ISEL – Instituto Superior de Engenharia de Lisboa
ADEETC – Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIM
Licenciatura em Engenharia Informática e Multimédia
Unidade Curricular de Projeto

Localização de Sons para Diferentes Topologias

(eventual) imagem ilustrativa do trabalho – *dimensão*: até 13cm x 4cm

Carlos Almeida 38415

Orientador(es)

Professor Joel Paulo

Professor Gonçalo Marques

Setembro, 2017

Resumo

Este projeto pretende comparar diversos algoritmos em ambito de localização de eventos sonóros em duas diferentes topologias. Estas topologias são a de Ambisonic, em que o aparelho contém 4 microfones unidirecionais todos centrados no mesmo ponto, e a topologia em triângulo, em que 3 (três) microfones estao separados entre si em forma de triângulo.

Na topologia Ambisonic segue-se uma abordagem baseada na energia dos diferentes sinais. Assim o calculo do angulo do evento sonóro é calculado unicamente na diferença de energias de cada microfone.

Na topologia em triângulo a abordagem é bastante diferente. Aqui o calculo do ângulo do evento é feito com base nos atrasos de chegada do som em cada par de microfones.

Para além de algoritmos de localização estudei e desenvolvei ainda algoritmos de filtros de ruído, detecção de eventos sonóros e separação de eventos.

O primeiro passo seria o de filtrar o sinal tentando remover ruído indesejado. Este processo revelou-se delicado no ambito deste projeto. Isto porque um sinal com ruído a mais dificulta a detecção de eventos assim como a sua localização mas remover ruído *a mais* pode provocar também em si erros de deteção e localização de eventos. O ideal, e desejado, seria desenvolver um algoritmo que retira-se apenas o ruído deixando "intacta" toda a informação proveniente dos eventos sonóros que queremos detectar e localizar.

O segundo passo foi o de detetar eventos para assim poder aplicar os algoritmos de localização apenas quando existe, de facto, algum evento sonóro. Seria redundante e enganador localizar ângulos em momentos sem eventos sonóros em que haveria apenas ruído de fundo.

O terceiro passo diz respeito à localização dos eventos. Neste projeto foi desprezado o factor do ângulo vertical (elevação). Assim, e sendo estes microfones propostos para funcionarem em ambiente de cidade ou também em espaços fechados, pudémos assumir que todos os eventos se localizavam no mesmo plano horizontal. Neste terceiro passo desenvolvi dois algoritmos diferentes, com base em documentos já existentes, um para cada topologia.

O quarto passo só foi acrescentado este ano e é o passo em que se efetua

a separação sonóra de eventos. Desta forma seria possível isolar cada um dos eventos, mesmo simultâneos, o que seria fundamental para poder efetuar-se uma boa classificação. Quanto a esta classifcação poderia ser humana, em que uma pessoa ouviria o sinal de cada evento isolado e então poderia identificá-lo, ou também poderia ser feita com algoritmos de classifcação por aprendizagem automática ou *Machine Learning*. Este algoritmos poderão ser uma etapa a ser explorada depois do termino deste projeto. O passo de classificação de sinais não será aboradado neste projeto ou relatório.

Índice

Lista de Figuras

Capítulo 1

Introdução

Este projeto pretende permitir a localização de eventos sonóros em ambiente urbano.

Passando a explicar o projeto e a sua utilidade, este trabalho pode, entre outras coisas, medir o ruído existente num dado local a uma dada hora ou dia, detetar a quantidade de tráfego existente numa determinada rua, detetar um disparo duma arma, detetar uma pessoa a gritar ou um acidente automóvel. Outro uso possível para este projeto seria o de usar a informação da localização para redirecionar uma câmara de video-vigilância.

Pretende-se, numa fase mais avançada, que este algoritmo possa estar ligado a uma central de bombeiros e polícia e que estes possam receber alertas para eventos sonóros mais relevantes para cada entidade.

Na totalidade, este projecto engloba duas arquiteturas que abordam este desafio. Foram feitos algoritmos e testes para uma topologia de 4 microfones unidirecionais, aparelho Ambisonic, e uma outra topologia baseada em 3 (três) microfones dispostos em triângulo. A maior diferença entre estas duas abordagens trata-se com o facto de a primeira abordagem calcular a localização do evento sonóro baseado na energia dos sinais enquanto a segunda trata a localização com base em tempos de atraso, entre cada par de microfones.

Foram também realizados testes com um algoritmo de separação de eventos sonóros. Esta característica pode ser muito importante, numa fase mais avançada do projeto, para fazer classifcaçao de eventos. Esta última fase iria pressupor o recurso a algoritmos de Aprendizagem Automática ou *Machine Learning* .

Apesar de na primeira abordagem ser possível recriar um espaço 3D, através dos 4 (quatro) microfones, essa capacidade não foi explorada tendo-me focado apenas na deteção e localização no plano horizontal (angulo azimute). Esta generalização (de que todos os eventos ocorrem no mesmo plano horizontal) nao é problemática num ambiente real pois, duma forma geral, podemos assumir que todos os eventos ocorrem ao nível do chão, não sendo particularmente relevante em nenhum caso saber a elevação (grau vertical) da ocorrência.

Neste projeto também não será, para já, abordada a questão do cálculo da distância a que ocorrem os eventos. Essa secção do projeto será explorada em trabalho futuro como estará referenciada mais à frente neste relatório na própria secção de "Trabalho Futuro".

Este trabalho vai estar dividido em 4 (quatro) processos. Redução de ruído, detecção de eventos, localização de eventos e separação de eventos.

Para a fase de "Remoção de Ruído" foram estudados os algoritmos de Subtracção Espectral e Filtro de Wiener. A Subtracção Espectral é mais rápida mas menos eficiente do que o Filtro de Wiener.

Para a fase de "Detecção de Eventos" foi estudado o algoritmo de *Spectral Flux* .

Para a fase de "Localização" foram estudados dois algoritmos. Um algoritmo baseado em energia dos sinais, para a arquitetura Ambisonic, e outro baseado em tempos de atraso, *gcc-phat* (*Generalized Cross Correlation with Phase Transform*) , 3.1.3 , para a arquitetura em triângulo. De notar que, das arquiteturas estudadas neste projeto (Ambisonic e arquitetura em triângulo) , ambos os algoritmos apenas funcionam nas respetivas arquieturas às quais foram aplicados. Ou seja, não se poderia ter aplicado algoritmos baseados em tempos de atraso na arquitetura Ambisonic nem aplicar algoritmos baseados na energia do sinal à arquitetura em triângulo.

Para a fase de "Separação de Eventos" foi aplicado um algoritmo de separação baseado no algoritmo de *ICA* (*Independent Component Analysis*), 3.1.4 , neste caso o *fastICA* .

Todos estes algoritmos serão explicados e exemplificados mais à frente.

O projeto tem esta forma porque faz sentido reduzir o ruído para uma melhor detecção de eventos. Os algoritmos de localização apenas fazem sentido ser aplicados em trechos de áudio que de facto contenham algum evento

sonóro. A separação é a última a ser processada pois o seu processamento *apaga* informação relevante para os algoritmos de localização.

Capítulo 2

Trabalho Relacionado

2.1 Detecção de Eventos

No trabalho [3] foram efetuadas várias abordagens para verificar como se desempenhavam diferentes fórmulas matemáticas no processo de detecção de eventos. Este projeto relaciona-se com o meu projeto no sentido em que neste projeto de localização de eventos é importante detetar em que momentos do sinal analisado se encontram os eventos sonoros relevantes e que zonas do sinal contem apenas ruído. Este projeto testa 5 (cinco) fórmulas de deteção de eventos. Partindo do princípio que todos os eventos num dado sinal áudio são visíveis na variação de amplitudes ou frequências, podemos concluir que, após processamento, será sempre possível obter um gráfico que nos mostra claramente aonde estão os eventos e a sua duração. O projeto por Simon Dixon fala então em 5 fórmulas:

- Spectral Flux (Subtração de Espectros consecutivos)
- Phase Deviation (diferença de fases em amostras consecutivas)
- Complex Domain (esta fórmula verifica a variação tanto de amplitudes como fase)
- Weigth Phase Deviation (esta fórmula é semelhante a Phase Deviation mas com pesos mais elevados atribuídos às fases com maiores amplitudes)

- Rectified Complex Domain (esta fórmula aplica uma retificação de onda a formula de Complex Domain atribuindo o valor 0 (zero) para valores abaixo dum determinado threshold)

Estas fórmulas são posteriormente sujeitas a alisamentos e são lhe aplicados thresholds para identificar as amostras que correspondem a eventos sonoros. Os thresholds aplicados não são universalmente melhores ou piores. Será sempre preciso adaptar os thresholds e parâmetros da melhor forma possível para cada caso. Este projeto conclui que a melhor fórmula para detetar eventos com menor erro possível é, duma forma geral, a primeira fórmula Spectral Flux, ou seja, a subtração espectral de amostras consecutivas. Este algoritmo encontra-se explicado em 3.1 .

$$g_\alpha(n) = max(f(n), \alpha g_\alpha(n-1) + (1-\alpha)f(n)) \tag{2.1}$$

Após o processamento efetuado em 3.9 ainda existe muito ruído para poderem ser detetados eventos sonóros eficientemente. Assim a função calculada em 3.9 passa por uma equacação (2.1) de "alisamento" ou *smoothing* .

2.2 Estimação da Direção de Eventos Sonóros em Arquitetura Ambisonic

Este trabalho relacionado envolve o desenvolvimento de um algoritmo de localização de eventos. Foi estudado a informação do sinal em função do tempo e da frequência. Isso é possível através das *STFTs - Short Time Fourier Transform*. Estas *STFTs* efetuam várias transformadas de fourier ao longo do tempo. Desta forma obtemos uma matriz m x n em que m representa as janelas de tempo enquanto n representa as janelas de frequência. A análise no espectro da frequência é o que permite a este estudo detetar mais do que um evento ao mesmo tempo.

O aparelho ambisonic, demonstrado na figura 2.1 , captura o som proveniente de todos os ângulos em 4 canais unidirencionais, cada um apontado para na sua direção especifica definidos na lista seguinte.

- RF (*right forward*) representa o microfone *frente direita*

Figura 2.1: Aparelho Ambisonic

- LF (*left forward*) representa o microfone *frente esquerda*
- RB (*Right Back*) representa o microfone *trás direita*
- LB (*left back*) representa o microfone *trás esquerda*

Estes 4 canais são depois processados, em 2.2 , para se obter sinais de energia nos respetivos eixos de referenciais, x , y e z , enquanto w representa a soma de todas as energias , ou seja representa um sinal omnidirecional.

$$\begin{cases} x(t) = 0.5 * ((LF - LB) + (RF - RB)) \\ y(t) = 0.5 * ((LF - RB) - (RF - LB)) \\ z(t) = 0.5 * ((LF - LB) + (RF - RB)) \\ w(t) = 0.5 * ((LF + LB) + (RF + RB)) \end{cases} \tag{2.2}$$

$$\begin{cases} Ix = Real(W^* * X) \\ Iy = Real(W^* * Y) \end{cases} \tag{2.3}$$

Em 2.3 são calculadas as matrizes Ix e Iy , que contêm os vectores de energia nos respectivos eixos. Nessa mesma equação, W^* é o conjugado de W e X e Y são as matrizes dos sinais x e y após $STFT$s.

Por sua vez, estas matrizes permitem-nos calcular os ângulos para cada janela de frequência e para cada janela de tempo, obtendo uma matriz de

mXn de ângulos, em que m representa as janelas de frequênica e n representa as janelas de tempo. A equaçaõ 2.4 ilustra esse mesmo processo.

$$\begin{cases} \alpha = tan^{-1}(\frac{-Iy}{-Ix}) & \text{se Iy} \geq 0 \\ \alpha = tan^{-1}(\frac{-Iy}{-Ix}) - 180 & \text{se Iy} \leq 0 \end{cases} \tag{2.4}$$

Esta matriz de ângulos em função da frequência e tempo permite-nos encontrar múltiplos eventos em simultâneo.

Em 2.5 temos a equação que nos dará o ângulo resultante.

$$N = \sum_{n=1,f=1}^{n=N,f=F} p(\alpha(t,f)|\theta) \tag{2.5}$$

2.3 Remoção de Ruído com Subtracção Espectral

Um outro trabalho realizado anteriormente que foi estudado, é referido em [1] que se centra no desenvolvimento do algoritmo de Subtração Espectral, sendo que se acentua mais especificamente sobre o uso deste mesmo filtro em sinais de fala. Apesar desse facto, o algoritmo pode-se aplicar a outros tipos de sinais de áudio. Este filtro acenta na subtração de magnitudes entre amostra de sinal e amostra de ruído.

O documento referido oferece um diagrama que simplifica bastante a perceção dos processos a seguir para construir este filtro de ruido (2.2).

Como se pode ver este diagrama mostra que usando uma amostra de ruído e subtraindo a sua magnitude á magnitude do sinal este perde o ruido existente, sabendo sempre que o ruido nunca será subtraído na sua totalidade, mas que este algoritmo pode oferecer um filtro relativamente eficiente para o "peso" de computação. Este digrama parte do princípio que o utilizador poderá indicar ao computador o que é ruido. Ou seja, o filtro não é adaptativo e precisa que lhe seja indicado "manualmente" o que são amostras de ruído, à partida. Assim o algoritmo só tem de subtrair a magnitude a todas as amostras do sinal. Este mecanismo está descrito mais á frente na secção 3.2 . Aí serão explicadas as alterações a este algoritmo para que se adaptasse a amostra de ruído.

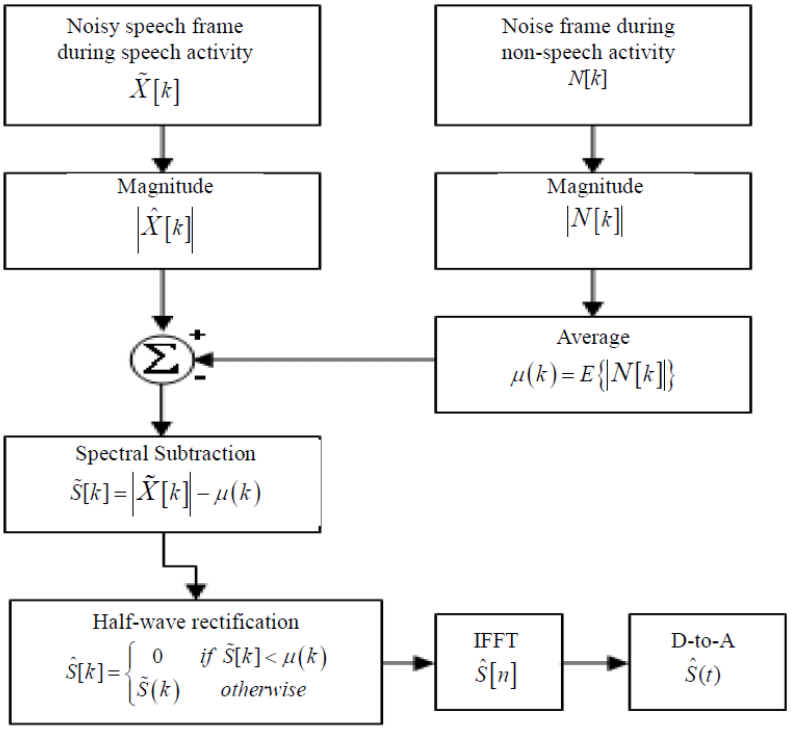


Figura 2.2: Diagrama de Subtração Espectral

Capítulo 3

Modelo Proposto

3.1 Fundamentos

Neste projeto foram estudadas duas abordagens à remoção de ruído. Uma com o algoritmo de Subtração Espectral e outro com o algoritmo de Filtro de Wiener. Contudo haveria uma falha nestes filtros se fossem aplicados, tal e qual a de definição original. Nenhum destes filtros é adaptativo por definição. Não ser um filtro adaptativo significa que a amostra de ruído que é usada na redução do ruído ao longo de todo o sinal, não é atualizada com o decorrer de qualquer iteração durante o funcionamento do algoritmo. Então foi preciso alterá-los de forma que a remoção de ruído se adaptasse às variações de ruído de fundo ao longo de hora(s) ou dia(s), em ambiente urbano. Mais à frente irei explicar o conceito de ambos os filtros usados.

Só numa fase mais avançada acabei por desenvolver este algoritmo de ruído adaptativo. Na secção 3.2 refiro este mesmo desenvolvimento de algoritmos de redução de ruído adaptativos.

Quanto à detecção de eventos optei por usar o algoritmo *Spectral Flux* que revelou ter a eficiência necessária para o desenvolvimento do projeto. Após a detecção dos "trechos" de áudio que continham eventos sonóros, estes "blocos" de amostras consecutivas eram passados ao algoritmo de localização onde eram identificado o ou os ângulos dos eventos existentes. Estes "blocos" tinham tamanho, em amostras, variável consoante as dimensões, do(s) evento(s), retornada pelo algoritmo. Havia um valor mínimo, de dimensão destes "blocos" de amostras, abaixo do qual os eventos eram classificados como sendo apenas ruído. Isto foi um passo necessário, pois acontecia com

alguma regularidade que o algoritmo retornasse um ou vários "pequenos eventos" que correspondiam a curtos momentos de maior ruído.

Em relação à localização de eventos para a arquitetura Ambisonic usei, como referido anteriormente, um algoritmo baseado nas diferentes energias dos sinais. Esse algoritmo está descrito nesta secção. Desenvolvi, também, uma fórmula diferente, da sugerida em 2 e referenciada em 2.4 , para calcular os múltiplos eventos que estará descrita na secção 3.2 .

Quanto à separação de eventos o algoritmo a ser usado foi o *fastICA* , explicado em 3.1.4 .

Os ficheiros de áudio usados para teste tinham frequência de amostragem de 48000 amostras por segundo.

3.1.1 Filtros de Ruído

Subtracção Espectral

Todos os filtros de ruído assumem que o sinal a ser processado consiste na soma de dois sinais diferentes. O ruído de fundo e o som do evento sonóro em si. Assim podemos definir uma fórmula de construção do sinal que nos irá ajudar a entender como reduzir o ruído existente no sinal.

$$x[n] = s[n] + r[n] \tag{3.1}$$

Em que x representa o sinal de áudio, r representa o ruído de fundo existente nos sinais de áudio captados e s representa o evento sonóro presente no trecho de áudio. Assim que conseguirmos definir a componente r podemos subtrair esta mesma ao sinal obtendo assim, apenas, a componente s . Subtração Espectral, por definição, é um algoritmo criado para ser aplicado em modo offline. Isto, porque o algoritmo necessita de saber que parte do sinal é apenas ruído. Com essa informação, irá então subtrair a magnitude da amostra, na frequência, de ruído a todas as frames do sinal. Desta forma foi necessario desenvolver este algoritmo de forma a que se adapte a cada momento de ruído no decorrer de todo o seu processamento. Na figura 2.2 é perceptível como funciona o algoritmo de subtração espectral. Como o esquema demonstra, é subtraído, ao sinal, uma amostra média de ruído. A amostra de ruído é "indicada" pelo utilizador. Após a subtração da amostra

de ruído é feita uma transformada de *Fourier* Inversa *IFFT* e recuparado o sinal de áudio sem, ou com menos, ruído.

Filtro de Wiener

Este filtro é, como foi dito anteiormente, mais eficiente do que o algoritmo de Subtração Espectral mas é também mais "pesado" em termos computacionais. Dentro dos filtros de Wiener decidi usar a abordagem TSNR (Two Step Noise Reduction). Este conceito consiste em calcular o nível de SNR *a posteriori* (Sginal to Noise Ratio) através da energia da frame atual e da amostra de ruído.

Este algoritmo assume que a primeira amostra analisada contém apenas ruído e para o resto do seu processamento usa essa mesma amostra como a amostra de ruído a remover ao resto do sinal.

Após termos a nossa amostra de ruído, calcula-se o *SNR a posteriori* pela formula 3.2 .

$$SNR_{posteriori} = (e/noise) - 1; \tag{3.2}$$

Onde e representa a energia do sinal e *noise*, como o nome indica, representa o ruído do sinal. Após este processo recorre-se a equaçao 3.3 para assegurar que o valor do *SNR* nunca é de facto igual a zero.

$$SNR_{posteriori} = max(SNR_{posteriori}, 0.1) \tag{3.3}$$

Em que 0.1 é simplesmente um valor baixo próximo de zero mas não igual a zero. Isto serve para prevenir divisões por zero. De seguida calcula-se o *SNR a priori* através da fórmula em 3.4 .

$$SNR_{priori} = \alpha * (e_{old}/noise) + (1 - \alpha) * SNR_{posteriori} \tag{3.4}$$

Em que e_{old} representa a energia calculada na iteração anterior.

$$\begin{cases} mag_x = |X| \\ mag_{new} = (\eta/(SNR_{priori} + 1)) * mag_x; \end{cases} \tag{3.5}$$

Em 3.5 calcula-se a nova magnitude do sinal em que mag_x representa o módulo da *FFT* do sinal analisado na iteração currente.

Em 3.6 calcula-se a atenuação do ruído sobre a energia do sinal, em que e_{new} define a energia calculada para a iteração currente.

$$tsnr = e_{new}/r; \tag{3.6}$$

Em que r representa o ruído de fundo. Os valores provenientes de 3.6 passam por uma função de ganho G que será posteriormente usado em 3.7

$$\begin{cases} mag_{new} = Gtsnr * mag_x \\ fft_x = mag_{new} * \exp i * phase_x \end{cases} \tag{3.7}$$

Em que $phase_x$ representa a fase retirada da FFT ao sinal a ser analisado nesta interação.

O resultado obtido em 3.7 é somando a um array, o qual no final representará o sinal, na sua totalidade, com ruído reduzido.

3.1.2 Detecção de Eventos

Neste projeto foi usado, para detecção de eventos, o algoritmo de *Spectral-Flux* mencionado em 2.1 . Assim, depois de calculadas a $STFT$ (*Short-Time Fourier Transform*) é aplicada a subtração de uma frame com a frame seguinte, isto para todas as frequências. É feita a soma de todos estes valores. Após este passo obtemos um gráfico com picos de amplitude correspondentes aos eventos sonóros, como mostra a figura **??**. Este gráfico é resultante da análise a um ficheiro de áudio que continha eventos sonóros de cães a ladrar.

Passo a explicar o algoritmo de *Spectral Flux* .

O algoritmo *Spectral Flux* consiste em medir a mudança de amplitudes em cada janela de frequência como mostra a equação 3.8 .

$$SF = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \tag{3.8}$$

Em que H indica o processo demonstrado na equação 3.9 .

$$H(x) = \frac{x + |x|}{2} \tag{3.9}$$

A figura 3.1 não apresenta, exatamente definidos, todos os eventos apresentando muitas irregularidades. Seria difícil aplicar qualquer tipo de *threshols* a este gráfico. Então esta lista de valores é "alisada" pela função

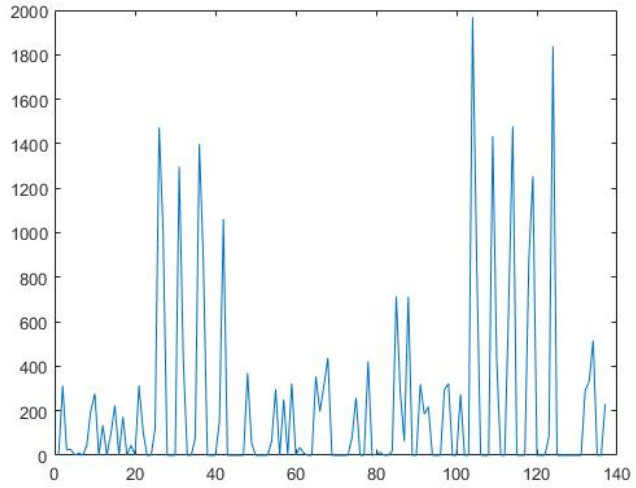
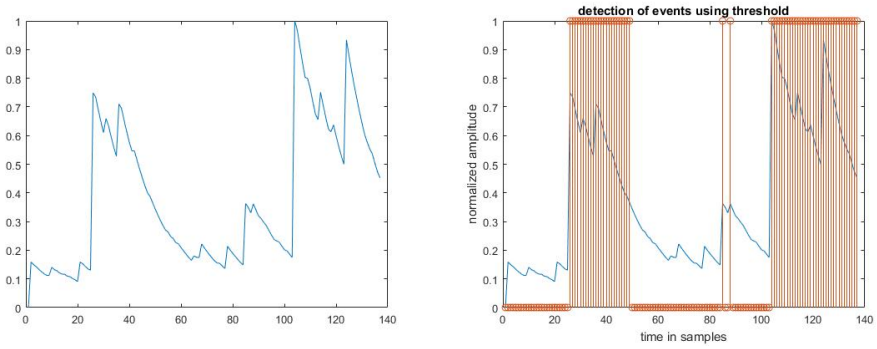


Figura 3.1: Detecção de Eventos - *Spectral Flux*

expressa em 2.1 , que soma parte da amostra atual com a amostra anterior, sendo que a amostra anterior tem sempre um *peso* muito elevado. O valor de α nessa mesma expressão deve ter valores entre 0 e 1, não inclusive, e propõe-se que tenha valores perto de 1 para um melhor desempenho como filtro passa-baixo. Esta função permite-me obter um gráfico com variações mais suaves e definir mais claramente cada evento e a sua duração. Sabendo em que zona do sinal analisado se encontram os eventos sonóros que queremos localizar, passa a não ser preciso analisar todo o sinal. Assim elimina-se, ou tenta-se eliminar o máximo possível, os momentos de apenas ruído, que geraria resultados aleatórios no processo de localizaçao.

Desta forma o conjunto de *frames* a ser analisada, em cada momento, adapta-se à dimensão do evento, sendo que existe um limite mínimo para a dimensão desta janela de análise. Desta forma se o evento detetado tiver duração (em número de amostras) abaixo dum determinado *threshold* este será desprezado, pois será interpretado como ruído de fundo. Assim os eventos que têm dimensão inferior a 30 milissegundos são eliminados. À partida, não haverá nenhum evento de dimensão no tempo inferior a esse *threshold*. Na figura 3.2 temos, à esquerda, a subtração de espectros consecutivos após um filtro passa-baixo, à direita, o gráfico que mostra, com linhas verticais laranja, os momentos dos eventos sonóros, sendo que no restante sinal de



(a) Passo 2 (b) Passo 3

Figura 3.2: Detecção de Eventos

áudio é assumido que só existe ruído de fundo.

3.1.3 Localização de Eventos Sonóros

Arquitetura Ambisonic (baseado em Energia do Sinal)

O excerto de sinal que chega a esta parte do algoritmo é apenas um determinado "bloco" de amostras consecutivas. Cada "bloco" de amostras identificado no passo anterior como sendo um evento, será analizado numa iteração de este algoritmo de localização. Apesar da a arquitetura Ambisonic permitir a localização do ângulo num espaço 3D, neste projeto apenas será usada a localização no plano horizontal, assumindo desta forma que geralmente todos os eventos sonóros ocorrem ao mesmo nível vertical.

Assim, neste projeto a formula de detecção do ângulo do evento propõe-se a analisar o sinal em função do tempo e da frequência. Podemos assumir que em cada janela de tempo e de frequência apenas um, dos eventos em simultâneo, sobressai. Desta forma podemos identificar todos os múltiplos eventos em simultâneo analisando todas as janelas de tempo e frequência, pois todos os eventos sobressaírão numa determinada janela.

Neste projeto também nao será aboradada a questão do cálculo da distância a que o evento ocorreu. Para a arquitetura Ambisonic seria preciso ter pelo menos dois destes aparelhos a funcionar em simultâneo para ser possível calcular a distância à qual ocorreu um determinado evento, pois só assim poderia ser feita uma "triângulação" . Assim, usando um *array* de microfones ao longo dum local (conforme é pretendido para este projeto no futuro)

poder-se-á contornar este problema.

Como foi dito, esta parte do algoritmo analisa o sinal em função do tempo e da frequência pelo que é preciso originar uma STFT para cada sinal audio. Serão analisados os sinais w, x e y. O sinal z, pelas razões mencionadas acima, não será relevante para o algoritmo. Após obter a $STFT$ para cada um dos sinais é aplicada uma formula que retorna uma matriz de ângulos. Esta matriz tem dimensão igual à matriz que retorna das $STFT$s. A fórmula aplicada é exemplificada em **??**

$$\begin{cases} Ix = real(W^*. * X) \\ Iy = real(W^*. * Y) \end{cases} \tag{3.10}$$

$$\theta = deg(\alpha(Iy + Ix * i)) \tag{3.11}$$

Em 3.11 , α representa o ângulo calculado, através do respetivo número imaginário, em radianos. deg representa a conversão de radianos para graus. Ix é a matriz de intensidades para o eixo dos XX . Em 3.10 , $W*$ é o conjugado da $STFT$ de W e X é corresponde à $STFT$ de x . A segunda fórmula é a respetiva fórmula para o eixo dos YY.

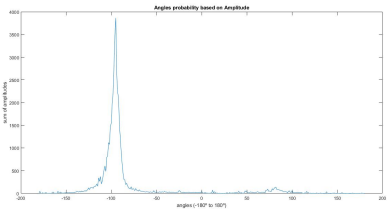
Com esta matriz de ângulos é possível então identificar o ângulo do evento a ser analisado. Haviam duas abordagens iniciais para estes cálculos. Calcular a probabilidade de um ângulo através dum histograma com os 360 graus ou calcular a probabilidade do mesmo baseado na soma das amplitudes desses ângulos.

$$\begin{cases} h(i) = h(i) + 1 \\ SAmp(i) = SAmp(i) + W(i) \end{cases} \tag{3.12}$$

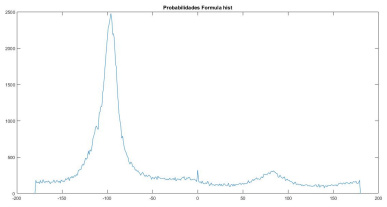
Em que h representa um histograma e $SAmp$ representa um fórmula, desenvolvida neste projeto, que faz a soma de amplitudes de W ($STFT$ do sinal w calculado em 2.2)

Como se pode verificar, o histograma baseado na soma de amplitudes tem, consideravelmente, menos ruído do que o histograma "normal" . Desta forma, neste projeto passou a ser usado a primeira opção em vez do histograma "normal" sugerido em 2.2 e noutros artigos relacionados semelhantes a este.

Ocasionalmente surgiu "picos" no gráfico que identificavam ângulos falsos. Ou seja, introduziam falsos positivos na deteção de ângulos. Isto poderia



(a) Soma das Amplitudes



(b) Histograma

Figura 3.3: Comparação de gráficos de ângulos para 'histograma' de amplitudes e histograma

acontecer por introdução duma componente de ruído de elevada amplitude na *frame*.

O gráfico resultante de todas estas fórmulas contem bastante ruído e, por isso, é passado por um método que reduzalgum ruído existente no gráfico.

Os picos detetados têm, assim, diferentes características que lhes permitem ser ignorados ou realçados. Se um pico for muito elevado, por comparação com os restantes, esse passa a ter mais relevo na identificação do ângulo de onde vem o evento. Nunca poderemos saber ao certo quantos eventos estão a ocorrer num determinado local a um determinado momento, por isso, não podemos descartar nenhum evento que se situe acima do *threshold*. Este *threshold* é definido pelo utilizador de acordo com o que for mais indicado para cada ambiente. Neste caso o *threshold* teve um determinado valor para os ambientes especificos de cidade em que os ficheiros foram gravados. Este mesmo poderia ter de ser ajustado para qualquer outro tipo de ambiente.

O desafio, nesta fase do projeto, passa por identificar quais os "picos" no gráfico que representam, de facto, eventos sonóros e aqueles que representam apenas ruído que por alguma razão sobressaiu naquele exacto momento. Aqui definir o *threshold* trará consequências na localização dos eventos. Um *threshold* demasiado elevado pode acabar por neutralizar eventos menos sonantes, mas importantes ainda assim, enquanto um *threshold* demasiado baixo poderá criar eventos falsos.

Após a detecção dos "picos" relevantes para os nossos casos de teste, os resultados eram apresentados num gráfico como ilustrado na figura 3.4 . Nesta figura o centro representa o ponto de coordenadas (0,0) num plano horizontal. Aqui existem dois traços que representam as direções de dois

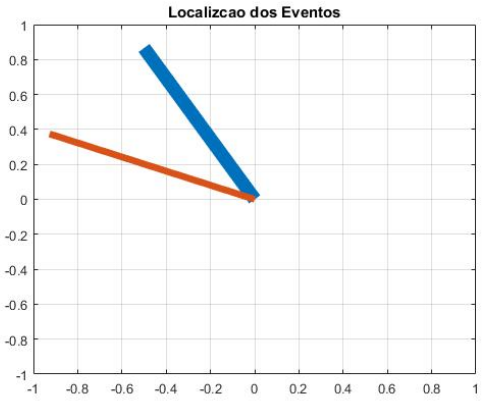


Figura 3.4: plano horizontal com dois eventos simultâneos

eventos sonóros detetados em simultâneo. As diferentes espessuras represetam um evento mais ou menos sonante, em que quanto mais espesso o traço mais sonante terá sido o evento.

Arquitetura em Triângulo (Baseado em Atrasos do Sinal)

Para esta arquitetura, o algoirtmo de localização desenvolvido foi o *GCC-PHAT* . Este algoritmo baseia-se na correlação entre dois sinais no tempo e na frequência. Essa correlação será usada numa fórmula onde serão aplicados atrasos/adiantamentos de tempo que definirão um gráfico com um ou mais picos que identificam tempos de atraso presentes no respetivo par de sinais. Estes tempos de atraso são medidos em amostras. Com essas amostras e com a frequência de amostragem é possível identificar o ângulo no qual incidiu a onda sonóra.

O primeiro passo é efetuar uma STFT (*Short Time Fourier Transform*) a cada um dos sinais. De seguida cálcula-se a correlação entre os dois sinais através da expressão em 3.13 .

$$R_{xx} = X(t,f) * Y(t,f)^H \tag{3.13}$$

E por último adiciona-se a fórmula que nos permite desenhar a correlação entre sinais ao longo dos diferentes tempos de atraso.

$$\phi^{gcc}(t,f,\tau) = \frac{R_{xx}}{|R_{xx}|} e^{-2\pi i f \tau} \tag{3.14}$$

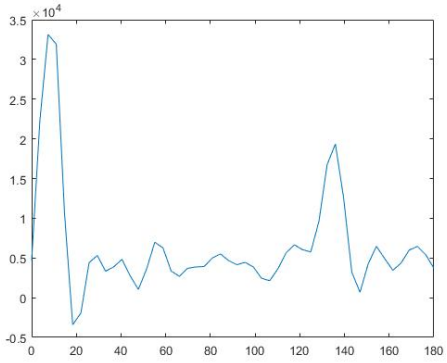


Figura 3.5: Correlaçaõ em função dos Tempos de Atraso

As expressões exemplificadas em 3.13 e 3.14 ocorrem uma vez por cada par de microfones. Sendo que nesta arquitetura existem 3 microfones, estas expressões ocorrem 3 vezes. Correlacionando desta forma os microfones 1 e 2, os microfones 1 e 3 , e os microfones 2 e 3.

Após este processamento obtemos uma matriz de dimensões TEMPOxFREQUENCIAxATRASOS que foi usada somando todos os resultados para as duas primeiras dimensões obtendo um vector final de dimensão 1xATRASOS. Se fizer o gráfico deste vector será possível identificar os tempos de atrasos com maior correlação entre cada par de sinais, como mostrado na figura 3.5.

Sendo que a topologia é de 3 microfones separados em forma de triângulo, cada par de microfones estará definido para o seu próprio referencial. Desta forma tive de ajustar os referenciais de cada par de microfones todos para a mesma disposição de referencial (figura 3.6). Assim os referenciais passaram a estar alinhados com o referencial indicado com a letra “A”.

Após ajustar os 3 referenciais posso sobrepor os 3, respetivos, gráficos de pares de microfones e visualizar onde está a maior incidência de “picos”.

Temos de ter em atenção os “picos” que representam ângulos falsos. Isto acontece porque cada par de microfones tem a capacidade de identificar o ângulo a que incidiu o sinal, mas não permite saber se veio da frente ou de trás dos microfones. A figura 3.7 ilustra essa particularidade desta topologia.

Esta ambiguidade é desfeita ao verificarmos os 3 pares de microfones, pois tempos de atraso falsos, ou eventos falsos, estarão presentes apenas num dos gráficos. Os eventos sonóros, que sejam de facto reais, estarão ilustrados

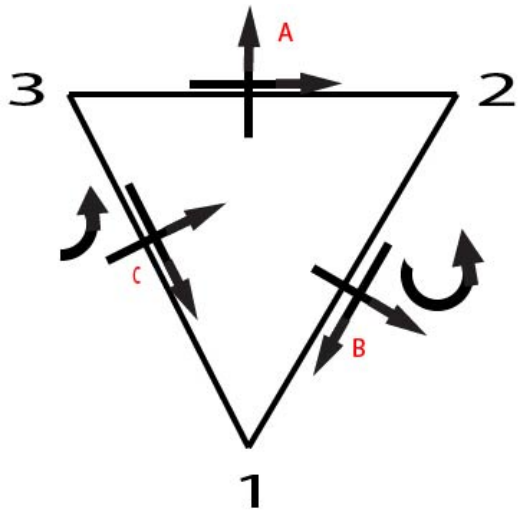


Figura 3.6: Ajuste de referenciais

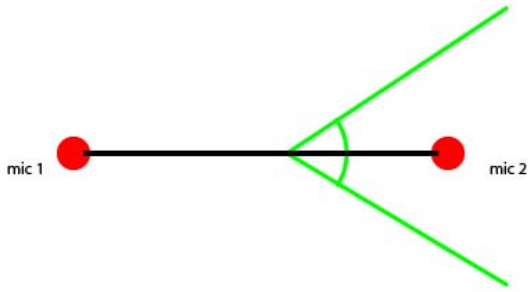


Figura 3.7: Ambiguidade de Ângulos em Pares de Microfones

como "picos" dos 3 gráficos.

3.1.4 Separação de Eventos por ICA

Neste projeto foi abordada a temática da separação de eventos sonóros. Este tipo de algoritmos é bastante relevante para este projeto pois será importante, em trabalho futuro, proceder à classificação automática de eventos. Desta forma é possível identificar programaticamente que tipo de evento sonóro ocorrera naquele preciso momento. Podem ser eventos variados como um acidente de viação, um disparo duma arma, cães a ladrar, sirenes, etc. Para uma classifcação mais exacta é necessária a separação dos eventos sonóros, pois eventos sobrepostos, mesmo que apenas sobreposto por ruído, são bastante mais dificeis de classificar.

Neste projeto, apenas foi usado o algoritmo *fastICA* que é uma versão de execução mais rápida em relação ao conceito básico do algoritmo de *ICA*.

Haveria outros algoritmos que poderiam ter sido estudados como por exemplo o algoritmo *DUET* . Esse e outros poderão ser estudados em trabalho futuro tal como referenciado na secção 5 .

Para termos noção de como funcionam os processos de mistura e separação estes processos estão exemplificados nas expressões 3.15 . A matriz A é uma matriz de mistura que simula a mistura que ocorre em ambiente real aquando da ocorrência de eventos sonóros pelos diferentes microfones em simultâneo. A matriz W representa a matriz de "pesos" que serão aplicados aos diferentes sinais (3 no caso da arquitetura em triângulo e 4 na arquitetura em Ambisonic) para que ocorra a separação de cada evento sonóro ocorrido.

Importante salientar que o número de eventos simultâneos separados nunca será superior ao número de sinais existentes. Desta forma, para a arquitetura Ambisonic seriam apenas possiveis de separar até 4 eventos em simultâneo enquanto que na arquitetura de triângulo seriam 3.

$$\begin{cases} y = Ax \\ x = W^T y \end{cases} \tag{3.15}$$

Este algoritmo, *fastICA* , consiste em 2 passos. Primeiro há uma centralização de dados seguido dum processo de *whitening*.

$$x_{i,j} = x_{i,j} - \frac{1}{M} \sum_{j'} x_{i,j'} \tag{3.16}$$

Com esta "centralização de dados" , referenciada pela expressão 3.16 , pretende-se que a média total de cada linha da matriz X tenha o valor esperado de 0.

Ainda como parte do primeiro passo, procede-se a fase de *whitening*. Esta fase pretende que as amostras de X sejam não correlacionadas entre si e tenham variância igual a 1, através da expressão em 3.17 .

$$X = ED^{-\frac{1}{2}}E^T X \tag{3.17}$$

Onde E define a matriz de *eigenvectors* e D define a matriz diagonal

O segundo passo consiste em separar de facto os evento sonóros. Assim é repetido um determinado processo o número de vezes que for definido até

termos a quantidade de eventos desejada. O processo de separação ocorre após processamento duma equação que vai obtendo valores para *pesos* que serão, no seguimento, aplicados ao sinal original. Enquanto esse conjunto de *pesos* continuar a retornar diferentes entre cada iteração seguida o processamento continua. A equação 3.18 demonstra como se obtém a matriz de *pesos*, *W*.

$$\begin{cases} W_p = \frac{1}{M} X_g (W_p^T X)^T - \frac{1}{M} g^{'}(W_p^T X) 1 W_p \\ W_p = W_p - \sum_{j=1}^{p-1} W_p^T w_j w_j \\ W_p = \frac{W_p}{|W_p|} \end{cases} \tag{3.18}$$

Como disse, este processo é repetido o número de vezes que for preciso para separar todos os eventos sonóros pretendidos, tendo em conta que se uma matriz tem n misturas então o *ICA* ou *fastICA* só poderão separar um máximo de n eventos sonóros.

Após o processo ilustrado em 3.18 falta apenas aplicar os ditos *pesos*, textitw, ao sinal original como demonstrado em 3.19.

$$\begin{cases} W = [w_1, w_2, ..., w_n] \\ S = W^{[}T]X \end{cases} \tag{3.19}$$

3.2 Abordagem

Este projeto, que na verdade se dividiu em dois, ao longo de dois anos, centrou-se sobretudo na lozalicação onde passei de facto a maior parte do meu tempo de trabalho.

Aqui apliquei diferentes fórmulas. Na topologia Ambisonic segui as fórmulas indicadas inicialmente pelo documento.

$$\begin{cases} Ix = real(W^{*}. * X) \\ Iy = real(W^{*}. * Y) \end{cases} \tag{3.20}$$

Em que W representa a $STFT$ do sinal w calculado em 2.2 . O documento referia também as seguintes fórmulas, para cálculo de ângulo.

$$\alpha(t,f) = \begin{cases} tan^{-1}(\frac{-I_y(t,f)}{-I_x(t,f)}) & I_y(t,f) > 0 \\ tan^{-1}(\frac{-I_y(t,f)}{-I_x(t,f)}) - 180 & I_y(t,f) <= 0 \end{cases} \tag{3.21}$$

Nesta fase alterei a equação 3.21 para a equação em 3.22.

$$azimuth = \theta(I_x i * I_y) \tag{3.22}$$

Em que θ identifica a função de cálculo de ângulo através de um número imaginário.

Esta formula retorna uma matriz de tamanho dimFrequência$x$ dimTempo. Esta matriz é uma matriz de ângulos calculados para todas essas mesmas janelas de frequência e tempo. Assim, após o uso desta fórmula calculei os histogramas. O histograma era a fórmula sugerida pelo documento referenciado em 2.2 .

Aqui, eu alterei ligeiramente a fórmula e em vez de somar o número de vezes de ocorrência de um ângulo, somei as amplitudes respetivas da matriz de energia total W em que W é a formula da $STFT$ aplicada ao sinal w, 2.2 . Desta forma fui somando, a cada iteracão, a amplitude 3.12 respetiva a cada ângulo em 3.22.

Em relação ao filtro de ruído de Wiener também fiz algumas alterações. É importante que a amostra de ruído vá sendo alteranda e adaptando conforme as "flutuações" do ruído de fundo presente numa cidade, ao longo de um dia inteiro ou mesmo de semanas ou meses.

Assim, para este projecto foi desenvolvida uma forma de atualizar a amostra de ruído, ao longo do tempo, de acordo com o ruído verificado em cada preciso momento. É feita uma média do módulo de todas as FFT calculadas para o trecho de áudio da iteraçao currente. Após fazer isto para a primeira amostra comparo este ruído com o sinal proveniente das iterações seguintes. Se essa média estiver entre um determinado *threshold* em relação à amostra de ruído então assiná-lo a amostra currente como se tratando de ruído. Aí adapto "suavemente" o o ruído a esta nova amostra de ruído através de um determinado *peso* como demonstrado em 3.23 .

$$n = \alpha * n + (1 - \alpha) * m_{|X|}; \tag{3.23}$$

Em que n representa a componente de ruído e m representa a média. O patametro α representa um *peso* entre 0 e 1 , não inclusive, e será ajustado pelo utiliador conforme desejado para um melhor desempenho.

3.3 Implementação do Modelo

Este projeto foi, todo ele, realizado em *Matlab*.

Em termos tecnológicos usei também, como ferramenta de edição dos áudios, o software Adobe Audition CC. Este permitiu-me todo o tipo de edição e montagem de ficheiros áudio bem como a gravação dos ficheiros de teste.

Neste projeto apliquei algoritmos de filtro de ruído, deteção de eventos, localização de sons em topologia Ambisonic e topologia em triângulo e ainda separação de eventos.

Aqui comparei o desempenho, em situações semelhantes, das duas topologias de microfones comparando a abordagem em função das energias dos sinais com a abordagem em função dos tempos de atraso.

Capítulo 4

Validação e Testes

Para este projeto foram necessários ficheiros áudio de teste gravados nas topologias ou arquiteturas usadas neste projeto. Infelizmente não haviam ficheiros de teste disponíveis na *web* que podessem ser usados aqui. Por isso foram gravados ficheiros de áudio de propósito para este projeto. Estas gravações foram feitas em câmara anecóica e em ambiente real, em locais fechados bem como em espaços abertos.

Os ficheiros usados incluem eventos sonóros de "cães a ladrar" , "buzina" , "travagem de carro" , "disparo", "musica" e "fala", com e sem ruído. O ruído, quando presente nos ficheiros, foi acrescentado programaticamente e também obtido por gravações em ambiente real. Estes ficheiros estavam todos amostrados a 48000 amostras/segundo. OS ficheiros de áudio foram gravados em 4 e 3 canais simultâneamente para as arquiteturas de Ambisonic e em Triângulo, respetivamente.

Para os filtros de ruído efetuei testes com ficheiros com ruído artificail e ruído real em ambiente de cidade.

Para o algoritmo de deteção de eventos, efetuei testes com todos os tipos de ficheiros áudio. OS melhores resultados foram, como seria de esperar, obtidos nos eventos que implicavam uma maior e mais abrupta mudança de amplitude. Assim os eventos de "cães a ladrar" e "disparo de arma" foram os que obteram eventos mais definidos. Ainda assim este algoritmo verificou bons resultados com todo o tipo de eventos sonóros testados.

Para a fase de localização de eventos, efetuei testes para as duas topologias (arquiteturas).

Para a topologia Ambisonic efetuei testes com sons provenientes de angu-

los 0, 90, 180 e -90 graus para todos os tipos de sons e em ambiente controlado (camara anecoica) ou ambiente de cidade.

Para a segunda topologia em triângulo testei o algoritmo com ficheiros de sons emitidos a 0, 40 e 110 graus. Estes testes foram efetuados com misturas virtuais, misturas reais em ambiente aberto e misturas reais em ambiente fechado (com maior reverbeção). Isto dá-nos uma percepção de como o algoritmo funciona conforme se vão alterando as condições de trabalho, desde ambientes mais controlados até ambientes mais realistas. Os resultados foram melhores para ambientes de misturas virtuais e piores para misturas reais em ambiente fechado.

Aqui os resultados foram claramente melhores para a arquitetura Ambisonic, revelando gráficos de ângulos muito mais definidos. Nos gráficos obtidos com a arquitetura e algoritmo Ambisonic os "picos" dos gráficos de localização são, duma forma geral, mais pronunciados e existe menos ruído. Ainda assim, em ambiente real aberto existe, nos dois casos, algum ruído. Enquanto no algoritmo para Ambisonic existe muito ruído, e por isso mesmo mais "picos" falsos, para a arquitetura em Triângulo e algoritmo *GCC-PHAT* existem "picos" mais pronunciados mas com maior margem de erro no ângulo que deveria ter sido identificado. Assim, mesmo sendo fácil identificar os "picos" na arquitetura em triângulo, essa identificação dos ângulos poderá estar mais longe do valor real do que no caso do Ambisonic.

Para a fase de separação foram testados os ficheiros de testes da ambas as arquiteturas. Ainda assim os resultados não divergiram muitos. Havia, à partida, a limitação da arquitetura em triângulo só poder separar 3 eventos sonóros no máximo, por o número de microfones ser também 3. Ainda assim a separação propriamente dita foi semelhante para ambos os cenários.

Os testes e resultados estão relatados nas secções seguintes.

4.1 Filtros de Ruído

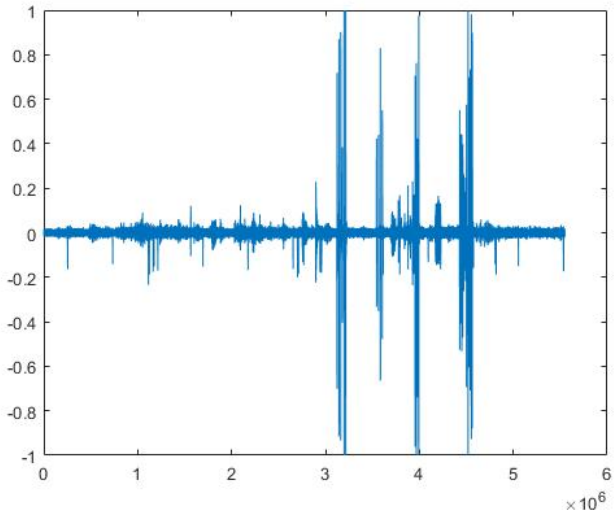
Neste projeto foi abordado a temática dos algoritmos de redução de ruído, neste caso, aplicados a ficheiros de áudio. Foram estudados os filtros de Subtração Espectral assim como o filtro de Wiener.

Na figura 4.1 são mostrados 3 sinais. O primeiro representa um sinal de áudio original sem qualquer filtro de ruído. Os 2 gráficos seguintes mostram

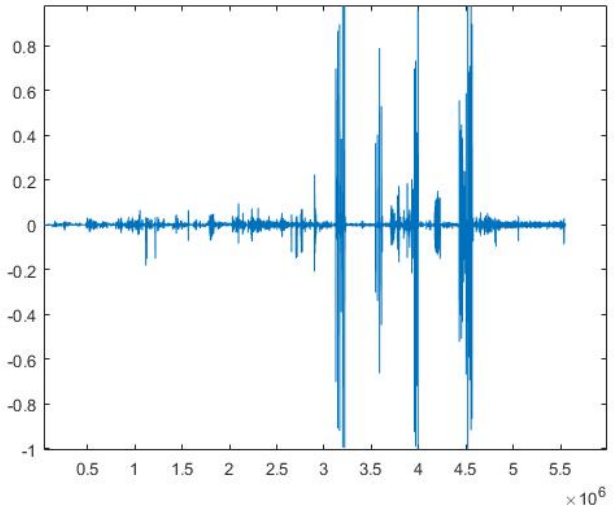
o mesmo sinal de áudio após filtro de Subtração Espectral e Filtro de Wiener respectivamente.

É importante salientar que há uma "linha ténue" que separa um bom filtro que retira apenas ruído dum filtro de ruído que remove ruído e também informação relevante do sinal para os algoritmos de localização. Por essa mesma razão, na maior parte dos testes realizados neste projeto foram feitos sem recurso a algoritmos de redução de ruído.

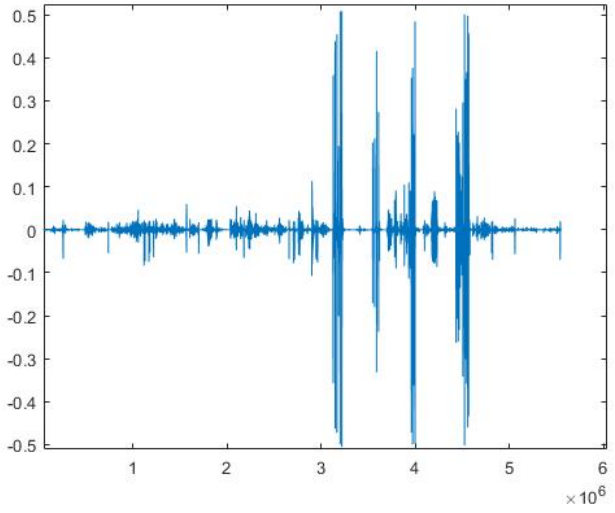
Os algoritmos de redução de ruído revelaram-se, ainda assim, sempre, ou quase sempre, muito úteis para o bom funcionamento do algoritmo de Deteção de Eventos, *Spectral Flux* .



(a) Sinal de Áudio



(b) Sinal após filtro de Subtração Espectral



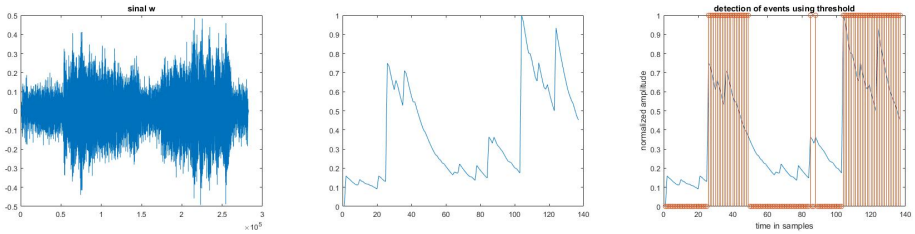
(c) Sinal após filtro de Wiener

Figura 4.1: Sinal de áudio para os diferentes filtros de ruído

4.2 Detecção de Eventos

Como descrito em secções anteriores, a deteção de eventos faz-se através do algoritmo de Spectral Flux, explicado em 2.1 e mais concretamente pela expressão em 3.8 .

Os gráficos na figura 4.2 mostra as diferentes fases do processo que ocorre para que haja a detecção de eventos. Primeiro temos um sinal de áudio com dois momentos de maior amplitude. De seguida temos a expressão *Spectral Flux* , aplicada tal como indicada na expressão **??** . Por último temos a normalização desta mesma expressão anterior à qual foi aplicada um *threshold* acima do qual o sinal é identificado como evento sonóro, e abaixo do qual é identificado com apenas ruído. Aqui o *threshold* é definido pelo utilizador da melhor forma que se adeque ao ambiente no qual os microfones se encontram.



(a) Sinal de Áudio (b) HSF após *smooth* (c) Deteção de Eventos

Figura 4.2: 3 Fases do processo para detetar eventos

Tendo este gráfico, defini um *threshold* que pode ser ajustado conforme o tipo de sons a ser analisado para um melhor desempenho. Aplicando esse *threshold* obtive o gráfico da figura 4.2c que mostra os trechos do sinal a serem analisados demarcados a laranja.

Os eventos, assinalados com as linhas vericais laranja, que sejam demasiado curtos são considerados resultados de ruído existente no sinal e são por isso descartados. Assim neste exemplo apenas dois eventos serão analisados, correspondendo aos trechos de maior largura.

4.3 Localização de Eventos

4.3.1 Arquitetura em Triângulo

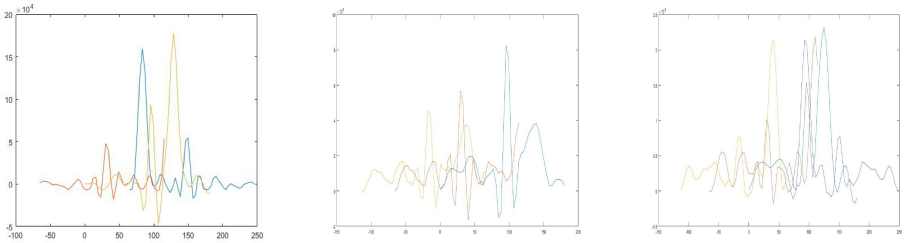
Como referido antes os testes foram realizados para angulos de 0, 40, e 110 graus.

Como explicado anteriormente, a correlação e consequente cálculo dos tempos de atraso é feita com cada par dos 3 microfones. Assim teremos 3 pares e 3 gráficos com que representaram os angulos cálculados. Nestes gráficos os picos (máximos, ou máximos relativos caso haja mais do que um evento sonóro) identificarão os angulos obtidos após os cáclulos.

Os ficheiros de teste foram originados em 3 ambientes diferentes. Primeiro efetuaram-se misturas virtuais, ou seja, ficheiros misturados programaticamente. Depois foram gravados ficheiros "reais" em ambiente fechado, dentro duma sala, e por ultimo foram gravados ficheiros com misturas reais em ambiente aberto. Estes diferentes ambientes mostrarão diferentes graus de realismos em testes. Consequentemente veremos como se comporta o algoritmo em ambientes mais "controlados" e ambientes mais realistas.

O esperado seria obter resultados mais concretos para misturas virtuais e resultados menos concretos para misturas reais em ambiente fechado.

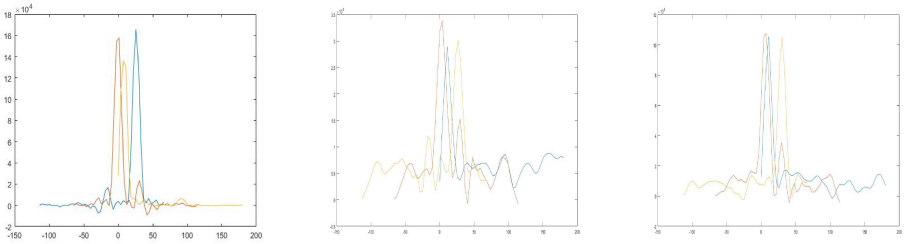
Passo a mostrar os resultados obtidos nas 3 situações para o mesmo tipo de ficheiros áudio. Assim na figura 4.3 estão ilustrados os gráficos obtidos para as 3 situações para um evento sonóro de fala.



(a) Mistura Virtual (b) Ambiente fechado (c) Ambiente aberto

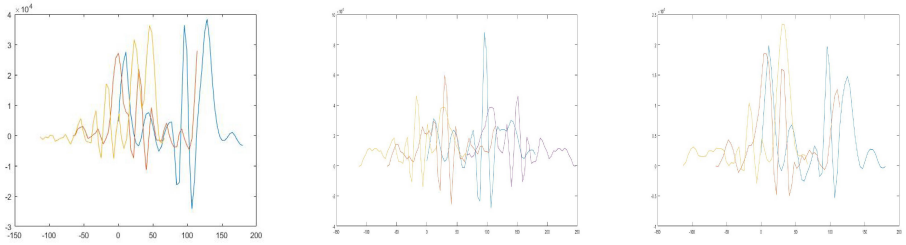
Figura 4.3: Ficheiro de "fala" nos 3 ambientes estudados - 1 evento a 110 graus - Topologia em Triângulo - evento de "fala"

Repito a demonstração mas agora para o ficheiro com sons de cães a ladrar e buzinas de carros (figura 4.4) .



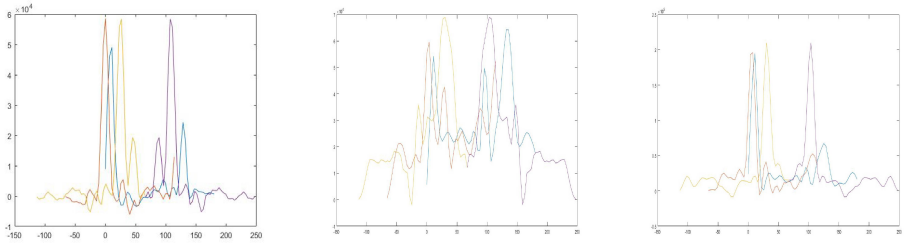
(a) Mistura Virtual (b) Ambiente fechado (c) Ambiente aberto

Figura 4.4: Ficheiro de "ladrar e buzina" nos 3 ambientes estudados - 1 evento a 0 graus - Topologia em Triângulo - evento "ladrar" seguido de "música"



(a) "fala" Mistura Virtual (b) "fala" em ambiente fechado (c) "fala" em ambiente aberto

Figura 4.5: Ficheiro de "fala" nos 3 ambientes estudados - 2 eventos simultâneos a 0 e 110 graus - Topologia em Triângulo - 2 eventos de "fala"



(a) Mistura Virtual (b) Ambiente fechado (c) Ambiente aberto

Figura 4.6: Ficheiro de "ladrar e buzina" nos 3 ambientes estudados - 2 eventos simultâneos a 0 e 110 graus - Topologia em Triângulo - eventos "ladrar" e "música"

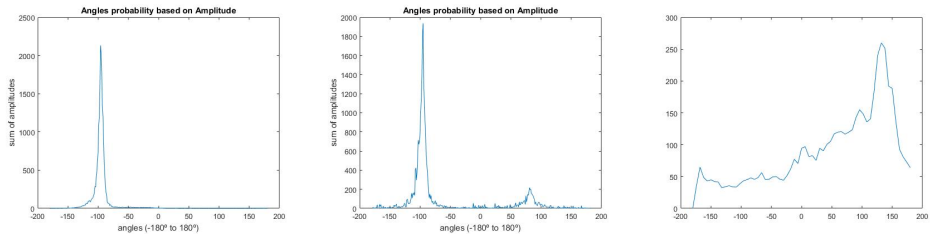
4.3.2 Arquitetura Ambisonic

Para a topologia Ambisonic efetuei diferentes testes com ruído de cidade adicionado virtualmente e também ficheiros gravados em ambiente real de cidade. Efetuei testes com eventos sonóros de cães a ladrar, travagem de carro e sirene.

Passo a mostrar a comparação entre resultdos obtidos em ambiente virtual sem ruído, ambiente virtual com ruído e ambiente real.

Na figura 4.7 são mostrados os gráficos para os 3 ambientes de testes para um único evento. Na figura 4.8 estão ilustrados, também, 3 gráficos para os referidos ambientes de teste mas para 2 eventos em simultâneo.

Como se pode verificar à medida que os testes se tornam mais reais, e menos de "laboratório" os resultados tornam-se mais ruidosos. Ainda assim o algoritmo foi capaz de classificar corretamente, na maior parte dos casos, os eventos sonóros presentes nos ficheiros gravados em ambiente real (cidade)
.



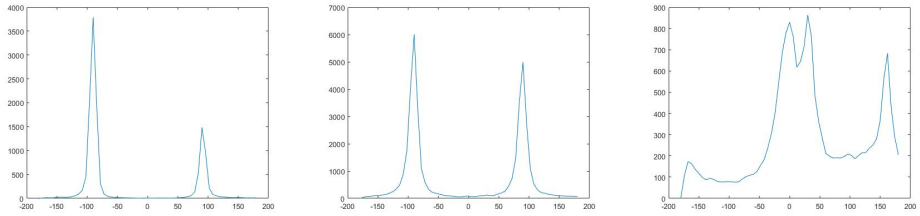
(a) Mistura Virtual sem Ruído - Evento a 270 graus - evento "disparo"

(b) Mistura Virtual com ruído - Evento a 270 graus - evento "disparo"

(c) Mistura Real - Evento a cerca de 155 graus - evento "buzina"

Figura 4.7: 3 Ambientes de teste para topologia Ambisonic com 1 evento único

Após obter estes gráficos foi necessário fazer um "filtro" que retornasse apenas os "picos" de maior relevância. Assim criei um algoritmo que apenas identificava um aspaspico no gráfico como sendo um evento se tivesse uma determinada mudança de amplitude em relação aos valores da sua vizinhança. Desta forma elimina-se o evento falso ("pico") presente, por exemplo, na figura 4.7b a cerca de 90 graus. Ou mesmo todos os pequenos "picos" presentes na figura 4.7c retornando apenas o "pico" com maior amplitude e maior mudança de amplitude em relação à sua vizinhança, a cerca de 155 graus.



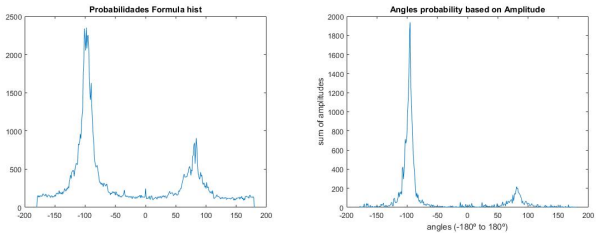
(a) Mistura Virtual sem Ruído - Eventos a 90 e 270 graus - eventos "tra- vagem" e "sirene"

(b) Mistura Virtual com ruído - Eventos a 90 e 270 graus - eventos "tra- vagem" e "sirene"

(c) Mistura Real a cerca de 25 e 155 graus - eventos "ladrar" e "buzina"

Figura 4.8: 3 Ambientes de teste para topologia Ambisonic com 2 eventos em simultâneo

Um outro aspecto aboradado neste projecto foi a maneira como era calculado o ângulo do evento na arquitetura Ambisonic. Assim, como referido anteriormente, foram apresentados dois métodos de cálculo presente nas expressões 3.12 . Os diferentes resultados otidos estão expostos na figura 4.9 . Como se pode verificar, o histograma proposto em 2.2 apresenta mais ruído e tornava-se em muitos casos menos perceptível de identificar os ângulos correctos.



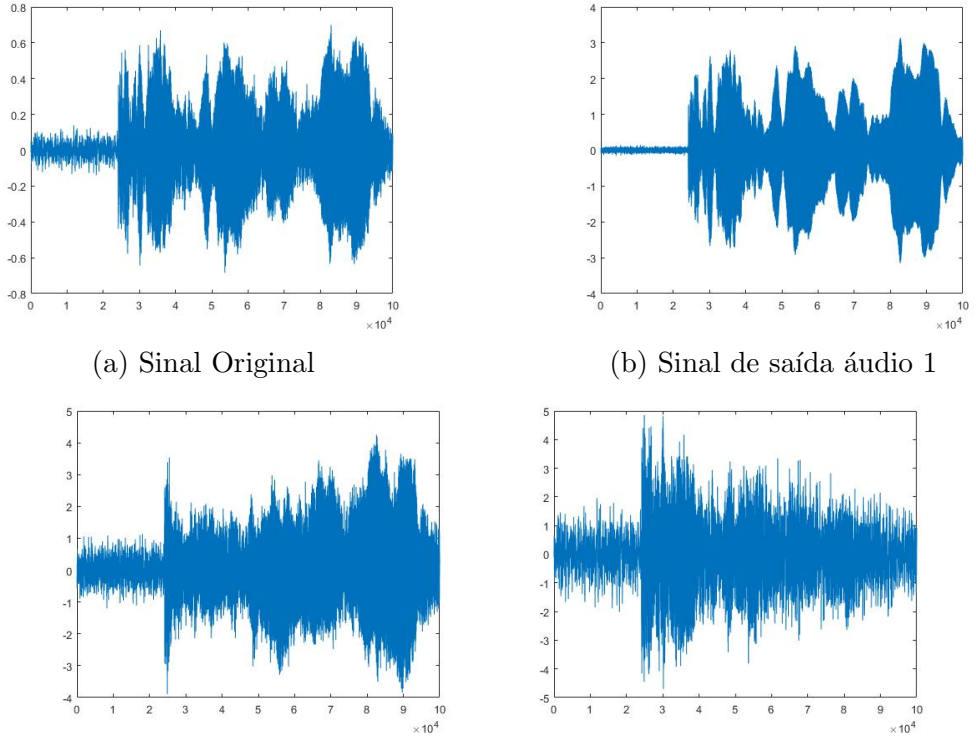
(a) Histograma Simples

(b) Histogrma com Soma de Amplitudes

Figura 4.9: Comparação de cálculos de ângulos com os diferentes Histogramas

4.4 Separação de Eventos

A separação de eventos, como referido anteriormente, foi efetuada através do aloritmo de *fastICA* .



(a) Sinal Original

(b) Sinal de saída áudio 1

(c) Sinal de saída áudio 2

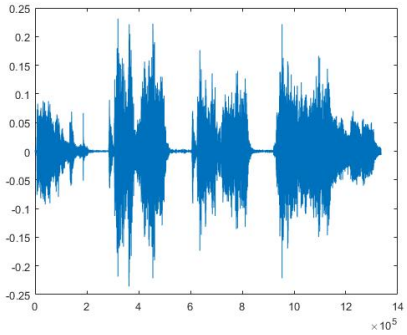
(d) Sinal de saída áudio 3

Figura 4.10: Separação por *fastICA* - Misturas Virtuais - 2 eventos - Topologia em Triângulo - eventos "Sirene" e "ruído de fundo em ambiente de cidade"

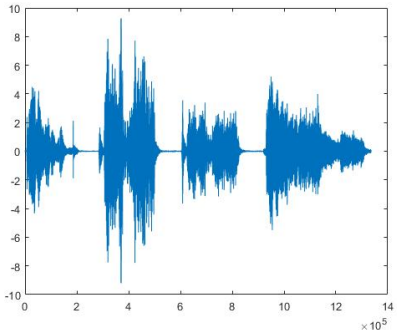
Foram efetuados testes para ambas as arquiteturas de microfones. Entre as duas, a maior diferença terá sido a de que na arquitetura Ambisonic existem 4 microfones enquanto na arquitetura em triângulo existem apenas 3. Como mencionado anteriormente, estes factos são relevantes porque os algoritmos de *ICA* separação um número máximo de eventos igual ao número de sinais analisados por este mesmo algoritmo.

Quando aplicamos o mesmo algoritmo a ficheiros com misturas reais, os resultados não são tão bons, como era expectável. Quando ouvidos os ficheiros de som à saída é possível distinguir qual foi o evento a ser sobressaído num específico sinal mas ainda assim acaba a ser razoavelmente audível todos os eventos sonóros presentes na origem, não isolando totalmente cada um dos

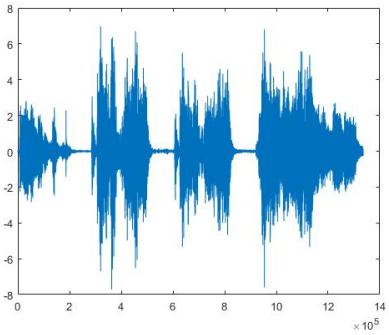
eventos sonóros.



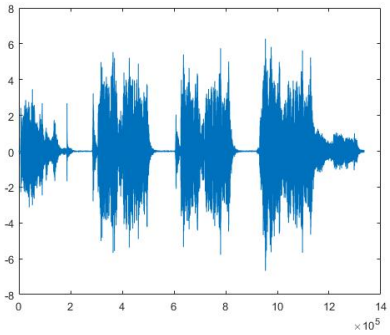
(a) Sinal Original



(b) Sinal à saida 1



(c) Sinal à saída 2



(d) Sinal à saída 3

Figura 4.11: Separação por *fastICA* - eventos "ladrar" , "buzina" e "sons metálicos" - Misturas Virtuais - 3 eventos - Topologia em Triângulo

Para a arquitetura Ambisonic, num primeiro teste, tentei separar quatro eventos sonóros, sendo eles som de "disparo" , "sirene" , "cães a ladrar" e "ruído de fundo".

A figura 4.14 mostra um dos 4 sinais originais de áudio. Nota que tratando-se dum ficheiro de áudio relativo à topologia Ambisonic o ficheiro de áudio contém 4 (quatro) canais. Deste modo o algoritmo *ICA* permite uma separação de 4 eventos sonóros máxima. Estes ficheiros de teste foram criados virtualmente em *software* próprio.

A figura 4.13 mostra o resultado obtido com um sinal com 4 eventos distintos após processamento por algoritmo *fastICA*.

O aspeto visual destes gráficos é bastante animador pois o algoritmo

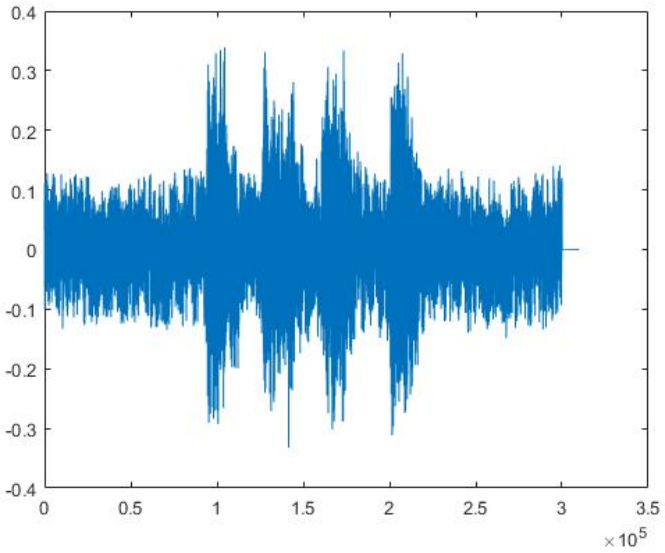


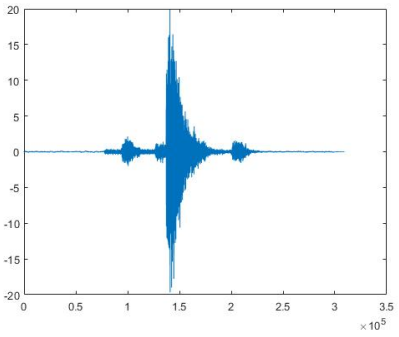
Figura 4.12: Sinal Áudio Original com 4 Sons Distintos - "disparo" , "sirene" , "ladrar" e "ruído de fundo"

fastICA retorna sinais bastante diferentes entre si, cada um salientado cada um dos eventos sonóros. Apesar desse facto, os resultados na prática não são assim tão esclarecedores como parecem ser a uma primeira vista da figura 4.13 mas ainda assim destacam com considerável relevância cada um dos eventos presentes nos ficheiros áudio originais.

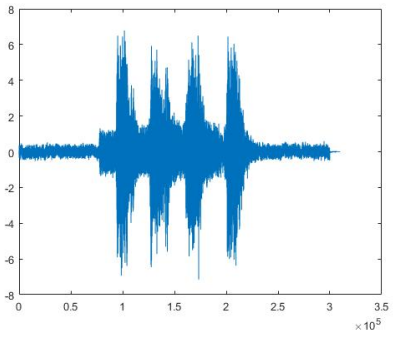
O nível de separação produzida neste teste seria, provavalmente, suficiente para fazer classificação de eventos através de algoritmos de *Aprendizagem Automática* ou *Machine Learning*. Isso será algo a ser desenvolvido numa fase mais avançada do seguimento deste projeto. Isto mesmo estará falado e referido na secção posterior deste projeto de *Conclusões e Trabalho Futuro*.

Agora mostro o resultado obtido para misturas reais gravadas em ambiente de cidade. Aqui havia apenas ruído de fundo e sons de cães a ladrar e buzina separadamente.

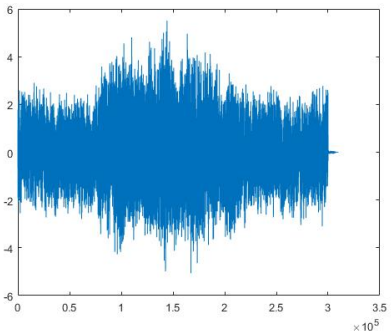
Apesar dos gráficos se diferenciarem bem entre si, o algoritmo *fastICA* salientou diferentes eventos em diferentes sinais de saída, o resultado prático não foi tão bom como no teste efetuado anteirormente para misturas áudio virtuais. Ainda assim penso que permitiria um bom desempenho de algoritmos de classificação aplicados aos ficheiros áudio respetivos aos gráficos da figura 4.15.



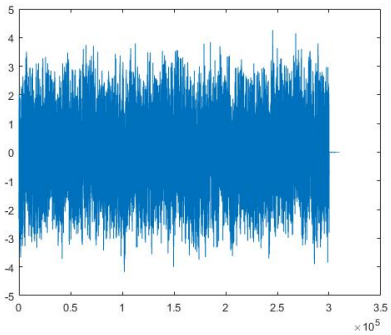
(a) Sinal à Saída 1



(b) Sinal à Saída 2



(c) Sinal à Saída 3



(d) Sinal à Saída 4

Figura 4.13: Separação por *fastICA* - Misturas Virtuais - 4 eventos - Topologia Ambisonic - eventos "disparo" , "sirene" , "ladrar" e "ruído de fundo"

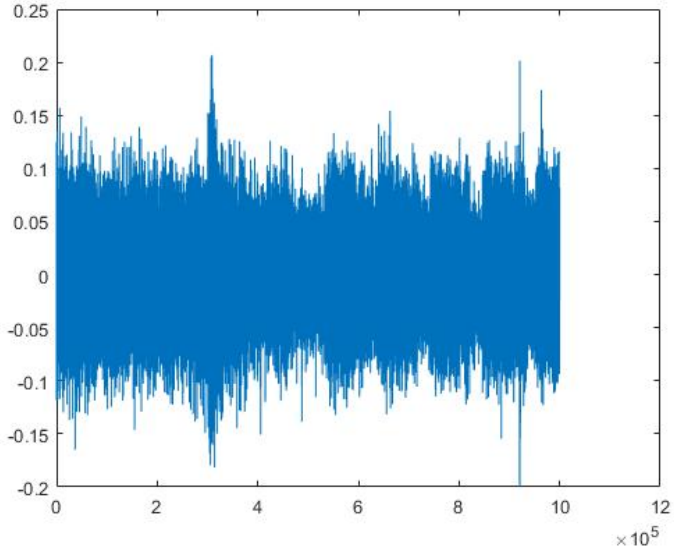
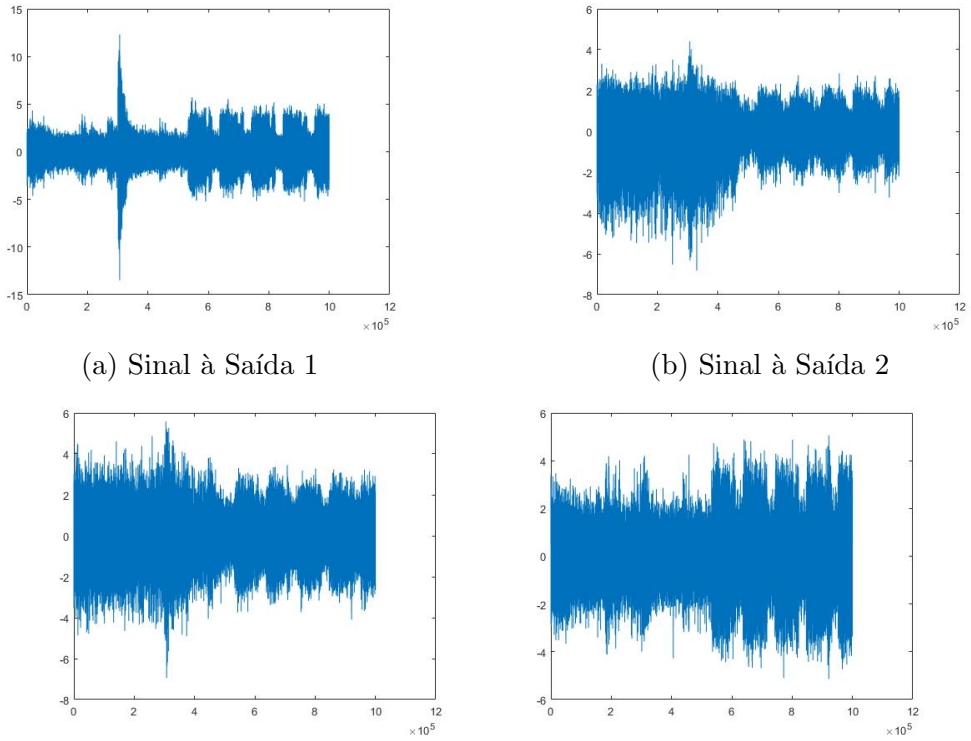


Figura 4.14: Sinal Áudio Original com 3 eventos - eventos "ladrar" , "buzina" e "ruído de fundo"



(a) Sinal à Saída 1

(b) Sinal à Saída 2

(c) Sinal à Saída 3

(d) Sinal à Saída 4

Figura 4.15: Separação por *fastICA* - Misturas Reais - 3 eventos - Topologia Ambisonic

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões

Após a conclusão deste trabalho posso verificar que a topologia Ambisonic é bastante mais eficaz para a localização de eventos. Se a isso somarmos o facto de que esta topologia pode localizar um evento sonóro num espaço 3D, calculando não só o ângulo horizontal (azimute) mas também o ângulo vertical (elevação), posso dizer que a topologia Ambisonic é vantajosa em relação à topologia em triângulo.

O algoritmo para o referido Ambisonic é também mais fácil de implementar e bastante mais rápido de executar. Quanto a esta topologia, posso dizer também que, o algoritmo de histograma baseado nas somas das amplitudes é mais eficiente do que o histograma simples (ver 3.12)

Quanto à topologia de triângulo é perceptível que o algoritmo usado, *GCC-PHAT* não conseguiu obter os resultados desejados, ficando bastante aquém do algoritmo e topologia Ambisonic. No trabalho futuro pretendo desenvolver o algoritmo de localização, baseado em tempos de atraso, que se orienta através dos *pesos*, w, de separação de eventos. Como se pode perceber este algoritmo tem como base o conteúdo teórico do algoritmo *ICA* mas que aqui seria aplicado à localização.

Quanto ao algoritmo de deteção de eventos posso concluir que é bastante eficaz a eliminar momentos de silêncio (apenas ruído). Assim posso dizer que o algoritmo de *Spectral Flux* é bastante preciso, sendo que, obviamente, funciona tão melhor quanto menos ruído houver. Para o algoritmo de deteção de eventos é vantajoso que haja filtros de ruído. O mesmo não se verifica

obrigatoriamente para os algoritmos de localização pois esta pode ser afetada pela falta de informação relevante. Informação essa que poderá ser recolhida sem querer no processamento dos filtros de ruído.

Desta forma acabei por efetuar a grande maioria dos meus testes de localização e separação de eventos sem recurrer a nenhum algoritmo de redução de ruído. Sendo que os resultados para essa mesma localização e separação de eventos ilustrada no capítulo anterior demonstram, exclusivamente, resultados para ficheiros de áudio aos quais não tinham sido aplicadas remoções de ruído.

5.2 Trabalho Futuro

No futuro pretendo prosseguir este projeto para desenvolver um algoritmo que identifique de forma mais eficiente e exacta a localização dos eventos.

Um outro objetivo muito importante é o de classificação de eventos. Esta secção já irá entrar na área de aprendizagem automática ou *Machine Learning* e será talvez a parte mais desafiante num projeto final que englobe esta área e a de processamento de sinal (DSP).

Para esta identificação de eventos seria essencial uma boa separação de eventos sonóros, abordada neste projeto através do algoritmo *fastICA*.

A classificação de eventos permitirá enviar avisos para centrais de bombeiros ou polícia caso o evento sonóro captado seja relevante para alguma destas entidades. Este objetivo seria a ideia final do primeiro projeto (abordagem de microfones Ambisonic) enquanto o projeto de topologia em triângulo tinha como um possível objetivo final o de redirecionar a direção duma câmara usada em video-vigilância.

Em trabalho futuro, espero também, desenvolver, como referido na secção anterior, o algoritmo de localização *TDOA - Time Difference of Arrival* - com base nos fundamentos do algoritmo *ICA*. Pela pesquisa que fiz e pelo documento que obti acerca destes algoritmos (são 3 algoritmos estudados no documento) posso presumir que os resultados serão melhores podendo, eventulamente, superar os resultados obtidos para o Ambisonic.

Os obejtivos finais são interessantes e úteis no contexto real pelo que seria cativante cumpri-los.

Bibliografia

[1] Noise Removal in Speech Processing Using Spectral Subtraction

[2] Estimation of Direction of Arrival of Multiple Sound Sources in 3D Space using B-Format

[3] Onset Detection