UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Criação de uma ferramenta para geração de impressões digitais de áudio

Rui Miguel Silva Santos

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Nuno Garcia
Coorientador: Prof. Doutor Bart van Rumste
Coorientador: Prof. Doutor Nuno Pombo

Covilhã, outubro de 2018

Dedicatória

Pai e Mãe, duas palavras tão pequenas mas com tanto significado e com enorme valor. Esta dissertação é dedicada a vocês. Pela forma que sempre acreditaram que era capaz e por terem estado ao meu lado, até aos dias de hoje, sempre que precisei. Obrigado por me terem dado esta oportunidade e pelos vossos sacrifícios. Este trabalho é vosso.

Agradecimentos

Eis que se está a dar por terminada mais uma etapa da minha vida e mais um objetivo cumprido. Foi um caminho bastante longo até aqui chegar e sozinho não era capaz de o percorrer.
Quero então agradecer aos meus pais, por todo o esforço que vocês tiveram de fazer para que o vosso filho pudesse percorrer um dos seus sonhos e ter meios para obter uma vida melhor e mais folgada, coisa que nem sempre foi possível para vocês. Obrigado por me terem dado o que vocês queriam e não puderam receber.
Agradecer a si, ”mãezinha”. Uma avó que foi sempre como uma segunda mãe. As suas experiências e valores transmitidos muito me ajudaram a ser o que sou hoje e foram uma parte importante para aqui chegar.
A ti, Beatriz por apesar de nem sempre nos entendermos da melhor forma, isso permitiu que eu crescesse e me tornasse melhor pessoa. Um obrigado pela força e pelos bons momentos que passámos.
Mariana, para ti, vai um enorme agradecimento. Um enorme agradecimento por me teres apoiado em todas as minhas decisões, por me teres sabido incutir o caminho que deveria percorrer. A tua capacidade de me incentivar a percorrer este caminho, mesmo nas etapas mais difíceis, é de louvar. Sabes bem o quão importante foste até aqui e sei que este caminho foi bastante mais fácil por estares ao meu lado. Obrigado por tudo e por seres quem és.
Miguel e Anita, por diferentes razões, foram bastante importantes para conseguir aqui chegar. Graças a vocês aprendi muito e muitos valores me transmitiram. Estiveram lá quando mais precisei e muito do que sou hoje deve-se ao vosso companheirismo e o que me ensinaram.
Agradecer à restante família por me terem acompanhado neste percurso e pela força que me deram para continuar.
Aos meus orientadores, ao Ivan Pires e ao André Pinho quero agradecer por me terem ajudado quando mais precisei, por me terem transmitido alguns dos vossos conhecimentos e pela vossa disponibilidade. Realçar, ainda, o professor Nuno Garcia por ser a pessoa que é e por termos conseguido ter uma boa relação graças à sua simplicidade, disponibilidade em ajudar os outros e a sua boa disposição. A todos, o meu obrigado.
Um obrigado ao meus amigos da Covilhã que me proporcionaram bons momentos e me possibilitaram, uma boa adaptação a esta cidade. Muito vos devo por todos os esforços que fizeram por mim.
Um obrigado ao NINF, à minha comissão de latada e aos meus caloiros por me terem dado a conhecer o lado associativo, por me terem dado a possibilidade de adquirir novos conhecimentos e por todos os momentos que passei convosco.
Aos meus colegas de curso que trabalharam comigo nestes anos e, em especial, aqueles que trabalharam comigo nesta investigação, um obrigado por tornarem os momentos de trabalho em momentos de boa troca de conhecimentos e que o trabalho fosse realizado mais ”facilmente”.
Aos meus amigos de toda a vida porque, apesar da distância, termos mantido esta boa amizade e por terem feito o esforço de quando ia ”aí a cima” estarmos juntos e me terem feito passar tão bons momentos. Vocês sabem o que significam.
Obrigado a todos por tornarem isto possível e por terem acreditado em mim.

Resumo

A utilização de aplicações com o intuito de realizar um reconhecimento de áudio tem crescido nos últimos tempos. Uma das aplicações mais reconhecidas, neste momento, é o *Shazam* que tem como objetivo o reconhecimento de músicas. Nesta é possível, ao realizar uma captura de áudio ficar-se a saber qual o nome da banda e da música em questão. Assim sendo, neste projeto são apresentados dois métodos capazes de construir um *audio fingerprint* e a sua classificação. Aqui, serão descritos todos os procedimentos realizados bem como realizada uma discussão dos testes e resultados obtidos pelo sistema desenvolvido. Está presente uma comparação relativamente ao desempenho observado pelos dois métodos desenvolvidos. Adicionalmente, é realizado um estudo relativo ao estado desta área.

Palavras-chave

Áudio, *Audio Fingerprint*, Amostra, Base de dados, *Bit Error Rate*, Captura, Característica, Classificação, Frequência, *Forward Fourier Transform*, *Hamming Window*, *Hanning Window*, *K-Nearest Neighbors*, *Mel Frequency Cepstral Coefficient*, Picos, Recolha, Rede Neuronal, Resultados, Ruído, Segmentação, Variância, *Windowing*.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
BER	Bit Error Rate
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Mode
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
IDFT	Inverse Discrete Fourier Transform
KNN	K-Nearest Neighbor
MFCC	Mel-Frequency Cepstral Coefficient
MVC	Model View Controller
SGBD	Sistema de Gestão de Base de Dados
SO	Sistema Operativo
SQL	Structured Query Language
STFT	Short Time Fourier Transform
UBI	Universidade da Beira Interior
XML	Xtensible Markup Language

Capítulo 1

Introdução

Nos dias de hoje temos verificado um enorme crescimento da utilização de dispositivos móveis e aproveitamento das suas capacidades. É sabido que existem alguns tipos de aplicações capazes de resolver dados problemas existentes no dia a dia das pessoas, sendo eles, por exemplo, dedicados à saúde, entretenimento ou para o dia a dia. O facto de ser possível a construção e existência deste tipo de aplicações muito se deve à existência de uma vasta gama de sensores incorporados nos dispositivos e a facilidade com que é possível conciliar o *hardware* com as funcionalidades existentes. Neste caso e, visto que o objetivo desta aplicação centra-se na análise de áudio, utilizar-se-á o microfone presente nos *smartphones*. Apesar de, neste momento, se verificar um enorme foco na construção de aplicações para dispositivos móveis é necessário referir que existe ainda a possibilidade de serem desenvolvidas diversas funcionalidades, com recurso ao som, para outro tipo de dispositivos.

É possível verificar uma crescente utilização de áudio para o desenvolvimento de diversos tipos de aplicações e a sua utilização tornou-se essencial para diferentes áreas. Algo como a realização de um diagnóstico ao estado do coração, pode ser possível através da construção de um sistema capaz de realizar uma análise ao batimento cardíaco do mesmo. Além deste exemplo existem diferentes funcionalidades capazes de serem desenvolvidas através da utilização do áudio. Uma *app* desenvolvida recentemente, com a designação de *Shazam*, tornou possível a execução de identificação de músicas.

O sucesso ou não de um método desenvolvido está muito dependente do processo de tratamento dos dados e extração de características dos mesmo. A este processo, é comummente designado pela construção de um *audio fingerprint*. Um *Audio Fingerprint* pode ser definido como sendo um resumo de um dado ficheiro de áudio. Este resumo é construido através da extração de características do mesmo. A junção das características forma um *Audio Fingerprint* sendo este único pois, por exemplo, a existência de duas músicas bastante semelhantes vai gerar *Audio Fingerprints* diferentes devido às características extraídas das mesmas serem distintas. Após a extração da ”impressão digital do áudio” torna-se possível a localização de ficheiros de áudio semelhantes ou a pesquisa de um trecho de áudio numa base de dados.

De forma a serem obtidos resultados satisfatórios torna-se essencial a construção do *audio fingerprint* de forma eficiente e, para esse fim, existem diferentes métodos aplicados para conseguir esse fim. De referir que na tecnologia anteriormente referenciada (Shazam), o método aplicado teve por base os dados obtidos por gráficos com domínio de frequência, o que nem sempre se verifica. No entanto, a maioria das técnicas utilizadas para a análise de áudio incidem sobre o tratamento das frequências da onda do som e, a partir daí, processa-se à execução da extração de características. Assim sendo, é necessário um estudo dos métodos existentes e que tipo de características podem ser retiradas para que seja possível, no fim, melhores condições para uma classificação e, consequentemente, melhores resultados obtidos. Por fim, referir que a realização de análise de áudio é algo bastante recente e ainda com, relativamente, poucos estudos desenvolvidos.

1.1 Objetivo

O principal objetivo deste projeto é o desenvolvimento de uma aplicação móvel que seja capaz de realizar identificação de áudio. A identificação de áudio será aplicada ao reconhecimento do ambiente em que o dispositivo se encontra.
Os principais objetivos deste projeto são os seguintes:

- Revisão de literatura sobre construção de um *audio fingerprint*;
- Estudo e análise de aplicações semelhantes;
- Estudo de técnicas de construção de um *audio fingerprint* e a sua classificação;
- Desenvolvimento de uma aplicação capaz de realizar a identificação do ambiente envolvido e no menor tempo possível.

1.2 Motivação

O propósito deste trabalho surge do interesse de um aluno da UBI (Universidade da Beira Interior) na construção de uma aplicação que seja capaz de realizar uma identificação do local em que o portador do dispositivo se encontra. Para tal, tornava-se essencial a realização de uma identificação do áudio do ambiente em que este se encontra envolvido. Neste trabalho deve, ainda, ser integrada a realização de análise de outros sensores disponibilizados pelos dispositivos móveis como o acelerómetro, magnetómetro e girosópio. Nos casos em que for possível utilizar-se-á, ainda, a recolha de dados Wi-Fi a que se encontra ligado. Por fim, referir que esta aplicação tem como motivação o desenvolvimento de um estudo cientifico.
A nível pessoal, este projeto despertou-me grande interesse por ser um tema bastante inovador, daí ser um grande desafio. A possibilidade de fazer algo inovador e diferente, adicionando a ser uma área que me permite adquirir novos conhecimentos e competências foram fundamentais para a escolha incidir neste projeto. Por fim, a possibilidade de poder efetuar uma aplicação para dispositivos móveis e trabalhar numa área da inteligência artificial, foram fundamentais para a minha escolha.

1.3 Estudos desenvolvidos

A área de geração de *audio fingerprints* é um processo que engloba diferentes passos para a sua conclusão. Algumas técnicas foram apresentadas para o seu desenvolvimento. O facto de serem aplicados diferentes métodos, não implica que sejam obtidos bons resultados. Assim sendo, foi realizada uma revisão sistemática de forma a ter uma percepção de que processos são realizados e qual a sua eficácia. Este estudo já publicado, pode ser visualizado e analisado na secção dos anexos A. Este estudo foi realizado com o intuito de perceber que resultados foram obtidos, nas mais diferentes técnicas, para se ter uma percepção de que escolhas se deveriam realizar no futuro.

1.4 Organização

A presente dissertação está organizada em vários capítulos que mostram, de forma sequencial, o processo de investigação desenvolvido. Neste primeiro capitulo foi descrito o projeto e apresentada uma breve definição do termo *audio fingerprint*, a sua motivação e seus objetivos. Os seguintes capítulos encontram-se organizados da seguinte forma:

- **Capitulo 2 - Estado da Arte:** Neste capítulo são apresentadas tecnologias já desenvolvidas e que serão utilizadas ao longo da presente dissertação. Inicialmente é realizada uma breve contextualização do estado atual no desenvolvimento de aplicação que têm por base a utilização do áudio. Em seguida, são apresentados diversos métodos, bem como a sua definição, já utilizados para as diferentes etapas necessárias para o desenvolvimento deste projeto. Por fim, são ainda apresentados, de forma sintetizada, alguns estudos já realizados e, mais à frente, são descritos alguns dos estudos mais relevantes.
- **Capitulo 3 - Tecnologias e Ferramentas Utilizadas:** Neste capítulo são apresentadas todas as tecnologias e ferramentas utilizadas para a construção e desenvolvimento da dissertação.
- **Capitulo 4 - Engenharia de Software:** Neste capítulo é discutida a Engenharia de Software para o desenvolvimento desta dissertação. São apresentados diversos diagramas, de forma a existir uma melhor compreensão, bem como uma análise dos requisitos do sistema.
- **Capitulo 5 - Implementação:** Este capítulo tem como objetivo a discussão de detalhes de implementação no desenvolvimento da dissertação.
- **Capitulo 6 - Testes e Resultados:** O intuito deste capítulo tem como objetivo a apresentação de uma discussão dos resultados obtidos após o desenvolvimento do projeto proposto. São apresentados alguns testes realizados do sistema desenvolvido.
- **Capitulo 7 - Conclusão e Trabalho Futuro:** O último capítulo refere-se às conclusões obtidas com a execução do trabalho desenvolvido e apresenta que funcionalidades podem ser integradas ou aspetos que devem ser alterados de forma a complementar o sistema desenvolvido ao longo da presente dissertação.

Capítulo 2

Estado da arte

2.1 Introdução

Nos dias de hoje tem-se verificado uma enorme aposta no estudo e desenvolvimento de aplicações que têm por base a análise do áudio. Este tipo de aplicações permite desenvolver diferentes *apps* com outros fins. Um dos sistemas construídos para a identificação de músicas foi o *Shazam*. Esta *app* obteve um enorme sucesso e é frequentemente utilizada pelos utilizadores.
Além de aplicações como o exemplo já referido anteriormente, (*Shazam*), é possível, ainda analisar se um coração tem um batimento saudável ou não, através da captura do seu batimento. Com isto, é pode concluir-se que através da recolha e análise do som é possível construir uma enorme panóplia de funcionalidades com diferentes fins como direcionadas para a saúde, por exemplo.

- **Shazam**: Esta aplicação para *smartphones* tem como objetivo a identificação de música ao utilizar o microfone incorporado em todos os *smartphones* para gravar uma amostra de mesma. Após este processo é criado um *audio fingerprint* e este é comparado aos armazenados numa base de dados existente. Caso a música seja reconhecida, é enviado para o utilizador vários dados relativos à mesma, como titulo e autor. Por fim, de referir que esta tecnologia, de forma a construir as impressões digitais, utiliza métodos que se baseiam em dados com o domínio de frequência.

Devido a estes factos, foi realizado um estudo relativo a este tema de forma a compreender que técnicas são utilizadas para os diferentes processos e que quais são efetuados até ser realizada a classificação da recolha de áudio. Rapidamente se percebe que é um processo complexo e com vários passos aplicados. É bastante importante, desde logo, a realização de um bom tratamento das ondas de áudio para que seja possível uma boa formação da mesma e uma boa extração de dados. De referir que o som pode ser representado na forma de um espectro. Este espectro pode ser representado de diferentes formas como por exemplo, com recurso à frequência ou em função do tempo. Neste caso, o tipo de espectro que for construído tem implicações nas características que são escolhidas como na qualidade destas e do resultado final. Um dos processos que tem, também, bastante importância e implicações é a construção do espectro pois pretende-se a construção de um espectro mais uniforme, com picos mais uniformizados e com uma variação da onda menor, o que faz com que exista uma melhor formação das curvas. Isto é obtido através da aplicação de vários métodos de transformação da onda. Uma melhor formação das curvas é definido como a diminuição dos picos nas curvas e a construção de picos mais suaves para que seja possível uma melhor análise dos dados.
Após este processo, e retiradas as características do espectro, com o uso do tempo ou frequência, conclui-se a construção do *audio fingerprint*. De forma a ser realizada a classificação pode ser necessária a realização de uma aprendizagem do sistema existindo, principalmente, dois grandes grupos: Aprendizagem supervisionada e a não supervisionada.

- **Aprendizagem Supervisionada:** Este conceito é utilizado quando um programa é treinado tendo por base um conjunto de dados pré-definido. O processo do treino é bastante importante, pois é com base no treino dos dados pré-definidos que o sistema vai tomar decisões quando recebe dados novos.

- **Aprendizagem Não-Supervisionada:** O conceito é usado quando se está presente num programa que pode, automaticamente, encontrar padrões e relações num determinado conjunto de dados.

Torna-se pertinente referir também que dentro destes dois grupos, existem vários tipos de classificadores que podem ser utilizados, sendo alguns mais adaptados para determinadas situações. Caso não se opte por esta última opção, existem ainda outras alternativas para a realização de testes, como por exemplo, a utilização do BER(Bit Error Rate).
Posto isto, e realizada uma breve revisão de que tipo de processos serão realizados, serão apresentados, em seguida, alguns dos métodos mais utilizados para a análise de áudio.

2.2 Fundamentos teóricos sobre Audio Fingerprint

Um *audio fingerprint* pode ser definido como uma síntese de um determinado ficheiro de áudio. Para esse resumo ser construído recorre-se à extração de características do espectro de áudio previamente construído. A extração das características faz com que uma impressão digital de áudio se torne única. Assim sendo, caso existam dois ficheiros de áudio com sons bastante semelhantes, a construção da sua caracterização é obrigatoriamente diferente, dado que as características extraídas serão, necessariamente, distintas. Após este passo, é possível a realização da pesquisa de um trecho de áudio numa base de dados ou a localização de ficheiros de áudio semelhantes.
Para a realização do processo de avaliação de um trecho de áudio existem, principalmente, dois processos fundamentais, segundo o sistema de reconhecimento de áudio apresentado em [TL03] que são: construção do *audio fingerprint* e o algoritmo de pesquisa como é possível observar na figura [TL03].

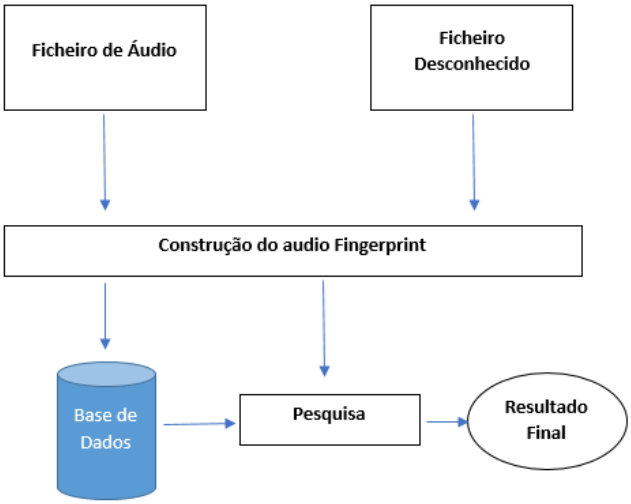

Figura 2.1: Sistema de reconhecimento de áudio.

Tendo por base a figura 2.1, podemos concluir que, de forma sintetizada, o processo consiste em construir uma base de dados com os *audio fingerprints* de diversos ficheiros já conhecidos. Esta irá permitir que quando o sistema de reconhecimento de áudio recolher sons e, após ser construído o *audio fingerprint* seja possível a realização de uma pesquisa na mesma base de dados para ser efetuada a devida classificação do ficheiro recebido e, assim, se obter o resultado final.
Realizando um foco, primeiramente, no processo de construção do *audio fingerprint*, de referir que existem vários processos a serem realizados e diversos métodos possíveis de implementação para a concretização destes passos.
Inicialmente, alguns estudos optam pela realização de um pré-processamento do sinal de áudio com a implementação de um filtro de forma a ser obtido um ótimo espectro para uma melhor construção do *audio fingerprint*. Dois dos filtros aplicados no pré-processamento são o *low pass filter* [MOK+16] e um *rectangular filter* [SBY06]. A aplicação destes filtros tem como objetivo principal a redução do ruído existente no sinal recebido. Sabendo que o ruído tem influência num futuro resultado, por vezes, pode ser necessária a aplicação deste tipo de filtros para a obtenção de melhores resultados. O que foi possível verificar é que a maioria dos estudos apresentados, ignoram este processo e realizam, imediatamente o passo designado por *windowing* existindo ainda a hipótese como apresentada em [SBY06], a aplicação de um filtro após a conclusão do passo anteriormente referido.
Outro fator de bastante importância incide em qual tamanho dos ficheiros deve ser utilizado. É possível verificar que a maioria dos estudos tem preferência na utilização de ficheiros com um tamanho total de 3 segundos. Este tamanho dos ficheiros é utilizado dado que se pensa que é o tamanho suficiente para a obtenção de resultados satisfatórios. Alguns estudos como [YCY14] optaram pela utilização de ficheiros com um tamanho de 5 segundos. Por fim, podemos ainda verificar que em [TL03] foi realizado um estudo com a utilização de ficheiros com tamanhos de 1, 2, 3, 4, 5 e 6 segundos. Neste estudo concluiu-se que com *frames* de 2 segundos já se consegue obter alguns resultados satisfatórios (95.6% de precisão), enquanto *frames* com 3 e 4 segundos obtiveram 97.8% de precisão e com ficheiros a partir de 5 segundos foi possível obter uma precisão de 100%. Por fim, referir outro aspeto importante no momento da realização das recolhas, sendo ele designado por *sample rate*. Este consiste no número de dados recolhidos por segundo através do dispositivo. Em [GBI06] este dado tem o valor de 5512Hz. Já no estudo desenvolvido em [HK02] optou-se por 5kHz, observando-se que o número de dados recolhidos é bastante semelhante ao estudo anterior apresentado. Outros valores também utilizados para o *sample rate* são o 44.1kHz, como foi utilizado em [KD13], e 8kHz usado em [WYWS12].

2.2.1 *Windowing*

Após a receção do ficheiro é realizado o já referenciado passo de *windowing*. A realização deste processo permite um melhor desempenho da aplicação da transformada futuramente utilizada. Isto deve-se ao facto de através das funções aqui implementadas permitirem a construção de ondas sinusoidais mais facilmente comparáveis. Assim, seguindo [HK02], é utilizada a função matemática designada por *Hanning Window* que pode ser observada em seguida, sendo que será aplicado para cada *frame t* a *Hanning Window w(t)* [CM10]:

$$w(t) = \begin{cases} 1 - \cos(2\pi t/T)/2, & 0 \leq t \leq T \\ 0, & t < 0 \ \ ou \ \ t > T \end{cases}, \qquad (2.1)$$

onde T é o intervalo de amostragem.
Aqui, são utilizados *frames* de tamanho de 3s sendo eles divididos em *"sub-frames"* de 0.37s. A implementação da função matemática anteriormente apresentada é então aplicada em cada *"sub-frame"* para se obter ondas sinusoidais que seja possível a realização de uma futura comparação e uma melhor extração de *audio fingerprints*. Resumidamente, após a conclusão do processo anterior, realiza-se a extração de *"sub-fingerprints"* a cada 11.6ms o que resulta em 32 *"sub-fingerprints"* a cada 0.37s. Realizando este processo, no final do processo são obtidos, aproximadamente, 256 *"sub-fingerprints"*.
Uma outra função comummente utilizada para o mesmo fim é a *Hamming Window* que é implementada em diversos estudos como em [XYW+12]. Em seguida é possível observar a descrição da mesma, sendo que em cada *frame* é aplicada a *Hamming Window* w(I)[XYW+12]:

$$w(I) = 0.54 - 0.46\cos[2\pi I/(N-1)], 0 \leq I < N, \quad (2.2)$$

sendo N o intervalo da amostragem.
Normalmente, aliado ao passo anteriormente já descrito e concluído o objetivo de se obter um espectro composto com ondas sinusoidais, é aplicada uma transformada de forma a ser obtida um espectro que facilite um futuro tratamento dos dados. Para tal, preferencialmente constrói-se um espectro com o domínio a ser a frequência ao invés de em função do tempo. De forma a ser possível obter um espectro com esta forma, é aplicada uma transformada. A transformada mais utilizada, tendo por base os estudos analisados, é a transformada de Fourier que é utilizada em estudos como [HK02] e [MSK14]. Existem diversos métodos de aplicação da transformada de Fourier, sendo uma das mais utilizadas, como em [HK02], a FFT(*Fast Fourier Transform*) que é apresentada em seguida [DGC+12] .

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk}, \quad (2.3)$$

sendo k um inteiro que varia de *0* a *N-1*.
Já em [GJ15] o processo de *windowing* é associado à utilização de uma transformada, neste caso a STFT(*Short Time Fourier Transform*), sendo esta definida em seguida.
Sendo ***x[n]*** um sinal de áudio de energia finita, o espectro é obtido através da divisão em blocos sobrepostos com comprimento l e calculando a transformada de Fourier em cada bloco.
Seja *m* cada bloco, então o ponto *N* da STFT ***x[n]*** é dado por [GJ15]:

$$F(m,k) = \sum_{n=0}^{\infty} x[m(l-o)+n] w_{l,o}[n] exp^{-j\frac{2\pi}{N}kn}, \quad (2.4)$$

onde $k, 0 \leq k < N$, representa a frequência, $w_{l,o}$ é a *hanning window* com comprimento *l* e sobreposição *o*.
Além das já referidas transformadas, existem outras opções, como utilizado em [KD13] que utiliza uma *Wavelet transform* enquanto em [GBI06] utiliza a BMW (*forward balanced multiwavelet*).

2.2.2 Extração de características

Após este processo, realiza-se a extração de características para que seja possível uma melhor classificação. Aqui, existem diversas possibilidades de características a utilizar tendo por base

o espectro de áudio com domínio de frequência. Em [MSK14] optou-se pela utilização de 12 coeficientes MFCC (*Mel Frequency Cepstral Coefficients*), excluindo o coeficiente de energia dado que a utilização deste último influência o desempenho do sistema.
Em [YP07] realizou-se um estudo de forma a perceber qual das características teria melhor desempenho, sendo que as que foram submetidas ao teste foram os coeficientes MFCC, os ZCPA (*zero-crossing with peak amplitudes*) e FBE (*Filter Bank-Energy*). A conclusão retirada desta experiência foi que para ficheiros com pouco ruído a eficiência dos sistemas foi bastante semelhante e com resultados bastante satisfatórios. O mesmo não se verifica com ficheiros com algum ruído dado que a utilização de ZCPA como características extraídas mostra um melhor desempenho, aproximadamente 80%, enquanto que as restantes apresentam um desempenho de 50%.
Já em [WYWS12] foram utilizados os picos do espectro de cada *frame*. Os picos são definidos como sendo o máximo local, neste caso, o máximo de frequência local num determinado intervalo de tempo.
Numa outra opção e que é utilizada em [SBY06] optou-se por extrair a média e a variância à matriz construída neste método. A média pode ser obtida da seguinte forma: Seja M a média resultante da matriz, k o bloco da matriz ASF (*Audio Spectrum Flatness*) [SBY06],

$$M(k,j) = \frac{1}{decim} \sum_{i=1}^{decim} V_{i,j} \quad , j = 1, ..., m\,, \qquad \textbf{(2.5)}$$

para $k = 1, ..., b$, onde m é o número de colunas da matriz ASF e a $V_{i,j}$ é um elemento da matriz ASF. Seja $\bar{X}_j$ a média de uma determinada coluna j da matriz ASF, S a variância resultante e o número do bloco da matriz ASF [SBY06]:

$$S(k,j) = \sqrt{\frac{1}{decim} \sum_{i=1}^{decim} (V_{i,j} - \bar{X})^2, j = 1, ..., m\,}, \qquad \textbf{(2.6)}$$

para $k = 1, ..., b$, onde m é o número de colunas da matriz ASF e a $V_{i,j}$ é um elemento da matriz ASF.
Em [XYW+12] realizou-se uma outra opção na escolha de que características seriam utilizadas no processo de caracterização. Essa opção recaiu na utilização de SSC (*Spectral Subband Centroid*), sendo estas normalizadas em seguida e, assim, os dados usados são entre 0 e 1.
Assim, o momento de sub-banda de ordem v na i-ésima sub-banda de um espectro de áudio P[k,m] é definido como [XYW+12]:

$$M_i^v[m] = \sum_{k=B_i+1}^{B_{i+1}} k^v P[k,m]\,, \qquad \textbf{(2.7)}$$

onde k, m e B_i representam o valor da frequência, o índice do *frame* e limite da sub-banda i. Com isto, a SSC pode ser definida seguindo a seguinte equação[XYW+12]:

$$C_i[m] = \frac{M_i^1[m]}{M_i^0[m]}\,. \qquad \textbf{(2.8)}$$

Outro estudo realizado foi o de [KD13] onde neste caso utiliza uma *wavelet transform*, optou

pelo uso de quatro características. Optou pela utilização da variância, ZCR (*zero-crossing Rate*), centroíde e a energia do espectro. A variância, neste contexto, é definida da seguinte forma [KD13]:

$$\sigma^2 = \frac{1}{N}\sum_{j=1}^{N}(cDj - c\bar{D})^2\,, \quad (2.9)$$

em que *cDj* é o j-ésimo coeficiente *cD* e $c\bar{D}$ é a média dos coeficientes.
Já o ZCR pode ser obtido da seguinte forma [KD13]:

$$ZCR_m = \frac{1}{2}\sum_{m}|sign(x(n)) - sign(x(n-1))|\,, \quad (2.10)$$

onde, *x(n)* é o n-ésimo valor do coeficiente *wavelet* no m-ésimo *frame* que corresponde a cA5 e cD5. Se *x(n)*=0, então *sign[x(n)]*=1 caso contrário *sign[x(n)]*=0.
O centroide num espectro de áudio é definido como sendo o centro de energia. Este pode ser calculado seguindo a seguinte equação [KD13]:

$$Centroide = \frac{\sum_{i=1}^{N} i|x(i)|^2}{\sum_{i=1}^{N}|x(i)|^2}\,, \quad (2.11)$$

onde, *x(i)* é o i-ésimo coeficiente *wavelet*.
A última característica extraída neste estudo é o calculo da energia da onde sendo obtido com a seguinte equação [KD13]:

$$Energy = \frac{1}{N}\sum_{i=1}^{N}|x(i)|^2\,, \quad (2.12)$$

onde, *x(i)* é o i-ésimo coeficiente *wavelet*.
De forma a efetuar o processo de caracterização, optou-se pela construção de *strings* de *bits* para que estas definissem os *frames* obtidos. A forma como essas *strings* são construídas podem ser observadas na descrição mais acima do estudo [KD13].
Um outro método que utiliza a construção de *strings* para a caracterização dos *frames*, é o proposto por [HK02], entre outros estudos que seguem esta mesma metodologia. Aqui, ao invés do método anterior que a construção das *strings* tinha por base a utilização de quatro características, este apenas utiliza uma: a energia de banda. Assim sendo, a construção da caracterização dos *sub-fringerprints* F é obtida com recurso à energia de banda E dos *frames* n e m, dada a seguinte fórmula [HK02]:

$$F(n,m)\begin{cases}1 \;\; se \;\; E(n,m) - E(n.m+1) - (E(n-1,m) - E(n-1,m+1)) > 0\\ 0 \;\; se \;\; E(n,m) - E(n.m+1) - (E(n-1,m) - E(n-1,m+1)) \leqslant 0\end{cases}. \quad (2.13)$$

2.2.3 Processo de identificação

Este processo tem como principal objetivo de, através da utilização das características extraídas em passos anteriores, realizar uma análise de dados que automatiza a construção de modelos analíticos. Através do uso da inteligência artificial que se baseia na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana.

Em [SSASS09] optou-se pela utilização do classificador KNN(K-Nearest Neighbor). Este método de classificação foi utilizado quer para o treino como para a fase de teste do sistema desenvolvido. Neste estudo verificou-se uma eficiência de 71%.
Já em [MK14] foi utilizado o classificador GMM(Gaussian Mixture Model) sendo aplicado quer para o treino como para a fase de testes. Com este classificador foi obtida uma eficiência entre 94 a 98%.
No estudo realizado por [HL12] a escolha do classificador caiu para o uso dos dois apresentados anteriormente, ou seja, o classificador GMM foi utilizado na fase de treino. Já o KNN foi utilizado para o passo de testar o sistema construído. Este sistema apresentou uma precisão de, aproximadamente, entre 77 e 89%.
Um estudo que optou, também pela utilização do classificador GMM para a realização do treino do sistema desenvolvido foi realizado em [MSK14]. No entanto, para a fase de teste optou-se pela construção e utilização de uma rede neuronal e verificou-se uma eficácia de 99%.
Um método bastante distinto dos anteriormente apresentados é a utilização do BER para a realização do processo de identificação. Em [DHL12], que tem por base o método proposto por [HK02], este classificador é utilizado após a construção de *strings* de *bits* para a caracterização do áudio recebido. Neste método é obtida uma eficiência de 92%.
Por fim, em [TL03] de modo escolhido para realizar a classificação do áudio recebido foi o uso da distância euclidiana, sendo obtidos resultados entre os 96 e os 100%. Esta distância pode ser obtida da seguinte forma [TL03]:

$$d_c(s_1, s_2) = \sum_{k=1}^{N} |x_k - y_k| , \quad (2.14)$$

onde *xk* e *yk* são os índices do conteúdo extraído nos pontos s1 e s2 respetivamente. Já *N* representa o comprimento do *fingerprint*.

2.3 Estado da arte

O facto de o interesse pelo estudo e desenvolvimento de aplicações na área de *audio fingerprint* ser ainda bastante recente, tem implicações em ainda existirem, relativamente, poucos métodos implementados de forma a contribuir para o desenvolvimento desta área. Apesar da sua juventude, isso não implica uma inexistência, nos dias de hoje, de aplicações e estudos de sucesso e de referência.
Esta secção pretende apresentar alguns dos estudos já realizados até aos dias de hoje, bem como a apresentação dos resultados derivados da aplicação dos métodos desenvolvidos. Assim sendo, na tabela em seguida estão apresentados, de forma sintetizada, os estudos realizados com o objetivo tirar partido da construção de *audio fingerprints*.

Tabela 2.1: Análise de estudos atualmente efetuados

Autor	Ano	Objetivo	Segmentação	Normalização	Representação Espectro	Características	Classificador	Resultados
Lahouari Ghouti, Ahmed Bouridane [GBI06]	2006	Construção de um método robusto que seja capaz de construir, analisar e classificar um *audio fingerprint*.	*Frames* de 3s com *sample rate* de 5515Hz. Nestes *frames* é aplicado as *Hanning Window* com sobreposição de 31/32.	Não aplicável	Transformada BMW (Balanced Multiwavelets)	Variação logarítmica da magnitude média das variações. Em seguida, é formada uma *string* de *bits* sendo eles atribuídos através de uma fórmula presente no artigo.	BER	Não são apresentados.
Yong Fan, Shuang Feng [TL03]	2003	Construção de uma base de dados de *audio fingerprints* e implementação de um método de pesquisa e reconhecimento do mesmo.	São testados *frames* com 1,2,3,4,5 e 6s onde o *sample rate* é 44.1kHz. Em seguida, aplica-se a *Hanning Window* com sobreposição 31/32.	Não aplicável.	FFT (Fast Fourier Transform)	Extração dos picos de frequência e diferença de tempo. As características extraídas são armazenadas numa *lookup table*.	Para classificar é usada a distância euclidiana.	Nesta experiência foram utilizadas 45 músicas. Concluiu-se que a partir de *frames* com 2s se consegue obter resultados positivos (95.6%) sendo que com *frames* a partir dos 5s se obtém eficácia de 100%.
M. Sert, B. Baykal, A. Yazici [SBY06]	2006	Pretende-se a construção de um método capaz de construir *audio fingerprints* da parte mais importante dos ficheiros de áudio, de forma a reduzir o espaço necessário de armazenamento.	Aplicação de *Hamming Window* para se obterem *frames* de 30ms.	Não aplicável	FFT	Variância e média	Utilização de uma fórmula presente no artigo para se classificar.	Obteve-se 93.8% de eficiência o que em comparação com o MFCC é bastante semelhante dado que se obteve 93%.

Autor	Ano	Objetivo	Segmentação	Normalização	Representação Espectro	Características	Classificador	Resultados
Sunhyung Lee, Dongsuk Yook, Sukmoon Chang[LYC13]	2013	Pretende-se uma melhoria na velocidade de pesquisa sem afetar a eficácia da mesma, usando o método proposto por *Haitsma*[HK02] para a construção do *audio fingerprint*.	O método apresentado foi implementado para *frames* com, aproximadamente, 3s sendo aplicada a *Hanning Window*.	Não aplicável.	FFT	Construção de uma *string* de *bits* com base na energia de banda. São atribuídos pesos ás músicas com a seguinte fórmula: $P = (T/S)$	BER	Este método de pesquisa apresentou eficiência de 80 a 99%.
Wei Xiong, Xiaoqing Yu, Wengen Wang, Wanggen Wan, Ram Swaminathan [XYW+12]	2012	Pretende-se a construção de um algoritmo de extração de *audio fingerprint* baseado em dynamic subband locating e SSC normalizada.	Aplicação de *Hamming Window* com sobreposição de 15/16 onde se obtêm *frames* de 0.371s.	É aplicada a normalização com a fórmula presente no artigo em Spectral Subband Centroid(SSC).	FFT	Dynamic subband locating.	Distância Euclidiana.	Verificaram-se melhores resultados com algum ruído (entre 80 e, aproximadamente, 100%) que sem ruído algum que se obteve, aproximadamente 60%.
B. M. Murphy, C. O'Driscoll, I. Korotchikova, G. B. Boylan1, G. Lightbody, W. P. Marnane [MOK+16]	2016	Aplicação de uma técnica de quantificação de *audio fingerprints* em dados EEG para tentar identificar secções de dados EEG usando apenas uma subsecção dos dados.	Existe um pré processamento onde é aplicado um *low pass filter*. É usado a *Hanning Window* com sobreposição de 31/32.	Não aplicável.	FFT	Gerada uma *string* de *bits*, com base na energia de banda, como definição do sinal recolhido e é armazenada numa *lookup table*.	BER	Os resultados obtidos estão dependentes da quantidade de ruído existente sendo que se obtêm melhores resultados com bastante ruído (aproximadamente 93%) que com pouco ruído (entre 48 a 65%).

Autor	Ano	Objetivo	Segmentação	Normalização	Representação Espectro	Características	Classificador	Resultados
Jin S. Seo, Minho Jin, Sunil Lee, Dalwon Jang, Seungjae Lee, Chang D. Yoo Dept [SJL+06]	2006	Apresentação de um método de extração de *audio fingerprint* baseado em centros geométricos de subbanda espectral normalizada.	É aplicado a *Hamming Window* com sobreposição de 50%	É efetuado nos *Spectral Subband Centroids*.	FFT	Utilização dos centros geométricos das 16 subbandas.	Distância euclidiana.	Foi possível concluir que este método obtém melhores resultados que com a utilização de MFCC.
Alexander Sinitsyn [Sin06]	2006	Criação de um sistema que seja capaz de detetar ficheiros de áudio duplicados.	Aplicação de *Hanning Window* com sobreposição de 31/32.	Não aplicável.	FFT	Construção de uma *string* de *bits*, usando a energia de banda, para definir a recolha. Esta atribuição é feita seguindo uma fórmula presente em [HK02].	BER	Não é apresentado.
Kyuwoong Hwang, Soo-Young Lee [HL12]	2012	Capacidade de construir uma aplicação capaz de desvendar em que cenário e ambiente o telemóvel do utilizador está envolvido.	Aplicação de *Hamming Window*.	Não aplicável.	FFT	Usos de 12 coeficientes MFCC e é excluído o coeficiente da energia.	De forma a treinar o sistema, é utilizado o GMM (Gaussian mixture mode) enquanto para testar é utilizado o classificador KNN.	Foram obtidos resultados de 77 a 89%, aproximadamente.

Autor	Ano	Objetivo	Segmentação	Normalização	Representação Espectro	Características	Classificador	Resultados
Sajad Shirali-Shahreza, Hassan Abolhassani, M. Hassan Shirali-Shahreza [SSASS09]	2009	Criação de um método de identificação do artista que tenha um tempo de pesquisa rápido.	São utilizados *frames* com tamanho entre 10 e 20ms.	É utilizada na normalização dos MFCC's.	IDFT	São extraídos os primeiros 10 a 20 coeficientes MFCC (é utilizada a DFT para a sua extração, ao invés da FFT), exceto o primeiro que representa a energia do sinal. A média e variância dos coeficientes normalizados são extraídos.	KNN	O sistema atingiu uma eficiência de 71%.
Mohanapriya. S. P, E. P. Sumesh, R. Karthika [MSK14]	2014	Construção de uma aplicação capaz de realizar uma deteção do som ambiental.	É utilizada a técnica designada por *Hamming Window*	Não aplicável	FFT	São extraídos 12 coeficientes MFCC estando excluído o coeficiente da energia.	É utilizado o GMM para a treinar o sistema, enquanto para a realização de testes foi desenvolvida uma rede neuronal.	Foi possível verificar uma eficiência de 98.9%.
M.Davidson Kamaladas, M.Maxina Dialin [KD13]	2013	Aplicação de um algoritmo de *audio fingerprint* baseado na transformade de wavelet para a extração de características.	É aplicada a *Hanning Window* com sobreposição de 28/32.	Não aplicável.	Wavelet Transform	Variância dos coeficientes, *Zero-crossing Rate*, centro geométrico da energia e energia da subbanda. Em seguida, construção de uma *string* de 8 *bits* seguindo uma fórmula presente neste artigo.	Não aplicável.	Não aplicável.

Autor	Ano	Objetivo	Segmentação	Normalização	Representação Espectro	Características	Classificador	Resultados
Mohanapriya. S.P, R.Karthika [MK14]	2014	Construção de uma aplicação capaz de analisar o som ambiental.	Aplicação da *Hamming Window*.	Não aplicável.	FFT	Uso de 12 coeficientes MFCC excluindo o coeficiente da energia.	GMM	Apresenta uma eficiência de 98%.
Arunan Ramalingam, Sridhar (Sri) Krishnan [RK05]	2005	Desenvolvimento de uma aplicação capaz de construir *audio fingerprints*. Existe, ainda, o objetivo de avaliar os recursos disponibilizados pela STFT(Short-Time Fourier Transform).	Uso de *frames* com 23ms e sobreposição de 50%.	Não aplicável.	STFT	Entropia de Shannon, Entropia Rényi, Centro geométrico do espectro, Largura de Banda do espectro, Energia do espectro, *Spectral flatness measure*, *Spectral crest factor*.	GMM	Apresenta uma eficiência de 99%.
Jaap Haitsma, Ton Kalker [HK02]	2002	Desenvolvimento de um sistema capaz de construir uma *audio fingerprint* e capaz de o classificar, como por exemplo, uma determinar o autor e o nome de uma determinada música.	Utilização de *Hanning Window* em *frames* de 3s com sobreposição de 31/32.	Não aplicável.	FFT	Construção de uma *string* de *bits*, com recurso a energia de banda, através de uma fórmula presente no artigo, e armazenada numa tabela.	BER	Não apresentado.

Apesar de a área de estudo da criação de um *audio fingerprint* ainda ser relativamente recente, verifica-se a existência de algumas metodologias para o seu desenvolvimento. Este processo é composto por diversas etapas como é possível verificar na tabela anterior. Assim sendo, e após uma análise da tabela 2.1 é possível verificar que a maioria dos autores optou pela utilização de *frames* com tamanho de 3 segundos. O estudo realizado em [TL03] teve o cuidado de realizar um teste com a implementação de um método mas com *frames* de diferentes tamanhos, compreendidos entre 1 e 6 segundos. Aqui foi possível concluir que a utilização de trechos com um tamanho a partir de 2 segundos consegue-se obter resultados satisfatórios(aproximadamente 96%) sendo que se a perfeição foi conseguida quando foram utilizados ficheiros de áudio com um tamanho mínimo de 5 segundos.

Antes da realização de outros processos, [MOK+16] realiza um pré processamento dos dados com a aplicação de um *low pass filter*. Para a obtenção de dados com domínio frequência é, em todos os casos, aplicada uma transformada. A transformada maioritariamente aplicada é a transformada de *Fourier* nas suas diversas vertentes, verificando-se um padrão na utilização da FFT. Dado que é aplicada a transformada de *Fourier* torna-se essencial a aplicação de janelas para evitar a existência de espaços vazios num espectro, visto que esta transformada se enquadra em dados periódicos e a descontinuidade poderia afetar o futuro desempenho. A construção de janelas é realizada através da utilização e duas funções matemáticas: *Hanning Window* e *Hamming Window*. A utilização destas funções matemáticas encontra-se bastante dividida dado que na tabela anterior se verifica o mesmo número de utilizações.

Para a realização da caracterização, trabalhando na sua maioria com dados não normalizados, verifica-se a existência de diversas opções. A opção mais utilizada consiste na construção de uma *string* de *bits*. Essa caracterização é realizada, na sua maioria, através de uma fórmula presente em [HK02], que conta com o recurso à extração da energia de banda. Um processo bastante semelhante ao anterior é o estudo [LYC13] que atribui diversos pesos às músicas. Outra caracterização bastante realizada é a extração dos coeficientes MFCC. De referir que, ao recorrer-se a esta forma, são extraídos 10 a 12 coeficientes sendo excluído o coeficiente da energia por uma questão de desempenho. Existem ainda outras possibilidades para a caracterização mas com menor utilização como é o exemplo da extração da variância, média ou centro geométrico (centroíde).

Por fim, para a realização do processo de caracterização dos ficheiros existem algumas opções. Os estudos que optaram, no processo anterior, pela construção de *strings* a escolha recai pela comparação entre elas com auxilio pelo BER ou distância euclidiana. Os estudos que optaram por uma caracterização decidiram utilizar *machine learning*. Existiu a opção pela utilização, quer para o treino do sistema como para a realização de testes, do KNN [SSASS09] ou então GMM [RK05]. Verifica-se ainda a opção pela utilização do GMM para a realização do treino do sistema e a escolha do KNN para o processo de classificação [HL12]. A opção restante verificada foi a utilização do método GMM para o treino do sistema desenvolvido e para a os testes optou-se pela implementação de uma rede neuronal.

2.4 Estudos relevantes

Este sub-capitulo tem como objetivo a apresentação e uma breve descrição de alguns estudos já realizados e com diferentes técnicas aplicadas. A escolha destes deve-se aos resultados obtidos e a tentativa de apresentar técnicas mais diversificadas possível. Segue-se a descrição de 5 estudos realizados.

A highly robust audio fingerprint system [HK02]

Neste estudo é apresentado um método bastante eficaz para uma extração e classificação de um *audio fingerprint*. Neste método optou-se pela utilização de características não semânticas ao invés das semânticas dado que os recursos semânticos nem sempre têm um significado claro e inequívoco, são mais difíceis de serem utilizados no computador e, por fim, estes recursos não são universalmente aplicáveis. Assim sendo, o processo principal para a extração de *audio fingerprint* tem por base a utilização do método *Hanning Window* e ainda a FFT para a realização da segmentação ficheiro recebido com tamanho de 3s com *sample rate* de 5kHz. Em seguida é realizada uma fragmentação no ficheiro com a obtenção de *frames* de 0.37 segundos. Nestes *frames* construídos são obtidos *sub-fingerprints* de 32 *bits* a cada 11,6ms. Concluído este processo verifica-se a obtenção de 256 *sub-fingerprints*.

A forma utilizada para a representação das impressões digitais foi o uso de uma *string* de *bits* aos invés da utilização de números reais ou funções criptográficas. A justificação para tal deve-se ao facto de este implicar uma menor complexidade de pesquisa e existir uma identificação confiável mesmo que a percentagem de *bits* não correspondentes seja elevada. Assim sendo, a construção da caracterização dos *sub-fringerprints* F é obtida com recurso à energia de banda E dos *frames* n e m, como é possível analisar na seguinte fórmula [HK02]:

$$F(n,m)\begin{cases}1 \;\; se \;\; E(n,m)-E(n.m+1)-(E(n-1,m)-E(n-1,m+1))>0\\0 \;\; se \;\; E(n,m)-E(n.m+1)-(E(n-1,m)-E(n-1,m+1))\leqslant 0\end{cases}. \quad (2.15)$$

De realçar que são as *strings* construídas e usadas para definição do ficheiro de áudio tem o tamanho de 32 *bits*. Após este processo é realizada a classificação com recurso ao BER.

Fingerprint Extraction of Audio Signal using Wavelet Transform [KD13]

Neste estudo é utilizado um novo algoritmo para a construção de um *audio fingerprint*. Aqui, apenas é apresentada a forma que é construído não sendo apresentada qualquer forma de classificação do mesmo.

Assim sendo, a divisão do ficheiro recebido inicialmente é realizada através da utilização da função *Hanning Window* com sobreposição de 28/32.

Devido à transformada de Fourier apenas fornecer informações de frequência para sinais não estacionários (sinais que não apresentam as mesmas componentes de frequência durante sua duração), optou-se pela utilização da *Wavelet Transform* dado que esta é capaz de fornecer informações de frequência e de tempo para este tipo de sinais.

Após estes passos anteriormente descritos e construidos os *sub-fingerprints*, procede-se à caracterização do áudio. Essa caracterização é efetuada com a utilização de *strings* de 8 *bits*. Para a definição dos 8 *bits* são utilizadas como características extraídas: a variância, *Zero-crossing Rate*, centro geométrico e a energia de banda. Assim sendo, os 5 dos primeiros 8 *bits*

são definidos através da seguinte fórmula que tem por base a utilização da variância [KD13]:

$$F1(n,m) = \begin{cases} 1 \ \ se \ \ \sigma(n,m) - \sigma(n,m+1) - (\sigma(n+1,m) - \sigma(n+1,m+1)) > 0 \\ 0 \ \ se \ \ \sigma(n,m) - \sigma(n,m+1) - (\sigma(n+1,m) - \sigma(n+1,m+1)) \leqslant 0 \end{cases}, \quad (2.16)$$

onde, F1(n,m) representa o valor do *bit* m no *frame* n.
Os restantes 3 *bits* são obtidos seguindo a seguinte fórmula [KD13]:

$$F2(n,c) = \begin{cases} 1 \ \ se \ \ Sc(n) \sum_i^N Sc(i) > 0 \\ 0 \ \ se \ \ Sc(n) \sum_i^N Sc(i) \leqslant 0 \end{cases}, \quad (2.17)$$

onde, F2(n,c) corresponde ao valor do *bit* relativo ao *Zero-crossing Rate*, centro geométrico e a energia de banda e Sc(n) representa *Zero-crossing Rate*, centro geométrico e a energia de banda para o *frame* n.
Se c=1, representa *Zero-crossing Rate*,
Se c=2, representa o centro geométrico do domínio *wavelet*,
Se c=3, representa a energia do domínio *wavelet*.

Environmental Audio Scene and Activity Recognition through Mobile-based Crowdsourcing [HL12]
Este estudo apresenta um método de deteção de identificação da atividade que está a ser realizada, naquele momento, pelo utilizador do *smartphone*. Esta identificação apresenta diversas dificuldades como o ruído de fundo existente nas atividades em questão ou como se o *smartphone* se encontra no bolso ou na mão do utilizador. Esta proposta desenvolvida tinha como principal objetivo ultrapassar estas adversidades e conseguir desenvolver um método capaz de desenvolver o reconhecimento de áudio.
O método apresentado para a construção do *audio fingerprint* consiste na implementação da função matemática *Hamming Window* a cada 32ms. Nesses *frames* criados é aplicada a FFT de forma a obter uma espectro com domínio de frequência. Tirando partido da aplicação dos passos anteriormente apresentados e obtido um espectro com domínio da frequência e segmentado, são extraídos 12 coeficientes MFCC, exceto o coeficiente da energia. Optou-se por excluir este coeficiente devido ao facto de a sua utilização iria implicar um pior desempenho no reconhecimento do som ambiental, onde o mesmo som pode ter energia diferente devido a, por exemplo, a distância entre a fonte e o microfone.
Para a realização da classificação, optou-se por um método de aprendizagem automática. A realização do treino do sistema desenvolvido é efetuado com recurso a GMM. Já para a realização de testes optou-se pela técnica do vizinho mais próximo, o KNN. Com este método, observou-se uma precisão entre os 77 e os 89%.

Scalable and Robust Audio Fingerprinting Method Tolerable to Time-stretching [GJ15]
O estudo desenvolvido tem como principal objetivo o desenvolvimento de um sistema de reconhecimento de áudio robusto e eficaz. Tem, ainda, como um dos principais objetivos que o sistema suporte o alongamento do tempo dos *frames* utilizados.
Para que tal seja possível, foi utilizado para o processo de *windowing* a função matemática *Hanning Window* com *frames* de 64ms e sobreposição de 50%. Em seguida, é aplicada a transformada de Fourier de forma a ser obtido um melhor espectro que será utilizado futuramente. Antes de serem retiradas as características que irão definir o áudio submetido a teste, é aplicado um *High*

Pass Filter sendo posteriormente extraídos os picos do espectro obtido. Os picos são definidos como sendo um máximo local. As caracterizações extraídas de cada áudio são inseridas numa tabela de *hash*.

Com recurso à tabela de *hash* construída, é efetuada a classificação. Este passo é realizado com recurso à técnica designada por: *longest common subsequence* (LCS). Esta técnica consiste em encontrar a maior sequência em comum entre dois *"hashs"* em comparação. A sequência maior encontrada é designada como resultado final para esse teste. Neste estudo, obteve-se, em alguns casos, resultados bastante melhores que os utilizados pelo *Shazam*.

A Music Identification System Based On Audio Fingerprint [TL03]

Este estudo foi realizado com a intenção de desenvolver um sistema capaz de efetuar uma identificação de musicas. Realizou-se uma experiência com *frames* de 1, 2, 3, 4, 5 e 6 segundos de forma a verificar se o tamanho destes influencia os resultados finais. Neste esquema foram utilizadas 45 músicas onde foi aplicado a função matemática *Hanning Window* com sobreposição 31/32. Foi utilizada a FFT de forma a obter dados com domínio de frequência.

Para a caracterização do áudio recebido, optou-se para a extração dos máximos locais de frequência do espectro(picos). Estes dados extraídos são armazenados numa *lookup table*. Para a realização dos testes é adotada a distância euclidiana, consultando a tabela já construída anteriormente. Assim, concluiu-se que, com a aplicação do sistema aqui proposto, a utilização de *frames* com tamanho a partir de 2 segundos consegue-se obter uma eficácia de 96% enquanto com *frames* com tamanho a partir de 5 segundos obtêm-se resultados com uma taxa de acerto de 100%.

An Industrial-Strength Audio Search Algorithm [Wan03]

Este estudo e método desenvolvido foi aplicado e deu como resultado a aplicação *Shazam*. Para tal, para cada amostra é processado um *audio fingerprint* sendo que, quer os ficheiros submetidos a teste como os que estão presentes na base de dados construída, passam pelo mesmo processo. Posto isto, são obtidos os picos de frequência de um espectro, anteriormente construído. O pico é definido como sendo o ponto mais alto entre os seus vizinhos. Em seguida, é construído um "mapa" com diversos pontos, sendo eles os pontos anteriormente recolhidos, abdicando-se do espetro. Todo o método desenvolvido em seguida tem por base este "mapa de constelação", como é designado.

Em seguida, são criados pontos de ancoragem. Estes pontos têm uma zona alvo a si associada. Cada ponto de ancoragem é emparelhado com pontos dentro da sua zona de ação em que cada par é registada a sua frequência e diferença temporal. De forma a ser armazenada esta informação, é criada uma estrutura de dados de 64 *bits*, 32 *bits* para o *hash* e 32 para o deslocamento de tempo e ID.

Após o passo anteriormente descrito ser aplicado à amostra, é pesquisada na base de dados possíveis *hashes* que sejam semelhantes e em termos de tempo, ou seja, uma sequência de *hashes* num ficheiro também deve ocorrer no mesmo tempo relativo.

Para a realização deste método, foram recolhidas amostras de cada música com 5, 10 e 15 segundos, num bar. É possível realizar uma classificação com apenas 1 ou 2% do *hash* construído. Por fim, é possível verificar um desempenho bastante satisfatório com bastante ruído, perto dos 100% de taxa de acerto.

2.5 Conclusão

O interesse pelo tema *audio fingerprint* é bastante recente e isso tem reflexos nos estudos realizados. No entanto, derivado às diversas funcionalidades que podem ser construídas através da utilização deste tipo de recursos, o número de estudos e desenvolvimento de aplicações tem crescido nos últimos anos. É possível verificar que, hoje em dia, existe uma aplicação de referência no contexto do *audio fingerprint* que é o Shazam. Esta teve um grande impacto na vida dos utilizadores devido à eficiência do sistema desenvolvido e isso é visível no número de indivíduos que optaram pela sua utilização, fazendo dessa aplicação um estudo de referência. O método desenvolvido [Wan03] é, atualmente, um dos mais importantes e uma referência nesta área.

Relativamente a outros métodos desenvolvidos até à data, muitos são semelhantes e optaram pela mesma estratégia. É possível verificar que o estudo desenvolvido por [HK02] serve de referência para diversos sistemas desenvolvidos dada a sua eficácia. Verifica-se ainda uma opção pela utilização de ficheiros com tamanho de 3 segundos, dado ser possível obter resultados satisfatórios e não sobrecarregar a memória disponível dos dispositivos, pois isso afetaria o seu desempenho. Constata-se ainda a opção pela utilização da transformada de *Fourier*, nas suas diversas formas o que dá a concluir que, possivelmente, é a transformada ideal a ser aplicada. Quanto ao processo de caracterização o processo que o maior número de autores optou foi a escolha pela construção de *strings*, realizando as respetivas comparações para a sua classificação. É possível verificar resultados satisfatórios em todos os métodos aplicados por esta opção. Adicionalmente, outra escolha que também se destaca é a hipótese de serem extraídos os coeficientes MFCC.

Por fim, relativamente ao processo de classificação, verifica-se a existência de duas hipóteses a implementar: a opção pela comparação de *strings*, para quem opte por construir *strings* de *bits* no processo de caracterização, ou implementar um sistema baseado em *machine learning*. É importante referir que se conseguiu obter bons resultados em qualquer que tenha sido a escolha realizada.

Em suma, de forma a serem obtidos resultados satisfatórios, é necessária uma ótima realização da construção do *audio fingerprint*. Qualquer falha que possa aqui existir irá influenciar a classificação e os respetivos resultados.

Capítulo 3

Tecnologias e Ferramentas Utilizadas

3.1 Tecnologias Utilizadas

Para a realização desta dissertação foram utilizadas diversas tecnologias para a sua realização, sendo elas as seguintes:

- Android;
- Java;
- SQL;
- XML(Xtensible Markup Language).

3.1.1 Android

Android é um sistema operativo baseado no sistema operativo *Linux* e desenvolvido pela empresa de tecnologia *Google*. A interface do utilizador é baseada na manipulação direta, sendo projetado principalmente para dispositivos móveis com ecrã sensível ao toque, como *SmartPhone* e *Tablets*. O utilizador interage com o SO (Sistema Operativo) através da manipulação de objetos virtuais, assim como de um teclado virtual. Apesar de ser maioritariamente utilizado em dispositivos móveis, é também utilizado em videojogos, computadores e outros dispositivos.

3.1.2 Java

Java é uma linguagem de programação orientada a objetos, desenvolvida na década de 90. Esta foi desenvolvida por um grupo de programadores tendo como membro principal o *James Gosling*, na empresa *Sun Microsustems*, sendo, atualmente, parte da empresa *Oracle*. Esta linguagem é compilada para um *bytecode*, que é executado numa máquina virtual, favorecendo, assim a sua portabilidade, ou seja, o mesmo código pode ser executado em diferentes dispositivos, desde que estejam suportados por uma máquina virtual Java. A partir da versão 8 do Java verifica-se a implementação de mecanismos funcionais. Java foi a linguagem de programação utilizada neste projeto visto que é a linguagem nativa do *Android*, que foi o ambiente de desenvolvimento escolhido.

3.1.3 SQL

SQL é uma linguagem de pesquisa declarativa direcionada para base de dados relacional. Muitas das características originais do SQL foram inspiradas na álgebra relacional.

3.1.4 XML

XML é uma linguagem de marcação que estende as funcionalidades da linguagem HTML (*Hypertext Markup Language*). O seu propósito é o de representar objetos num formato comum para garantir uma interpretação única. A XML é utilizada nativamente pelo SO *Android*. Este utiliza o padrão MVC (*Model View Controller*) para implementação de aplicações, sendo que o aspeto visual é definido em um ou mais ficheiros XML separados do código. A utilização do XML é consequência do desenvolvimento da aplicação ser direcionada para *Android*.

3.2 Ferramentas Utilizadas

3.2.1 Android Studio

O *Android Studio* é um IDE (*Integrated Development Environment*) para desenvolvimento de software na plataforma *Android*. Este foi anunciado pela *Google* no ano de 2013. O *Android Studio* está disponibilizado gratuitamente seguindo a licença *Apache* 2.0, encontrando-se disponível para *Windows*, *Linux* e *Mac OS X*.
Esta plataforma de desenvolvimento foi a escolhida para o desenvolvimento da aplicação móvel desenvolvida nesta dissertação.

3.2.2 NetBeans

O *NetBeans* é um IDE gratuito de desenvolvimento de software que suporta linguagens como Java, C, C++, PHP, Ruby. O IDE pode ser executado em plataformas *Windows, Linux* e *MacOS*.
Como ferramenta de desenvolvimento, o *NetBeans* foi utilizado, principalmente, para criar módulos de software e poder executa-los isoladamente. Assim sendo, após a execução, foram devidamente implementadas na plataforma de desenvolvimento *Android Studio*.

3.2.3 SQLite

O SQLite consiste numa biblioteca em linguagem C que implementa uma base de dados SQL embutido. Ao se proceder à utilização da biblioteca SQLite é possível obter acesso à base de dados SQL sem executar um processo SGBD (Sistema de Gestão de Base de Dados) separado.
O SQLite não é uma biblioteca cliente usada para conectar com um grande servidor de base de dados, mas sim o próprio servidor. A biblioteca SQLite lê e escreve diretamente no arquivo de banco de dados no disco.
O SQLite é caracterizado por não serem necessárias dependências externas, não necessitar de instalação, configuração ou administração e ser um mecanismo de armazenamento seguro com transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Por fim, o SQLite apenas suporta bases de dados abaixo de 2 terabytes.

Capítulo 4

Engenharia de Software

4.1 Introdução

A Engenharia de Software é uma área da computação que tem como objetivo a especificação, desenvolvimento e manutenção de sistemas de *software*. Esta é que irá dar as instruções para o desenvolvimento da aplicação, seguindo as especificações necessárias para uma boa construção e que se pensam necessários para o desenvolvimento do sistema.
A Engenharia de Software engloba a utilização de modelos abstratos e precisos que permitem a especificação, projeção, implementação e manutenção do *software*.
Em suma, a Engenharia de Software serve de base para a criação de um sistema, independentemente de qualquer que este seja.
Este trabalho de mestrado consiste no desenvolvimento de um método de criação de *audio fingerprinting*, e a descrição da construção desta aplicação está feita com detalhe no Capítulo 5, uma vez que esta aplicação não apresenta complexidade do ponto de vista de engenharia de software, antes apresenta complexidade algorítmica, e como tal, necessidade de testes e validação.
A base de dados contendo os diferentes *audio fingerprints* será depois usada numa aplicação de teste, cuja natureza é necessariamente simples. Este capítulo descreve a Engenharia de Software da aplicação de teste.

4.2 Enquadramento

Esta dissertação tem como objetivo o desenvolvimento de uma aplicação para dispositivos móveis *Android* que seja capaz de realizar uma análise e identificação de áudio, ou seja, construção de uma ferramenta de *audio fingerprinting*. Para tal, são desenvolvidos dois métodos distintos, de forma a perceber se existe diferença nos resultados obtidos e qual se adequa mais a este problema. De forma a realizar esta tarefa, decidiu-se aplicar apenas para determinados locais que o utilizador frequenta no seu dia a dia para permitir uma melhor perceção da eficácia dos métodos aplicados dado que o número de amostras será menor. Posto isto, com recurso ao microfone existentes nos *smartphones*, o utilizador deverá ser capaz de realizar uma captura de som. Quando esta estiver finalizada, o sistema deve ser capaz de executar a análise da recolha e apresentar, no dispositivo, o resultado da respetiva identificação.

4.3 Requisitos do Sistema

A aplicação desenvolvida deverá ser capaz de realizar os seguintes requisitos de forma a satisfazer todas as necessidades do utilizador:

4.3.1 Requisitos Funcionais

RF1. Realizar capturas de áudio através do dispositivo;

RF2. Capacidade de realizar análises do áudio recolhido;

RF3. Apresentar ao utilizador o ambiente em que este se encontra.

4.3.2 Requisitos Não Funcionais

RNF1. Executar a aplicação num tempo satisfatório;

RNF2. Executar em qualquer dispositivo que possua o SO *Android*;

RNF3. Possuir um aspeto agradável e com uma fácil utilização;

RNF4. Realizar uma metodologia de armazenamento que não sobrecarregue a memória do dispositivo;

4.4 Diagrama Casos de Uso

O utilizador, através da aplicação, terá acesso a algumas funcionalidades sendo elas as seguintes: Realização de uma captura de áudio e a visualização do resultado da análise da captura realizada, ou seja, visualização do ambiente em que está envolvido.

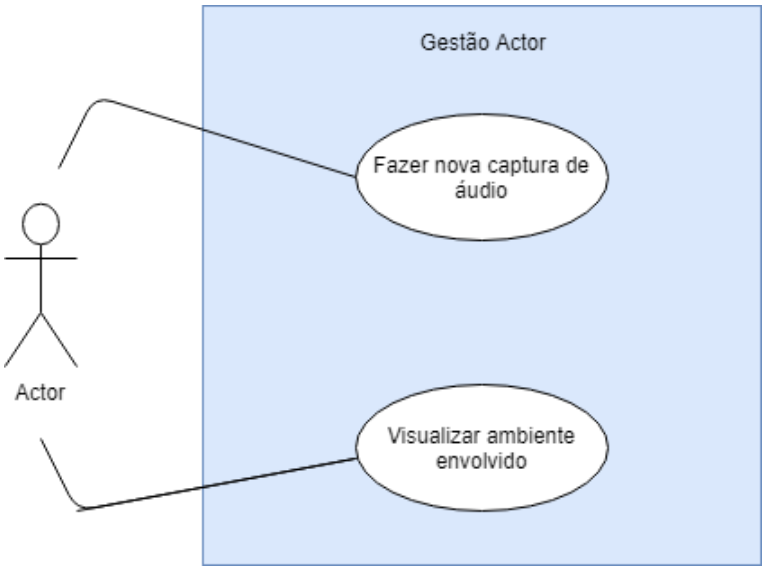

Figura 4.1: Diagrama de Casos de Uso.

4.5 Casos de Uso

4.5.1 Fazer Captura de Áudio

Este caso permite que seja iniciado uma recolha de áudio com recurso ao microfone do *smartphone*, após ser premido o botão que está disponibilizado para o utilizador, como demonstrado no *mockup* apresentado na figura 4.2.

Figura 4.2: *Mockup* página inicial.

Após a realização da captura, o sistema deve ser capaz de realizar a respetiva análise da recolha e a construção de um *audio fingerprint*.

4.5.2 Visualizar ambiente envolvido

Após ter sido terminada a tarefa descrita na secção anterior, o sistema é capaz de realizar uma classificação e apresentar ao utilizador o resultado obtido. Esta tarefa poderá ser observada da seguinte forma, como se visualiza na figura 4.3.

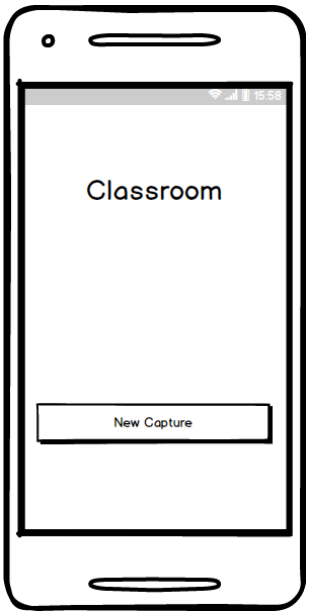

Figura 4.3: *Mockup* visualização do resultado final.

Após a visualização do resultado, o utilizador pode optar pela realização de uma nova captura ao premir o botão presente no *layout*.

4.6 Diagrama Entidade Relação

No contexto do problema apresentado verificou-se necessária a implementação de uma base de dados dada a necessidade de armazenar os dados extraídos das amostras recolhidas. Este modelo apenas será utilizado num dos métodos apresentado dado que, o outro método não tem necessidade de ser suportado por uma base de dados. É extremamente necessário que o modelo de dados construído seja capaz de satisfazer todas as necessidades do sistema, não só no presente bem como no futuro.
Dado este factos, optou-se pela utilização de duas tabelas: Ambiente e Fingerprint. A primeira tem como objetivo registar todos os ambientes que irão fazer parte da amostra e alvo de estudo. Esta tabela possui dois atributos, sendo o primeiro um número inteiro designado por ID, sendo chave primária, irá funcionar como um identificador do ambiente. O outro atributo é designado por nome, que é a designação do ambiente armazenado.
Relativamente à tabela Fingerprint, esta tem como objetivo armazenar as características extraídas a cada recolha de áudio que será utilizada como amostra dos ambientes em estudo, bem como o identificador do ambiente. Assim sendo, esta tabela possuí três atributos. O primeiro tem o nome de ID e é utilizado para a mesma função do ID da tabela anteriormente descrita. Outro atributo é designado como Hash, sendo uma string que tem como objetivo armazenar as características extraídas do ficheiro, neste caso é uma string composta por 8192 bits. O último atributo é o id_ambiente que é utilizado para identificar a que ambiente pertence o *fingerprint* extraído.
Por fim, é possível visualizar, em seguida, o modelo entidade relação construído para esta dissertação.

Figura 4.4: Diagrama Entidade Relação.

Capítulo 5

Implementação

5.1 Estrutura da aplicação

Neste sub-capítulo será apresentada e discutida a estrutura da aplicação desenvolvida nesta dissertação. Na figura 5.1 é apresentado um esquema de um dos métodos que irá ser implementado para a construção deste sistema.

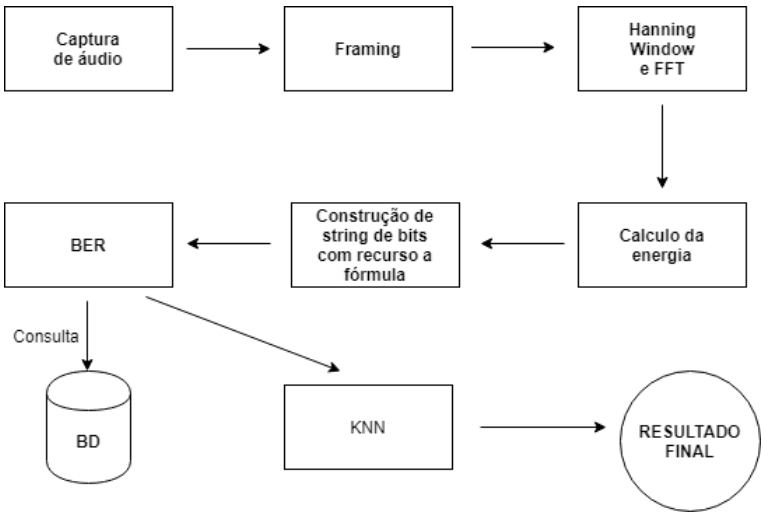

Figura 5.1: Estrutura do 1º método desenvolvido.

Analisando a figura 5.1 é possível verificar a estrutura da aplicação desenvolvida. De forma resumida, o funcionamento da mesma inicia-se com a realização da captura de áudio com recurso ao microfone incorporado nos dispositivos móveis. As recolhas são realizadas com um tamanho de 3 segundos. Após a captura, o ficheiro de 3 segundos é dividido em 8 *frames*, sendo que cada *frame* tem tamanho de 0.37 segundos. Em seguida, aplica-se o processo de *windowing* com recurso à função matemática *Hanning Window*. Após este passo os dados extraídos são transformados em dados com domínio frequência, através da aplicação da FFT.
Concluído o processo anteriormente apresentado, inicia-se a construção do *audio fingerprint*. A caracterização das recolhas é realizada com a construção de *strings* de *bits*. Esta definição é conseguida com recurso à equação 2.13 que tem por base o calculo da energia que pode ser obtida através da fórmula 2.12. Posto isto, são construídos 256 sub-*fingerprints* de 32 *bits*. Em seguida é utilizado o BER para a construção de uma lista de *audio fingerprints* candidatos. Neste passo, o BER recorre à base de dados previamente construída onde estão armazenadas as caracterizações das amostras recolhidas. Em seguida, é aplicado o classificador KNN tendo por base a lista previamente construída.
Relativamente ao segundo método implementado, é apresentado na figura 5.2 um esquema da sua implementação.

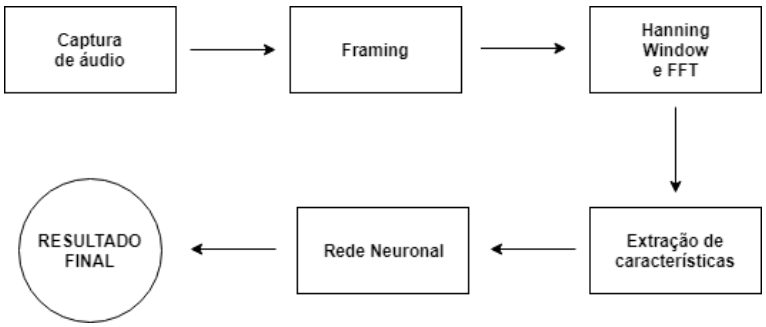

Figura 5.2: Estrutura do 2º método desenvolvido.

Com recurso à figura 5.2 é possível visualizar um esquema do segundo método a teste nesta dissertação. Á semelhança do método anterior, são utilizadas recolhas com um tamanho de 3 segundos. Em seguida, estes dados são divididos por 8 *frames*, observando-se que o tamanho de cada *frame* é de 0.37 segundos. Processa-se, em seguida, a aplicação da função matemática *Hanning Window* e a FFT. Este último passo tem como objetivo a transformação dos dados com o domínio a passar a ser a frequência. Após o processamento dos dados e a segmentação, são extraídas características para ser construído o *audio fingerprint*. As características extraídas foram o valor máximo de cada um dos 8 *frames* e o cálculo do MFCC em todo o ficheiro. Por fim, é realizada a classificação com recurso a uma rede neuronal implementada e anteriormente treinada.

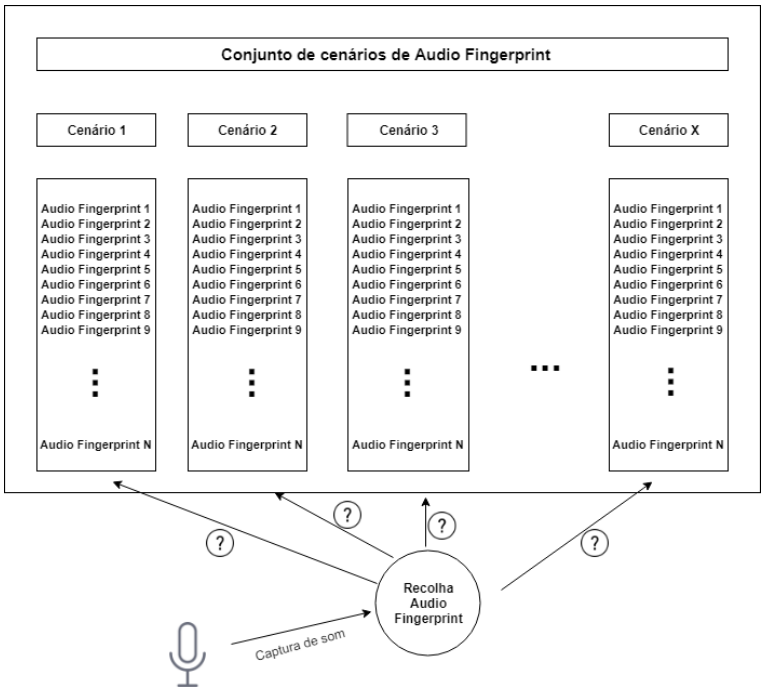

Figura 5.3: Estrutura da base de dados de *Audio Fingerprints* e esquema de funcionamento.

Após verificação do esquema apresentado na figura 5.3, é possível visualizar que após a realização da captura de áudio e a construção do *audio fingerprint* pretende-se realizar a sua classificação. Para tal passo é necessária a consulta de uma base de dados composta por *Audio Fingerprints*. Na base de dados construída existem *N audio fingerprints* em cada um dos *X* cenários. Dado este facto, é realizada uma pesquisa na base de dados de forma a ser encontrado o cenário mais semelhante à captura recolhida. Assim sendo, e com uma explicação superficial,

após se realizar a captura e a construção do seu *audio fingerprint*, são consultados todos os *audio fingerprints* de todos os cenários e é escolhido o que apresentar maior semelhança com o que está submetido a teste. Pretende-se, assim, que a aplicação seja capaz de identificar com um grau de certeza elevado, qual o ambiente físico que o utilizador se situa.

5.2 Organização dos dados recolhidos

Para a realização desta dissertação foi necessária a recolha de amostras dos ambientes físicos que iriam ser submetidos a teste. Só com os *audio fingerprints* das amostras recolhidas é que se torna possível efetuar a classificação e concluir, assim, o objetivo deste trabalho. Durante o desenvolvimento do sistema foi decidido que não iria ser utilizar uma biblioteca de recolhas já existentes mas sim, construir uma nova biblioteca de raiz dos ambientes escolhidos com o tamanho das recolhas a ser de 3 segundos e o mais diversificadas possível dentro do mesmo ambiente. Os ambientes escolhidos para teste deste sistema foram os 7 espaços físicos que estão enumerados em seguida:

- Autocarro;
- Biblioteca;
- Café/Bar;
- Carro;
- Comboio;
- Praia;
- Sala de aula.

A escolha por estes ambientes deve-se ao facto de serem espaços que os possíveis utilizadores frequentam no seu dia a dia e pela divergência do tipo de ambiente entre alguns espaços físicos. Para cada espaço foram capturadas entre 150 e 200 amostras para a construção da base de dados. Com isto, a base de dados construída conta com, aproximadamente, 1300 *audio fingerprints* no seu conjunto.
Para o primeiro método implementado, que necessita de um armazenamento dos dados, optou-se por armazenar os *audio fingerprints* numa base de dados SQLite, ao invés de um armazenamento em ficheiros, dado que um armazenamento em base de dados tem as seguintes vantagens associadas:

- Portabilidade: é compatível com qualquer plataforma dos temos que correm;
- Bom desempenho e estabilidade;
- Maior organização e mais facilidade de acesso aos dados.

Verificadas as vantagens de um armazenamento numa base de dados, esta foi construída seguindo o modelo entidade relação apresentado na figura 4.4, este foi implementado da seguinte forma:

```
CREATE TABLE Ambient(
id integer primary key not null,
nome varchar(30) not null
);

CREATE TABLE Fingerprint (
id integer PRIMARY KEY NOT NULL,
id_ambient integer NOT NULL,
hash longtext (8192) NOT NULL,
FOREIGN KEY (id_ambient) REFERENCES Ambient (id));
```

Listing 5.1: Criação da Base de Dados.

5.3 Detalhes de Implementação

Nesta secção serão apresentados alguns detalhes de implementação dos dois métodos propostos no desenvolvimento do sistema.
Inicialmente, é realizada a captura de áudio através da utilização do microfone presente nos dispositivos, sendo ela armazenada num ficheiro de texto. A captura terá comprimento de 3 segundos dado que diversos estudos defendem que com esta escolha é possível obter resultados bastantes satisfatórios e recolher informação necessária para uma futura avaliação. O ficheiro de texto criado é armazenado numa diretoria automaticamente criada no dispositivo, estando aí disponível enquanto for necessário para a execução da aplicação. Quando esse ficheiro é classificado, optou-se por o eliminar do dispositivo, de forma a não sobrecarregar a memória do mesmo.

```
private Handler Handler = null;
private Runnable runnableCode = null;
private AudioRecord audioRecord;
String data ="";
private static int SAMPLE_RATE_CD = 5000;
private static int audioSource = MediaRecorder.AudioSource.MIC;
private static int channelConfig = AudioFormat.CHANNEL_IN_MONO;
private static int audioFormat = AudioFormat.ENCODING_PCM_16BIT;
private int bufferSizeInBytes;
private int frameByteSize;
private byte sData[];
private static final String PATH_NAME = Environment.
    getExternalStorageDirectory().getAbsolutePath() + "/MsAudioFiles/";

public void onCreate() {
    super.onCreate();
    this.bufferSizeInBytes = AudioRecord.getMinBufferSize(
    SAMPLE_RATE_CD, channelConfig, audioFormat);
    this.frameByteSize = bufferSizeInBytes / 2;
```

```
18      this.audioRecord = new AudioRecord(audioSource, SAMPLE_RATE_CD,
    channelConfig, audioFormat, bufferSizeInBytes);
        this.sData = new byte[frameByteSize];
20      HandlerThread thread = new HandlerThread("MS Sound Capture",
    android.os.Process.THREAD_PRIORITY_BACKGROUND);
        thread.start();
22      Looper serviceLooper = thread.getLooper();
        this.Handler = new Handler(serviceLooper);
24      this.audioRecord.startRecording();

26      this.runnableCode = new Runnable() {
            @Override
28          public void run() {
                writeAudioDataToFile();
30          }
        };
32  }
```

Listing 5.2: Captura de áudio através do dispositivo.

No código apresentado anteriormente é possível visualizar como é que é iniciada a captura de áudio através do microfone dos dispositivos, sendo executado como um serviço. Isso é conseguido com recurso à ferramenta *AudioRecord* disponibilizada pelo *Android*. A captura dos ficheiros é realizada através de um canal *Mono* com frequência 5kHz. Estes dados são armazenados num *array*. Com recurso à função *"writeAudioDataToFile"* esses dados são passados para um ficheiro criado na diretoria indicada na variável *"PATH_NAME"*. Assim sendo, nessa função, o ficheiro é escrito da seguinte forma:

```
try {
2       audioRecord.read(sData, 0, frameByteSize);
      BufferedWriter buf = new BufferedWriter(new FileWriter(file, true));
4       buf.append(Arrays.toString(sData).replace("[", "").replace("]", "")
    .replace(", ", "\n"));
        buf.newLine();
6       buf.close();
}
```

Listing 5.3: Escrever num ficheiro.

Após este passo, é obtido um ficheiro de texto com os dados recolhidos.
Obtido o ficheiro de áudio de 3 segundos, é necessário um processamento do mesmo. Assim, este é segmentado em 8 partes iguais, sendo obtidos 8 *frames*. A cada *frame* é aplicada uma função de "janela" com o objetivo de atenuar as descontinuidades no inicio e no fim de cada um dos 8 *frames* construído. Este processamento tem influência na eficácia da aplicação da transformada de *Fourier* (aplicada após este passo). Para a realização deste processo, optou-se pela utilização da função *Hanning Window*, em vez da *Hamming Window*. Devido à sua definição, esta tem a característica de reduzir a amplitude, permitindo assim uma suavização do sinal. Esta permite, também, a inexistência de descontinuidades de sinal dado que, pela sua definição, nas duas extremidades tem valor zero. De forma a comprovar isto, a figura 5.4 demonstra

o comportamento desta função num determinado espaço de tempo, neste caso, 1 segundo.

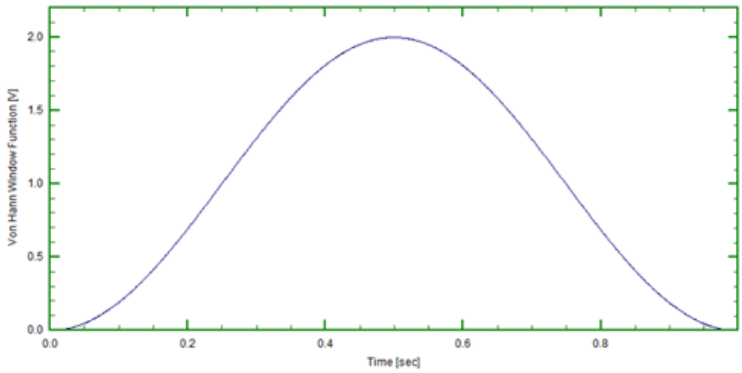

Figura 5.4: Representação da função *Hanning Window*.

Assim, a função *Hanning Window* pode ser definida da seguinte forma:

```
protected float value(int length, int index) {
    return 0.5f - 0.5f * (float)Math.cos ((TWO_PI * index )  / (length
    -1f));
}
```

Listing 5.4: *Hanning Window*.

A geração de uma função *Hanning Window* com tamanho *length* é conseguida através do seguinte método que implementa a função anteriormente apresentada, em cada posição do *array samples* com tamanho *length*.

```
public float[] generateCurve(int length) {
    float[] samples = new float[length];
    for (int n = 0; n < length; n++) {
        samples[n] = 1f * value(length, n);
    }
    return samples;
}
```

Listing 5.5: Aplicação da *Hanning Window*.

Após este passo, é aplicada a transformada de *Fourier*, neste caso a FFT, em cada um dos 8 *frames* já criados. O objetivo da sua aplicação é a transformação dos dados em domínio de frequência ao invés do tempo. Com este método é conseguida uma redução do ruído existente nos dados. A FFT foi aplicada da seguinte forma:

```
WindowFunction hann= new HannWindow();
FFT fft= new FFT(frame1.length, hann);
fft.forwardTransform(frame1);
fft.forwardTransform(frame2);
fft.forwardTransform(frame3);
```

Listing 5.6: Aplicação da FFT.

Concluído este processo e obtidos dados com o domínio de frequência, processa-se à caracterização das recolhas. Dado que, após este passo, os dois métodos implementados têm abordagens diferentes, explicar-se-ão os mesmos em secções diferentes.

5.3.1 1º Método:

Para o processo de construção do *audio fingerprint* e respetiva análise, optou-se pelo método proposto em [HK02]. Pretende-se, então, construir uma *string* com 8192 *bits* e que funcionam como um *hash* criptográfico dessa recolha. O objetivo passa pela construção de 256 "*sub-fingerprints*", cada um composto por 32 *bits*, obtendo-se no final a referida *string* composta por 8192 *bits*. A atribuição dos *bits* consegue-se com o recurso ao calculo da energia de cada "*sub-fingerprint*", como é possível visualizar na fórmula 2.13, sendo a energia obtida através da equação apresentada na fórmula 2.12.
Sendo o cálculo da energia definido com a função *"calculateEnergy"*, esta é calculada da seguinte forma:

```
public static float calculateEnergy(float[]input, int i, int j){
float energy=0;
int aux=i;
for(int h=i;h<j;h++){
    energy+=input[h];
}
energy=(float)Math.pow(energy, 2);
int N=j-aux;
energy=energy/N;
return energy;
}
```

Listing 5.7: Cálculo da Energia.

Dado que são necessários 256 "*sub-fingerprints*", é necessário "dividir" cada um dos 8 *frames* já existentes em 32 partes dado que, $32 * 8 = 256$. É com essa intenção que as variáveis *i* e *j*, passadas por parâmetro, são utilizadas. Assim sendo, os dados do *array input*, que corresponde a um dos 8 *frames*, são utilizados e é calculada a energia desse "*sub-frame*" com inicio em *i* e termino em *j*. Assim sendo, são construídos 32 "*sub-fingerprints*" de 32 bits em cada *frame* já existente. Em seguida, é aplicada a fórmula já referida usufruindo do cálculo da energia, anteriormente explicada. Caso a energia seja maior que 0 é atribuído o *bit* "1", acontecendo o oposto é entregue o *bit* "0". Para isso, é utilizada a função "*stringCaractization*" que tem o papel de realizar o papel anteriormente descrito e retornar a *String* construída.

```
public static String stringCaractization(float[]frame, float[] frame2,
    String finalString){
int aux3=frame.length/1024;
int flag=0;
float energy=0;
float auxEnergy1, auxEnergy2;
for(int h=0; h<32;h++){
```

```
    auxEnergy1=calculateEnergy(frame, aux3*flag, aux3*(flag+1))-
    calculateEnergy(frame, aux3*(flag+1), aux3*(flag+2));
    auxEnergy2=calculateEnergy(frame2, aux3*flag, aux3*(flag+1))-
    calculateEnergy(frame2, aux3*(flag+1), aux3*(flag+2));
    energy=auxEnergy1-auxEnergy2;
    if(energy>0)
        finalString=finalString+"1";
    else
        finalString= finalString + "0";
}
```

Listing 5.8: Construção do "*hash*" da captura de áudio.

Após ser efetuada a caracterização da recolha, é necessário proceder-se à classificação do mesmo. Neste passo optou-se por conjugar a utilização do BER e do KNN. O primeiro consiste na realização de uma comparação entre a *string* construída, de 8192 *bits*, para a recolha e as que estão armazenadas na base de dados construída previamente. O resultado devolvido é o número de *bits* diferentes. A contagem do número de *bits* diferentes entre duas *strings* é efetuada da seguinte forma:

```
public static int countDiff(String string1, String string2){
int count=0;
for (int i = 0; i < string1.length(); i++) {
    if(!(string1.substring(i, i+1).equals(string2.substring(i, i+1))))
        count++;
}
return count;
}
```

Listing 5.9: Contagem do número de *bits* diferentes na comparação de duas *strings*.

Uma das opções de pesquisa seria o cálculo da diferença, seguindo o processo apresentado anteriormente, para todos os *audio fingerprints* presentes na base de dados. Isto consistia em comparar o *audio fingerprint* construído com toda a base de dados. Obtidas as diferenças para todos os *audio fingerprints* presentes na base de dados, estas são inseridas numa *ArrayList* sendo, posteriormente, ordenados por ordem crescente tendo como referência as diferenças anteriormente calculadas. Esse processo é realizado da seguinte forma:

```
Fingerprints=databaseAccess.getAllFingerprints();
ArrayList<DiffFingerprints> Diferencas = new ArrayList<DiffFingerprints
    >();
int minDiff=0;
int posMin=0;
int diff=0;

for (int i = 0; i < Fingerprints.size(); i++) {
    diff=countDiff(string1, Fingerprints.get(i).getHash());
    DiffFingerprints diferenca= new DiffFingerprints(i,Fingerprints.get
    (i).id_ambient, diff);
```

```
10      Diferencas.add(diferenca);
    }
12  Collections.sort(Diferencas, new Comparator<DiffFingerprints>() {
    @Override
14      public int compare(DiffFingerprints p1, DiffFingerprints p2) {
            return new Integer(p1.diff).compareTo(p2.diff);
16      }
    });
18  }
```

Listing 5.10: Contagem do número de *bits* diferentes na comparação de duas *strings* e respetiva ordenação.

Sendo o *array Fingerprints*, o conjunto de todos os dados recolhidos existentes na base de dados, este é utilizado para a realização da pesquisa. Cada elemento existente em *Fingerprints* é comparado com o construído para a atual captura com recurso à função *counDiff*.
Obtidas as diferenças e já ordenadas, implementa-se o classificador KNN. Para tal são selecionados os $7 * K + 1$ elementos mais semelhantes, ou seja, os que possuem menor diferença em comparação com o *audio fingerprint* da captura. Opta-se por escolher este número de elementos dado que são submetidos a teste 7 ambientes e imaginemos que $K = 3$. No pior dos casos, podem ser necessários 15 *audio fingerprints* pois, caso os outros 6 ambientes vejam correspondidos 2 *audio fingerprints* é o 15º *audio fingerprint* mais próximo que irá tomar a decisão de qual ambiente a ser escolhido e, assim, finalizar a classificação. Em seguida, é possível visualizar como é feita a verificação se um ambiente já possui K correspondências.

```
for (int i=0;i<lenAmbients.length;i++){
2       if (lenAmbients[i]==1){
            total[0]++;
4           if(total[0]==k) {
                ambiente = 1;
6           break;
        }
8   }
```

Listing 5.11: Excerto da implementação do classificador KNN.

Outra opção testada pode ser definida como sendo a construção de uma *substring* para a captura e para todos os *audio fingerprints* existentes na base de dados. Por parâmetro é passada uma variável inteira designada por *size*. Assim sendo, a *string* é composta pelos elementos $size * n$ a $size * n + 1$, sendo *n* um número par ou 0. Imaginemos que $size = 20$, as *substrings* iriam ser compostas com os *bits* nas posições de 0 a 20, de 40 a 60, 80 a 100 e assim sucessivamente. Isto permite que o número de *bits* a comparar seja menor e, possivelmente, uma pesquisa mais rápida.

```
public String windowFrameBER(String string1, int size){
2   ArrayList<Fingerprint> Fingerprints= new ArrayList<Fingerprint>();
    Fingerprints=databaseAccess.getAllFingerprints();
4   String finalString1="";
    String finalString2="";
6   int flag;
```

```
for (int j = 0; j < Fingerprints.size(); j++) {
    finalString1="";
    finalString2="";
    flag=0;
    for(int i=flag;flag+(2*size)<string1.length();i+=flag+(2*size)){
        String aux=Fingerprints.get(j).getHash().substring(flag, flag
   +(2*size)-1);
        finalString1=finalString1.concat(aux);
        finalString2=finalString2.concat(string1.substring(flag, flag
   +(2*size)-1));
        flag=flag+(2*size);
    }
}
}
```

Listing 5.12: Constrção de *substrings* para o método anterior.

Em seguida, procede-se da mesma forma que a primeira opção apresentada para a classificação, com recurso ao KNN, ou seja, é construída uma lista com os $7 * K + 1$ *audio fingerprints* mais semelhantes ao da captura e em seguida aplica-se o KNN.

Dado que, teoricamente, o uso dos métodos anteriores iriam sobrecarregar bastante a memória do dispositivo e o tempo de pesquisa seria bastante elevado, com crescimento exponencial, optou-se pela implementação de outros métodos de forma a realizar os testes. Estes métodos têm como objetivo a construção de uma lista de candidatos com recurso à utilização do BER e, em seguida, procede-se ao uso do classificador KNN. A ideia consistiu em construir um *array* de *audio fiingerprints* candidatos. Esses candidatos são escolhidos com a existência de um padrão com o *audio fingerprint* submetido a teste. Foram testados diversos padrões, sendo eles descritos em seguida.

O primeiro método testado consiste em construir uma *substring* da captura a teste, com tamanho *size* dos primeiros *size bits* presentes no seu *audio fingerprint*. Em seguida, é procurado na base de dados *audio fingerprints* com início exatamente igual à *substring* construída, como se visualiza na figura 5.5.

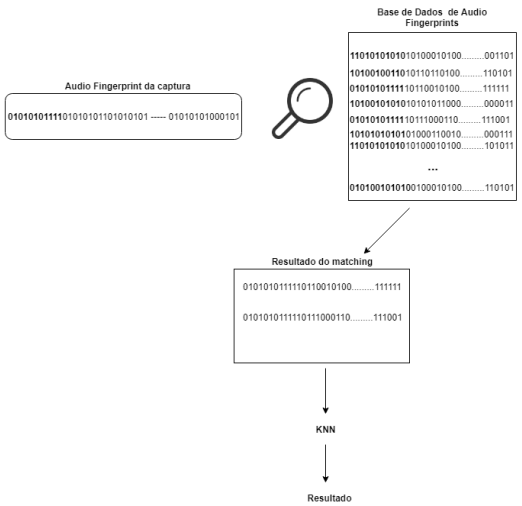

Figura 5.5: Procura de *substrings* no inicio de um *audio fingerprint*.

Na figura anterior é possível visualizar a construção de uma *substring* (representada a negrito) e em seguida é procurada na base de dados *audio fingerprints* com inícios iguais. Esse resultado é obtido da seguinte forma:

```
String query = "SELECT  * FROM  Fingerprint where hash Like '" +
    subString + "%';";
```

Listing 5.13: Procura de uma *substring* no inicio do *audio fingerprint*.

Caso não seja encontrado nenhum resultado, a variável *size* é decrementada em uma unidade até ser encontrado algum resultado. Após serem obtidos os resultados, é aplicado o classificador KNN apenas nos *audio fingerprints* selecionados na filtragem anterior. Com isto, é devolvido o ambiente do *audio fingerprint* mais semelhante.
O seguinte método é bastante semelhante ao que foi descrito anteriormente. São extraídos os primeiros *size bits* do *audio fingerprint* submetido a teste mas, a pesquisa em vez de ser no início das *string* presentes na base de dados, é procurada em qualquer parte da dos *audio fingerprints*, como é exemplificado na figura 5.6.

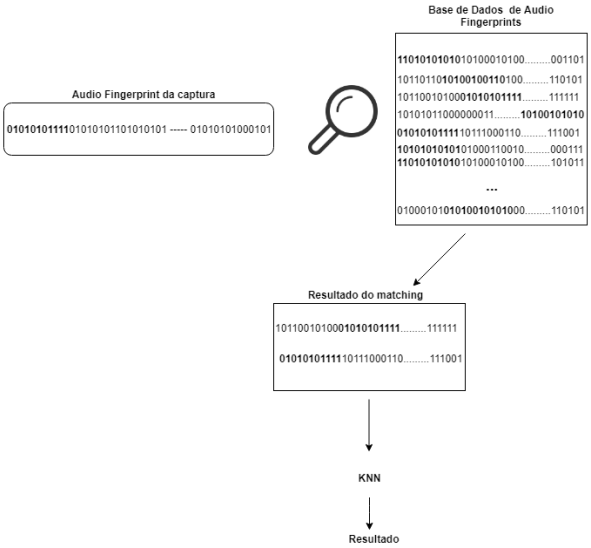

Figura 5.6: Procura de *substrings* iguais ao inicio do *audio fingerprint* da captura, em qualquer parte dos que estão armazenados na base de dados.

Na figura anterior é possível verificar que é construída uma *substring* e é procurada, na base de dados, em qualquer parte dos *audio fingerprints* aí presentes. Pesquisa é efetuada da seguinte forma:

```
String query = "SELECT  * FROM  Fingerprint where hash Like '%" +
  subString + "%';";
```

Listing 5.14: Procura de uma *substring* em qualquer parte do *audio fingerprint*.

Após este passo, realiza-se exatamente o mesmo passo que o método anterior para realizar a classificação.
O último método submetido a teste consiste em construir uma *substring* de tamanho *size* com os *bits* presentes, exatamente, no meio do *audio fingerprint* construído para a captura a teste. Em

seguida, procura-se por um segmento de *bits* exatamente igual ao da *substring* construída sendo devolvido uma lista com os candidatos. Após este passo, processa-se à utilização do classificador KNN, como nos métodos anteriores, na lista dos *audio fingerprints* candidatos. Este processo está exemplificado na figura 5.7.

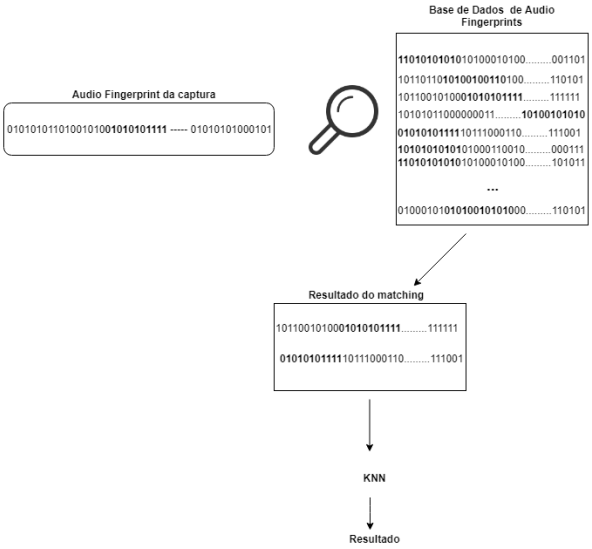

Figura 5.7: Procura de *substrings*, na base de dados, em qualquer parte do *audio fingerprint* que seja igual ao *bits* presentes a meio do *audio fingerprint* submetido a teste.

5.3.2 2º Método

Este método aplicado tem uma metodologia diferente do que foi descrito anteriormente. Desde logo pelas características que foram escolhidas para a construção do *audio fingerprint*. Ao invés da utilização de uma *String* de bits, como foi escolhido no outro método, optou-se pela extração de valores numéricos dos dados recolhidos e previamente tratados. A primeira opção e extração prende-se com a extração do valor máximo existente em cada um dos 8 *frames*. Este valor, além se ser o valor máximo presente no *frame*, pode significar o maior pico aí presente devido ao seu valor. Um pico define-se como sendo o valor máximo local, ou seja, os valores presentes à sua direita e esquerda são menores que ele. Assim, são extraídos 8 picos presentes no ficheiro de 3 segundos. Este processo é conseguido da seguinte forma:

```
public static float calcMax(float[]data){
    ArrayList<Float> dataList= new ArrayList();
    for(int i=0; i<data.length;i++){
        dataList.add(data[i]);
    }
    float max=Collections.max(dataList);
    return max;
}
```

Listing 5.15: Cálculo dos valores máximos presentes num *frame*.

Obtidos os 8 picos presentes em 3 segundos, procede-se à extração de 26 coeficientes MFCC. Estes são obtidos seguindo os seguintes passos:

```
 MFCC mfcc = new MFCC(5000, 16384, 26, true);
double[] input_mfcc = new double[16384];
for(int i = 0; i < 16384; i++) {
    input_mfcc[i] = data[i];
}
double[][] mfcc_coefs = mfcc.process(input_mfcc);
```

Listing 5.16: Extração dos 26 coeficientes MFCC.

Após serem obtidos as características escolhidas, um total de 34, pretende-se a realização da classificação. Para este passo foi treinada uma rede neuronal utilizando a *framework Encog*. O *Encog* é uma estrutura para *machine learning* disponível para Java e .Net. Este, possuí diversos algoritmos de aprendizagem automática mas o que se destaca mais são as suas redes neuronais.

Para uma rede neuronal ter a capacidade de efetuar testes e ser capaz de obter bons resultados a nível de classificação, esta tem de passar por um processo de treino, onde lhe são disponibilizados algumas amostras e a rede ”aprende” com base nesses dados. De forma a fornecer os dados no formato ideal para teste, são inseridas as 34 características num ficheiro .csv. À frente das características tem de estar presente um identificador do ambiente correspondente. Essa identificação é efetuada com 7 bits atribuídos a cada ambiente, sendo que 6 têm o valor ”0” e apenas um tem o valor ”1”. O valor ”1” apenas está presente na posição que corresponde a esse ambiente, por exemplo: o ambiente bar é identificado como ”1000000”, a praia como ”0100000”, autocarro ”0010000” e assim sucessivamente. Podemos, assim, construir um ficheiro .csv que terá um formato semelhante ao seguinte:

```
160.38910708736734,-25.87021206261843,-38.12655274223837,
2.451338940886094,-15.383032355404907,1.8615082680816286,
-8.555939271574204,1.1233200055300534,-5.481464923773832,
0.5043878269119197,-3.5755869223384247,0.41031208482811277,
-2.488244245640113,0.2833129925468083,-1.3396151996605021,
0.4426641614366158,-0.9661613725843667,-0.14947029458263095,
-0.7340574814604401,-0.10596460182684908,-3.4807610113236203E-13,
0.10596460182682177,0.734057481460532,0.14947029458263494,
0.9661613725843983,-0.44266416143665976,5502.1147,4840.0024,
6751.504,5787.274,4593.2305,4673.2095,5084.695,5173.758,1,0,0,0,0,0,0
```

Listing 5.17: Exemplo de uma linha do ficheiro .csv para treino da rede neuronal.

podendo-se observar que cada dado inserido é separado por uma virgula, de forma a separar-lo e identificá-lo.
Posto isto, processa-se o já mencionado treino da rede neuronal. O treino de uma rede prende-se com esta a aprender com os dados inseridos e, esse treino, é realizado até ser obtido um erro escolhido pelo utilizador, neste caso 10%.

```
String trainingSetFileName = "training/soundTrainingMaxMFCC.csv";
String trainingSetFileNameSaved = "test/sound.eg";
int inputsCount = 34;
int outputsCount = 7;

public void run() {
    MLDataSet trainingSet = TrainingSetUtil.loadCSVTOMemory(CSVFormat.
    ENGLISH, trainingSetFileName, false, inputsCount, outputsCount);
    FeedForwardPattern pattern = new FeedForwardPattern();
    pattern.setActivationFunction(new ActivationSigmoid());
    pattern.setInputNeurons(inputsCount);
    pattern.setOutputNeurons(outputsCount);
    BasicNetwork feedforwardNetwork = (BasicNetwork) pattern.generate()
    ;
    Train trainMain= new ResilientPropagation(feedforwardNetwork,
    trainingSet, 0.01, 0.001);
    int epoch = 0;
    do {
        trainMain.iteration();
        epoch++;
    } while (trainMain.getError()>0.1 && epoch <30000);
    saveObject(new File(trainingSetFileNameSaved), feedforwardNetwork);
    Encog.getInstance().shutdown();
}
```

Listing 5.18: Treino da rede neuronal

No código anteriormente apresentado é possível visualizar todos os processos que são efetuados para a realização do treino da rede neuronal. Inicialmente, é possível visualizar a construção de um *MLDataSet* para treino com a definição de 34 *inputs*, as características, e 7 *outputs*, a definição do ambiente. Após isso, é construída uma *feedForwardPattern* com a utilização de uma função sigmoide na camada oculta. Uma rede *feed Forward* define-se como sendo uma rede neuronal que apenas se move numa direção, em frente, não realizando um ciclo. Por fim, realiza-se o treino através de *ResilentPropagation* com um $learningRate = 0.01$ para que a rede aprenda lentamente e não depressa demais, pois isso pode interferir no desempenho futuro da rede neuronal. São realizadas iterações na rede até que o erro atinja os 10%, sendo guardado após a obtenção desta meta, num ficheiro, a definição da rede.
Tendo a rede neuronal treinada, está pronta a ser submetida a teste. Os testes podem ser realizados da seguinte forma.

```
String trainingSetFileNameSaved = "test/sound.eg";
String teste ="testing/BarMFCC.csv";
int inputsCount = 34;
int outputsCount = 7;

public void run() {
    MLDataSet testSet = TrainingSetUtil.loadCSVTOMemory(CSVFormat.
```

```
    ENGLISH, teste, false, inputsCount, 0);
     BasicNetwork loadedNetwork = (BasicNetwork) loadObject(new File(
    trainingSetFileNameSaved));
     boolean adlFound=false;
     for(MLDataPair pair : testSet){
         final MLData output =loadedNetwork.compute(pair.getInput());
         if(output.getData(0)>= output.getData(1)
             && output.getData(0) >= output.getData(2)
             && output.getData(0) >= output.getData(3)
             && output.getData(0) >= output.getData(4)
             && output.getData(0) >= output.getData(5)
             && output.getData(0) >= output.getData(6)
             && output.getData(0) >0){
                 System.out.println("Bar");
                 adlFound=true;
         }
     }
```

Listing 5.19: Realização de testes à rede neuronal

Segundo o excerto de código anterior, começa-se por abrir o ficheiro que define a rede neuronal, após ter sido treinada. Em seguida, cria-se um *MLDataPair* com, apenas, as características dos ficheiros que estão submetidos a teste, seguindo-se o processamento deles através da rede neuronal, esta retornando 7 valores. Estes valores, compreendidos entre 0 e 1, são as probabilidades de ocorrer cada um dos ambientes, sendo que a primeira corresponde ao bar, a segunda à praia e assim sucessivamente. Obtidas as probabilidades, verifica-se qual ambiente tem uma maior probabilidade, face aos dados submetidos a teste, terminando aqui o processo de classificação.

5.4 Manual de instalação

De forma a ser instalada a aplicação deverá ser executado no dispositivo o ficheiro ".apk" da aplicação desenvolvida. A instalação desta aplicação apenas deverá ser testada em dispositivos suportados pelo SO *Android* dado que foi para este ambiente que a aplicação foi desenvolvida. De forma a ficar concluída a aplicação, o utilizador deverá dar permissão para a possibilidade de armazenamento no dispositivo e para recolha de áudio através do microfone presente no dispositivo. Concluídos estes passos, a instalação deverá ser bem sucedida e a aplicação estará pronta a ser utilizada.

5.5 Manual de utilização

Após a aplicação entrar em execução é apresentada a página inicial onde estão disponíveis para o utilizador dois botões: iniciar captura e sair da aplicação. É apresentada ainda uma mensagem com o conteúdo "*Waiting*" de forma a informar o utilizador que a aplicação está pronta a ser utilizada e que está à espera de alguma ação por parte dele (Ver figura 5.8).

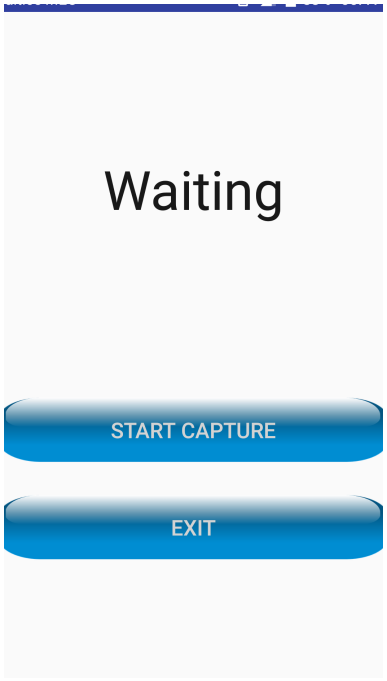

Figura 5.8: Página inicial da aplicação.

Caso prima o 2º botão, a aplicação é terminada, caso contrário, é iniciada a captura de áudio para teste. O utilizador é avisado do inicio deste processo através de um *Toast* e com a mensagem apresentada ser "*Recording*". Verifica-se ainda que os botões ficam "bloqueados" para o utilizador enquanto o sistema não terminar a recolha e a sua classificação de forma a não ser interrompido nenhum processo que esteja em execução (Ver figura 5.9).

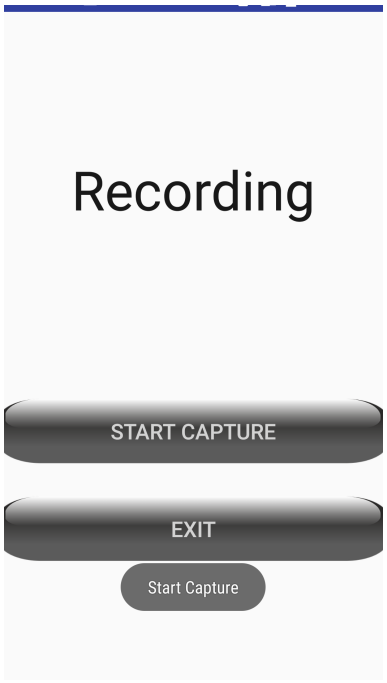

Figura 5.9: Realização de uma recolha.

Terminando este processo é apresentada a avaliação realizada pelo sistema, neste caso foi classificado como "*Library*", e é dada a possibilidade ao utilizador de sair da aplicação ou iniciar

uma nova captura repetindo-se, neste caso, o procedimento anteriormente descrito (Ver figura 5.10).

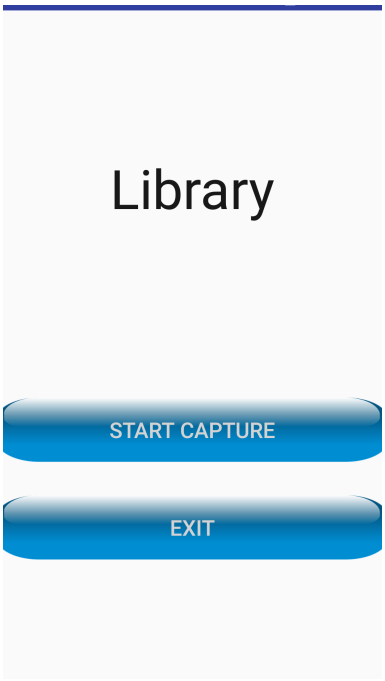

Figura 5.10: Apresentação do resultado.

Capítulo 6

Testes e Resultados

Este capítulo tem como objetivo a apresentação de todos os testes realizados e os respetivos resultados. Pretende-se, a realização de uma discussão dos resultados obtidos. Inicialmente, irão ser apresentados os testes e resultados do sistema desenvolvido e, por fim, a verificação dos requisitos propostos no capítulo 4.

6.1 Resultados da Aplicação

No desenvolvimento desta dissertação, optou-se pela implementação de dois métodos distintos para ser possível a obtenção de uma perceção do mais adequado para este temática e em que situações estes apresentam fragilidades. Dado que o processo de segmentação ser o mesmo e os métodos se basearem nas frequências obtidas, as diferenças entre eles estão presentes na construção do *audio fingerprint* e na sua classificação. Enquanto o primeiro método construído usufruí de características não semânticas e beneficia da construção de uma *String* de 8192 *bits*, que pode facilitar o processo de classificação, o segundo método é construído com características matemáticas com recurso à extração dos picos e dos MFCC. Também no processo de classificação se verificam enormes divergências: no primeiro usufruí-se do BER e, após esse processo, utiliza-se o classificador KNN enquanto que, no segundo processo têm-se por base a utilização de *machine learning* com o treino e utilização de uma rede neuronal. É importante referir que os dados utilizados para treino e para teste dos dois sistemas, foram exatamente os mesmos. Estes dados foram recolhidos através de um *smartphone*. Dada a diversidade dos métodos desenvolvidos, os resultados dos mesmos serão apresentados em secções distintas.

6.1.1 1º Método

O primeiro método apresentado tem como recurso, para a construção do *audio fingerprint*, o cálculo da energia. É com base nesse cálculo que o *audio fingerprint* é construído e obtida a *String* de 8192 bits. Este é o processo realizado para a geração do *audio fingerprint*. Para a realização da classificação é utilizado o BER para a construção de uma lista de candidatos e, em seguida, é aplicado o classificador KNN. Como apresentado no capítulo 5, foram desenvolvidas diversas formas de construir a lista de candidatos, sendo elas testadas em seguida e apresentados os seus resultados. Dado que, a lista de candidatos, normalmente, tem um número reduzido de *audio fingerprints*, em média entre 20 e 30, optou-se por utilizar $K = 15$, ou seja, a pesquisa é realizada apenas nos 15 "vizinhos" mais próximos. Nas formas testadas em que não é realizada uma pré-seleção, optou-se por utilizar, igualmente, $K = 15$ para ser possível analisar o desempenho, das diferentes formas nas mesmas condições. De forma a tornar a apresentação dos resultados mais clara, estes serão divididos em dois grupos: o primeiro com os dois métodos que não apresentam nenhuma forma de filtragem, ou seja, consideram toda a base de dados e a pesquisa é realizada sobre ela. O outro grupo apresenta os resultados das três formas testadas para construir a lista de *audio fingerprints* candidatos.

6.1.1.1 Sem lista de candidatos

Neste grupo estão presentes os dois métodos que foram testados em que não existia uma lista de candidatos e, assim sendo, era calculado o BER para toda a base de dados. Apresentando, os resultados da primeira forma apresentada, na tabela 6.1, em que esta consistia na utilização de todos os *audio fingerprints* construídos.

Tabela 6.1: Matriz de confusão para a utilização de todos os *audio fingerprints* e estes com 8192 *bits*.

	Bar	**Praia**	**Autocarro**	**Carro**	**Sala de aula**	**Biblioteca**	**Comboio**	**TAXA ACERTO**
Bar	**13**	2	1	9	2	1	2	43%
Praia	3	**11**	1	2	4	6	3	37%
Autocarro	2	1	**10**	4	2	3	8	33%
Carro	6	1	4	**10**	1	3	5	33%
Sala de Aula	4	1	3	2	**12**	7	1	40%
Biblioteca	1	4	6	2	6	**10**	1	33%
Comboio	0	2	6	4	2	4	**12**	40%
TAXA ACERTO	44%	50%	32%	30%	41%	29%	38%	**37%**

É possível verificar que, após a realização dos testes verifica-se uma taxa de acerto de 37%, com um tempo de procura, num *smartphone*, a rondar os 12 segundos. É um tempo bastante elevado para a realização da pesquisa. Este acontecimento deve-se ao facto de ser utilizada toda a base de dados para a construção a classificação. Relativamente aos resultados obtidos, pode-se verificar que existe uma ”confusão” do sistema entre os meios de transporte testados (autocarro, carro e comboio). Este facto pode dever-se à semelhança do ruído existente nestes ambientes. Verifica-se, também, que existe uma relação entre o Bar e o Carro. Por fim, é possível visualizar resultados, relativamente, semelhantes entre a biblioteca e uma sala de aula.

O segunda forma apresentada é a utilização de metade dos *bits* presentes num *audio fingerprint*, ou seja, 4096 *bits*. No entanto, continuam a ser utilizados todos os *audio fingerprints* presentes na base de dados. Neste, foram obtidos os resultados apresentados na tabela 6.2.

Tabela 6.2: Matriz de confusão para a utilização de todos os *audio fingerprints* e estes com 4096 *bits*.

	Bar	Praia	Autocarro	Carro	Sala de aula	Biblioteca	Comboio	TAXA ACERTO
Bar	**11**	3	0	8	3	2	3	37%
Praia	2	**12**	1	3	5	3	4	40%
Autocarro	0	2	**9**	5	4	4	6	30%
Carro	7	0	4	**11**	1	4	3	37%
Sala de Aula	3	2	2	4	**11**	7	1	37%
Biblioteca	1	5	6	3	4	**11**	2	37%
Comboio	1	1	5	2	3	5	**13**	43%
TAXA ACERTO	44%	48%	33%	31%	38%	31%	41%	**37%**

Com esta forma de pesquisa, visualizam-se resultados bastante semelhantes aos que foram obtidos no método anterior. Ao contrário do que podia ser expectável, para a realização da pesquisa, este método utilizava, em média 20 segundos: números nada aceitáveis para uma pesquisa num telemóvel pois procuram-se resultados imediatos. Era expectável um tempo de execução menor dado que os *bits* a serem comparados eram metade do que no outro método. No entanto, durante a construção da *substring* verifica-se um enorme dispêndio de tempo. Relativamente aos resultados obtidos, além de a taxa de acerto ser igual ao método anterior, verificam-se os mesmos padrões em termo de classificação.
Realizando uma breve reflexão sobre o desempenho dos dois métodos, é possível verificar que o tempo utilizado para a realização da pesquisa não é aceitável para um telemóvel, dadas as exigências dos utilizadores. A taxa de acerto verificada nos dois métodos é negativa. Uma das razões para tal acontecimento prende-se com o facto se existir alguma semelhança entre os ambientes escolhidos para teste.

6.1.1.2 Com lista de candidatos

Nesta secção estão presentes os resultados que se visualizaram ao testar este método com a construção de uma lista de candidatos, ou seja, é efetuada uma pré seleção antes de ser aplicado o classificador. Para tal, são construídas *substrings* e é procurado na base de dados excertos iguais. O tamanho das *substrings* utilizado foi o de 20 *bits*, para todas as formas utilizadas para a construção da respetiva lista. A primeira forma de construção da lista de candidatos utilizada foi a verificação de quais *audio fingerprints* tinham um inicio exatamente igual ao que está submetido a teste. Com a sua utilização foram obtidos os resultados que é possível visualizar na tabele 6.3.

Tabela 6.3: Matriz de confusão para a construção da lista de candidatos com base nos inícios dos *audio fingerprints*.

	Bar	**Praia**	**Autocarro**	**Carro**	**Sala de aula**	**Biblioteca**	**Comboio**	**TAXA ACERTO**
Bar	**14**	1	2	7	3	1	3	47%
Praia	1	**16**	2	3	3	2	3	53%
Autocarro	1	2	**15**	1	2	2	7	50%
Carro	6	0	2	**17**	1	2	2	57%
Sala de Aula	2	1	1	4	**17**	4	1	57%
Biblioteca	0	1	2	1	3	**22**	1	73%
Comboio	1	0	5	4	1	3	**16**	53%
TAXA ACERTO	56%	76%	52%	41%	57%	61%	48%	**56%**

Com esta forma de a lista de candidatos ser construída verifica-se um aumento da taxa de acerto em comparação às técnicas anteriormente apresentadas, apresentando-se positiva: cerca de 56%. Apesar de positiva, continua a ser uma percentagem baixa para a avaliação dos sistema. Continua-se a verificar uma taxa negativa para a classificação do ”Bar”. Uma das razões pode ser a existência de enorme ruído e levar à ineficácia do método. Relativamente aos meios de transporte, os resultados apesar de serem positivos, continua a verificar-se uma ”confusão” entre eles. A exceção encontra-se na classificação do ”Carro” que tem algumas classificações atribuídas ao ”Bar”. Uma razão para tal facto pode dever-se ao facto de grande parte das recolhas efetuadas no ambiente ”Carro” estar presente música de fundo. Destacar, o bom funcionamento do método relativamente ao ambiente ”Biblioteca” que apresenta uma taxa de acerto de 73%. Por fim, o método de pesquisa é executado, em média, em 3 segundos

A segunda forma testada para construção da lista de *audio fingerprints* candidatos consiste em procurar a *substring* inicial do *audio fingerprint* submetido a teste e verificar se se encontra algum exatamente igual, em qualquer parte, nos *audio fingerprints* presentes na base de dados. Os resultados obtidos neste teste podem ser visualizados na tabela 6.4.

Tabela 6.4: Matriz de confusão para a construção da lista de candidatos com base na procura de padrões iguais ao inicio do *audio fingerprint* submetido a teste.

	Bar	Praia	Autocarro	Carro	Sala de aula	Biblioteca	Comboio	TAXA ACERTO
Bar	**12**	2	3	6	4	1	3	40%
Praia	1	**15**	3	3	2	2	4	50%
Autocarro	1	2	**16**	2	1	2	6	53%
Carro	7	0	2	**18**	0	2	1	60%
Sala de Aula	3	1	2	3	**15**	5	1	50%
Biblioteca	1	0	1	2	4	**21**	1	70%
Comboio	1	0	6	4	1	2	**16**	53%
TAXA ACERTO	46%	75%	48%	47%	57%	60%	50%	**54%**

Com visualização da tabela 6.4 é possível constatar que os resultados são relativamente semelhantes aos apresentados na forma anterior, na tabela 6.3. No entanto, o tempo de execução deste método é mais elevado em comparação ao anterior, em média, um tempo total de 6 segundos.
Por fim, é testada uma quinta e última forma de ser construída a lista de candidatos. Esta consiste em procurar, nos *audio fingerprints* presentes na base de dados, os que têm um padrão igual aos *bits* presentes a meio do *audio fingerprint* gerado para a captura que está a ser testada. com isso, os resultados obtidos podem ser observados na tabela 6.5.

Tabela 6.5: Matriz de confusão para a construção da lista de candidatos com base na procura de padrões iguais ao inicio do *audio fingerprint* submetido a teste.

	Bar	Praia	Autocarro	Carro	Sala de aula	Biblioteca	Comboio	TAXA ACERTO
Bar	**11**	2	4	7	4	0	2	37%
Praia	2	**16**	2	2	3	2	3	53%
Autocarro	1	3	**14**	3	0	2	7	47%
Carro	6	1	2	**16**	1	2	2	53%
Sala de Aula	2	2	3	3	**13**	6	1	43%
Biblioteca	1	1	2	2	2	**20**	2	67%
Comboio	0	0	7	2	1	3	**17**	57%
TAXA ACERTO	44%	64%	41%	46%	54%	57%	50%	**51%**

Analisando a tabela 6.5 é possível constatar que, das três formas apresentadas para a construção de lisas candidatas, esta é a que obteve pior percentagem de acerto. É, também, a que

apresenta pior desempenho ao que ao tempo de execução se refere, apesar de muito próximo ao anterior. Este facto deve-se a ter de se realizar uma pesquisa em qualquer parte dos *audio fingerprints* presentes na base de dados.

6.1.2 2º Método

A implementação do 2º método apresentado tem uma metodologia diferente para a construção do *audio fingerprint*. Para este passo, optou-se pela extração dos máximos globais de cada um dos 8 *frames* construídos. Além da extração dos 8 máximos, foram extraídos 26 coeficientes MFCC do ficheiro de 3 segundos. Assim sendo, o *audio fingerprint* de cada um dos ficheiros de áudio é composto por, no total, 34 características. Para o processo de classificação, foi treinada uma rede neuronal *feed Forward*, com recurso à *framework Encog*. Nesta aplicação deste método, foram obtidos os resultados apresentados na tabela 6.6.

Tabela 6.6: Matriz de confusão com o treino de uma rede neuronal

	Bar	**Praia**	**Autocarro**	**Carro**	**Sala de aula**	**Biblioteca**	**Comboio**	**TAXA ACERTO**
Bar	**10**	3	2	8	3	1	3	33%
Praia	1	**18**	3	1	1	2	3	60%
Autocarro	1	2	**17**	1	1	3	5	57%
Carro	7	1	2	**15**	1	3	1	50%
Sala de Aula	1	1	2	2	**16**	4	2	53%
Biblioteca	0	1	3	1	0	**23**	2	77%
Comboio	1	0	8	1	1	1	**18**	60%
TAXA ACERTO	48%	69%	46%	52%	70%	62%	53%	**56%**

Após análise da tabela 6.6 é possível verificar que a taxa de acerto deste método é relativamente baixa em relação ao que podia ser esperado. Verificam-se maus resultados para o ambiente "Bar". Uma razão para tal facto pode ser que os coeficientes MFCC não se adequam a ambientes com bastante ruído. Continua a verificar-se uma confusão entre alguns dos ambientes. Relativamente ao tempo de execução, verifica-se que este método é o mais rápido dos que foram implementados, sendo obtidos resultados quase instantaneamente.

6.2 Discussão dos resultados

Apresentados os resultados de todos os testes realizados nesta dissertação, é possível verificar que nenhum dos métodos aplicados apresenta uma taxa de acerto satisfatória. Para tal facto podem haver algumas justificações. Começando pela primeira metodologia apresentada, esta apresentou cinco formas diferentes de implementação. Uma metodologia que se baseou num *audio fingerprint* composto por *Strings* de 8192 bits e que fazia uso disso mesmo para calcular a distância entre os mesmos e, em seguida, classificar com recurso ao KNN.

As primeiras duas tentativas apresentadas consideravam todos os *audio fingerprints* para a realização da pesquisa, apesar de se saber que não era o formato ideal para a realização da pesquisa, dado ao tempo necessário para a sua execução e que este tinha crescimento exponencial, isto é, quanto maior fosse a base de dados dos *audio fingerrints* maior iria ser o tempo despendido para a conclusão da pesquisa. O mesmo se pode verificar ao serem considerados os 8192 *bits* das impressões digitais de áudio pois a pesquisa foi realizada, em média, em 12 segundos. Uma tentativa de melhorar este aspeto foi considerar-se, apenas, metade dos *bits* que um *audio fingerprint* é composto. Pensava-se que, como não iriam existir tantos *bits* a comparar por cada *audio fingerprint*, o tempo despendido iria ser menor. O mesmo não se verificou. A razão para tal facto deve-se a ser necessária a construção de *subStrings* para todos os que estavam presentes na base de dados. O tempo despendido para esse processo tem uma enorme influência no tempo total gasto para a realização da pesquisa. Relativamente aos resultados obtidos nestes métodos, possuem ambos uma taxa negativa. Os resultados a nível de tempo despendido para a sua conclusão e os resultados de classificação não são os mais indicados. Verifica-se uma taxa de acerto negativa em todos os ambientes, ou seja, estes dois métodos não se adequam a nenhum dos ambientes testados.
Dado que era expectável uma pesquisa bastante demorada com as formas anteriormente apresentadas, optou-se por fazer uma filtragem antes de ser processada a classificação. A opção encontrada para esta filtrada foi a construção de uma lista de candidatos. Esta lista é construída através de alguns padrões. Para tal, foram testadas três diferentes formas para a sua construção. A primeira tem por base a verificação da existência de semelhanças no inicio dos *audio fingerprints*. Na sua execução verificou-se uma melhoria em termos de desempenho dado que apenas foram utilizados, em média, 3 segundos. Esta proposta apresenta uma taxa de acerto de 56%, valor bem acima das propostas que não possuíam construção de uma lista de candidatos.
As outras formas de construção de uma lista de candidatos e, assim, efetuar uma pré seleção, são bastante semelhantes. Ambas consistem na pesquisa de padrões em qualquer parte dos *audio fingerprints*. A diferença entre eles é que num a comparação é efetuada com base no início da impressão digital de áudio e a outra recorre-se aos *bits* presentes a meio do mesmo. Dada a semelhança entre eles, verifica-se uma enorme semelhança ao que ao tempo de execução diz respeito. No entanto, o tempo de execução é superior aos 3 segundos apresentados no formato anterior. Isto deve-se ao facto de, ao contrário do formato anterior, para a realização da pré seleção, efetuar comparações apenas no inicio das impressões digitais, este tem de procurar um padrão em qualquer parte dos 8192 *bits* que compõem um *audio fingerprint*. Relativamente aos resultados obtidos, estes foram bastante semelhantes à proposta anterior, tendo um valor positivo, no entanto, não satisfatório.

Fazendo uma avaliação deste método, em nenhum dos casos foram obtidos resultados satisfatórios. Nos métodos aplicados sem a realização de uma pré seleção, foram verificados os piores resultados para este método. Foi nesses dois casos que se verificou o maior tempo de execução da pesquisa. No que se refere às formas que utilizam lista de candidatos, apresentam resultados, relativamente, semelhantes entre eles. A forma que apresentou melhor taxa de acerto foi a que apenas verifica os inícios das impressões digitais de áudio, para a realização da pré seleção. Todas as formas apresentadas demonstram uma confusão entre os ambientes que envolvem meios de transporte. Este facto pode dever-se a ao ruído envolvente ser bastante semelhante. Isso é visível nas tabelas 6.2 e 6.3 em que o ambiente ”Autocarro” é confundido com ”Comboio” um número considerável de vezes e, vice versa. Adicionalmente, verifica-se uma confusão entre os ambientes ”Bar” e ”Carro”. Uma justificação possível para este acontecimento consiste em

algumas das amostras recolhidas no ambiente ”Carro” conter música de fundo e aí levar a essa confusão dado que num bar existe bastante ruído devido à música envolvente. Por fim, referir a elevada taxa de acerto ao que o ambiente ”Biblioteca” diz respeito. Por ser um ambiente bastante silencioso pode ter levado a não ser confundido com outros ambientes.

A outra opção implementada nesta dissertação teve uma opção completamente distinta para a construção de um *audio fingerprint*. A opção foi a de extrair oito máximos presentes num ficheiro de áudio de 3 segundos e os seus 26 coeficientes MFCC. Para a realização da classificação foi levada a treino uma rede neuronal. Na utilização deste metodologia, verificou-se uma pesquisa bastante rápida, quase instantânea, o que contrasta com o outro método implementado nesta dissertação. No que toca aos resultados obtidos relativamente à taxa de acerto, verificou-se uma percentagem de 56%. Apesar das taxa de acerto, não é um valor satisfatório pois estava-se à espera de resultados mais elevados. É visível que nos ambientes ”Bar” e ”Carro” existe uma enorme confusão entre eles pois verifica-se uma taxa de acerto de 33 e 50%, respetivamente. Estes dois ambientes apresentam alguma semelhança entre eles e a justificação para tal pode ser a apresentada para o método anterior, dado o ruído existente nos dois locais. É possível verificar uma enorme eficácia, 77 %, para o ambiente que envolve a ”Biblioteca” dado o reduzido ruído aí existente. Os outros ambientes têm uma taxa de acerto semelhante, entre os 53 e 60%, destacando-se, novamente, a confusão existente entre o ”Autocarro” e ”Comboio”.

A tarefa de construção de um *audio fingerprint* é deveras complexa e está dependente de diversos fatores para ser obtido um método com sucesso. No entanto, para a serem obtidos bons resultados não basta a aplicação de um bom método. Neste caso, foram testados dois métodos completamente distintos e os resultados obtidos não foram os ideais. O mesmo não quer dizer que os métodos implementados não sejam capazes de bons resultados, precisam é de aperfeiçoamento. Uma das razões possíveis para a obtenção destes resultados pode dever-se ao facto de as recolhas para a base de dados ter sido efetuada com recurso ao microfone de um *smartphone*. Um microfone que seja capaz de efetuar uma boa recolha, sem ruído e com uma perceção do ambiente que se encontra pode melhorar, e muito, a qualidade da construção de um *audio fngerprint*. Posto isto, o *smartphone* utilizado não possuía as características ideais para a realização desta tarefa.
Outra razão possível, e referida no capitulo 2, prende-se com a distância a que a captura é realizada do ambiente que queremos capturar. A variação da distância interfere diretamente na qualidade das capturas e pode interferir na construção de uma impressão digital de áudio. Assim, não é garantido que as recolhas efetuadas tenham sido feitas, todas, à mesma distância do centro do ambiente e com a distância ideal para a realização da recolha.
Uma terceira razão que pode ter influenciado os resultados obtidos tem a ver com o número de amostras recolhidas. Este fator foi mais visível no treino da rede neuronal pois, possivelmente, com um número maior de dados para treino, ter-se-iam obtido resultados mais satisfatórios pois a rede neuronal iria ter mais informação o que levava a uma aprendizagem com maior sucesso. Este facto também influenciou o primeiro método aplicado pois com mais amostras, a lista de candidatos seria composta por um maior número de *audio fingerprints* e a aplicação do classificador KNN podia ser aplicado com um ”K” mais elevado.
Um fator decisivo para este resultado tem a ver com os ambientes escolhidos para teste. A escolha de locais semelhantes, como se verifica com o autocarro e o comboio, em que os níveis de ruído são bastante semelhantes, fez com que se verifica-se uma confusão entre os dois locais. Uma consideração a ter em conta é o facto de ambos os métodos não apresentar um

bom desempenho para ambientes com muito ruído. Em nenhum dos testes foi obtida uma taxa positiva para o Bar, sendo este confundido com o ambiente vivido num carro. Para o caso do segundo método, a razão pode dever-se ao facto de serem utilizados os MFCC para a construção das impressões digitais de áudio, já que em [YP09] é concluído que a utilização de MFCC para construção de um *audio fingerprint* obtém bons resultados em ambientes com pouco ruído, não se verificando o mesmo com capturas com bastante ruído.

6.3 Verificação dos Requisitos

Nesta secção pretende-se a realização de uma verificação se os requisitos, vistos como necessários no capítulo 4, foram cumpridos com sucesso. Nesse capítulo foram apresentados requisitos funcionais e não funcionais para o desenvolvimento e sucesso do sistema implementado. Esses requisitos eram vistos como os extremamente necessários nesta aplicação. Assim sendo, serão, sem seguida, efetuados testes de forma a verificar quais os que foram cumpridos.

Começando por fazer a verificação dos requisitos funcionais:

RF1: Realizar capturas de áudio através do dispositivo
Para a realização deste requisito, foi necessária a utilização do microfone do dispositivo. Através desta ferramenta, foi possível realizar capturas como foi apresentado no capítulo 5. Os dados são armazenados num ficheiro de texto para futura utilização. Posto isto, este requisito foi cumprido com sucesso.

RF2: Capacidade de realizar análises do áudio recolhido
De forma a cumprir este requisito, verificou-se necessária a construção de *audio fingerprints* para todas as capturas efetuadas. Este construção é a caracterização das recolhas e o que permite, em seguida efetuar análise das recolhas. Apesar de os resultados obtidos não serem os melhores, é possível fazer análise das recolhas, apesar de poder ser melhorado. Assim sendo, este é um requisito que pode e deve ser melhorado no futuro.

RF3: Apresentar ao utilizador o ambiente em que este se encontra
Este requisito pretendia que fosse apresentado ao utilizador o resultado da pesquisa efetuada. Essa parte foi cumprida com sucesso dado que a aplicação desenvolvida é capaz de o fazer. No entanto, pretende-se que a aplicação produza resultados mais eficazes, dado que ainda tem uma taxa de erro elevada. Assim sendo, pretende-se que, no futuro, a apresentação do ambiente que o dispositivo está envolvido seja efetuada com uma melhor taxa de acerto.

Feita a verificação dos requisitos funcionais, é realizado, em seguida a verificação dos requisitos não funcionais.

RNF1: Executar a aplicação num tempo satisfatório
Neste requisito pretendia-se que a aplicação executa-se num termo razoável para o utilizador. Tendo em conta que são necessários 3 segundos para a recolha, seria essencial que a pesquisa e classificação fossem realizadas quase instantaneamente. Isso só é conseguido com a aplicação da rede neuronal, enquanto no outro método testado, a pesquisa mais rápida é efetuada em 4 segundos.

RNF2: Executar em qualquer dispositivo que possua o SO *Android*
Dado que o SO *Android* possuí diversas versões e atualizações, é necessário, ao construir uma aplicação, verificar se esta executa com sucesso nas mais diversas versões. A aplicação foi testada em *smartphones* com o SO *Android* 5.0.2 e 6.0.1 executando com sucesso. Foi ainda instalada, com recurso a um emulador presente no *Android Studio*, com a versão 7 do *Android*, tendo executado com sucesso. Assim sendo, este requisito foi cumprido.

RNF3: Possuir um aspeto agradável e com uma fácil utilização
A *design* da aplicação foi desenvolvido com o intuito de ser *user-frendly* e simples. É possível verificar que o aspeto da aplicação é simples e de fácil utilização. Apesar de possuir alguns aspetos onde melhorar, pode-se considerar que o objetivo foi cumprido.

RNF4: Realizar uma metodologia de armazenamento que não sobrecarregue a memória do dispositivo
A utilização da memória disponível num dispositivo tem interferência no desempenho do mesmo. Assim sendo, verifica-se uma necessidade de tentar ocupar o menos espaço da memória possível. Assim sendo, quando a aplicação é utilizada, os dados da recolha realizada são armazenados no dispositivo num ficheiro de texto. No entanto, quando a aplicação faz a classificação do mesmo, o ficheiro é apagado do dispositivo. Posto isto, é possível verificar que o requisito foi concluído com sucesso.

Capítulo 7

Conclusão e Trabalho Futuro

7.1 Conclusão

Após o desenvolvimento desta dissertação, é possível concluir que o processo de construção de um *audio fingerprint* e a sua classificação é uma tarefa complexa. Além de ser necessário a aplicação de um método que se adeque ao contexto da mesma, existem outros fatores que têm um papel decisivo para o sucesso do sistema, como o facto de ser fundamental que os ficheiros recolhidos sejam de alguma qualidade. Além do descrito anteriormente, é necessário que as amostras recolhidas sejam capturadas à mesma distância do ambiente "principal" e desejado e que sejam efetuadas com *hardware* de qualidade. É possível verificar que, apesar de recente, este tema possuí alguns métodos desenvolvidos e com, relativamente, bons resultados.
No que toca à dissertação, o seu principal objetivo prendia-se com o desenvolvimento um método que fosse capaz de gerar uma impressão digital de áudio. O mesmo foi conseguido, sendo desenvolvidas, não uma, mas duas técnicas diferentes para o mesmo fim. Estas metodologias foram aplicadas com sucesso. O mesmo não se pode concluir com os resultados obtidos após a classificação dos mesmos, uma vez que era esperada uma taxa de acerto mais elevada nos dois métodos. Estes resultados permitem concluir que existe bastante por onde melhorar.
Alguns aspetos que falharam nesta dissertação e que levaram a não ser obtido um sucesso completo prendem-se com os ambientes escolhidos para teste. O facto de a escolha efetuada conter locais com um paradigma bastante semelhante em que têm movimentos muito "monótonos", levou a uma confusão do sistema em relação a alguns ambientes. Este facto levou a que os resultados não fossem os mais satisfatórios. Uma outra conclusão diz respeito com o facto de a utilização dos MFCC poder não ser a escolha ideal para ambientes com bastante ruído, apesar de nenhum dos métodos aplicados ter obtido bons resultados com o local "Bar".
Um fator que interferiu foi a qualidade das amostras não ter sido a melhor e a necessidade de efetuar um maior número de recolhas. Acredita-se que com a correção destes dois fatores, verificar-se-ão resultados mais satisfatórios e de encontro ao que era expectável.
Por fim, o facto de o microfone do *smartphone* utilizado não se encontrar nas melhores condições e, adicionalmente, não se ter tido em conta a que distância as amostras eram recolhidas, terão tido um impacto direto no sucesso do sistema desenvolvido.
Em suma, é possível concluir que o objetivo principal desta dissertação foi cumprido com sucesso dado que foram aplicadas diferentes técnicas para a geração de um *audio fingerprint*. Adicionalmente, foi realizado um estudo aprofundado do estado da arte que permitiu retirar ilações do caminho que deveria ser. No entanto, é visível nos resultados obtidos que existe muito por melhorar, de forma a verificar-se um sucesso absoluto do sistema desenvolvido.

7.2 Trabalho Futuro

Apesar de ser concluído o objetivo principal desta dissertação, existem diversos aspetos a melhorar para serem obtidos resultados mais satisfatórios. Assim sendo, num futuro próximo deverão ser aplicadas as seguintes medidas para a obtenção do sucesso total do sistema:

- Realização de capturas com melhor qualidade;
- Possuir um número maior de recolhas que o atual, por cada ambiente;
- Adicionar mais locais para teste;
- Realização de mais testes ao sistema desenvolvido;
- Aplicação de uma rede neuronal diferente e, possivelmente, mais eficaz;
- Analisar as características escolhidas para a geração do *audio fingerprint* são as mais adequadas e se podem ser acrescentadas mais.

Bibliografia

[CM10] K. F. Chen and S. L. Mei. Composite interpolated fast fourier transform with the hanning window. *IEEE Transactions on Instrumentation and Measurement*, 59(6):1571–1579, June 2010. 7

[DGC+12] Jeric Ryan R De Josef, Bobby D. Gerardo, Ma Beth S Concepcion, Kyle Gabriel V Oreta, and Yung Cheol Byun. Ultrasonic key recognition: Security algorithm for precomposed high frequency sound as a mode of unlocking a security lock. *Proceedings - 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, ICIS 2012*, pages 655–660, 2012. 8

[DHL12] Ngoc Q.K. Duong, Christopher Howson, and Yvon Legallais. Fast second screen TV synchronization combining audio fingerprint technique and generalized cross correlation. *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin*, pages 241–244, 2012. 11

[GBI06] Lahouari Ghouti, Ahmed Bouridane, and Mohammad K. Ibrahim. A fingerprinting system for musical content. *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, 2006:1989–1992, 2006. 7, 8, 12

[GJ15] Jacob George and Ashok Jhunjhunwala. Scalable and robust audio fingerprinting method tolerable to time-stretching. *International Conference on Digital Signal Processing, DSP*, 2015-Septe:436–440, 2015. 8, 19

[HK02] Jaap Haitsma and T Kalker. A highly robust audio fingerprinting system. *in Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR'02)*, pages 107–115, 2002. 7, 8, 10, 11, 13, 14, 16, 17, 18, 21, 35

[HL12] Kyuwoong Hwang and Soo Young Lee. Environmental audio scene and activity recognition through mobile-based crowdsourcing. *IEEE Transactions on Consumer Electronics*, 58(2):700–705, 2012. 11, 14, 17, 19

[KD13] M D Kamaladas and M M Dialin. Fingerprint extraction of audio signal using wavelet transform. *Signal Processing Image Processing Pattern Recognition (ICSIPR), 2013 International Conference on*, pages 308–312, 2013. 7, 8, 9, 10, 15, 18, 19

[LYC13] Sunhyung Lee, Dongsuk Yook, and Sukmoon Chang. An Efficient Audio Fingerprint Search Algorithm for Music Retrieval. 59(3):652–656, 2013. 13, 17

[MK14] S. P. Mohanapriya and R. Karthika. Unsupervised environmental sound recognition. *International Conference on Embedded Systems, ICES 2014*, (Ices):44–48, 2014. 11, 16

[MOK+16] B. M. Murphy, C. O'Driscoll, I. Korotchikova, G. B. Boylan, G. Lightbody, and W. P. Marnane. Application of audio fingerprinting to Neonatal EEG. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2016-Octob:912–915, 2016. 7, 13, 17

[MSK14] S P Mohanapriya, E P Sumesh, and R Karthika. Environmental sound recognition using Gaussian mixture model and neural network classifier. *2014 International Conference*

on Green Computing Communication and Electrical Engineering (ICGCCEE), pages 1–5, 2014. 8, 9, 11, 15

[RK05] Arunan Ramalingam and Sridhar Krishnan. Gaussian Mixture Modeling Using Short Time Fourier Transform Features for Audio Fingerprinting. *Computer Engineering*, 1(4):5–8, 2005. 16, 17

[SBY06] M Sert, B Baykal, and A Yazici. A Robust and Time-Efficient Fingerprinting Model for Musical Audio. *2006 IEEE International Symposium on Consumer Electronics*, pages 1–6, 2006. 7, 9, 12

[Sin06] A Sinitsyn. Duplicate Song Detection using Audio Fingerprinting for Consumer Electronics Devices. *Consumer Electronics, 2006. ISCE '06. 2006 IEEE Tenth International Symposium on*, pages 1–6, 2006. 14

[SJL+06] Jin S. Seo, Minho Jin, Sunil Lee, Dalwon Jang, Seungjae Lee, and Chang D. Yoo. Audio fingerprinting based on normalized spectral subband moments. *IEEE Signal Processing Letters*, 13(4):209–212, 2006. 14

[SSASS09] Sajad Shirali-Shahreza, Hassan Abolhassani, and M. Hassan Shirali-Shahreza. Fast and scalable system for automatic artist identification. *IEEE Transactions on Consumer Electronics*, 55(3):1731–1737, 2009. 11, 15, 17

[TL03] Alfie Tan and Kok Leong. A Music Identification System Based on Audio Content Similarity By. *October*, (October):1–195, 2003. 6, 7, 11, 12, 17, 20

[Wan03] Avery Li-Chun Wang. An Industrial Strength Audio Search Algorithm. *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR 203), Baltimore, Maryland (USA), 26-30 October 2003*, pages 7–13, 2003. 20, 21

[WYWS12] Hongxue Wang, Xiaoqing Yu, Wanggen Wan, and Ram Swaminathan. Robust audio fingerprint extraction algorithm based on 2-D chroma. *ICALIP 2012 - 2012 International Conference on Audio, Language and Image Processing, Proceedings*, pages 763–767, 2012. 7, 9

[XYW+12] Wei Xiong, Xiaoqing Yu, Wengen Wang, Wanggen Wan, and Ram Swaminathan. Audio fingerprinting based on dynamic subband locating and normalized SSC. *ICALIP 2012 - 2012 International Conference on Audio, Language and Image Processing, Proceedings*, pages 772–776, 2012. 8, 9, 13

[YCY14] Guang Yang, Xiaoou Chen, and Deshun Yang. Efficient music identification by utilizing space-saving audio fingerprinting system. *Proceedings - IEEE International Conference on Multimedia and Expo*, 2014-Septe(Septmber), 2014. 7

[YP07] Won Jung Yoon and Kyu Sik Park. A robust mobile-based music information retrieval system. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, 2007. 9

[YP09] Wonjung Yoon and Kyusik Park. A noise robust content-based music retrieval system. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pages 4–5, 2009. 55

Apêndice A

Anexos

A.1 Artigo Científico

sensors

MDPI

Review

Recognition of Activities of Daily Living Based on Environmental Analyses Using Audio Fingerprinting Techniques: A Systematic Review

Ivan Miguel Pires [1,2,3], Rui Santos [1,3], Nuno Pombo [1,3,4], Nuno M. Garcia [1,3,4,*], Francisco Flórez-Revuelta [5], Susanna Spinsante [6], Rossitza Goleva [7] and Eftim Zdravevski [8]

1 Instituto de Telecomunicações, Universidade da Beira Interior, 6201-001 Covilhã, Portugal; impires@it.ubi.pt (I.M.P.); rui_17_santos@hotmail.com (R.S.); ngpombo@ubi.pt (N.P.)
2 Altranportugal, 1990-096 Lisbon, Portugal
3 ALLab—Assisted Living Computing and Telecommunications Laboratory, Computing Science Department, Universidade da Beira Interior, 6201-001 Covilhã, Portugal
4 ECATI, Universidade Lusófona de Humanidades e Tecnologias, 1749-024 Lisbon, Portugal
5 Department of Computer Technology, Universidad de Alicante, 03690 Sant Vicent del Raspeig, Alicante, Spain; francisco.florez@ua.es
6 Department of Information Engineering, Marche Polytechnic University, 60121 Ancona, Italy; s.spinsante@univpm.it
7 Department of Informatics, New Bulgarian University, 1618 g.k. Ovcha kupel 2 Sofia, Bulgaria; rgoleva@gmail.com
8 Faculty of Computer Science and Engineering, University Ss Cyril and Methodius, 1000 Skopje, Macedonia; eftim.zdravevski@finki.ukim.mk
* Correspondence: ngarcia@di.ubi.pt; Tel.: +351-966-637-9785

Received: 28 November 2017; Accepted: 5 January 2018; Published: 9 January 2018

Abstract: An increase in the accuracy of identification of Activities of Daily Living (ADL) is very important for different goals of Enhanced Living Environments and for Ambient Assisted Living (AAL) tasks. This increase may be achieved through identification of the surrounding environment. Although this is usually used to identify the location, ADL recognition can be improved with the identification of the sound in that particular environment. This paper reviews audio fingerprinting techniques that can be used with the acoustic data acquired from mobile devices. A comprehensive literature search was conducted in order to identify relevant English language works aimed at the identification of the environment of ADLs using data acquired with mobile devices, published between 2002 and 2017. In total, 40 studies were analyzed and selected from 115 citations. The results highlight several audio fingerprinting techniques, including Modified discrete cosine transform (MDCT), Mel-frequency cepstrum coefficients (MFCC), Principal Component Analysis (PCA), Fast Fourier Transform (FFT), Gaussian mixture models (GMM), likelihood estimation, logarithmic moduled complex lapped transform (LMCLT), support vector machine (SVM), constant Q transform (CQT), symmetric pairwise boosting (SPB), Philips robust hash (PRH), linear discriminant analysis (LDA) and discrete cosine transform (DCT).

Keywords: acoustic sensors; fingerprint recognition; data processing; artificial intelligence; mobile computing; signal processing algorithms; systematic review; Activities of Daily Living (ADL)

1. Introduction

The identification of Activities of Daily Living (ADL) [1] is of utmost importance to build Enhanced Living Environment and Ambient Assisted Living solutions [2,3], or to allow the development of Personal Digital Life Coaching systems [4]. To achieve this, several authors have proposed the development of solutions based on mobile devices (e.g., smartphones) [5–8] for several reasons, the most prominent being the adoption ratios of these devices, its increasing computing power and memory, and the fact that these devices already come equipped with a plethora of sensors that can be used to sense and feed data to ADL identification systems.

Despite the increasing complexity of ADL identification systems, the recognition of the surrounding environment is limited because of the restrictions of some location sensors. For instance, Global Positioning System (GPS) sensors, can only be reliably and accurately used in outdoor scenarios. Likewise, proximity sensors, radar sensors, Passive Infra-Red (PIR) sensors and alike require significant installation effort, thus are not widely used in real scenarios which require ADL identification. As proposed in previous works [9–11], an ADL identification framework should also be able to integrate data from the sound of the environment into the ADL identification module in order to allow the system to sense the environmental sounds, to determine the type of environment, and to increase the accuracy of the overall ADL identification solution.

Most mobile devices are equipped with a microphone that can capture an acoustic signal. This signal can be processed using audio fingerprinting techniques, allowing the system to find a match between the collected signal and a database of well-known audio fingerprints. This might facilitate an increase in the accuracy of recognition of the environment where ADLs are performed.

Several methods may be used to carry out audio fingerprinting, performing the pre-processing of the acoustic data (e.g., Fast Fourier Transform (FFT)), extracting relevant features, and after that, obtaining a classification or recognition (e.g., Support Vector Machine (SVM)).

This review summarizes the existing methods in the literature related to audio fingerprinting techniques for the application in a system that uses mobile technology for the recognition of the environment. While acknowledging that the methods here presented are very diverse and have been tested with different data sets and different feature extraction techniques, in order to estimate which method may provide better results in a mobile computational device, this paper also presents a comparison between the different methods and features.

The remainder of this paper is organized as follows: Section 2 presents the methodology for this review; the methods discovered in the literature are presented in Section 3; Section 4 discusses different methods, and finally, Section 5 present conclusions of this review.

2. Methodology

2.1. Research Questions

The primary questions of this review were as follows: (RQ1) What is audio fingerprinting? (RQ2) Which audio fingerprinting techniques are useful to identify the environment of daily activities? (RQ3) Which are the audio fingerprinting techniques feasible for their use in mobile devices?

2.2. Inclusion Criteria

Studies assessing ADLs using audio fingerprinting techniques were included in this review if they met the following criteria: (1) audio fingerprinting techniques adapted to mobile devices; (2) audio fingerprinting techniques used for the detection of the environment of ADL; (3) using mobile devices; (4) the accuracies of the audio fingerprinting techniques presented are reported; (5) were published between 2002 and 2017; and (6) were written in English.

2.3. Search Strategy

The team searched for studies meeting the inclusion criteria in the following electronic databases: IEEE Xplore, and ACM Digital Library. Every study was independently evaluated by eight reviewers (IP, RS, NP, NG FR, SP, RG and EZ), and its suitability was determined with the agreement of all parties. The studies were examined to identify the characteristics of audio fingerprint and its suitability for application with mobile devices for the identification of ADL.

2.4. Extraction of Study Characteristics

The following data was extracted from the studies and tabulated (see Tables 1 and 2): year of publication, population for the application of the algorithm, purpose of the study, devices used, and study outcomes of the algorithm for audio fingerprinting. For all cited studies in Tables 1 and 2, the experiments were conducted in laboratory settings. We additionally verified whether the raw data and source code are available, either publically or per request, by emailing the corresponding author of each study.

Table 1. Study Analysis.

Paper	Year of Publication	Population	Purpose of the Study	Devices	Raw Data Available	Source Code Available
			ACM			
Sui et al. [12]	2014	2500 pieces of 8 s advertisement audios, and randomly select 200 pieces of audio in the existing database and 50 pieces of other irrelevant audio as test audio	To search for audio in the database by the content rather than by name	Mobile Phone (Android)	No	No
Liu [13]	2012	100,000 MP3 fragments	To create an MP3 sniffer system that includes audio fingerprinting	Not mentioned	Yes	Only for feature extraction
Liu et al. [14]	2011	10,000 MP3 fragments	Proposes an MP3 fingerprint system for the recognition of several clips	Not mentioned	The same data as [13]	The same source code as [13]
			IEEE			
Tsai et al. [15]	2016	Multi-channel audio recordings of 75 real research group meetings, approximately 72 h of meetings in total	Proposes an adaptive audio fingerprint based on spectrotemporal eigenfilters	Mobile phones, tablets or laptop computers	Yes	No
Casagranda et al. [16]	2015	1024 samples	Proposes an audio fingerprinting method that uses GPS and acoustic fingerprints	Smartphone	No	No
Nagano et al. [17]	2015	Approximately 1,518,177 min (25,303 h) of songs	Proposes a method to accelerate audio fingerprinting techniques by skipping the search for irrelevant signal sections	Not mentioned	Yes	No
Ziaei et al. [18]	2015	1062 10 s clips	Proposes a method to analyze and classify daily activities in personal audio recordings	Not mentioned	Yes	No

Table 1. *Cont.*

Paper	Year of Publication	Population	Purpose of the Study	Devices	Raw Data Available	Source Code Available
George et al. [19]	2015	1500 audio files	Proposes an audio fingerprinting method based on landmarks in the audio spectrogram	Computer	No	No
Kim et al. [20]	2015	6000 television advertisements with a total time of 1110 h	Proposes a television advertisement search based on audio fingerprinting in real environments	Television	No	No
Seo [21]	2014	1000 songs with classic, jazz, pop, rock, and hip-hop	Proposes a binary audio fingerprint matching, using auxiliary information	Not mentioned	No	No
Rafii et al. [22]	2014	Several songs with a duration between 6 and 9 s	Proposes an audio fingerprinting method for recognition of some clips	Computer and Smartphone	No	No
Naini et al. [23]	2014	1000 songs	Proposes an audio fingerprinting method based on maximization of the mutual information across the distortion channel	Not mentioned	No	No
Yang et al. [24]	2014	200,000 songs	Proposes a music identification system based on space-saving audio fingerprints	Not mentioned	No	No
Yin et al. [25]	2014	958 randomly chosen query excerpts	Proposes an audio fingerprinting algorithm that uses compressed-domain spectral entropy	Not mentioned	No	No
Wang et al. [26]	2014	100,000 songs	Proposes an audio fingerprinting method that uses GPUs	Not mentioned	No	No
Lee et al. [27]	2014	3000 TV advertisements	Proposes a high-performance audio fingerprint extraction method for identifying Television commercial advertisement	Television	No	No
Shibuya et al. [28]	2013	1374 television programs (792 h in total)	Proposes a method of identifying media content from an audio signal recorded in reverberant and noisy environments using a mobile device	Smartphone, tablet, notebook, desktop, or another mobile device	No	No
Bisio et al. [29]	2013	20 sounds	Proposes the Improved Real-Time TV-channel Recognition (IRTR) method	Smartphone	No	No
Lee et al. [30]	2013	1000 songs as positive samples and 999 songs as negatives	Proposes a method that speeds up the search process, reducing the number of database accesses	Not mentioned	No	No

Table 1. *Cont.*

Paper	Year of Publication	Population	Purpose of the Study	Devices	Raw Data Available	Source Code Available
Bisio et al. [31]	2012	100,000 songs	Proposes an audio fingerprint algorithm adapted to mobile devices	Smartphone	No	No
Anguera et al. [32]	2012	Several datasets	Proposes an audio fingerprinting algorithm that encodes the local spectral energies around salient points selected among the main spectral peaks in a given signal	Not mentioned	No	No
Duong et al. [33]	2012	300 real-world recordings in a living room	Proposes an audio fingerprinting method that combines the Fingerprinting technique with Generalized cross correlation	iPad	No	No
Wang et al. [34]	2012	20 music clips with 5 s	Proposes an audio fingerprinting algorithm for recognition of some clips	Not mentioned	No	No
Xiong et al. [35]	2012	835 popular songs	Proposes an audio fingerprinting algorithm based on dynamic subband locating and normalized spectral subband centroid (SSC)	Not mentioned	No	No
Deng et al. [36]	2011	100 audio files	Proposes an audio fingerprinting algorithm based on harmonic enhancement and SSC of audio signal	Not mentioned	No	No
Pan et al. [37]	2011	62-h audio database of 1000 tracks	Proposes an audio feature in spectrum, local energy centroid, for audio fingerprinting	Not mentioned	No	No
Martinez et al. [38]	2011	3600 s of several real-time tests	Presents an audio fingerprinting method with a low-cost embedded reconfigurable platform	Computer	No	No
Cha [39]	2011	1000 songs	Proposes an indexing scheme and a search algorithm based on the index	Computer	No	Only pseudo-code for fingerprint matching
Schurmann et al. [40]	2011	7500 experiments	Proposes an audio fingerprinting method for the recognition of some clips	Computer	No	No
Son et al. [41]	2010	500 popular songs	Proposes an audio fingerprinting method using sub-fingerprint masking based on the predominant pitch extraction	Mobile devices	Yes	No

Table 1. *Cont.*

Paper	Year of Publication	Population	Purpose of the Study	Devices	Raw Data Available	Source Code Available
Chang et al. [42]	2010	17,208 audio clips	Presents a sub-Nyquist audio fingerprinting system for music recognition, which utilizes Compressive Sampling (CS) theory	Not mentioned	No	No
Umapathy et al. [43]	2007	213 audio signals	Proposes an audio feature extraction and a multi-group classification using the local discriminant bases (LDB) technique	Not mentioned	No	No
Kim et al. [44]	2007	100 Korean broadcast TV programs	Proposes an audio fingerprinting method for identification of bookmarked audio segments	Computer	No	No
Sert et al. [45]	2006	approximately 45 min of pop, rock, and country songs	Proposes an audio fingerprinting method from the most representative section of an audio clip	Not mentioned	No	No
Ramalingam et al. [46]	2006	250 audio files	Proposes and audio fingerprinting method using several features	Not mentioned	No	No
Ghouti et al. [47]	2006	Two audio contents perceptually similar	Proposes an audio fingerprinting algorithm that uses balanced multiwavelets (BMW)	Not mentioned	No	No
Cook et al. [48]	2006	7,106,069 fingerprints	Proposes an audio fingerprinting algorithm for the fast indexing and searching of a metadata database	PDA or computer	Yes	No
Seo et al. [49]	2005	8000 classic, jazz, pop, rock, and hip-hop songs	Proposes an audio fingerprinting method based on normalized SSC	Not mentioned	No	No
Haitsma et al. [50]	2003	256 sub-fingerprints	Proposes to solve larger speed changes by storing the fingerprint at multiple speeds in the database or extracting the fingerprint query at multiple speeds and then to perform multiple queries on the database	Not mentioned	No	No
Haitsma et al. [51]	2002	256 sub-fingerprints	Proposes an audio fingerprinting system for recognition of some clips	Not mentioned	No	No

Table 2. Study summaries.

Paper	Outcomes
	ACM
Sui et al. [12]	The authors propose a two-level audio fingerprint retrieval algorithm to satisfy the demand of accurate and efficient search for advertisement audio. Based on clips with 8 s of advertisements, the authors build a database with 2500 audio fingerprints. The results show that the algorithm implemented with parallel processing yields a precision of 100%.
Liu [13]	The authors create an MP3 sniffer system and test it with multi-resolution local descriptions. The system has a database of 100,000 MP3 tones and authors report that the system has high performance, because 100 queries for identifying unknown MP3 tones took less than 2 s to be processed
Liu et al. [14]	The authors describe an MP3 fingerprinting system that compares the normalized distance between two MP3 fingerprints to detect a false identification. The authors identify the possible features of the song and build a large database. For the identification, the authors test the near neighbor searching schemes and compare with the indexing scheme, which utilizes the PCA technique, the QUery Context (QUC)-tree, and the MP3 signatures. The conclusions show that the system has a maximum average error equals to 4.26%.
	IEEE
Tsai et al. [15]	The authors propose a method for aligning a set of overlapping meeting recordings, which uses an audio fingerprint representation based on spectrotemporal eigenfilters that are learned on-the-fly in an unsupervised manner. The proposed method is able to achieve more than 99% alignment accuracy at a reasonable error tolerance of 0.1 s.
Casagranda et al. [16]	The authors propose an audio fingerprinting algorithm based on the spectral features of the audio samples. The authors reported that the algorithm is noise tolerant, which is a key feature for audio based group detection.
Nagano et al. [17]	The authors propose an approach to accelerate fingerprinting techniques and apply it to the divide-and-locate (DAL) method. The reported results show that DAL3 can reduce the computational cost of DAL to approximately 25%.
Ziaei et al. [18]	The authors propose a method to analyze and classify daily activities in personal audio recordings (PARs), which uses speech activity detection (SAD), speaker diarization, and a number of audio, speech and lexical features to characterize events in daily audio streams. The reported overall accuracy of the method is approximately 82%.
George et al. [19]	The authors propose an audio fingerprinting method that is tolerant to time-stretching and is scalable. The proposed method uses three peaks in the time slice, unlike Shazam, which uses only one. The additive noise deteriorates the lowest frequency bin, decreasing the performance of the algorithm at higher additive noise, compared to other algorithms.
Kim et al. [20]	The authors propose a Television advertisement search based on audio peak-pair hashing method. The reported results show that the proposed method has respectable results compared to other methods.
Seo [21]	The authors propose an asymmetric fingerprint matching method which utilizes an auxiliary information obtained while extracting fingerprints from the input unknown audio. The experiments carried out with one thousand songs against various distortions compare the performance of the asymmetric matching with the conventional Hamming distance. Reported results suggest that the proposed method has better performance than the conventional Hamming distance.
Rafii et al. [22]	The authors propose an audio fingerprinting system with two stages: fingerprinting and matching. The system uses CQT and a threshold method for fingerprinting stage, and the Hamming similarity and the Hough Transform for the matching stage, reporting an accuracy between 61% and 81%.
Naini et al. [23]	The authors present a method for designing fingerprints that maximizes a mutual information metric, using a greedy optimization method that relies on the information bottleneck (IB) method. The results report a maximum accuracy around 65% in the recognition.
Yang et al. [24]	The authors propose an efficient music identification system that utilizes a kind of space-saving audio fingerprints. The experiments were conducted on a database of 200,000 songs and a query set of 20,000 clips compressed in MP3 format with different bit rates. The author's report that compared to other methods, this method reduces the memory consumption and keeps the recall rate at approximately 98%.

Table 2. *Cont.*

Paper	Outcomes
	IEEE
Yin et al. [25]	The authors propose a compressed-domain audio fingerprinting algorithm for MP3 music identification in the Internet of Things. The algorithm achieves promising results on robustness and retrieval precision rates under various time-frequency audio signal distortions including the challenging pitch shifting and time-scale modification.
Wang et al. [26]	The authors propose parallelized schemes for audio fingerprinting over GPU. In the experiments, the speedup factors of the landmark lookup and landmark analysis are verified and the reported overall response time has been reduced.
Lee et al. [27]	The authors propose a salient audio peak pair fingerprint extraction based on CQT. The reported results show that the proposed method has better results compared to other methods, and is suitable for many practical portable consumer devices.
Shibuya et al. [28]	The authors develop a method that uses the quadratically interpolated FFT (QIFFT) for the audio fingerprint generation in order to identify media content from an audio signal recorded in a reverberant or noisy environment with an accuracy around 96%.
Bisio et al. [29]	The authors present an improvement of the parameter configuration used by the Philips audio fingerprint computation algorithm in order to reduce the computational load and consequent energy consumption in the smartphone client. The results show a significant reduction of computational time and power consumption of more than 90% with a limited decrease in recognition performance.
Lee et al. [30]	The authors propose an audio fingerprint search algorithm for music retrieval from large audio databases. The results of the proposed method achieve 80–99% search accuracy for input audio samples of 2–3 s with signal-to-noise ratio (SNR) of 10 dB or above.
Bisio et al. [31]	The authors present an optimization of the Philips Robust Hash audio fingerprint computation algorithm, in order to adapt it to run on a smartphone device. In the experiments, the authors report that the proposed algorithm has an accuracy of 95%.
Anguera et al. [32]	The authors present a novel local audio fingerprint called Masked Audio Spectral Keypoints (MASK) that is able to encode, with few bits, the audio information of any kind in an audio document. MASK fingerprints encode the local energy distribution around salient spectral points by using a compact binary vector. The authors report an accuracy around 58%.
Duong et al. [33]	The authors presented a new approach based on audio fingerprinting techniques. The results of this study indicate that a high level of synchronization accuracy can be achieved for a recording period as short as one second.
Wang et al. [34]	The authors present an audio fingerprinting algorithm, where the audio fingerprints are produced based on 2-Dimagel, reporting an accuracy between 88% and 99%.
Xiong et al. [35]	The authors propose an improved audio fingerprinting algorithm based on dynamic subband locating and normalized Spectral Subband Centroid (SSC). The authors claim that the algorithm can recognize unknown audio clips correctly, even in the presence of severe noise and distortion.
Deng et al. [36]	The authors propose an audio fingerprinting algorithm based on harmonic enhancement and Spectral Subband Centroid (SSC). The authors build a database with 100 audio files, and also implement several techniques to reduce the noise and other degradations, proving the reliability of the method when severe channel distortion is present. The results report an accuracy between 86% and 93%.
Pan et al. [37]	The authors propose a method for fingerprinting generation using the local energy centroid (LEC) as a feature. They report that the method is robust to different noise conditions and, when the linear speed is not changed, the audio fingerprint method based on LEC obtains an accuracy of 100%, reporting better results than Shazam's fingerprinting.
Martinez et al. [38]	The authors present a music information retrieval algorithm based on audio fingerprinting techniques. The size of frame windows influences the performance of the algorithm, e.g., the best size of the frame window for shorts audio tracks is between 32 ms to 64 ms, and the best size of the frame window for audio tracks is 128 ms.
Cha [39]	The author proposes an indexing scheme for large audio fingerprint databases. The method shows a higher performance than the Haitsma-Kalker method with respect to accuracy and speed.

Table 2. *Cont.*

Paper	Outcomes
	IEEE
Schurmann et al. [40]	The authors propose a fuzzy-cryptography scheme that is adaptable in its noise tolerance through the parameters of the error correcting code used and the audio sample length. In a laboratory environment, the authors utilized sets of recordings for five situations at three loudness levels and four relative positions of microphones and audio source. The authors derive the expected Hamming distance among audio fingerprints through 7500 experiments. The fraction of identical bits is above 0.75 for fingerprints from the same audio context, and below 0.55 otherwise.
Son et al. [41]	The authors present an audio fingerprinting algorithm to recognize songs in real noisy environments, which outperforms the original Philips algorithm in recognizing polyphonic music in real similar environments.
Chang et al. [42]	The authors introduce the Compressive Sampling (CS) theory to the audio fingerprinting system for music recognition, by proposing a CS-based sub-Nyquist audio fingerprinting system. Authors claim that this system achieves an accuracy of 93.43% in reducing the sampling rate and in the extraction of musical features.
Umapathy et al. [43]	The authors present a novel local discriminant bases (LDB)-based audio classification scheme covering a wide range of audio signals. After the experiments, the obtained results suggest significant potential for LDB-based audio classification in auditory scene analysis or environment detection.
Kim et al. [44]	The authors develop a system that retrieves desired bookmarked video segments using audio fingerprint techniques based on the logarithmic modified Discrete Cosine Transform (DCT) modulation coefficients (LMDCT-MC) feature and two-stage bit vector searching method. The author's state that the search accuracy obtained is 99.67%.
Sert et al. [45]	The authors propose an audio fingerprinting model based on the Audio Spectrum Flatness (ASF) and Mel Frequency Cepstral Coefficients (MFCC) features, reporting and accuracy of 93% and 91%, respectively.
Ramalingam et al. [46]	The authors propose a method to create audio fingerprints by Gaussian Mixtures using features extracted from the short-time Fourier transform (STFT) of the signal. The experiments were performed on a database of 250 audio files, obtaining the highest identification rate of 99.2% with spectral centroid.
Ghouti et al. [47]	The authors propose a framework for robust identification of audio content by using short robust hashing codes, which applies the forward balanced multiwavelet (BMW) to transform each audio frame using 5 decomposition levels, and after the distribution of the subbands' coefficients into 32 different blocks, the estimation quantization (EQ) scheme and the hashes are computed.
Cook et al. [48]	The authors propose a system that allows audio content identification and association of metadata in very restricted embedded environments. The authors report that the system has better performance than the method based on a more traditional n-dimensional hashing scheme, but it achieves results with 2% less accuracy.
Seo et al. [49]	The authors propose an audio fingerprinting method based on the normalized Spectral Subband Centroid (SSC), where the match is performed using the square of the Euclidean distance. The normalized SSC obtains better results than the widely-used features, such as tonality and Mel Frequency Cepstral Coefficients (MFCC).
Haitsma et al. [50]	The authors present an approach to audio fingerprinting, but it has negligible effects on other aspects, such as robustness and reliability. They proved that the developed method is robust in case of linear speed changes.
Haitsma et al. [51]	The authors present an approach to audio fingerprinting, in which the fingerprint extraction is based on the extraction of a 32-bit sub-fingerprint every 11.8 millis. They also develop a fingerprint database and implement a two-phase search algorithm, achieving an excellent performance, and allowing the analytical modeling of false acceptance rates.

3. Results

As illustrated in Figure 1, our review identified 115 papers that included three duplicates, which were removed. The remaining 112 works were evaluated in terms of title, abstract, and keywords, resulting in the exclusion of 50 citations. Full text evaluation of the remaining 62 papers resulted in the exclusion of 22 papers that did not match the defined criteria. The remaining 40 papers were

included in the qualitative synthesis and the quantitative synthesis. In summary, our review examined 40 papers.

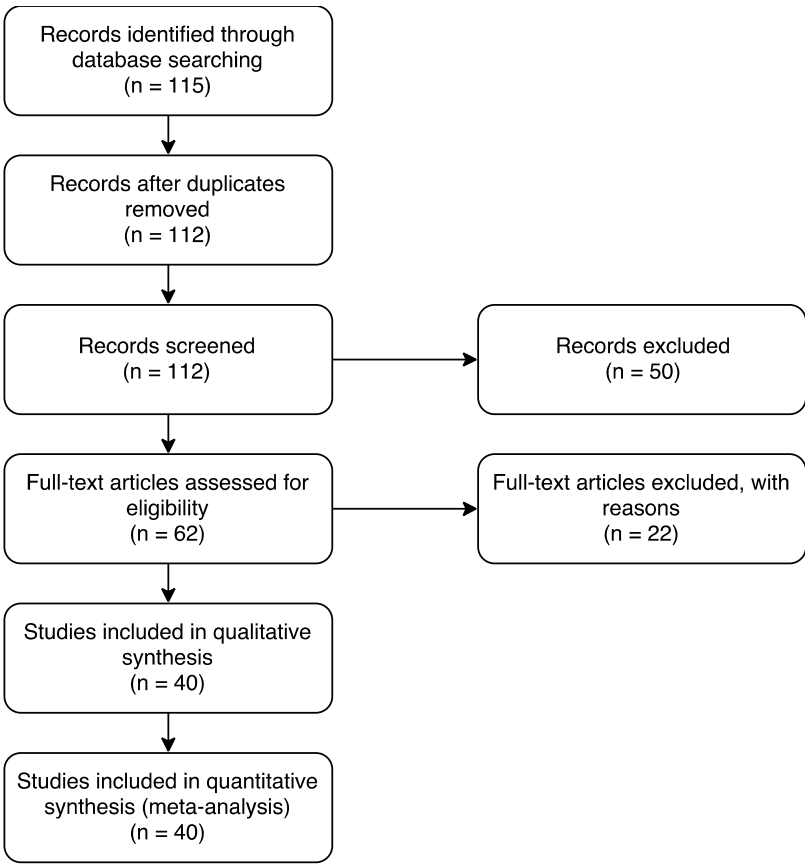

Figure 1. Flow diagram of identification and inclusion of papers.

We suggest that interested readers refer to the original cited works to find relevant information about the details of the methods analyzed in this review. Table 1 shows the year of publication, population, purpose of the study, devices, and settings of the selected papers. Table 2 shows study aims and results. As shown in Table 1, all studies have been performed in controlled environments (laboratory). The major part of the studies was performed between 2011 and 2016 with a total of 29 studies (73%), where five studies were in 2011 (13%), five studies in 2012 (13%), four studies in 2013 (10%), eight studies in 2014 (20%), six studies in 2015 (15%), and one study in 2016 (3%). Some studies indicate the devices used: eight studies used computer microphones (23%), 10 studies used mobile devices (25%), and two studies used a television (5%).

Methods for Audio Fingerprinting

In [12], the authors created a system that implements the framing, Fast Fourier Transform (FFT), calculation of the spectrum modules, extraction of two kinds of audio fingerprinting, and two level search of two kinds of audio fingerprinting. The two kinds were extracted calculating the sum of the spectrum modulus of every frame, getting the sum of global spectrum modulus in two stages. The authors reported that, when the signal noise rate (SNR) is 10 dB, the two level algorithm, with parallel processing, reports a precision of 100% [12].

In [14], several MP3 features were extracted, such as the Single local description, the Multiple local description, the Modified discrete cosine transform (MDCT), the Mel-frequency cepstrum coefficients (MFCC), the MPEG-7 descriptors, and the chroma vectors, using the Principal Component Analysis (PCA) technique to reduce the dimensionality and QUery Context (QUC)-tree to search for songs.

The tests of the methods described in [14] were performed with 10,000 MP3 fragments, reporting a maximum average error equals to 4.26%, which represents an accuracy around 96%. In [13], the same authors extracted the same features and used the same techniques, but they also used the MP3 signatures for the implementation of the audio fingerprinting method, performing tests with 100,000 MP3 fragments, which reported the errors and accuracies obtained are equals to the reported in [14].

Tsai et al. [15] presented a method to calculate audio fingerprints with 6 steps, namely compute spectrogram, collect context frames, apply eigenfilters, compute deltas, apply threshold, and bit packing. The authors reported that the developed method is more robust than the three other fingerprints (e.g., Shazam, Masked Audio Spectral Keypoints (MASK), and Panako), achieving an accuracy of 99.2% [15].

Another audio feature named local energy centroid (LEC) is used in [37] to obtain a representation of audio signals in noisy condition. The method for audio fingerprinting has several steps. First, the audio is downsampled to 8 kHz and segmented into frames, and then FFT is employed to obtain the spectrum. Later, the spectrum is optimized by applying weighted window functions with different size. Then, the LEC is saved and the amplitude components are removed, obtaining an audio spectrum that can be represented by sparse LEC set of coordinates [37]. The authors reported that the method is robust to different noise conditions, and when the linear speed is not changed, the audio fingerprint method based on LEC reports an accuracy of 100% [37].

In [36], the authors proposed an audio fingerprinting algorithm that starts with the application of low-pass filter to the audio signal and resampling to eliminate the high-frequency noise and other audio components that are perceptually insignificant for human auditory system. Afterwards, the audio is framed and weighted by Window function, and the FFT is applied [36]. Next, the Spectral Subband Centroid (SSC) is calculated and the approach of harmonic enhancement is adopted to estimate the predominant pitch of audio signal [36]. Finally, the normalized SSC is masked by the predominant pitch, and the proposed algorithm is resistant to some kinds of signal degradations in varying degrees, reporting an accuracy between 86% and 93% [36]. The authors of [35] also used the normalized SSC for the creation of an audio fingerprinting algorithm. The algorithm is structured in several phases, such as: pre-processing, framing, implementing the FFT to transform audio signals from time to frequency domain, implementing the dynamic subband locating, and applying the normalized SSC, obtaining, at the end, the audio fingerprint [35]. With the fingerprints created, the authors reported an accuracy up to 80% in normal conditions [35]. The authors of [49] also proposed an audio fingerprinting algorithm using SSC, starting with the conversion to mono and downsampling the audio to 11,025 Hz. After the downsampling, the audio signal is windowed by Hamming window (typically 371.5 ms) with 50% overlap and transformed into the frequency domain using FFT [49]. Afterwards, the audio spectrum is divided into 16 critical bands, and the frequency centroids of the 16 critical bands are used as a fingerprint of the audio frame [49], reporting an accuracy around 60% with MP3 and Random start, and an accuracy around 100% with Equalization.

Another algorithm is presented in [50] that consists of the modification of an existing algorithm named Streaming Audio Fingerprinting (SAF), which apply the framing and the FFT, create energy 33 bands, and then, apply a filter and a threshold. The modification of the algorithm consists of increasing the number of the energy bands, and three new steps between the creation of energy bands and the application of a filter and threshold: auto-correction, filter and the creation of a subsample [50]. The authors reported that the algorithm is robust in case of linear speed changes [50].

In [28], the audio fingerprinting methods proposed has several steps, these are framing, application of FFT or quadratically interpolated FFT (QIFFT), time averaging, peak detection, quadratic interpolation, sinusoidal quantification, frequency-axial discretization, and time-axial warping. A fingerprint that represents the distribution of pseudosinusoidal components in the time-frequency domain is generated, showing results with an accuracy around 96% and precision of 100% [28].

In [51], the authors proposed a new fingerprint algorithm based on the streaming approach, where the audio signal is segmented into overlapping frames, the FFT should be applied, and after that, the Human Auditory System (HAS) is used, reporting an accuracy of 100% for the recognition of pop-music.

In [45] is proposed a system for audio fingerprinting that starts with preprocessing and framing of the audio signal. Afterwards, a general feature extraction paradigm, extended with a descriptor based on structural similarity analysis with MPEG-7 Audio Spectrum Flatness (ASF), is applied to the signal [45]. The last step, before the fingerprint construction, consists of the structural analysis that results only the feature vector of the expressive audio piece [45]. At the end, the authors reduce the dimension of the ASF feature vector in the fingerprint construction stage based on the MPEG-7 Audio Signature (AS) description scheme [45], reporting an accuracy around 93%.

The authors of [46] proposed an audio fingerprinting scheme with several stages, such as preprocessing, framing, feature extraction, Gaussian mixture models (GMM) modelling, likelihood estimation, and comparison with a fingerprinting database. In the preprocessing stage, the audio signal is converted to a standard format (16-bit, pulse code modulation (PCM)) [46]. In the framing stage, the audio signals are divided into frames of length equals to 23 ms [46]. During feature extraction, the authors used the STFT, extracting several features, such as Shannon entropy, Rényi entropy, Spectral centroid, Spectral bandwidth, Spectral band energy, Spectral flatness measure, Spectral crest factor, and Mel-frequency cepstral coefficients (MFCC) [46]. Afterwards, the GMM models are applied, using the probability density function (PDF), and the Expectation-Maximization (EM) [46]. Among the features used, spectral centroid gives the highest identification rate of 99.2% [46].

The authors of [47] presented an audio fingerprint extraction algorithm, consisting of: downsampling of the input audio content of 3 s to obtain a sampling rate of 5512 Hz; applying the framing division on the downsampled content using Hamming window with an overlap factor of 31/32; computing the forward balanced multiwavelet (BMW) to transform for each audio frame using five decomposition levels; dividing the subbands' coefficients into 32 different blocks; applying the estimation quantization (EQ) scheme using a neighbouring window of five audio samples; computing the log variances of the magnitudes of the subbands' coefficients; computing the mean value of all the log variances for each audio frame; and at the end, extracting the sub-hash bit. Authors report that the performance of the algorithm degrades as the compression rate increases.

In [48], the authors proposed an algorithm with two stages named indexing and search. The indexing is based in the construction of zone tables using the Search by Range Reduction (SRR) threshold values [48]. The search is based on the SRR test, calculating the Itakura distance between two fingerprints, and comparing it with values in the zone tables [48]. An accuracy of around 98% is reported.

The authors of [43] proposed an algorithm with training and testing phases. For the training phase, the authors started with the wavelet packet decomposition, and developed a local discriminant bases (LDBs)-based automated multigroup audio classification system, which focuses on identifying discriminatory time-frequency subspaces [43]. The testing phase consists of the construction of a new wavelet tree, feature extraction, and implementation of a linear discriminant analysis (LDA) [43]. The extracted features include MFCC, spectral similarity, timbral texture, band periodicity, linear prediction coefficient derived cepstral coefficients (LPCCs), zero crossing rate, MPEG-7 descriptors, entropy, and octaves [43]. The authors of [43] reported that the average classification accuracy was between 91% and 99% [43].

The authors of [44] presents a video retrieval system (VRS) for Interactive-Television as like internet protocol television (IPTV), which implements an audio fingerprint feature of long-term logarithmic modified discrete cosine transform (DCT) modulation coefficients (LMDCT-MC) for audio indexing and retrieval, and implements two-stage search (TSS) algorithm for fast searching. In the first stage of TSS, candidate video segments are roughly determined with audio index bit vectors (IBV)

and then the optimal video clip is obtained by fingerprint bit vectors (FBV). An accuracy of 99.67% is reported in [44].

In [41] an audio fingerprint method using sub-fingerprint masking based on the predominant pitch extraction is proposed. It increases the accuracy of the audio fingerprinting system in a noisy environment dramatically, while requiring much less computing power compared to the expanded hash table lookup method. When applied to an audio signal without noise, the reported accuracy is 97.4%.

The authors of [42] presented a sub-Nyquist audio fingerprinting system for music recognition, which utilizes Compressive Sampling (CS) theory to generate a compact audio fingerprint, and to achieve significant reduction of the dimensionality of the input signal, compared to Nyquist sampling methods [42]. The average accuracy of this method is 93.43% under various distorted environments.

In [38], the authors presented an algorithm based on fingerprinting techniques implemented in a low-cost embedded reconfigurable platform. It utilizes the FFT implementation from the CUFFT library, based on the Fastest Fourier Transform in the West (FFTW) algorithm. This approach normalizes and frames the audio signal, computes the correlation and cross correlation, and applies a derivative of the audio signal. An accuracy of 94% is reported.

The authors of [39] created a fingerprint database of songs and focused on the problem of effective and efficient database search. The authors proposed a new indexing scheme that overcomes the limitations of Haitsma-Kalker's method and Miller's k-ary tree method, adopting the inverted file as the underlying index structure and developing the techniques to apply it to the effective and efficient audio fingerprinting problem. An accuracy higher than 97% is reported in [39], which is the performance of the Haitsma-Kalker's method.

The authors of [40] explored a common audio-fingerprinting approach with the implementation of FFT, and taken into account the noise in the derived fingerprints by employing error correcting codes and applying statistical tests. Testing with several sample windows of Network Time Protocol (NTP)-based synchronization recordings, authors of [40] reported an accuracy between 60% and 70%.

The authors of [31] created a system based on a client-server architecture able to recognize a live television show using audio fingerprinting. To create audio fingerprints, FFT is computed to obtain the power spectrum, which is integrated over a pre-defined set of non-overlapping, logarithmically spaced frequency bins and eventually squared to obtain an energy measure [31]. The likelihood estimation based on the cross-correlation function was used for comparison of the audio fingerprints. An accuracy of around 95% is reported in [31].

The authors of [32] presented an audio fingerprint method named Masked Audio Spectral Keypoints (MASK), which encodes the acoustic information existent in audio documents and discriminates between transformed versions of the same acoustic documents and other unrelated documents. The MASK fingerprint extraction method is composed of several tasks: time-to-frequency transformation, where the input signal is transformed from the time domain to the spectral domain, and transformed into Mel-scale; salient spectral points search; local mask application around each of the salient points; grouping of the different spectrogram values into regions; and the MASK fingerprint encoding and storage. The averaged energy values of each one of these spectrogram regions are compared to construct a fixed length binary descriptor. Authors of [32] report an accuracy around 58%.

In [33], the authors implemented an audio fingerprinting algorithm based on fingerprint extraction and matching search, adapting the well-known Philips' algorithm. The fingerprint extraction derives and encodes a set of relevant audio features, which need to be invariant to various kinds of signal distortion, including background noise, audio compression, and A/D conversion [33]. The matching search finds the best match between these fingerprints and those stored in the database [33]. The implemented audio fingerprint extraction method uses FFT, and extracts several features, such as: mel-frequency cepstral coefficients (MFCC), spectral centroid or spectral flatness [33]. The audio fingerprinting method reports an accuracy of 95% and a precision of 100% [33].

The authors of [34] proposed an audio fingerprinting system with several characteristics, including robustness, granularity, and retrieval speed, reporting an accuracy between 88% and 99%. The structure of the audio fingerprinting implemented is the same as all other algorithms presented in [34], applying the FFT and an High-pass filter. The authors used the local maximum chroma energy (LMCE) to extract the perception features of Tempo-Frequency domain [34].

The work presented in [29] reviews the state-of-the-art methods for improving the power consumption and computation speed to make the smartphone implementation. It also proposed the Improved Real-Time TV-channel Recognition (IRTR), which is a fingerprint extraction method aimed at recognizing in real time what people are watching on TV without any active user interaction. The identification using the audio fingerprint is performed using a likelihood estimation [29]. The audio fingerprinting method implements linear transform and feature extraction, with several steps: the audio is recorded and divided into frames with overlap factor; each frame is filtered by means of a Hamming window function; the application of the FFT and the squared modulus; the spectrum is divided into logarithmically spaced frequency bins and the energy is computed for each bin; and the nervy of band of each frame is denoted. An accuracy about 95% is reported in [29].

In [30], an audio fingerprinting algorithm is proposed for efficient retrieval of corresponding or similar items from large audio databases, which improves the of the database search compared to the algorithm used in Haitsma's method, without impairing the accuracy of the search results. The approach implements the FFT, the extraction of candidate songs via lookup table, the assignment of weights to candidate songs, and the database search [30], while reporting an average accuracy around 81%.

The authors of [21] proposed an audio fingerprinting algorithm, which improves binary audio fingerprint matching performance by utilizing auxiliary information. The proposed matching method is based on Philips robust hash (PRH) for audio signal; Asymmetric Fingerprint Matching for PRH using the Magnitude Information, which consists of Normalization of the Subband-Energy Difference; and Fingerprint Matching Based on the Likelihood Ratio Test [21]. The proposed method yields better performance than the conventional Hamming distance [21].

The authors of [22] proposed an audio fingerprinting constituted by two stages: fingerprinting and matching. The fingerprinting module uses a log-frequency spectrogram based on the Constant Q Transform (CQT), and an adaptive thresholding method based on two-dimensional median filtering [22]. The matching uses the Hamming similarity and the Hough Transform [22]. The reported accuracy is between 61% and 81%.

The authors of [23] presented a method for the construction of audio fingerprints based on: maximization of the mutual information across the distortion channel; using the information bottleneck method to optimize the filters; and quantizers that generate these fingerprints. The method starts with the application of the short time Fourier transform (STFT), and capturing the Spectral Sub-band Centroids (SSC) using 16 bins on the Bark scale. The generated features with [23] result in a maximum accuracy of around 65%.

The authors of [24] implemented an audio fingerprinting algorithm composed by several steps: downsampling to 5 kHz, segmenting frames every 11.6 ms, applying the FFT, calculating the frequency bands energies, and finally, calculating the fingerprints. A recall around 98% is reported.

In [25], the authors presented an audio fingerprinting algorithm based on the compressed-domain spectral entropy as audio features, showing strong robustness against various audio signal distortions such as recompression, noise interference, echo addition, equalization, band-pass filtering, pitch shifting, moderate time-scale modification, among others. The algorithm includes four steps: granule grouping, frequency alignment between long and short windows, coefficients selection and subband division, and MDCT spectral entropy calculation and fingerprint modelling [25]. It reports an accuracy above 90%.

In [26], the authors presented the implementation of an audio fingerprinting system, using graphic processing units (GPUs). The system starts with the extraction of landmarks using FFT, and continues

with the landmark extraction, lookup, and analysis. The authors explored the use of one thread for one hash key, and one block for one hash key, reporting an accuracy around 80.96%, when there are 100,000 songs in the database [26].

The authors of [27] proposed a high-performance audio fingerprint extraction method for identifying TV commercial advertisement. The audio fingerprint extraction consists of a salient audio peak pair fingerprints based on constant Q transform (CQT). The algorithm obtains the audio fingerprints through five main steps: preprocessing; application of the CQT; application of the Mean Subtraction of Logarithmic CQT Spectrum; application of the CQT Based Salient Peak Detection using forward and Backward Filtering; and finally, application of the Fingerprint Generation using CQT Peak Pair. The reported recognition accuracy of the method based on CQT, presented in [27], is around 89.8%.

The authors of [16] used a smartphone and create an audio fingerprinting algorithm based on the joint usage of GPS and acoustic fingerprints. The authors created an audio fingerprinting algorithm with noise tolerance, assessing it under several conditions [16]. The algorithm starts with the calculation of the audio sample spectrogram using the STFT, and then calculates audio sample spectrogram using the Hamming window and a high overlap [16]. Next, it takes only the first 40 frequency bins, as most of the useful audio features are in that bandwidth, averaging the logarithmic amplitude in each bin [16]. Afterwards, for each frequency bin, a 16-bit fingerprint is calculated [16]. The 16-bits fingerprint is then stored with the associated frequency and time [16]. For the comparison of the audio fingerprints, the Hamming distances between each fingerprint are calculated, looking for a minimum [16]. An accuracy of around 86% is reported.

In [17], the authors proposed an approach to accelerate fingerprinting techniques by skipping the search for irrelevant sections of the signal and demonstrate its application to the divide and locate (DAL) audio fingerprint method. The method in DAL starts with the extraction of the time-frequency power spectral applied for the signals, normalizing each logarithmic power [17]. Afterwards, the normalized data is decomposed into a number of small time-frequency components of uniform size, and thus, the computational cost and memory usage are reduced in the fingerprint data [17]. The authors verified that with a reduced search threshold, the accuracy of the recognition is around 100% [17].

The authors of [18] created a method to analyze and classify daily activities in personal audio recordings (PARs). The method applies: speech activity detection (SAD), speaker diarization systems, and computing the number of audio speech and lexical features [18]. It uses a TO-Combo-SAD (Threshold Optimized Combo SAD) algorithm for separating speech from noise [18]. The Principal Component Analysis (PCA) is first applied for dimensionality reduction, and then, the remaining features are supplied to a multi-class support vector machine (SVM) with radial basis function (RBF) kernel for model training and evaluation [18]. The authors performed recognition of faculty meeting, research meeting, staff meeting, alone time, and conference call, reporting accuracies between 62.78% and 84.25% [18].

In [19], the authors proposed an audio fingerprinting method, based on landmarks in the audio spectrogram. The algorithm is based on the audio hashing of frequency peaks in the spectrogram [19]. It starts with the application of the FFT, thresholding the data, applying a high pass filter, identifying the local maximums and finding the peaks of the spectrogram [19]. The performance of the algorithm decreases at higher additive noise in comparison with other algorithms [19], reporting an accuracy around 96.71%.

In [20], the authors proposed a robust TV advertisement search based on audio fingerprinting in real environments. This algorithm has several steps, such as preprocessing, logarithmic moduled complex lapped transform (LMCLT), two-are segmentation using adaptive thresholding based on median filtering, detection of prominent LMCLT spectral peaks, and fingerprinting generation using LMCLT peak pair [20]. The method applies adaptive peak-picking thresholding method to extract more salient and distinct peak pairs for comparing the query fingerprint with the original fingerprints, and the authors reported an accuracy of 86.5% [20].

4. Discussion

This review confirms the findings of previous studies related to the use of audio fingerprinting techniques for identification of the environment related to the different ADLs. We consider that many of the reviewed works raise important issues regarding the concept of Open Science, including, but not limited to, Reproducibility and Verifiability of the research results, and Comparability of similar research. Many of them were evaluated on unpublished data and did not publish their source code, although when commercial solutions are in perspective, a necessary degree of confidentiality is understandable. Regarding validation and comparability, only six studies used raw data available online or published its research data online. Likewise, only three studies presented some parts of the code used of the experiments. In addition, the studies that used data that is now publicly available, did not publish the research source code, making the validation of the results and further comparative research an impossible task. Therefore, we suggest to the audio fingerprinting community to become better at sharing raw data and algorithms, so as to be able to recreate and evaluate the soundness of previous studies.

Nevertheless, assuming the results of the presented research studies are comparable, Tables 3–5 present a summary of the Features and Methods ordered by the number of identified studies that use these features and methods.

Tables 3 and 4 present the distribution of the extracted features and methods implemented in the analyzed studies, verifying that FFT is one of the most widely used feature extraction method, because it extracts the frequencies from the audio signal, and the other most used features include thresholding, normalized Spectral Subband Centroid (SSC), Mel-frequency cepstrum coefficients (MFCC), maximum, local peaks and landmarks, Shannon entropy, Rényi entropy, MPEG-7 descriptors, Spectral bandwidth, Spectral flatness measure, Modified discrete cosine transform (MDCT), Constant Q Transform (CQT), Short-time Fourier transform (STFT), average, and the maximum and minimum. These features were used in a large part of the analyzed studies [12,14,19,24,26,28–31,33–35,38,49–51], and with them, the reported accuracy is greater than 80%, as presented in Table 3.

For Tables 3 and 4, the accuracies that are equal or higher than 99% are shown in a different background color (yellow).

Table 3. Distribution of the features extracted in the studies.

Features	Average Accuracy of Features	Number of Studies
Fast Fourier Transform (FFT)	93.85%	16
Thresholding	90.49%	6
Normalized spectral subband centroid (SSC)	93.44%	5
Mel-frequency cepstrum coefficients (MFCC)	97.30%	4
Maximum	87.57%	3
Local peaks and landmarks	82.32%	3
Shannon entropy	99.10%	2
Rényi entropy	99.10%	2
MPEG-7 descriptors	97.50%	2
Spectral bandwidth	97.10%	2
Spectral flatness measure	97.10%	2
Modified discrete cosine transform (MDCT)	93.00%	2
Constant Q transform (CQT)	85.40%	2
Short-time Fourier transform (STFT)	84.50%	2
Average	83.00%	2
Minimum	83.00%	2
Sum of the spectrum modulus of every frame	100.00%	1
Sum of global spectrum modulus in two stages	100.00%	1
Local energy centroid (LEC)	100.00%	1
Time-frequency power spectral	100.00%	1

Table 3. *Cont.*

Features	Average Accuracy of Features	Number of Studies
Long-term logarithmic modified discrete cosine transform (DCT) modulation coefficients (LMDCT-MC)	99.67%	1
Bit packing	99.20%	1
Spectral band energy	99.20%	1
Spectral crest factor	99.20%	1
Spectral similarity	99.00%	1
Timbral texture	99.00%	1
Band periodicity	99.00%	1
Linear prediction coefficient derived cepstral coefficients (lpccs)	99.00%	1
Zero crossing rate	99.00%	1
Octaves	99.00%	1
Single local description	96.00%	1
Multiple local description	96.00%	1
Chroma vectors	96.00%	1
MP3 signatures	96.00%	1
Time averaging	96.00%	1
Quadratic interpolation	96.00%	1
Sinusoidal quantification	96.00%	1
Frequency-axial discretization	96.00%	1
Time-axial warping	96.00%	1
Logarithmic moduled complex lapped transform spectral peaks	86.50%	1
Correlation coefficient	70.00%	1
Matching score	70.00%	1

Table 4. Distribution of the methods implemented in the studies.

Methods	Average Accuracy of Methods	Number of Studies
Other methods	90.78%	15
Two level search algorithm	99.84%	2
Likelihood estimation	97.10%	3
Principal Component Analysis (PCA)	90.13%	2
Hamming distances between each fingerprint	83.50%	2
Streaming audio fingerprinting (SAF)	100.00%	1
Human auditory system (HAS)	100.00%	1
Divide and locate (DAL)	100.00%	1
Gaussian mixture models (GMM) modelling	99.20%	1
Local discriminant bases (LDBS)-based automated multigroup audio classification system	99.00%	1
Linear discriminant analysis (LDA)	99.00%	1
Local maximum chroma energy (LMCE)	99.00%	1
Expanded hash table lookup method	97.40%	1
Query Context (QUC)-tree	96.00%	1
Improved Real-Time TV-channel Recognition (IRTR)	95.00%	1
Sub-Nyquist fudio fingerprinting system	93.43%	1
Logarithmic moduled complex lapped transform (LMCLT) peak pair	86.50%	1
TO-Combo-SAD (Threshold Optimized Combo SAD) algorithm	84.25%	1
Support vector machine (SVM)	84.25%	1
Hough Transform between each fingerprint	81.00%	1
Masked audio spectral keypoints (MASK)	58.00%	1

On the other hand, as verified in Table 4, a large part of the analyzed studies [12,14,19,24,26,28–31,33–35,38,49–51] do not mention the name of the method applied, presenting only the features used. Regarding the undifferentiated methods, the most used methods are the two level search algorithm, the likelihood estimation, the Principal Component Analysis (PCA), and the Hamming distances between each fingerprint, reporting accuracies also higher than 80%.

Table 5 presents in a matrix format, the average of the averages of the accuracies in Methods vs. its Features. This table is a mere comparison exercise, as there are not enough studies to sustain a valid analysis of the use of different features with different methods. On the other hand, this table assumes that these results are comparable, and moreover, that any method or algorithm can be used with any set of features, which of course, is a very wide, and possibly not true assumption. Nevertheless, Table 5 shows, in a colored background the match between features and methods. For example, for method SAF (Streaming Audio Fingerprinting) the set of used features are Fast Fourier Transform, Thresholds and Energy bands, whose mean accuracies in the found studies are not higher than 99%. Also, for example for the method GMM (Gaussian Mixture Models Modelling), besides the 4 highlighted features that were used, this method uses additionally 5 other sets of features.

Taking Table 5 into consideration, one can identify Shannon's Entropy as the feature that is most used in the most accurate number of studies. Arguably, this table may propose new combinations of features and methods that can be used to devise audio-fingerprinting solutions.

For a particular use, the methods to be implemented must be chosen according to their complexity, the computational power of the use case scenario, and to the purpose of its intended use. This review is focused on the use of mobile devices, but only three of the reviewed works argue that they use methods that need low resources (see Table 1). Only 19 studies compared the implemented methods with other methods published in the literature and present their accuracy, claiming an increased accuracy in the recognition of the environment using audio fingerprinting.

According to the results of this review, the use of the mobile devices for the application of audio fingerprinting techniques is limited, because of the restrictions these devices impose, i.e., low power processing and battery capacity. Thus, only 10 of the analyzed studies utilize mobile devices with local processing or server-side processing of the data acquired from the mobile devices. In the case of the server-side processing, the use of the mobile devices implies a constant and stable network connection, which is not a trivial requirement both from technical perspective, but also because of battery life implications. To some extent, the using Fog and Mist Computing paradigms could overcome the challenges of the client-server architectures. The creation of lightweight techniques should be explored, as they could be executed on mobile devices (i.e., edge-nodes). The models could be recalibrated offline on the server occasionally, and then, as pre-trained models to be seamlessly redeployed on mobile devices.

In conclusion, only one of the reviewed studies [38] can achieve reliable performance with reduced computational cost and memory usage. It utilizes the FFT implementation in the CUFFT library, divide and locate (DAL) audio fingerprint method, and sub-fingerprint masking based on the predominant pitch extraction methods. However, other methods could be implemented on mobile devices with some restrictions. Nonetheless, they could be amended to utilize more lightweight implementations of the underlying libraries, or by sacrificing floating point precision, for instance.

Table 5. Potential accuracies for the top most accurate methods vs. top most mean accurate features (mean accuracies equal or higher than 99%, according to its authors).

		SAF	**HAS**	**DAL**	**TLS**	**GMM**	**LDBS**	**LDA**	**LMCE**
		100.00%	***100.00%***	***100.00%***	***99.84%***	***99.20%***	***99.00%***	***99.00%***	***99.00%***
Local energy centroid (LEC)	***100.00%***	100.00%	100.00%	100.00%	99.92%	99.60%	99.50%	99.50%	99.50%
Sum of global spectrum modulus in two stages	***100.00%***	100.00%	100.00%	100.00%	99.92%	99.60%	99.50%	99.50%	99.50%
Sum of the spectrum modulus of every frame	***100.00%***	100.00%	100.00%	100.00%	99.92%	99.60%	99.50%	99.50%	99.50%
Time-frequency power spectral	***100.00%***	100.00%	100.00%	100.00%	99.92%	99.60%	99.50%	99.50%	99.50%
Long-term logarithmic modified discrete cosine transform (DCT) modulation coefficients (LMDCT-MC)	***99.67%***	99.84%	99.84%	99.84%	99.76%	99.44%	99.34%	99.34%	99.34%
Bit packing	***99.20%***	99.60%	99.60%	99.60%	99.52%	99.20%	99.10%	99.10%	99.10%
Spectral band energy	***99.20%***	99.60%	99.60%	99.60%	99.52%	99.20%	99.10%	99.10%	99.10%
Spectral crest factor	***99.20%***	99.60%	99.60%	99.60%	99.52%	99.20%	99.10%	99.10%	99.10%
Rényi entropy	***99.10%***	99.55%	99.55%	99.55%	99.47%	99.15%	99.05%	99.05%	99.05%
Shannon entropy	***99.10%***	99.55%	99.55%	99.55%	99.47%	99.15%	99.05%	99.05%	99.05%
Band periodicity	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%
Linear prediction coefficient derived cepstral coefficients (lpccs)	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%
Octaves	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%
Spectral similarity	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%
Timbral texture	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%
Zero crossing rate	***99.00%***	99.50%	99.50%	99.50%	99.42%	99.10%	99.00%	99.00%	99.00%

5. Conclusions

This review identified and described the methodologies used for audio fingerprinting that can be applied to mobile technologies. Forty-seven studies were examined and the main findings are summarized as follows:

- (RQ1) the audio fingerprinting is defined as the ability to recognize the scenario in which a given audio was collected and involved in, based on various methods.
- (RQ2) Several techniques have been applied to implement audio fingerprinting methods, including Fast Fourier Transform (FFT). Support Vector Machine (SVM), QUery Context (QUC)-tree, spectral subband centroid (SSC), Streaming Audio Fingerprinting (SAF), Human Auditory System (HAS), Gaussian mixture models (GMM) modelling, likelihood estimation, linear discriminant analysis (LDA), Compressive Sampling (CS) theory, Philips robust hash (PRH), Asymmetric Fingerprint Matching, and TO-Combo-SAD (Threshold Optimized Combo SAD). These techniques yield high accuracy, and the use of mobile devices does not influence the predictive performance, allowing the use of these techniques anywhere, anytime.
- (RQ3) All of the methods presented in RQ2 can be implemented on mobile devices, but the methods that require lower computational resources are FFT with the CUFFT library, divide and locate (DAL) audio fingerprint method, and sub-fingerprint masking based on the predominant pitch extraction.

In addition, this review highlights the application of audio fingerprinting techniques on mobile or other devices with limited computational and battery resources. Some limitations of this review should be mentioned. First, the authors chose to exclude studies that are not focused on audio fingerprinting techniques. Second, the studies that do not utilize mobile devices have been excluded. These exclusions were performed with the analysis of the abstract and then, the full text of the papers. Finally, only English-language publications were included.

Based on the analysis, we conclude that the most used methods are undifferentiated methods, two level search algorithms, likelihood estimation, Principal Component Analysis (PCA), and Hamming distances between each fingerprint. The conclusion is that the use of statistical methods reports results with an accuracy higher than 80%. Furthermore, the most used features are Fast Fourier Transform (FFT), Thresholding, normalized spectral subband centroid (SSC), Mel-frequency cepstrum coefficients (MFCC), maximum, local peaks and landmarks, Shannon entropy, Rényi entropy, MPEG-7 descriptors, Spectral bandwidth, Spectral flatness measure. Modified discrete cosine transform (MDCT), Constant Q Transform (CQT), Short-time Fourier transform (STFT), average, and minimum, which also result in accuracies greater than 80%.

As future work, the extraction of features based on audio fingerprinting will be implemented in order to develop a system for the recognition of ADLs and their environments, presented in [9–11]. As presented in Table 3, the accuracy is always higher than 80%. Then, we should consider the most used features, including FFT, MFCC, average, maximum, and minimum, in order to better handle the recognition of the environment. The implementation of this framework is part of the development of a personal digital life coach [4].

Acknowledgments: This work was supported by FCT project **UID/EEA/50008/2013** (*Este trabalho foi suportado pelo projecto FCT UID/EEA/50008/2013*). The authors would also like to acknowledge the contribution of the COST Action IC1303—AAPELE—Architectures, Algorithms and Protocols for Enhanced Living Environments.

Author Contributions: All the authors have contributed with the structure, content, and writing of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Foti, D.; Koketsu, J.S. *Pedretti's Occupational Therapy: Practical Skills for Physical Dysfunction*, 7th ed.; Activities of daily living; Mosby: St. Louis, MI, USA, 2013; pp. 157–232.
2. Garcia, N.M.; Rodrigues, J.J.P. *Ambient Assisted Living*; CRC Press: Boca Raton, FL, USA, 2015.
3. Dobre, C.; Mavromoustakis, C.X.; Goleva, R.L. *Ambient Assisted Living and Enhanced Living Environments: Principles, Technologies and Control*; Butterworth-Heinemann: Oxford, UK, 2016; p. 552.
4. Garcia, N.M. A Roadmap to the Design of a Personal Digital Life Coach. In *ICT Innovations 2015*; Springer: Cham, Switzerland, 2016.
5. Da Silva, J.R.C. *Smartphone Based Human Activity Prediction*; Faculdade de Engenharia: Porto, Portugal, 2013.
6. Bieber, G.; Luthardt, A.; Peter, C.; Urban, B. The Hearing Trousers Pocket—Activity Recognition by Alternative Sensors. In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, Crete, Greece, 25–27 May 2011.
7. Kazushige, O.; Miwako, D. Indoor-outdoor activity recognition by a smartphone. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; p. 537.
8. Ganti, R.K.; Srinivasan, S.; Gacic, A. Multisensor Fusion in Smartphones for Lifestyle Monitoring. In Proceedings of the 2010 International Conference on Body Sensor Networks, Singapore, 7–9 June 2010.
9. Pires, I.M.; Garcia, N.M.; Pombo, N.; Flórez-Revuelta, F. From Data Acquisition to Data Fusion: A Comprehensive Review and a Roadmap for the Identification of Activities of Daily Living Using Mobile Devices. *Sensors* **2016**, *16*, 184. [CrossRef] [PubMed]
10. Pires, I.M.; Garcia, N.M.; Flórez-Revuelta, F. Multi-sensor data fusion techniques for the identification of activities of daily living using mobile devices. In Proceedings of the ECMLPKDD 2015 Doctoral Consortium, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, 7–11 September 2015.
11. Pires, I.M.; Garcia, N.M.; Pombo, N.; Flórez-Revuelta, F. Identification of Activities of Daily Living Using Sensors Available in off-the-shelf Mobile Devices: Research and Hypothesis. In *Ambient Intelligence-Software and Applications, Proceedings of the 7th International Symposium on Ambient Intelligence (ISAmI 2016), Seville, Spain, 1–3 June 2016*; Springer International Publishing: Cham, Switzerland, 2016.
12. Sui, D.; Ruan, L.; Xiao, L. A Two-level Audio Fingerprint Retrieval Algorithm for Advertisement Audio. In Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia, Kaohsiung, Taiwan, 8–10 December 2014; pp. 235–239.
13. Liu, C.-C.; Chang, P.-F. An efficient audio fingerprint design for MP3 music. In Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia, Ho Chi Minh City, Vietnam, 5–7 December 2011; pp. 190–193.
14. Liu, C.-C. MP3 sniffer: A system for online detecting MP3 music transmissions. In Proceedings of the 10th International Conference on Advances in Mobile Computing, Bali, Indonesia, 3–5 December 2012; pp. 93–96.
15. Tsai, T.J.; Stolcke, A. Robust and efficient multiple alignment of unsynchronized meeting recordings. *IEEE/ACM Trans. Audio Speech Lang. Proc.* **2016**, *24*, 833–845. [CrossRef]
16. Casagranda, P.; Sapino, M.L.; Candan, K.S. Audio assisted group detection using smartphones. In Proceedings of the 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), Torino, Italy, 29 June–3 July 2015.
17. Nagano, H.; Mukai, R.; Kurozumi, T.; Kashino, K. A fast audio search method based on skipping irrelevant signals by similarity upper-bound calculation. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015.
18. Ziaei, A.; Sangwan, A.; Kaushik, L.; Hansen, J.H.L. Prof-Life-Log: Analysis and classification of activities in daily audio streams. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015.
19. George, J.; Jhunjhunwala, A. Scalable and robust audio fingerprinting method tolerable to time-stretching. In Proceedings of the 2015 IEEE International Conference on Digital Signal Processing (DSP), Singapore, 21–24 July 2015.
20. Kim, H.G.; Cho, H.S.; Kim, J.Y. TV Advertisement Search Based on Audio Peak-Pair Hashing in Real Environments. In Proceedings of the 2015 5th International Conference on IT Convergence and Security (ICITCS), Kuala Lumpur, Malaysia, 24–27 August 2015.

21. Seo, J.S. An Asymmetric Matching Method for a Robust Binary Audio Fingerprinting. *IEEE Signal Process. Lett.* **2014**, *21*, 844–847.
22. Rafii, Z.; Coover, B.; Han, J. An audio fingerprinting system for live version identification using image processing techniques. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014.
23. Naini, R.; Moulin, P. Fingerprint information maximization for content identification. In Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014.
24. Yang, G.; Chen, X.; Yang, D. Efficient music identification by utilizing space-saving audio fingerprinting system. In Proceedings of the 2014 IEEE International Conference on Multimedia and Expo (ICME), Chengdu, China, 14–18 July 2014.
25. Yin, C.; Li, W.; Luo, Y.; Tseng, L.-C. Robust online music identification using spectral entropy in the compressed domain. In Proceedings of the Wireless Communications and Networking Conference Workshops (WCNCW), Istanbul, Turkey, 6–9 April 2014.
26. Wang, C.C.; Jang, J.S.R.; Liou, W. Speeding up audio fingerprinting over GPUs. In Proceedings of the 2014 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 7–9 July 2014.
27. Lee, J.Y.; Kim, H.G. Audio fingerprinting to identify TV commercial advertisement in real-noisy environment. In Proceedings of the 2014 14th International Symposium on Communications and Information Technologies (ISCIT), Incheon, South Korea, 24–26 September 2014.
28. Shibuya, T.; Abe, M.; Nishiguchi, M. Audio fingerprinting robust against reverberation and noise based on quantification of sinusoidality. In Proceedings of the 2013 IEEE International Conference on Multimedia and Expo (ICME), San Jose, CA, USA, 15–19 July 2013.
29. Bisio, I.; Delfino, A.; Lavagetto, F.; Marchese, M. A Television Channel Real-Time Detector using Smartphones. *IEEE Trans. Mob. Comput.* **2015**, *14*, 14–27. [CrossRef]
30. Lee, S.; Yook, D.; Chang, S. An efficient audio fingerprint search algorithm for music retrieval. *IEEE Trans. Consum. Electron.* **2013**, *59*, 652–656. [CrossRef]
31. Bisio, I.; Delfino, A.; Luzzati, G.; Lavagetto, F.; Marchese, M.; Fra, C.; Valla, M. Opportunistic estimation of television audience through smartphones. In Proceedings of the 2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Genoa, Italy, 8–11 July 2012.
32. Anguera, X.; Garzon, A.; Adamek, T. MASK: Robust Local Features for Audio Fingerprinting. In Proceedings of the 2012 IEEE International Conference on Multimedia and Expo, Melbourne, Australia, 9–13 July 2012.
33. Duong, N.Q.K.; Howson, C.; Legallais, Y. Fast second screen TV synchronization combining audio fingerprint technique and generalized cross correlation. In Proceedings of the 2012 IEEE International Conference on Consumer Electronics (ICCE-Berlin), Berlin, Germany, 3–5 September 2012.
34. Wang, H.; Yu, X.; Wan, W.; Swaminathan, R. Robust audio fingerprint extraction algorithm based on 2-D chroma. In Proceedings of the 2012 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 16–18 July 2012.
35. Xiong, W.; Yu, X.; Wan, W.; Swaminathan, R. Audio fingerprinting based on dynamic subband locating and normalized SSC. In Proceedings of the 2012 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 16–18 July 2012.
36. Jijun, D.; Wan, W.; Yu, X.; Pan, X.; Yang, W. Audio fingerprinting based on harmonic enhancement and spectral subband centroid. In Proceedings of the IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2011), Shanghai, China, 14–16 November 2011.
37. Pan, X.; Yu, X.; Deng, J.; Yang, W.; Wang, H. Audio fingerprinting based on local energy centroid. In Proceedings of the IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2011), Shanghai, China, 14–16 November 2011.
38. Martinez, J.I.; Vitola, J.; Sanabria, A.; Pedraza, C. Fast parallel audio fingerprinting implementation in reconfigurable hardware and GPUs. In Proceedings of the 2011 VII Southern Conference on Programmable Logic (SPL), Cordoba, Argentina, 13–15 April 2011.
39. Cha, G.H. An Effective and Efficient Indexing Scheme for Audio Fingerprinting. In Proceedings of the 2011 5th FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE), Crete, Greece, 28–30 June 2011.

40. Schurmann, D.; Sigg, S. Secure Communication Based on Ambient Audio. *IEEE Trans. Mob. Comput.* **2013**, *12*, 358–370. [CrossRef]
41. Son, W.; Cho, H.-T.; Yoon, K.; Lee, S.-P. Sub-fingerprint masking for a robust audio fingerprinting system in a real-noise environment for portable consumer devices. *IEEE Trans. Consum. Electron.* **2010**, *56*, 156–160. [CrossRef]
42. Chang, K.K.; Pissis, S.P.; Jang, J.-S.R.; Iliopoulos, C.S. Sub-nyquist audio fingerprinting for music recognition. In Proceedings of the Computer Science and Electronic Engineering Conference (CEEC), Colchester, UK, 8–9 September 2010.
43. Umapathy, K.; Krishnan, S.; Rao, R.K. Audio Signal Feature Extraction and Classification Using Local Discriminant Bases. *IEEE Trans. Audio Speech Lang. Proc.* **2007**, *15*, 1236–1246. [CrossRef]
44. Kim, H.G.; Kim, J.Y.; Park, T. Video bookmark based on soundtrack identification and two-stage search for interactive-television. *IEEE Trans. Consum. Electron.* **2007**, *53*, 1712–1717.
45. Sert, M.; Baykal, B.; Yazici, A. A Robust and Time-Efficient Fingerprinting Model for Musical Audio. In Proceedings of the 2006 IEEE International Symposium on Consumer Electronics, St Petersburg, Russia, 28 June–1 July 2006.
46. Ramalingam, A.; Krishnan, S. Gaussian Mixture Modeling of Short-Time Fourier Transform Features for Audio Fingerprinting. *IEEE Trans. Inf. Forensics Secur.* **2006**, *1*, 457–463. [CrossRef]
47. Ghouti, L.; Bouridane, A. A fingerprinting system for musical content. In Proceedings of the 2006 14th European Signal Processing Conference, Florence, Italy, 4–8 September 2006.
48. Cook, R.; Cremer, M. A Tunable, Efficient, Specialized Multidimensional Range Query Algorithm. In Proceedings of the 2006 IEEE International Symposium on Signal Processing and Information Technology, Vancouver, BC, Canada, 28–30 August 2006.
49. Seo, J.S.; Jin, M.; Lee, S.; Jang, D.; Lee, S.; Yoo, C.D. Audio fingerprinting based on normalized spectral subband centroids. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 05), Philadelphia, PA, USA, 23 March 2005.
50. Haitsma, J.; Kalker, T. Speed-change resistant audio fingerprinting using auto-correlation. In Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 03), Hong Kong, China, 6–10 April 2003.
51. Haitsma, J.; Kalker, T.; Oostveen, J. An efficient database search strategy for audio fingerprinting. In Proceedings of the 2002 IEEE Workshop on Multimedia Signal Processing, St.Thomas, VI, USA, 9–11 December 2002.