatives (FN) are not retrieved matching segments that according to ground truth have their matches in reference dataset

For the allowed tolerance values of start/end positions of retrieved matches $\tau \in \{\pm 0.5, 0.6, 0.7, 0.8, \dots 2\}$ in seconds, recall and precision were calculated accordingly.

5.1.4 Results and state-of-the-art comparison

In Section 3, we presented a brief overview of the state-of-the-art in audio matching and came to the conclusion that an approach capable of adequately addressing the target application scenarios does not exist. We therefore selected an algorithm that offers the closest solution to our problem and includes temporal alignment, which is described in [27]. To compare our results we use the python implementation by Dan Ellis [7].

In order to use Ellis's implementation for our purposes, we used the supported option for reporting the position of every match. We ask for the five best matches to be returned, as this is the maximum number of matching segments we expect. We also used the supported option for a preciser match counting. In classic matching, these options are less relevant, since the goal is usually to only detect one reference file from the database, based on one best match. Hence, the algorithm requires, as an input from the user, a number of best matches to be retrieved. This information is, by definition, unknown in our target applications, but for the sake of evaluation, we provided it here.

Figure 13 shows the results of the proposed algorithm and Ellis's implementation of Wang's algorithm [27], which were obtained using the datasets and the evaluation metrics described in Sections 5.1.2 and 5.1.3, respectively. Our proposed algorithm has achieved significantly better results: For an allowed start/end segment tolerance of $\pm 1.7s$, the proposed algorithm achieved an average precision of 0.88 and an average recall of 0.94, while the state-of-the-art approach at the same tolerance achieved 0.42 average precision and 0.47 average recall.

In Figs. 14 and 15 precision and recall are presented separately for every editing action done upon audio content: removal, splicing, replacement and insertion. Our proposed approach performs equally well for the editing actions of removal, splicing and insertion, where it detects correctly most of the partial matches (precision and recall above 0.9) already at start/end tolerance of 0.8. For the action of replacement, localization of exact start/end

Fig. 13 Recall on the left and Precision on the right: full line presents results of proposed algorithm and dashed line results of Dan Ellis's Shazam implementation

Fig. 14 Precision: more detailed presentation of results per different editing action, full line presents results of proposed algorithm and dashed line results of Dan Ellis's Shazam implementation

points of partial matches becomes a bit more demanding and our approach achieves 0.9 precision and recall at start/end tolerance of 1.7. The state-of-the-art approach performed slightly better for the editing action of removal than for the splicing and insertion, while for the action of replacement it delivered very unsatisfying results.

A lot of music pieces have repeated parts or even consist of bits that keep unchanged for longer periods. During the creation of test files, we randomly select excerpts of reference files. We store only the location, where the excerpt was taken from, in ground truth, without knowing if the same part/sequence was repeated elsewhere. This made the accurate localization of segments very challenging, and explains the higher number of false positives that the matching algorithms returned in our evaluation set up.

5.2 Algorithm settings for fine time granularity

5.2.1 Algorithm set up

With the settings presented in the previous subsection we can trigger positive matches after few seconds of replicated content. This coarse granularity is insufficient for application scenarios in domain of multimedia forensics where we are interested in detecting replicas lasting only a fraction of a second: A simple “yes”, copy-pasted adequately, can entirely

Fig. 15 Recall: more detailed presentation of results per different editing action, full line presents results of proposed algorithm and dashed line results of Dan Ellis's Shazam implementation

change the meaning of a sentence or conversation upside down. Therefore, we introduce next settings to achieving a better time resolution:

- length of STFT window: $d_{stft} = 40ms$
- hop size for the STFT $h_{stft} = 4ms$
- Number of frequency bands $M = 30$
- Amount of content withing sub-matrices (step 4.): $d_{sm} = 12ms$
- Amount of audio content per binary sub-matrix (step 6.): $d_{smb} = 108ms$
- Hop size for binary sub-matrices (step 6.) $h_{smb} = 12ms$

This improved fingerprint granularity comes with the higher computational cost which is absolutely acceptable for applications within forensics domain where only one or mostly few audio files have to be analyzed. Moreover, in case of copy-move forgery detection where we compare a single audio file against itself, we have a symmetrical matching matrix and only the half of it will be analyzed with the retrieval algorithm presented in Section 4.2.

5.2.2 Test set up

The dataset we employed addressed the use case of copy-move forgery detection. As a basis for generation of the tampered dataset we use Free Spoken Digit Dataset (FSDD) by

Jackson et al. [13]. FSDD is a dataset of a spoken digits by a four different speakers. The recordings of sentences with digits zero to nine are done in fifty different takes for every of four speakers. After recording, digits are split in such a way that no leading nor trailing silence is present. Finally, 50 audio files for each of 10 digit and from every of 4 speakers, sum up in total 2000 recordings.

Starting from this dataset we generate full sentences with 10 digits, among which 2 digits are exact copies of each other, and the 8 other ones are all different. Such sentences are good examples of copy-move forgeries with available ground truth, which makes them suitable for our experiment. On this way for every of 50 takes per person we select randomly five digits to be copied and five to be replaced. That's how from every take per person five tampered audio files are created. In the end we have 1000 of tampered audio files in dataset T .

In this application scenario we should expect that forger would add a slight noise over whole tampered recording in order to cover possible processing traces. Hence, we add white noise with SNR of 50, 40 and 30 dB to all audio files in dataset T and created 3 more datasets T_{SNR50} , T_{SNR40} and T_{SNR30} .

Furthermore it is to expect that a forger could intentionally add environmental noise only to a duplicate, to better adapt it to the part of the track where it is copied to. Likewise, a forger could slightly adjust the noise threshold of the copied segment to make it less similar to its original version. Hence, we add to our experimental set up three more dataset $T_{segSNR50}$, $T_{segSNR40}$ and $T_{segSNR30}$ where the source digit for the copy is unaltered, and its replica is corrupted by white noise with target SNR of 50, 40 and 30 dB respectively. The overview of tampered datasets used in our experiment can be find in Table 1.

5.2.3 Evaluation metrics and results

Every test file was analyzed looking for traces of copy-move forgery, resulting in retrieved matching excerpts or in an empty list. The first case corresponds to True Positive or one or more False Positives while the second case corresponds to False Negative. Definition of what TP, FP and FN are is the same as in 5.1.3. In this case, however, reference and test file are always the same audio file and the True Positives that we want to detect are the source digits of copy move operations and their replicas. Furthermore, the allowed tolerance for start/end positions of retrieved matches τ is 0.15 seconds.

Table 1 Tampering detection: Datasets

Label	Description
T	1000 copy-move forgeries generated out of the 2000 recordings in FSDD [13]
T_{SNR50}	Recordings from T corrupted by AWN with a target SNR of 50dB
T_{SNR40}	Recordings from T corrupted by AWN with a target SNR of 40dB
T_{SNR30}	Recordings from T corrupted by AWN with a target SNR of 30dB
$T_{segSNR50}$	1000 copy-move forgeries generated out of FSDD, in which the source digit for the copy is unaltered, and its replica is corrupted by AWN with target SNR of 50dB
$T_{segSNR40}$	Same as $T_{segSNR50}$, but with SNR=40dB
$T_{segSNR30}$	Same as $T_{segSNR50}$, but with SNR=30dB

Table 2 Tampering detection: Results

Dataset	Statistics (#)		
	Precision	Recall	FNR
T	0.978	0.981	0.018
T_{SNR50}	0.979	0.982	0.018
T_{SNR40}	0.977	0.980	0.020
T_{SNR30}	0.935	0.942	0.058
$T_{SNRseg50}$	0.977	0.980	0.020
$T_{SNRseg40}$	0.973	0.986	0.014
$T_{SNRseg30}$	0.955	0.969	0.031

From collected number of TP, FP and FN precision, recall and false negative rate are calculated accordingly and presented in Table 2.

The algorithm achieves very high performance over the clean dataset tampered with copy-move forgery T . When additive white noise is added with constant characteristics on the complete file, precision and recall are stable for a SNR of 50dB, and drop only slightly for SNRs of 40dB and 30dB. More importantly, the copy-move forgeries are also detected in case of mismatched background noises, i.e. on the datasets $T_{segSNR50}$, $T_{segSNR40}$ and $T_{segSNR30}$.

Hence, the algorithm can be considered robust for all conditions relevant for scenario of tampering via copy-move forgery. Inferring that with slight changes in our fingerprinting approach we were able to address one more application domain with a specific time granularity requirement and at the same time achieve exceptionally good results.

To the best of our knowledge, other approaches for copy-move forgery detection as [12, 16, 25] require a pre-segmentation of the content in words or syllables: They are unable to locate the copied segments, and can only confirm if two suspicious time intervals are copies of each other or not, assuming that the pre-segmentation is correct.

Any comparison would thus have been unfair, since these algorithms are inherently performing a different task than the one they were devised for, i.e., match/non-match discrimination in spite of localization. Moreover, the segmentation methods used by the authors in their experiments could not be replicated, and this issue would have added a further negative bias to their effectiveness.

6 Conclusion and future work

In this paper, we have introduced several application scenarios within four application domains. All of them require detection and accurate localization of unknown partial audio matches within files or datasets, and they cannot be adequately addressed with state-of-the-art audio matching approaches.

We proposed a new method for this purpose, which is based on fingerprinting and a suitable matching algorithm, it provides two distinct variants / settings to address the needs of two very different requirement sets:

- Variant 1 addresses the majority of applications domains (archive management, broadcast/stream analysis, and media search), where potentially large datasets are to be

analyzed, and computational efficiency plays an important role. As evident from Section 5.1, this variant performed significantly better than the Dan Ellis's implementation [7], one of the most cited query-based audio matching approaches [27], especially with respect to localization, replacement (cut & paste) editing action, and auto-detection of the amount of partial matches.

- Variant 2 addresses the media forensics domain, where fine-grain analysis down to fractions of seconds is a key requirement. As evident from Section 5.2, our proposed method achieves very good performance regarding detection and localization of copy-move forgery, regardless of whether noise was added over the whole audio track or only over the duplicated audio excerpt.

In summary, we have shown that by using modified settings, our approach can be successfully applied for various applications scenarios, including in the media forensics domain.

As for future work, we plan to address requirements regarding robustness against noise, which depending on the circumstances, can be relevant for some of the described application scenarios. This goes especially for the case of mixing, for which we plan to combine our proposal matching algorithm with other fingerprinting formats / methods. Furthermore, we plan to further improve and integrate audio phylogeny analysis, thereby extending e.g. [18], in order to get a more complete picture not only of content reuse, but also of its processing history. Lastly, we are going to explore further possibilities to improve the computational efficiency of partial matches detection, e.g. by using techniques such as Locality-sensitive hashing (LSH).

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Allamanche E, Herre J, Hellmuth O, Fröba B (2001) Content-based identification of audio material using mpeg-7 low level description. In: International symposium on music information retrieval (ISMIR), Bloomington, IN, USA, pp 197–204
2. Anguera X, Garzon A, Adamek T (2012) Mask: Robust local features for audio fingerprinting. In: IEEE international conference on multimedia and expo (ICME), Melbourne, Australia, pp 455–460
3. Bisio I, Delfino A, Lavagetto F, Marchese M (2015) A television channel real-time detector using smartphones. *IEEE Trans Mob Comput* 14(1):14–27
4. Cano P, Battle E, Kalker T, Haitsma J (2005) A review of audio fingerprinting. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology* 41(3):271–284
5. Chandrasekhar V, Sharifi M, Ross DA (2011) Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications. In: ISMIR international conference on music information retrieval, Miami, FL, USA, pp 801–806
6. Covell M, Baluja S Known-audio detection using waveprint: spectrogram fingerprinting by wavelet hashing. In: IEEE international conference on acoustics, speech, and signal processing, ICASSP, Honolulu, HI, USA, pp I-237–I-240

7. Ellis D (2014) Audfprint: Landmark-based audio fingerprinting. <https://github.com/dpwe/audfprint>
8. Fischler MA, Bolles RC (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun ACM* 24(6):381–395
9. Gomez E, Cano P, Gomes LCT, Batlle E, Bonnet M (2002) Mixed watermarking-fingerprinting approach for integrity verification of audio recordings. In: *Proceedings of the international telecommunications symposium*, Natal, Brasil
10. Haitsma J, Kalker T (2002) A highly robust audio fingerprinting system. In: *ISMIR International conference on music information retrieval*, Paris, France, pp 107–115
11. Héritier M, Gupta V, Gagnon L, Boulianne G, Foucher PCS (2009) Crim ' s content-based copy detection system for trecvid. In: *NIST TREC video retrieval evaluation (trecvid) conference*, Gaithersburg, MD, USA
12. Imran M, Ali Z, Bakhsh ST, Akram S (2017) Blind detection of copy-move forgery in digital audio forensics. *IEEE Access* 5:12843–12855
13. Jackson Z, César S, Flaks J, Pan Y, Nicolas H, Thite A (2018) Free spoken digit dataset (FSDD) v1.0.8. <https://github.com/Jakobovski/free-spoken-digit-dataset/tree/v1.0.8>
14. Jégou H, Delhumeau J, Yuan J, Gravier G, Gros P (2012) BABAZ: A large scale audio search system for video copy detection. In: *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Kyoto, Japan, pp 2369–2372
15. Ke Y, Hoiem D, Sukthankar R (2005) Computer vision for music identification. In: *IEEE computer society conference on computer vision and pattern recognition (CVPR)*, San Diego, CA, USA, pp 597–604
16. Liu Z, Lu W (2017) Fast copy-move detection of digital audio. In: *IEEE international conference on data science in cyberspace (DSC)*, Shenzhen, China, pp 625–629
17. Maksimovic M, Aichroth P, Cuccovillo L (2018) Detection and localization of partial audio matches. In: *International conference on content-based multimedia indexing (CBMI)*, La Rochelle, France, pp 1–6
18. Maksimovic M, Cuccovillo L, Aichroth P (2017) Phylogeny analysis for MP3 and AAC coding transformations. In: *IEEE international conference on multimedia and expo (ICME)*, Hong Kong, China, pp 1165–1170
19. Malekesmaeili M, Ward RK (2012) A novel local audio fingerprinting algorithm. In: *IEEE international workshop on multimedia signal processing (MMSP)*, Banff, AB, Canada, pp 136–140
20. Mihçak MK, Venkatesan R (2001) A perceptual audio hashing algorithm: a tool for robust audio identification and information hiding. In: *ACM international workshop on information hiding (IHW)*, Pittsburgh, PA, USA, pp 51–65
21. NIST (2020) Trec video retrieval evaluation: TRECVID. <https://www-nlpir.nist.gov/projects/trecvid/>
22. Nucci M, Tagliasacchi M, Tubaro S (2013) A phylogenetic analysis of near-duplicate audio tracks. In: *IEEE international workshop on multimedia signal processing (MMSP)*, Pula, Italy, pp 99–104
23. Ouali C, Dumouchel P, Gupta V (2014) A robust audio fingerprinting method for content-based copy detection. In: *International workshop on content-based multimedia indexing (CBMI)*, Klagenfurt, Austria, pp 1–6
24. Ouali C, Dumouchel P, Gupta V (2016) Fast audio fingerprinting system using GPU and a clustering-based technique. *IEEE/ACM Trans Audio, Speech & Language Processing* 24(6):1106–1118
25. Qi Y, Yang R, Huang J (2015) Copy-move detection of audio recording with pitch similarity. In: *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, South Brisbane, QLD, Australia, pp 1782–1786
26. Sonnleitner R, Widmer G (2016) Robust quad-based audio fingerprinting. *IEEE/ACM Trans Audio, Speech & Language Processing* 24(3):409–421
27. Wang A (2003) An industrial strength audio search algorithm. In: *ISMIR international conference on music information retrieval*, Washington, D.C., USA

Milica Maksimović has received her M.Sc. in Electrical Engineering and Computer Science at the University of Novi Sad (Serbia). She is currently pursuing her PhD at Ilmenau University of Technology (Germany). The main focus of her research are methods for the detection of manipulation and re-usage of audio-visual content. Since 2014, she is working as a researcher at Fraunhofer Institute for Digital Media Technology in the research group media distribution and security.

Patrick Aichroth worked as an IT freelancer and software developer, before becoming a researcher at Fraunhofer IDMT in 2003. Since 2006, he is head of the media distribution and security research group at Fraunhofer IDMT. He has been involved in many industrial and publicly funded research projects, and is especially interested in applying technology to solve trust and information overload problems, and privacy/security issues for media applications.

Luca Cuccovillo has received his M.Sc. in Computer Science and Engineering at the Politecnico di Milano (Italy). He is currently pursuing his PhD at Ilmenau University of Technology (Germany), on the topic of audio signal analysis for multimodal tampering detection. He is working as a researcher at Fraunhofer IDMT since 2013, in the research group of media distribution and security.